Project TA – Peter (cspeter@cse.ust.hk): Handling questions before the deadline
Grader TA - Lin ZHANG (lzhangbv@cse.ust.hk): Grading and appeal handling after the deadline

COMP3511 Operating System (Fall 2022)

PA3: Page-Replacement Algorithms

Released on 16-Nov-2022 (Wednesday) Due on 06-Dec-2022 (Tuesday) at 23:59

Introduction

A page-replacement algorithm is used to decide which frames to replace. Several page replacement algorithms are discussed in the virtual memory lecture. In this project assignment, we require students to implement 3 fundamental page-replacement algorithms: First-In-First-Out (FIFO), Optimal (OPT), and Least Recently Used (LRU). Upon completion of the project, students should be able to understand the details of these 3 page-replacement algorithms.

You are highly recommended to attend the corresponding project-related lab.

Program Usage

The program name is `page_replacement`. Here is the sample usage:

```
$> ./page_replacement < input.txt > output.txt
```

`$>` represents the shell prompt.
`<` means input redirection, which is used to redirect the file content as the standard input
`>` means output redirection, which is used to redirect the standard output to a text file
Thus, you can easily use the given test cases to test your program and use the `diff` command to compare your output files with the sample output files.

Getting Started

`page_replacement_skeleton.c` is provided.
You should rename the file as `page_replacement.c`

You can add new constants, variables, and helper functions
Necessary header files are included. You should not add extra header files.
Some constants and helper functions are provided in the starter code.
Please read the skeleton code carefully.

Assumptions

- There are at most 10 different frame numbers (i.e., frame 0-9)
- The maximum number of available frames is 10
- The maximum size of the queue data structure (used in FIFO) is 10
- The maximum length of the reference string is 30 (defined in the starter code)

Input and Output Format

The input parsing is given in the skeleton code.
Empty lines and lines starting with # are ignored
A sample input file:

```
# COMP3511 PA3 (Fall 2022)
# Page replacement algorithm
# Empty lines and lines starting with '#' are ignored

algorithm = FIFO

frames_available = 3

reference_string_length = 20

reference_string = 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
```

The output consists of 2 regions:

- Displaying the parsed values from the input
- Displaying the steps of the algorithm and the total page fault.

The sample output file based on the sample input file. Given the reference string, there are 15 page-faults after running the FIFO page-replacement algorithm. You can find a similar example in the lecture notes.

```
algorithm = FIFO
frames_available = 3
reference_string_length = 20
reference_string = 7 0 1 2 0 3 0 4 2 3 0 3 2 1 2 0 1 7 0 1
7: 7
0: 7 0
1: 7 0 1
2: 2 0 1
0: No Page Fault
3: 2 3 1
0: 2 3 0
4: 4 3 0
2: 4 2 0
3: 4 2 3
0: 0 2 3
3: No Page Fault
2: No Page Fault
1: 0 1 3
2: 0 1 2
0: No Page Fault
1: No Page Fault
7: 7 1 2
0: 7 0 2
1: 7 0 1
Total Page Fault: 15
```

First-In-First-Out (FIFO) Page Replacement Algorithm

A queue data structure is given in the starter code. You should use a queue to implement the FIFO page-replacement algorithm.

You should refer to the lecture notes for the details of FIFO page-replacement algorithm.

Optimal (OPT) Page Replacement Algorithm

There is no specific data structure for the OPT page-replacement algorithm. However, OPT may not give a unique solution (i.e., more than one possible solution giving the same number of page faults). It happens if there are page faults near the end of the algorithm.

To avoid multiple solutions, if there are more than one choice of the victim frame, we should choose a frame with the smallest frame number. For example, if frame 0 and frame 1 are victim frames, you should choose frame 0.

You should refer to the lecture notes for the details of OPT page-replacement algorithm.

Least Recently Used (LRU) Page Replacement Algorithm

There are 2 possible implementations of LRU page-replacement algorithm: Counters implementation and Stack implementation.

In this project, we have a small number of frames (i.e., frame 0-9), so the counters implementation is more suitable. You can create an array of integers to store the current count of the frames. So, there is no specific data structure for the LRU page-replacement algorithm if you use the counters implementation.

If you prefer to have the stack implementation, you can implement your own stack data structure, which is quite similar to the queue data structure given in the starter code.

You should refer to the lecture notes for the details of LRU page-replacement algorithm.

Compilation

The following command can be used to compile the program

```
$> gcc -std=c99 -o page_replacement page_replacement.c
```

The option `c99` is used to provide a more flexible coding style (e.g., you can define an integer variable anywhere within a function)

Sample test cases

Several test cases (both input files and output files) are provided. We don't have other hidden test cases.

The grader TA will probably write a grading script to mark the test cases. Please use the Linux `diff` command to compare your output with the sample output. For example:

```
$> diff --side-by-side your-outX.txt  sample-outX.txt
```

Sample Executable

The sample executable (runnable in a CS Lab 2 machine) is provided for reference. After the file is downloaded, you need to add an execution permission bit to the file. For example:

```
$> chmod u+x page_replacement
```

Development Environment

CS Lab 2 is the development environment. Please use one of the following machines (`csl2wk`**`XX`**`.cse.ust.hk`), where **`XX`**`=01…40`. The grader will use the same platform.

In other words, *"my program works on my own laptop/desktop computer, but not in one of the CS Lab 2 machines*" is an invalid appeal reason. Please test your program on our development environment (not on your own desktop/laptop) thoughtfully before your submission, even you are running your own Linux OS. Remote login is supported on all CS Lab 2 machines, so you are not required to be physically present in CS Lab 2.

Marking Scheme

1. (100%) Correctness of the 10 provided test cases. We don't have other hidden test cases, but we will check the source codes to avoid students hard coding the test cases in their programs. Example of hard coding: a student may simply detect the input and then display the corresponding output without implementing the program
2. Make sure you use the Linux diff command to check the output format.
3. Make sure to test your program in one of our CS Lab 2 machines (not your own desktop/laptop computer)
4. Please fill in your name, ITSC email, and declare that you do not copy from others. A template is already provided near the top of the source file.

***Plagiarism**: Both parties (i.e., students providing the codes and students copying the codes) will receive 0 marks. Near the end of the semester, a plagiarism detection software (JPlag) will be used to identify cheating cases. **DON'T** do any cheating!*

Submission

File to submit:

`page_replacement.c`

Please check carefully you submit the correct file.

In the past semesters, some students submitted the executable file instead of the source file. Zero marks will be given as the grader cannot grade the executable file.

You are not required to submit other files, such as the input test cases.

Late Submission

For late submission, please submit it via email to the grader TA.
There is a 10% deduction, and only 1 day late is allowed
(Reference: Chapter 1 of the lecture notes)