CS433 Programming Assignment 5

Simulation of Page Replacement Algorithms

Overall Goal

For this assignment, you are to write a simulation to analyze different memory page replacement algorithms. You need to implement a simple simulator of page replacement policies, assuming a single running process and a fixed size physical memory.

Overview

You simulator should allow the page size and physical memory size (# of address bits) as command-line arguments. The simulation should support at least three page replacement policies: Random, FIFO, and LRU. The page size must be a number of power of 2 between 256 (2^8) and 8192 (2^13) bytes inclusively. The physical memory size is also power of 2.

Assume that when the simulation start, the physical memory is empty, i.e. all page table entries are invalid. As they are brought in, the simulator should make sure that the total number of pages in memory should never exceed the physical memory size.

The simulator should read a sequence of logical memory references from a file (references.txt). This file contains a list of virtual (logical) memory byte addresses accessed by a program. In this simulation, the maximum virtual memory address is 128 MB (2^{27}bytes) and the number of addresses in the file is 5,000,000. In this simulation, the least significant bit is used to differentiate between a read access (0, i.e. even number) or write access (1, i.e. odd number). For example, the value 1220149 means a write reference to memory location 1220148. Those addresses can be assumed to be aligned at a two-byte boundary, i.e., they are always even. We can safely play this simple trick because the minimum page size is 256 bytes, so the eight least significant bits of an address are irrelevant to the page number anyway.

Your program should keep track of pages in the memory and free frames. Therefore you need to maintain a pagetable data structure, which can be easily implemented as an array of page-table entries, and a free-frame list. The number of pages is dependent on the page size. In the pagetable entry data structure, you need additional fields, e.g. valid and dirty bits, besides the mapped frame number of the page. For each memory reference, find out its page number and whether this page is in the main memory. If the memory reference is a write, this page is marked dirty. If this page is not in the main memory, a page fault is generated and this page is loaded into the main memory. However, if the main memory is full, a victim page must be selected and evicted according a page replacement algorithm. Your simulation is to compare different page replacement algorithms. Notice you may need to keep track of additional information such as last page access time for the LRU (least recently used) algorithm.

Required Output

- All programs should print your name as a minimum when they first start.
- Beyond that, this program should collect and print the following statistics for different algorithms:
  - The total number of memory references.
  - The total number of page faults.
  - The total number of flushes, i.e., cases when the victim page was dirty.
  - The total time it took the simulator to produce the results.
- In addition to a program printout, the results of algorithm should be analyzed and written in the reprot. You should thoroughly test your program with several different page sizes (256 - 8192 bytes) and physical memory sizes (e.g. 4 MB, 16 MB, 32 MB, 64 MB), analyze the data and summarize results and conclusions in the report.

Extra Credits

(up to 10 points) Implement additional page replacement algorithms, e.g the optimal algorithm and Enhanced Second-Chance (pg 378 of the textbook), and compare them to the algorithms in the simulation.