CS471: Operating System Concepts
Fall 2012
Course Project
Due: December 3, 2012
Points: 100

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

The project deals with three problems related to operating systems:

1. Producer-consumer problem (50 points)
2. Virtual Memory Management problem (50 points)

The background material required to solve these problems has been covered in chapters 6 and 8-9 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 using the submit command in Unix ***submit cs471*** command. 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)

---------- **PRODCONS** ***(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)
----------A report that summarizes your findings from the experiments

---------- **VMEMMAN** ***(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)
----------A report that summarizes your findings from the experiments

Details of each of these problems are in the appendix.

Appendix

Problem 1: Producer-Consumer Problem

Here, we have a set of **p** producers and **c** consumers, each running as a single thread. They are synchronized via shared buffer of size **b** (i.e., it can accommodate b items). Each buffer item contains the following information: Sales Date (DD/MM/YY), store ID (integer), register# (integer), sale amount (float). Each item represents a sales record from a specific cashier register in a particular location of a retail-chain. Thus, each producer reports sales from a specific store location. Each consumer represents an entity that reads sales records and computes sales statistics locally. Each buffer item is consumed by one-and-only-one consumer. When all sales records have been read (indicated by a special flag set by another designated thread), each consumer adds its local statistics to the global statistics (in the shred space). It also prints its own local statistics along with its ID. In addition, your main program (parent process) prints the overall (global) statistics.

The statistics to be maintained are:
Store-wide total sales
Month-wise total sales (in all stores)
Aggregate sales (all sales together)
Total time for simulation (from begin to end)

Each producer produces records randomly. Assume that the DD field is 1-30, MM is 01-12, and YY is always 06. Store IDs are in the 1 to **p** range (where **p** is the number of producers). The register numbers range from 1-10 for any store. The sale amount in each item can range between 0.50 and 999.99. Each producer generates its record with random data. Run the program until 10,000 items are produced by all producers together. Obviously, the number of items produced so far (by all producers) need to be maintained in shared memory. Each producer is assigned a fixed store ID when it is created. It has the following structure:

```
 While the total number of items generated by all producers is less than 10,000
do
{
   Randomly generate DD, MM, register#, sale amount.
   Create a sales record and place it in the shared buffer.
    Increment the number of records count (in the shared memory)
    Randomly sleep for 5-40 milliseconds
}
```

Your report should clearly indicate the places where shared variables and semaphores were employed by your code to manage shared variables. Run your program for p=2, 5,

and 10 (i.e., try all three possible values for number of producers). Similarly, run it for c=2, 5, and 10. So there will be 9 runs in total. Your report should compile the results from all 9 runs and make a comparison of the time. Use pseudo random number generators to generate data.

Problem 2: Virtual Memory management problem

Compare the performance of the following page replacement algorithms: FIFO, LRU (Least recently used), MRU (most recently used), and optimal. You will be provided with a file containing the virtual addresses that are being referenced by a single program (a process). Run your program with the following parameters:

Page size: 512, 1024, 2048 (words)
Number of **frames** allocated to the process: 4, 8, 12
(So you will have 9 runs, with each page size and **number of frames** combination. Each run contains statistics for each of the four page replacement algorithms.
You must collect and print the following statistics

Page Size #of pages Page replacement ALG Page fault percentage

Your report must show a summary of runs and your conclusions.