Electrical and Computer Engineering Department
ENCS339 Operating Systems, Second Semester, 2015-2016
Project 2 Instructor: Dr. Adnan H. Yahya, Due: Monday, May 23, 2016

Virtual Memory Management Simulation

This project is about memory management and virtual memory in particular. It consists of writing a simulator for experimenting with page replacement algorithms. The project will be done in **groups of up to 3 students**. The project is due on **May 23, 2016**.

General:

The primary goal for the project is to implement and experiment with page replacement algorithms. To do this, you will write a paging **simulator**. It will read in a set of data files specifying the page traces for individual jobs and will need to simulate the paging requirements of those programs. The trace file could also be generated using a random number generator that produces random page numbers for each job. The value of each number is within the size of the program (address space) which we assume given.

We will assume that we have n programs, where N is determined by observing the least significant numbers of the University ID numbers: max + plus min + 2 (but no less than 5). So if your numbers are (129, 78, 31) then the degree of multiprogramming is 8+1+2 = 11. If your numbers are (120, 71, 32) then the degree of multiprogramming is 5 (2+0=2<5).

We assume that the processes are run Round Robin and that the rate of page generation is steady/fixed and that the number of memory accesses for a job is proportional to its length, which is given at the start. So a job with 2 hours duration generates double the memory traces of the job of one hour and half the traces of a job of 4 hours duration. The process duration is given at the start or generated randomly (between 1 and 10 hours only).

Scheduling Data Format

Scheduling files contain a trace of process start time and CPU time needed (duration). The format of these files could be as follows:

PID	Start	Duration	Size
P1	0	8.58	102720
P2	3	1.36	137024
P3	4	0.03	96

Memory Traces

Memory trace files contain a list of addresses accessed by a program in Hexadecimal. The addresses are truncated to virtual **page numbers** by removing the lower 12 bits (assuming which page size??), which makes the files smaller. The process size limitation needs to be observed

Scheduling

You will need to simulate a simple scheduling algorithm that allows programs to alternate running. For this project, you only need implement round robin between runnable processes with fixed time Quantum. The simulator defines the quantum through the number of references seen (a clear approximation).

Your scheduler will keep track of the current time in terms of cycles, where a cycle is one memory reference in the memory trace. If needed you can assume 1000 cycles in a second.

Context switching should take 5 cycles. After a context switch, you should start simulating memory references from where you left off.

The scheduler should report elapsed time (time between when it started and when it completed) for each process. In addition, at the end of the simulation (when all processes have completed), it should report the TA (Turnaround) and W (wait) for the processes.

You should experiment with different quanta values.

Page Replacement

Your simulated machine will have a fixed number of frames (physical pages) to share between running processes. When the memory required exceeds the available physical memory, you will need to use disk storage.

You need to keep track of which pages are in physical memory and which are on disk. It takes 300 cycles to move a page from disk to memory. Moving pages from memory to disk is free, as this can be overlapped with computation. For each cycle that process executes, you should check whether the virtual page is in physical memory. If so, advance to the next address. If it is not in physical memory, you must simulate a **page fault** by:

1. Context switching to a new process
2. Putting the current process on a blocked queue
3. Reading the page off disk

4. Making the process ready again when the page comes back off disk

You must keep track, for each process, which pages are in physical memory and which are on disk. Assume that all pages start off on disk and you are doing demand paging. Because writing pages to disk is free, you don't have to worry about whether a page is dirty or not (and in fact, the trace does not indicate whether a memory reference is a read or a write).

Report the number of page faults experienced by each process and the total number of page faults experienced across all processes.

You should experiment with total physical memory sizes of 40 pages, 200 pages, and 500 pages.

Page replacement policies

You should simulate 2 different page replacement policies, the odd ones in the list below (1,3) or the even ones in the list below (2,4) based on the max value of the first digit of the team ID numbers, again:

1. **FIFO**
2. **Second-change FIFO**
3. **LRU**
4. **Clock**

For each trace, you should try both page replacement policies to see how they compare.

Reporting

A report of about 5 pages describing your system and explaining your results needs to be submitted.
Turn in your code of the running program through Ritaj.

Evaluation:
The evaluation will be based on the correctness of the program, thequality of the interface and the report. You may need to present your report and working program to the instructor.

Any Problems:
Consult the instructor during office hours or send a Ritaj message.

Good luck