Wayne State University

CSC 4421 - Winter 2017
Computer Operating Systems Labs
Lab 9 - Process Scheduling Simulator

Instructor 001: David Warnke
Instructor 003: Rui Chen

Points Possible: 100
Due: April 3, 2017

Submission

Every red star represents something you need to submit. Add each item with a red star to a .doc, .docs, or .pdf file, and **submit just that file to blackboard**. All images should be imbeded in the file submitted. Do not submit a zipped folder.

Tasks

Task 1* (50 points)

Process Simulator In this task, you are required to run a process simulator and put your answers to the following questions in a file Task_1_lastname_firstname.docx. Use one file with all of the answers and screen-shots. Upload the file to Blackboard(no need to zip).

1. Download process scheduler from the following web site.

   `http://vip.cs.utsa.edu/simulators/`

2. Use "sudo apt-get install" in the command line to install c shell and java. You should use install "default-jre" and "csh". If you don't know how to install java in linux, please google it. Or you can use Windows.

3. Type "csh" to change into c shell mode. Start the simulator by run "./runps" or in Windows system, you should click runps.bat file.

4. *By default, the simulator will run two experiments. The first experiment uses FCFS scheduling algorithm, and the second one uses SJF. The program creates lots of logs. Some information is shown as*

soon as the experiment is done. Some other inforamtion can be accessed from various other buttons on the gui, such as "All(#)" which shows process information, and "Show All Table Data" which shows some information on each of the runs. Click the 'Run Experiment' button in the simulator. Explore the simulator and answer the following questions:

(a) * How many events were created for each experiment?

(b) * How many milliseconds did each take?

(c) * In each experiment, how many processes are created?

5. Restart the simulator again, this time you will only run the FCFS algorithm (by clicking Run so that it changes from "Run All" to "Run: myrun 1").

   (a) Look at the process information by clicking the 'All(#)' button in the simulator.

   (b) Look at the CPU history information by clicking the 'CPU History(#)' button in the simulator.

   (c) * Explain how process 30 is scheduled in relation to the other processes. Note that it is scheduled more than once due to I/O, so explain each time it is scheduled.

6. Restart the simulator again, this time you will only run the SJF algorithm (by clicking Run so that it changes from "Run: myrun 1" to "Run: myrun 2").

   (a) Look at the process information by clicking the 'All(#)' button in the simulator.

   (b) Look at the CPU history information by clicking the 'CPU History(#)' button in the simulator.

   (c) * Explain how process 30 is scheduled this time.

   (d) * Explain why process 30 is scheduled differently compared to the process 30 in the last experiment, considering the different algorithms used.

7. Restart the simulator again. Run both scheduling algorithms at once (Run All). Draw the following figures (by clicking on the "Graph Type" button until it shows the appropriate graph, and then clicking the "Draw" button.)

   (a) * Draw the waiting time, waiting time box and waiting ratio figures. Include the figures in your file.

      i. * Which algorithm has shorter average waiting time? Explain the reason.

      ii. * Why is the waiting time for the process with the greatest wait for SJF larger than FCFS?

      iii. * Give some comments about FCFS's and SJF's waiting time in your own words. i.e. Pros and Cons.

   (b) * Draw 'turnaround' and 'turnaround box' figures. Include the figures in your file.

      i. * What does turnaround time mean?

      ii. * From the figures, which algorithm has lower minimum turnaround time? Explain the reason.

   (c) * Draw 'Gantt Chart' for both FCFS and SJF. Include the figures in your file.

Task 2

Read the following guide about the process scheduling simulator. Learn how to change simulation settings.

Specifying an Experiment

An experiment is specified by two files. Each file has a name consisting of a base name and an extension. The files are:

- The experiment file with extension **.exp** specifies a number of experimental runs. Each run is specified by a base name and a possible list of modifications.
- The experimental run file, which is also called the run file for short, has the extension **.run** and contains information about the scheduling algorithm to use, the processes to run, and when the processes arrive.

Time

The simulator uses a virtual time which is represented by a floating point value of unspecified units. When the simulator is started, the time is set to 0.0. The simulator is event driven, and events that occur at the same time may occur in any order.

Processes

A process is specified by the following information:

- Arrival time
- Total CPU time
- CPU burst time distribution
- I/O burst time distribution

Probability Distribution

The distribution supported by the simulator are

- Constant distribution. Key word: constant
- Exponential distribution. Key word: exponential
- Uniform distribution. Key word: uniform

The structure of a run file

An experimental run contains all of the information needed to run the simulator on one collection of processes. An experimental run specifies a scheduling algorithm, a collection of processes, and when the processes arrive. A simple format for an experimental run file is like :

name //experimental run name
comment //experimental run description
algorithm //algorithm description
numprocs //number of processes
firstarrival //first arrival time
interarrival //inter-arrival distribution
duration //duration distribution
cpuburst //cpu burst distribution
ioburst //io burst distribution
basepriority //base priority

