Project TA – Peter (cspeter@cse.ust.hk): Handling questions before the deadline
Grader TA – CHAI, Di (dchai@cse.ust.hk): Grading and appeal handling after the deadline

COMP3511 Operating System (Spring 2023)

PA3: Page-Replacement Algorithms

Released on 26-Apr-2023 (Wednesday) Due on 09-May-2023 (Tuesday) at 23:59

Introduction

In the virtual memory lecture, various page replacement algorithms are discussed, which are used to determine the frames to be replaced. For this project assignment, students are required to implement four fundamental page-replacement algorithms: First-In-First-Out (FIFO), Optimal (OPT), Least Recently Used (LRU), and Second Chance/Clock (CLOCK). By completing this project, students will gain an understanding of the intricacies of these algorithms. We strongly advise students to attend the relevant lab sessions for this project.

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 number of available frames ranges from a minimum of 2 to a maximum of 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:

```
# 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:

```
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

You should use a queue to implement the FIFO page-replacement algorithm. A queue can be implemented using a 1D array. You can implement some helper functions such as enqueue and dequeue.

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.

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.

Second Chance (CLOCK) Page Replacement Algorithm

The clock page replacement algorithm is quite simple. It makes use of a 1D array (named as `second_chance`, tracking whether pages have been accessed) and an integer variable (named as `pointer`, tracking the next target). Initially, all values of the `second_chance` array are initialized as `0`, and the `pointer` value is `0`.

If the page does not exist, we check whether the frame pointed by pointer is empty (i.e., the frames are not full yet). If so, we fill in the frame and move the pointer cyclically (i.e., if it is not the last frame, the pointer will be moved to the next frame. If it is the last frame, the pointer will be moved to the first frame).

If the frame pointed by pointer is non-empty, we will traverse the frames cyclically starting from the pointer, marking all corresponding `second_chance` elements as `0`, until we find a frame with the `second_chance` element equal to `0`. Then, we fill in the frame and move the pointer cyclically.

If the page already exists, we set its corresponding entry in `second_chance` to `1` and keep the `pointer` unchanged.

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)

Given Test Cases

Several test cases (both input files and output files) are provided.

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
```

Hidden Test Cases

The hidden test cases are like the given test cases. Different input values will be used. The main purpose of the hidden test cases is to avoid students hard coding all test cases in their code. For example, if the grader finds out that a student passes all given test cases but fails all hidden test cases, the grader may further investigate the code and check whether hard coding occurs.

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. Make sure you use the Linux diff command to check the output format.
2. 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.
3. **Correctness of the given test cases (50 marks)**
    1. The given test cases are equally weighted.
    2. The sum will be normalized to 50 marks.
    3. There won't be partial credits for each test case.
4. **Correctness of the hidden test cases (50 marks)**
    1. The hidden test cases are equally weighted.
    2. The sum will be normalized to 50 marks.
    3. There won't be partial credits for each test case.

Plagiarism

***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. Canvas will rename the submitted file if you submit more than once, so you don't need to worry about the renaming in Canvas.

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)