CS 470 - Operating Systems
Spring 2019 - Memory Management Project

40 points
Out: March 29, 2019
Due: April 12, 2018 (Friday, day of Exam 2)

The purpose of this project is to gain an appreciation of the work involved in managing virtual memory, and to compare different page replacement policies for virtual memory management. The assignment is to write a simple virtual memory simulation.

Problem Statement

Develop a simulation program to investigate the relative effectiveness of the FIFO, LRU, and OPT page replacement strategies. Your program should measure the number of page faults generated by each strategy.

Your simulator should read a reference string from a file. The reference string items will be of the form: `<PID> <page ref>`, where PID is a process ID and page ref is a page of that process' logical memory. There will be one reference per line of the file. For example, the following might be the beginning of an input file:

```
1 45
1 46
1 45
1 44
1 45
2 45
2 46
2 47
2 47
2 46
```

This reference string corresponds to Process 1 accessing its pages 45, 46, 45, 44, and 45, then Process 2 accessing its pages 45, 46, 47, 47, and 46. We will assume a pure demand paging system (i.e., no pre-fetching). The simulation also must allow the number of physical frames to be specified. So for example, if there are 3 or more frames available, the above reference string will cause 6 page faults for all three strategies (one for each time a new page is referenced). If there are only 2 frames available, then there would be 6 page faults for LRU and OPT, and 7 page faults for FIFO.

The input file name and the number of physical frames may be entered interactively or on the command-line. The output for this simulation is the number of page faults generated for each replacement strategy.

For the purposes of making this simulation manageable, assume a program has 2 megabytes (2048 kilobytes) of virtual memory divided into 1K pages (i.e., a program can generate page references from 0 to 2047), and there will no more than 150 processes (PIDs 0 to 149). We also will assume a **global** replacement strategy. That is, when a page fault occurs, the victim page can be taken from any process. We will ignore the issue of dirty pages by assuming that all victim pages have not been modified.

Assignment (40 points)

(20 points) This project is to be done individually or in pairs**.** Each student/pair must implement a simulation meeting the specifications above for each page replacement algorithm with a physical memory size (in frames) **that can be set for each run of the program**.

The simulator may be written in any language for either Linux (projects must run on csserver) or Windows (projects using any IDE available in CS Lab are acceptable). However, the instructor can provide assistance only for C/C++ projects, and maybe Java projects. **Provide a makefile that will make your project if it needs to be compiled.** The file names and frame number value may be provided to the simulation either as command-line arguments or interactively. Hardcoding the test file names or frame number values into the program is not acceptable.

(10 points) Provide a high-level discussion of the program describing the functionality of the major components of the program and how they interact with each other, and a more detailed discussion for the data structures and algorithms for the memory management portion of the program. In particular, discuss how the system keeps track of which process a frame belongs to, and how a replacement page is determined for each replacement strategy. If the program does not meet all of the project requirements, describe which parts are missing and what you think should be done to implement them.

(10 points) Each student must provide the results of running their simulation using the test files provided on csserver in **`/home/hwang/cs470/project4`** (to be released on or before April 3) for 10, 50, and 100 physical frames. Each student/pair should produce a summary table showing the number of page faults generated for each strategy for each number of frames (nine values in all) for each file. The files will exhibit varying degrees of locality and number of processes. A README file will be placed in the directory explaining each file.

In addition, **each** student should answer the following questions (i.e., **provide two sets of answers** if doing the project in a pair):

1. Which replacement strategy would you choose and why? You should consider both the end results and the effort it took to implement each strategy. Discuss what your results show about the relative merits of FIFO, LRU, and OPT for the different input files.
2. What aspect of memory management did you find most difficult to implement?
3. What aspect of memory management did you find least difficult to implement?
4. What, if anything, would you change in your current design?
5. What, if anything, did you find interesting or surprising about page replacement algorithms that you did not know before doing this project?

What to Submit

Create a **tarfile** or **zipfile** containing the following:

- The well-documented code source code for your project. If you are submitting a Windows project, submit the entire project folder.
- A makefile to make your project, if needed
- A single document **in PDF format** containing:
    - instructions on how to build and/or run the program
    - the discussion of the functional design of your project
    - the answers to the questions above **from each student**
- The summary results table may be pasted into the PDF document above, or submitted as a separate textfile or PDF document.

Submit your archive using the submission system (http://submission.evansville.edu) on the due date (**to only one account of a pair**) . The grading script only will accept submissions. It will not run anything.