CS / SE 149

Operating Systems

Spring Semester 2014

Department of Computer Science
San José State University
Instructor: Ron Mak

Assignment #4

Assigned: Thursday, March 5
Due: Friday, March 20 at 11:59 pm
100 points, team assignment

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. You should not need to make any process management system calls.

Swapping

Assume main memory has 100 MB for swapping with variable-sized partitions. Processes have randomly and evenly distributed sizes of 5, 11, 17, and 31 MB. Processes have randomly and evenly distributed durations of 1, 2, 3, 4, or 5 seconds.

Write programs that simulate the **first fit**, **next fit**, **best fit**, and **worst fit** memory allocation algorithms. Do not perform memory compaction.

Run each 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. Generate enough new random processes before each run so that the job queue is never empty. According to the algorithm, allocate them into memory in the order that you generated them.

For each algorithm, print the average number of processes that were successfully swapped in (but not necessarily completed). Each time a process is swapped in, or a process completes and therefore is removed from memory, print a memory map, e.g., **`AAAAA....BBBBBBBB..CCCC`** where the characters are the process names (one character per MB) and the dots are holes (one per MB). Indicate which process entered or left. For an entering process, also print its size and duration.

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

Extra credit (up to 10 points)

Rerun all your algorithms, but this time, after 30 (simulated) seconds, do one **memory compaction**. On average for each algorithm, how many MB of memory were copied during compaction? How does compaction change the average number of processes that were successfully swapped in?

Paging

You are running a process that consists of 10 pages numbered 0 through 9. Physical memory has 4 page frames. There are always 10 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. In other words, there is a 70% probability that for a given i, Δi will be -1, 0, or +1. Otherwise, $|\Delta i| > 1$. Page references should wrap from 9 back to 0.

Suggested procedure:

- Given page reference i, to compute the next page reference, first generate a random number r from 0 through 9.
- If $0 \le r < 7$, then generate a random Δi to be -1, 0, or +1.
- For $7 \le r \le 9$, randomly generate $2 \le \Delta i \le 8$.
- The next value of i is $i + \Delta i$. The value of i wraps around from 9 to 0.

Simulate the **FIFO**, **LRU**, **LFU**, **MFU**, and **random pick** page replacement algorithms. Run each algorithm 5 times, 100 page references each time, to compute an **average hit ratio** of pages already in memory. For each reference, print the page numbers of the pages in memory and which page (if any) needed to be paged in and which page was evicted.

What to turn in

Email a zip file to ron.mak@sjsu.edu that contains:

- Your source files.
- Your output.
- A short report describing your results. Note clearly in your report if you did the extra credit.

Name the file after your team, such as SuperCoders`.zip`. Your subject line should be `CS 149-`*section* `Assignment #4` *team name*

Don't forget to CC all your team members.