Here is a sample experimental run file that must be stored in the file **myrun.run**.

```
name myrun
comment This is a sample experimental run file
algorithm SJF
numprocs 20
firstarrival 0.0
interarrival constant 0.0
duration uniform 500.0 1000.0
cpuburst constant 50.0
ioburst constant 1.0
basepriority 1.0
```

Listing 1: myrun.run

This file specifies a run of 20 processes using the shortest job first algorithm. The first process arrives at time 0.0. The inter-arrival times are all 0.0, so all processes arrive at the same time. Each process has a duration (total CPU time) chosen from a uniform distribution on the interval from 500 to 1000. All processes have a constant cpu burst time of 50 and a small constant I/O burst time of 1.0. The base priority must be present in the file, but it is currently not used by the simulator.

The structure of an exp file

An experiment specifies a number of experimental runs that are to be made, compared and analyzed. There are an arbitrary number of lines specifying experimental runs to be made.

The simulator allows you to do this by specifying the same experimental run in each case and giving a new value to one or more parameters. The general format for a run line in the experiment file consists of the word **run** followed by the name of an experimental run, followed by a list of modifications to that experimental run. So it looks like this

name	experiment_name	
comment	experiment_description	
run	run_name1	optional_modification_list1
...		
run	run_namen	optional_modification_listn

Here is an example experiment file that must be stored in the file **myexp.exp**

```
name myexp
comment This experiment contains 3 runs
run myrun
run myrun cpuburst uniform 10 90
run myrun cpuburst exponential 50
```

Listing 2: myexp.exp

This experiment file makes three runs. All of the runs are based on the run file **myrun.run** above. In the second run the CPU burst distribution is changed to be uniform in the interval from 10 to 90. In the third run the CPU burst is an exponential distribution with mean 50. Here is a list of the key words you can modify:

- **numprocs**: the number of processes to create.
- **firstarrival**: the arrival time of the first process, a floating point number.
- **basepriority**: the base priority of the processes created. This base priority is not used unless the simulator is using priorities.
- **interarrival**: the distribution of the interarrival times of the processes.
- **duration**: the distribution of the total CPU time used by the processes.

- **cpuburst**: the distribution of the CPU burst times of the processes.
- **ioburst**: the distribution of the I/O burst times of the processes.
- **algorithm**: the scheduling algorithm to use.

Scheduling Algorithms Supported

- Round Robin (RR):
- First-Come/First-Served (FCFS): Processes are taken from the ready queue in the order that they arrived. Once a process is using the CPU, it stays there until its CPU burst expires. It then goes into the I/O waiting queue until its I/O burst expires.
- Shortest Job First (SJF): The next process to be removed from the ready queue is the one with the shortest CPU burst time. If there is more than one process with the smallest time, the one that arrived first is taken.

A scheduling algorithm is specified with a string and a possible optional floating point parameter depending on the scheduling algorithm. The following strings are used to specify scheduling algorithms:

- **RR** quantum represents the Round Robin algorithm with the given quantum.
- **FCFS** represents the First-Come/First-Served algorithm.
- **SJF** represents Shortest Job First scheduling.

Task 3* (25 points): FCFS and SJF

- Modify the `myexp.exp` file to have 2 run lines that will use the following algorithms:

  FCFS, SJF, On each run line, make sure that both the algorithm and key are appropriately set.

- Now modify the myrun.run file so that there will be a total of 20 processes with the following properties:

  - Do not change the seed line of the run file.
  - All processes arrive at time 0.0.
  - All processes have a constant duration of 100.
  - All have constant I/O bursts of 10.
  - All have basepriority 1.0.
  - The first 10 processes have CPU bursts uniformly distributed between 2 and 8.
  - The last 10 processes have CPU bursts uniformly distributed between 50 and 60.

- Run the simulator using FCFS and SJF and draw a 'Gantt chart' for each of the two (). Add these two figures along with the revised myexp.exp and myrun.run files to your file.

- Did you find any difference from those two figures? Briefly explain the impact of these two scheduling algorithms?

Task 4* (25 points): Round Robin

- Modify the `myexp.exp` file to have 4 run lines that will use the following algorithms:

  RR 1, RR 10, RR 15, RR 50, RR 100 and RR 101. On each run line, make sure that both the algorithm and key are appropriately set like this:

  ```
  run myrun algorithm RR 100 key "RR 100"
  ```

- Now modify the myrun.run file so that there will be a total of 20 processes with the following properties:

  - Do not change the seed line of the run file.
  - All processes arrive at time 0.0.
  - All processes have a constant duration of 1000.
  - All processes have constant CPU bursts of 101.
  - All have constant I/O bursts of 100.

- Run the simulator using the revised experiment setting. Draw the 'Gantt chart' for each run. Submit these 4 figures along with the revised myexp.exp and myrun.run files.

- Briefly describe the Round Robin algorithm, and the impact of using different quanta. Also note that this simulator does not take into account context switch overhead.