CS 4328 / CS 5305

Operating Systems

Programming Assignment # 3

Page Replacement Algorithms

Assigned: Friday, 11/5/21

Due: Thursday, 12/2/21 at 11:55 PM

Last day accepted (late): Saturday, 12/4/21 at 11:55 PM

This project is going to take a good amount of work and time, so please start early. You would not be able to finish if you start a few days before the due date. **Late submissions beyond two days will NOT be accepted, since enough time is needed to grade all programs.** Leave the last few days for documentation, further testing and formatting the results.

Please read this description carefully and contact me if you have any questions. You may discuss this project with other students. However, you must write your own code and your own report.

1. Overview

In this project, we are going to write a program that implements several virtual-memory page replacement algorithms*. The goal of this project is to compare and assess the impact of these algorithms on the number of page faults incurred across a varying number of physical-memory page frames available.*

1.1. Page replacement Algorithms

We are going to implement the following page replacement algorithms that we discussed in class:

1. First-In, First-Out (FIFO)
2. Least Recently Used (LRU)
3. Optimal (OPT)

Implement the algorithms so that the number of page frames available can be passed in as an argument.

1.2. Performance Metric

We are interested in computing the following metric:

• The number of **page faults** incurred

2. The Process

First, generate a random page-reference string, of 100 items, where the virtual page numbers range from 0 to 49.

Apply the same page-reference string to each algorithm, and record the number of page faults incurred by each algorithm.

Assume that demand paging is used. Structure the program to run so that you can vary the number of physical-memory frames available from 1 to 30.

You will need to run the 3 different algorithms, each for the 30 different values of physical-memory frames available. This is a total of 90 runs (each using the same 100-page-reference string) with the goal to calculate the number of page faults for each run.

3. Submission details

Submission shall be done using the **Assignments** tool on the Canvas website for this class. Please submit a single zip file including all your files.

Name your file **program3_xxxxx.zip** where xxxxx is your TX State NetID.

Submissions shall include:

- the program's source code, and
- a report containing:
  - a brief overview of the design and implementation,
  - instructions for how to compile and run the program on one of the CS Linux servers, and
  - the results of the runs and their interpretation (more below).

The report shall include a single plot for the above metric. The plot on the x-axis will vary the number of frames available, and represent the number of page faults on the y-axis, with a different-color line for each of the three algorithms.

See an example plot on chapter 10, page 404 (Fig. 10.11, "Graph of page faults versus number of frames."). It's also on the slide deck for chapter 10, slide 31.

You can write your program in any of these languages (C, C++, Python or Java), however, it is your responsibility to ensure it runs on the CS Linux servers with a command line – nothing graphical (GUI-based) or IDE. Please indicate clearly how to compile and run your program.

You may form a **2-person team** and work together on your programming assignment. If you submit a program that is the result of team work, you must list the name of the other contributor in the project file. The penalty for not citing the collaborator will be -30% for that assignment. **Each student must submit their own file (with their own NetID), even if it is the same file as the other student.**

4. Grade breakdown

20% of the grade is on developing the correct design and data structures.

30% of the grade is on obtaining the correct results (i.e., the metric and plot) for the algorithms.

20% getting the program to compile and run as expected.

30% of the grade is on the report (overview, explanation of the results, providing the compile and run instructions).