CS 452 Winterim Lab Assignment – Process Scheduling

Introduction

Modern operating system employ scheduling algorithms that are based on the round-robin concept as described in class. The scheduling policy is also based on ranking processes according to their priority. Complicated algorithms are used to calculate the current priority of a process and scheduling decisions are based on the value of each process's priority.

For this lab, you are to implement the Multi-level Feedback Queue Scheduler (MFQS)

Program Efficiency

Your program must be efficient, i.e., it must not take a long time to compute the result, especially during a stress test. The following is a breakdown of the grading of this project:

Design (Efficiency):	20%
Correctness:	40%
Stress Test:	40%

MFQS

You are to implement the scheduler described in class. Note that whenever a tie has to be resolved, the priority of the process is considered. Listed below are some of the parameters that must be considered in your simulation.

0 Number of queues:
variable, upper bound = 5 (ask user to input number)

1 Scheduling algorithms for each queue:
a. Round Robin for all except last queue (FCFS).
b. Time quantum: doubled for each subsequent queue below it.

2 Method used to determine when to demote a process:
Processes that used up their time quantum and still cannot complete are demoted.

3 Ageing: when a process waits in a queue for more than some specified ageing interval value (value is prompted), it is promoted to the next queue up. Apply this only to the last queue, i.e., only processes waiting in the last queue are permitted to age up, i.e., processes in any other queue do ***not*** age up.

Statistics To Collect for All Algorithms

0. Process information for Gannt chart construction, e.g., start/end time spent in CPU, etc.
1. Average Waiting Time.
2. Average Turnaround Time (all these times must be calculated in the program)

Process Characteristics as Input Parameters

The following is an example of the information that will be provided to you in your project demonstration phase:

P_ID	Burst	Arrival
1	8	0
2	4	2

I will be providing you with a file that has the information described above during the demo (in that order). There is a test file in the **/tmp** directory of **phoenix1.cs.uwec.edu** of size 1MB for you to stress test your program.

Project Requirements

Your program should allow the user to allow users to enter interactively input values for process ID, burst times, and arrival times of each process.

To be eligible for extra credit, your program should draw the Gannt chart

What do I look for in the output?

1. A chronological set of statistics - in the absence of a Gannt chart - displaying the entire simulation run of the MFQS algorithm. Shown below is an example of the expected format of the output. Failure to follow the format will result in a deduction of 10 points.

```
Running MFQS ……

Enter number of queues: 5
Enter time quantum for top queue: 4
Enter ageing interval: 4

Process 1: arrives @  0
Process 1  runs @     0
Process 2: arrives @  2
Process 1: switched @ 4
Process 2: runs @     4
      :
      :
Process 1: finished @ 8
      :
Ave. wait time = xxxxx      Ave. turnaround time = xxxxx
Total processes scheduled = 1000
```

Project Submission

The following must be submitted:

1. Name your source code file as follows:
   <lastname><first initial>.cc, e.g., **smitha_hossb.cc**

2. Drop your file in the W drive:

   W Drive → c s → Tan → cs452 → Winterim_MFQS_Lab

CS 362 – Operating Systems Lab Assignment Cover Page

Name(s) _____ Lab Assignment No.: 2

1. Include a copy of your source code, a design document (if required), and a sample run
2. Fill in the table of contents including the names of routines and the corresponding pages.
3. Indicate the status of your program by checking one of the boxes.
4. Submit the assignment at the beginning of the class on the due date.

Program Status: (check one box)
□ Program runs with user-defined test cases. □ Program runs with some errors.
□ Program compiles and runs with no output. □ Program does not compile.

Workload:	Written By			
Design document				
Paper				
Test cases				
Program modules (classes)	Module name	Written by	Module name	Written by
List only major modules				
that are typically more				
than 50 lines of code				

Grading:	Points	Score
Program correctness	40	
Code Readability	10	
Inline documentation	10	
Design	25	
Technical Paper	15	

Late Penalty	-20 per day
Total	100

Honor Code

Department of Computer Science

University of Wisconsin – Eau Claire

As members of the University of Wisconsin – Eau Claire community and the computer science discipline, we commit ourselves to act honestly, responsibly, and above all, with honor and integrity in all areas of campus life. We are accountable for all that we say, read, write, and do. We are responsible for the academic integrity of our work. We pledge that we will not misrepresent our work nor give or receive illicit aid. We commit ourselves to behave in a manner which demonstrates concern for the personal dignity, rights and freedoms of all members of the community and those that depend on the expertise we possess.

For all course work in the Department of Computer Science, students will write and sign (if printed) the following: "**I have abided by the Department of Computer Science Honor Code in this work**."

I accept responsibility to maintain the Honor Code at all times.

Name _____

Signature _____

Date _____