CS 3305: Operating Systems
Department of Computer Science
The University of Western Ontario
Programming Assignment 4
Spring, 2015

Purpose:

The goals of this assignment are the following:

- Understand basic operating system concepts related to virtual memory
- Understand and simulation two page replacement policies
- Compare the performance of such policies

Description:

A *memory trace* is a list of memory addresses referenced by the instructions of the processes. This assignment requires that you compare the results of Least Recently Used (LRU) and Least Frequently Used (LFU) on memory traces using a simulator.

You are to build a simulator that reads a memory trace and simulates the action of a virtual memory system that uses a single level page table. As the simulator examines each referenced memory address, it should check to see if the corresponding page is loaded. If not, it should choose a page to remove from memory. You should always choose an empty available frame over a frame that is not available e.g., you only replace a frame if there is no frame that is available.

Please note that in a real system, if the page to be replaced is dirty, it must be saved to disk and the new page is to be loaded into memory from disk. The page table is then updated. However, this assignment is just a simulation. You should update the frame table and the page table. You do not actually need to read and write data from disk.

A sample trace is provided on the web page. You may want to create other traces for testing and debugging purposes.

The executable should be called *simulator* and the input arguments are the following

- The number of frames to be used in the simulation;
- The name of the file with the memory trace. You may assume that each line of the file is a page reference and that page references may vary from 0 to 1023.
- Either LRU (indicating that LRU should be used) or LFU (indicating that LFU should be used).

An example line is the following: 4 trace LRU

The output of the program should be the number of page faults encountered.

Output
After completion, your program is to provide the number of page faults encountered and TLB misses.

Details

1. You will need data structures representing the page table, set of frames and the TLB. We recommend an array for each one of these. Each entry of the array should represent information needed for the simulation. For example, a page table entry should consist of information for each page.
2. Entry information should be represented using a `struct`. For example, a page entry may look something like this:

   ```
   typedef struct pageInfo{
      int frameNumber;
      …….
    }pageInfoEntry
   ```

   The page table would declared as follows:

   ```
    pageInfoEntry *pageTable;
   ```

3. Use a `malloc` to allocate space for an array e.g.,

   ```
   pageTable=(pageInfoEntry*)malloc(sizeof(pageInfoEntry)*pageTableSize);
   ```