CS471: Operating System Concepts
Course Project - Due: December 08, 2024
Fall 2024 Points
100

You may work in teams of 1 or 2 members. Choose your own team members.

Provide a video explanation of your code that does not exceed ten minutes in length and show how your code is executed in the video. Submit your code and the recorded video in Canvas. ***If you do not submit the video or fail to demonstrate the execution of your code in the video, you will lose 20% of your marks.***

The project deals with two problems related to operating systems:

1. CPU Scheduling (50 points)
2. Process Synchronization (50 points)

The background material required to solve these problems has been covered in chapters 3-6 of the textbook. You can choose either **Java or C++** as a programming language for the project. Solutions to each of these problems will be graded separately. You are required to submit the following by the deadline. You will submit a single directory that has two subdirectories for each of the problems. The directory you submit will have the following structure:

CS471PROJECT (main dir)

---------- **Recorded Video**

---------- **CPUSCHED** ***(subdir 1)***

---------- README file with directions (make sure to include actual commands that can be copied and pasted to run)

---------- SOURCE code (well documented)

---------- Executable code

---------- Sample input data files

---------- Sample output file(s)

---------- **PRODUCER-CONSUMER** ***(subdir 2)***

---------- README file with directions (make sure to include actual commands that can be copied and pasted to run)

---------- SOURCE code (well documented)

---------- Executable code

---------- Sample input data files

---------- Sample output file(s)

Details of each of these problems are in the appendix.

Appendix

Problem 1: CPU Scheduling

This part of the projects simulates a CPU scheduler. Since it is a simulation, there are no real processes to be scheduled. Instead, you simulate the arrival of new processes. Whenever a new process arrives (in simulation) into the ready queue, the CPU scheduler is invoked. Each simulated process has the following parameters: <Process ID, Arrival time, Priority, CPU burst units>. Each of these is an integer parameter.

For example, if a process <100, 20, 2> is read by your simulator, it means a new process P arrived at the system at simulation time 100, it executes for a CPU burst time of 20 units before finishing and leaving the system, and its priority is 2. At the time of invocation of the scheduler, the user indicates the type of scheduling to be enforced. You are required to implement the following three scheduling types:

1. FIFO
2. Priority with preemption

Each run will handle scheduling of 500 (simulated) processes. In other words, as soon as the number of processes that have completed CPU execution reaches 500, you can stop running the program and print the following statistics. All times are expressed in terms of CPU burst units. So they are not actual elapsed time (in msec) but simulated times.

Statistics for the Run

Number of processes: 500

Total elapsed time (for the scheduler):

Throughput (Number of processes executed in one unit of CPU burst time):

CPU utilization:

Average waiting time (in CPU burst times):
Average turnaround time (in CPU burst times): **Average response time (in CPU burst times)**:

Since there are only 500 processes, it may be easy to read all 500 processes data first (from datafile1-txt), and store the records in a queue in the program, and then process them. This is an easy way to do it. But you can choose any method you like.

The following formulas will be utilized for calculating Problem 1:

Total elapsed time: The amount of time from initiation to termination of the application.

Throughput (Number of processes executed in one unit of CPU burst time) so you can calculate it using the formula: Total burst time(It is the sum of all given burst length) / Total number of processes

CPU Utilization: The CPU Utilization measures the percentage of the time that the CPU is performing work so you can calculate it using the formula: Total Burst Time/ Total elapsed time

Average waiting time: total waiting time / Number of processes

Turnaround time = Burst time + Waiting time or
Turnaround time = Exit time - Arrival time
Average turnaround time: Total turnaround time / No of processes

Response time = Time at which the process gets the CPU for the first time - Arrival time **Average response time:** Total response time / No of processes

Problem 2: Producer-Consumer Problem

This follows the description in pages 253-257 of the 9th edition of the textbook. Develop the producer-consumer problem using Pthreads or Winn32 API or using any other library of your preference. You can code the program in **Java/C++** language of your choice. Test the program with several inputs (the three parameters to the main program are shown in the textbook). Definitely, test the following parameters. **Try with different sleep times**. Measure the performance of the time in terms of overall turnaround time. Tabulate your results along with the parameters used. Finally, summarize your results and give an explanation for the results.

Test case	Number of producers	Number of consumers
1	1	1
2	4	1

3	16	1
4	1	2
5	4	2
6	16	2
7	1	4
8	4	4
9	16	4
10	1	16
11	4	16
12	16	16