Santa Clara University

Department of Computer Engineering

Advanced Operating Systems (COEN 383)

Project-4 (6 pts)

Instructor: Ahmed Ezzat

Swapping and Paging

The purpose of this assignment is to explore the various memory management algorithms for swapping and paging. You will write a series of small simulation programs in Java, C, or C++ (your choice). As in Assignment #2, you will generate simulated processes <Process Name, Process size in pages, arrival time, service duration> using random number generator. You should not need to make any process management system calls.

Workload Generation

Assume main memory has 100 MB with page size of 1 MB. Processes have randomly and evenly distributed sizes of **5, 11, 17, and 31** MB. Processes have randomly and evenly distributed service durations of **1, 2, 3, 4, or 5** seconds.

Assume that free pages are structured as a linked list, and when a process needs a free page we pick the page from the head of the free linked list.

Generate around **150** new random processes into a Job Queue linked list sorted based on arrival time. A job at the head of the Job Queue is taken out and assigned memory to run if there is at least 4 free pages that can be assigned to that job (i.e., initially only 25 jobs will run. When a Job run, we need initially one page (page-0) and then the process will make a memory reference to another page in that process and we need to page-in the referenced page from disk if the page is not already in memory. Once a jobs are in memory, the processes run independently and simultaneously for their durations.

Paging:

Assume as an example you are running a process that consists of **11** pages numbered 0 through 10. Assume when start executing this process there is only free 4 pages Physical memory frames. There are always 11 page frames available on disk. When the process starts, none of its pages are in memory.

The process makes random references to its pages. Due to **locality of reference**, after referencing a page i, there is a 70% probability that the next reference will be to page i, i-1, or i+1. i wraps around from 10 to 0. In other words, there is a 70% probability that for a given i, Δi will be -1, 0, or +1. Otherwise, $|\Delta i| > 1$.

Suggested procedure

- To compute the next i, first generate a random number "r" from 0 through 10.
- If $0 \leq r < 7$, then generate a random Δi to be -1, 0, or +1.
- For $7 \leq r \leq 10$, randomly generate the new page reference "j," $2 \leq |\Delta i| \leq 9$ [$0 \leq j \leq i-2$ or $i+2 \leq j \leq 10$]

Process Execution

A process will always start at page-0, then every 100 msec the process references another memory location using the above locality of reference algorithm. Process continues referencing memory till its duration expires. For each memory reference we need to collect statistics <time-stamp in seconds, process Name, page-referenced, Page-in-memory, which process/page number will be evicted if needed>

Simulator

Simulate the **page replacement** algorithms FIFO, LRU, LFU, MFU, and random pick:

1. Run simulator 5 times, each to complete the **one minutes**, and compute the hit/miss ratio of pages referenced by the running jobs for each run. Then get average of 5 runs.
2. Run the simulator for 100 page references, and for each reference, print the <time-stamp in seconds, process Name, page-referenced, if-Page-in-memory, which process/page number will be evicted if needed>

Simulation Execution

Run each page replacement algorithm 5 times simulating 1 minute each time to get an average of the number of processes successfully swapped into memory (but not necessarily completed) during the minute.

For each replacement algorithm, print the average number of processes that were successfully Swapped-in. Memory map for the 100 pages is defined as <AAbbbACCAAbbbbbbbbCCC**.... ddddddddddd... 33333**> where the characters are the process names (one character per MB) and the dots are holes (one per MB).

Each time a process is **swapped in** (start running) or completes (exits and therefore is removed from memory), print a record <time stamp, process name, Enter/exit, Size in Pages, Service Duration, Memory-map>.

For each replacement algorithm, print the average number of processes (over the 5 runs) that were successfully swapped-in (started execution).

What to turn in

Email your answers to the grader. The email should include the following:

- Your source files.
- Your output.

Your subject line should be **COEN383 Assignment #4, Group-n**. Cc all your group members so that the grader can do a "Reply all" when needed. This is a group assignment; all group members will receive the same score.