COEN 346 Programming Assignment #3

The main objective of this assignment is to simulate operating system's virtual memory management, process scheduling and concurrency/synchronization control as discussed during the Lab sessions.

The following features should be considered for this simulation:

1- Processes should be simulated by threads in the application. Each process has a starting time and duration (burst time). "processes.txt" is the input file which contains the number of processes N followed by N lines. Each line contains the process start time and duration.

2- **Process Scheduler:** A process scheduling thread should be implemented based on the "**Round Robin**" policy that schedules the processes as discussed during the lab sessions. In this assignment (#3) we assume that the simulated system has two CPUs (or CPU cores). Therefore, two processes can be executed at the same time, i.e. the scheduler must select two processes (from the head of the waiting queue) for execution during each cycle. Additionally, the time slice must be set to 3000 ms.

3- **Virtual Memory Management:** Virtual memory consists of a fixed size main memory and unlimited disk space. The main memory is divided into frames while the virtual memory is divided into pages. A page has the same size as a frame. The size of the main memory will be given as an input in a file called "memconfig.txt". This file contains a number indicating number of frames in main memory. The disk space is assumed to be unlimited. Notice that frames in the main memory should be simulated by an array (or vector) in the actual computer physical memory, while virtual memory pages are simulated by an array stored in a text file such as "vm.txt", which must be accessed every time we need to access the disk space.

4- Processes try to store, release and retrieve "variables" to/from the memory. Each page contains only one variableId and its value. Of course, working with memory can only happen when the process is running. Each variable in the memory has a Last Access property which is a time stamp indicating the last time this variable was accessed.

5- Virtual memory manager should be implemented with a set of APIs (functions) as follows:

   a. **memStore (string variableId, unsigned int value)**: This instruction stores the given variableId and its value in the first unassigned spot in the memory.

   b. **memFree (string variableId)**: This instruction removes the variableId and its value from the memory, so the page which was holding this variable becomes available for storage.

c. **memLookup (string variableId)**: If the variableId exists in the main memory it returns its value. If the variableId is not in the main memory but exists in disk space (i.e. page fault occurs), then it should move this variable into the memory. Notice that if no spot is available in the memory, program needs to swap this variable with the least recently accessed variable, i.e. the variable with smallest Last Access time, in the main memory. Finally, if the variableId does not exist in the main memory and disk space, the API should return -1.

6- The processes should continuously pick one command from a given list of commands located in the input file called "commands.txt". The processes should then call the suitable API of virtual memory manager based on the picked command.
   a. **memStore** [variableId] [value]: e.g. "memStore 10 5"
   b. **memFree** [variableId]: e.g. "memFree 10"
   c. **memLookup** [variableId]: e.g. "memLookup 10"

7- Output of the program should be a text file "output.txt" which includes the following events:
   a. Threads start, resume, pause and finish.
   b. Each instruction executed by each thread
   c. Page swapping between the main memory and the disk space

Programming Tips:

- In order to protect critical sections and ensure mutual exclusion use appropriate tools, semaphores for instance, within the body of each function.
- The program must work with arbitrary number of threads and arbitrary memory size.

The following can be the list of sample input / output files.

Input Files:

- processes.txt

  3
  2 1
  1 2
  2 2

- memconfig.txt

  2

- “commands.txt”

  Store 1 5

  Store 2 3

  Store 3 7

  Lookup 3

  Lookup 2

  Release 1

  Store 1 8

  Lookup 1

  Lookup 2

Output File:

Clock: 1000, Process 2: Started.

Clock: 1000, Process 2, Resumed

Clock: 1310, Process 2, **Store**: Variable 1, Value: 5

Clock: 1730, Process 2, **Store**: Variable 2, Value: 3

Clock: 2000, Process 2, Paused

Clock: 2000, Process 1, Started.

Clock: 2000, Process 1, Resumed

Clock: 2000, Process 2, Resumed

Clock: 2350, Process 1, **Store**: Variable 3, Value: 7

Clock: 2570, Memory Manager, **SWAP**: Variable 3 with Variable 1

Clock: 2580, Process 2, **Lookup**: Variable 3, Value: 7

Clock: 3000, Process 2, Paused

Clock: 3000, Process 2, Finished

Clock: 3000, Process 1, Paused

Clock: 3000, Process 1, Finished

Clock: 3000, Process 3: Started.

Clock: 2000, Process 3, Resumed

Clock: 3400, Process 3, **Lookup**: Variable 2, Value: 3

Clock: 3830, Process 3, **Release**: Variable 1

Clock: 4000, Process 3, Paused

Clock: 4400, Process 3, **Store**: Variable 1, Value 8

Clock: 4900, Memory Manager, **SWAP**: Variable 1 with Variable 3

Clock: 4910, Process 3, **Lookup**: Variable 1, Value 8

Clock: 5000, Process 3: Paused.

Clock: 5000, Process 3: Finished.

The assignment should be done in a group of two students. The deliverables consist of a well-commented code and a report. Do not include your code in the report.

The report should be at least two pages with the following sections:

- Name of group members
- High level description of the code (description of the methods/functions/threads/data structures and the flow of the program).
- A detailed conclusion, discussing you experience with concurrency control in simulating virtual memory management. You also need to analyze your program and discuss how you enable mutual exclusion in your program and the efficiency of your technique.

This assignment will count for 40% of your total mark for programming assignments. For this assignment 80% of the mark is dedicated to the code and 20% to the report.

The code and the report should be submitted through the EAS website (https://fis.encs.concordia.ca/eas/). The deliverables should be submitted through EAS by Sunday April 5th, 11 pm.