CSC360/660 Operating Systems

Project #5: Simulation of Page Replacement Strategies

Objective

The purpose of this programming project is to explore page replacement algorithms. This can be accomplished by developing a simple simulator that implements various page replacement algorithms and determining the number of page faults for a given reference string. A secondary objective of this assignment is to reinforce good software project design by using multiple source code modules in your solution.

Project Specifications

Develop a simulator program that will enable you to compare and contrast the operation of various page replacement strategies discussed in class (plus an additional page replacement algorithm discussed here). For a given page reference string, your program will output the number of page faults for a given page replacement algorithm. The name of your executable must be "simpager".

Input to your program will be from standard input. There is to be no "user prompts" in your program. Program output will be to standard output. Your program must follow a standardized input format. The first line is the page reference string. Each number in the page reference string is separated by whitespace and is terminated by a new line. The page reference string is on 1 (one) and only 1 (one) line. The second line is the number of frames allocated to a specific process. The remaining lines will be string mnemonics; one for each page replacement algorithm.

Output from your program will be the following. Echo the page reference string up to 20 page references per line. Echo the number of frames allocated to the process. Print the page replacement algorithm "mnemonic" and the number of page faults. Although your program reads from standard input and writes to standard output, it is suggested that you have several program data files that you can re-direct standard input and thus have your program read.

Example Input:

```
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
3
FIFO
LRU
```

Example Output:

```
% simpager < testcase1.txt
Page Reference String:
7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
Number of Frames: 3
FIFO: 15
LRU: 12
```

The page replacement algorithms and their respective mnemonics to be implemented are the following:

1. **FIFO** - First in first out page replacement. See text for description.
2. **LRU** - Least recently used page replacement See text for description.
3. **OPT** - Optimal page replacement. See text for description.

4. **RAND** - Random page replacement. This is an easy to implement, low-overhead page replacement strategy. Under this strategy, each page in main memory has an equal likelihood of being selected for replacement. One problem with RAND is that it may accidentally select as the next page to replace the page that will be referenced next. A benefit of RAND is that it makes replacement decisions quickly and fairly. However, because of this "hit-or-miss" approach, RAND is rarely used in practice.

Assessment and Grading

This is an individual assignment. Your program must be written using C/C++ or Java or Python (no extra points this time). Comment and document all code submitted and follow the documentation guidelines described here. Use good programming practices by implementing procedures and functions where necessary. You are not required to use the STL in your solution (but it is STRONGLY SUGGESTED!!!!!). This project is worth 100 points.

Project Submission

Please upload your project on Blackboard.