Programming Assignment 3 – Virtual Memory Page Replacement Policies

Due: 11:59 p.m., Tuesday, December 1, 2015

In this assignment, you will write a program that simulates the virtual memory page replacement algorithms described in section 8.2 of the text book, pages 362 to 367: Optimal (OPT), Least Recently Used (LRU), First-In-First-Out (FIFO), and the single use-bit one-handed clock policy (CLOCK).

This project will require you to generate three separate executables: a page replacement simulator vmsim, and two page reference generators: vmgenU and vmgenWS.

1. Specifications of the page replacement simulator vmsim

- vmsim must accept three command-line arguments in the following order:
  - (a) the total number of physical memory frames (maximum 100),
  - (b) an input filename where a sequence of page references is stored,
  - (c) the chosen algorithm (OPT, LRU, FIFO, or CLOCK).

  The four possible values for (c) are the strings “opt” or “OPT”, “lru” or “LRU”, “fifo” or “FIFO”, and “clock” or “CLOCK”. If vmsim is run with the wrong arguments or no arguments, it should print out usage instructions and exit. For example:

  ```
  > vmsim 5 vmrefs.txt LRU
  ```

  The format of the input file must be a simple ASCII sequence of integers in the range 0 to 99 separated by spaces, for example: 51 7 34 0 8 45 21. No symbols or formatted text, just a pure sequence of space-separated integer numbers. This file can be created by hand or generated by either the vmgenU or vmgenWS programs (see section 2 and 3 below).

  vmsim will first read all the memory references from the input file and store them in a local array. Then, it will play back these references one by one and print out for each reference the current allocation state of physical memory frames in the following format:

  ```
  34: [51| 7|34|  |  ]
  ```

  This line means that after using page 34, frames 0, 1 and 2 are occupied by pages 51, 7 and 34, and frames 3 and 4 are empty. The Frames list must start with an open square bracket [, and end with a close square bracket]. Individual frames within the frame list must be separated with a vertical bar |. One-digit page numbers should have an extra space to the left so that frame numbers are always 2 characters wide (2 spaces for an empty frame). Each page fault should be signaled by an F character two spaces to the right of the closed bracket, for example:

  ```
  45: [45| 7|34| 0| 8] F
  ```

This indicates that a page fault occurred when referencing page 45 and that this page was loaded into frame 0, replacing whichever page was located there beforehand.

After processing all the memory references, vmsim should finally print the total number of page faults and the miss rate (page faults divided by number of references). It should start counting page faults and page references only after all frames have been initially filled (see book, Fig. 8.14), e.g., with 3 frames that means starting at the 4th step. Use this printout format:

```
Miss rate = 237 / 997 = 23.78%
```

2. Specifications of the page reference generator vmgenU

vmgenU must accept three command-line arguments in the following order: (a) the range of page references (maximum 100), (b) the length of the sequence of page references and (c), the name of the file that will be generated. (If vmgenU is run with the wrong arguments or no arguments, it should print out usage instructions and exit.) For example:

```
>vmgenU 10 200 vmrefs.dat
```

vmgenU will then generate a sequence of page references of the desired length containing random page numbers ***uniformly*** drawn between 0 and the range minus one (i.e., 200 page numbers between 0 and 9 in the example above). vmgenU must write this sequence into the file given as input.

3. Specifications of the page reference generator vmgenWS

vmgenWS must accept five command-line arguments in the following order: (a) ws_size: the size of the working set, (b) LB – the lower bound on the number of pages to generate in a given working set exhibiting locality of reference, (c) UB – the upper bound on the number of pages to generate in a given working set exhibiting locality of reference, (d) the range of page references (maximum 100), (e) the length of the sequence of page references and (f), the name of the file that will be generated. (If vmgenWS is run with the wrong arguments or no arguments, it should print out usage instructions and exit.) For example:

```
>vmgenWS 5 8 15 25 200 vmrefs.dat
```

vmgenWS will then generate a sequence of page references of the desired length (i.e., 200) containing random page numbers generated as follows:

1. generate num_gen – randomly generate a number between 8 (LB) and 15 (UB).
2. generate 5 (ws_size) page reference numbers in range 0 to 25
3. randomly generate num_gen of page references from the set of page references found in 2 (i.e., if num_gen is 12, we would generate 12 numbers from the set of 5 numbers from step 2).
4. repeat steps 1-3 until 200 (num) page references have been generated.

vmgenWS must write this sequence into the file given as input.

I remind you of the programming guidelines, including, among other things: appropriately structured and properly commented the code, not including object code or executables in your submission, naming the makefile exactly makefile, naming the readme file exactly readme, and writing a reasonably detailed UNIX-like readme documentation. Your documentation should describe the problem's background, the programs' usage and parameters, and a sample of a typical outcome.

Your programs must compile without warnings and run properly under Linux on the systems used in the CSE labs. You may develop and test your programs on your own UNIX machine, but it is your responsibility to make sure that they also work properly on the Linux installation of the Linux Lab in Lopata 400. There will be a penalty if they don't.

Notes:

- To compile your program you should use something like:

  ```
  g++ vmsim.cpp
  g++ vmgenU.cpp
  g++ vmgenWS.cpp
  ```

- Programming style matters. I will take off points if your program exhibits poor programming style.
- Project submissions should be the same as before.