Programming Project 05

Memory Management Simulator

CS 441/541 – Fall 2017

Project Available		Nov. 7
Component	**Points**	**Due Date (at 11:59 pm)**
Notification of Group Participation		Nov. 10
Project 4		Nov. 21
Program Correctness	85	
Style, Documentation & Testing	15	
Project Total	100	

Late submissions will be subject to the late policy in the syllabus. Your project must be written in C (not C++). No credit will be given for projects written in any programming language other than C. Your project must be able to be compiled using a Makefile by typing the command `make` without any additional arguments in the project directory.

You may work in groups of at most two. Use the D2L groups feature to create your group by the deadline above.

Packaging & handing in your projects

You will turn in a single compressed archive of your completed project directory to the appropriate Autolab project. The compressed archive should be downloaded from the BitBucket website, which is a `.zip` archive. If, for some reason, you cannot download the source from the BitBucket website then turn in a manually compressed archive of the project as a zip file.

Overview

In this project, you will implement a memory management simulator for page replacement. You will be reading from a file the following information: number of page references, page being referenced, and if that page was written to or read from. A command line parameter will pick the file name, and set the number of frames, as needed.

The following command line argument(s) can occur in **any order** on the command line. All other command line options can be considered file inputs. Your program needs to only handle processing the first file encountered, but be aware that other command line parameters (e.g., `-f`) may occur after the file name(s).

- Optional: `-f #`
  The number of frames to use in the algorithm analysis. If this parameter is not provided then your program will run for the full range of frames. The number of frames can range from 1 to 7, inclusive.

You will need to implement the following page replacement algorithms (in this order).

1. **Optimal (OPT)**
2. **First-In, First Out (FIFO)**

3. **Least Recently Used (LRU)**
4. **LRU Second Chance (LRU SC)**
5. **LRU Enhanced Second Chance (LRU ESC)**

File Format

The file format provides all of the information for the pages that you will need to run these algorithms - the file describes the *reference string*. The first line of the file contains a single number identifying the number of page references that are contained in the file. The number of page references will range from 1 to an undefined upper bound (your program will need to handle a **large** number of references). You may assume that the file is formatted correctly.

The subsequent lines in the file each describe one page reference page number (integers ranging from 0 to 9, inclusive), if the page is written to or read from (indicated by the character 'W' or 'R' - may be in upper or lower case). For example, the `4 R` in the example below indicates that page `4` is read. And, the `2 W` in the example below indicates that page `2` is written. Do not assume that all pages are referenced in every file.

The order of the pages in the file is their ordering in the reference string. In the example below, the pages are assembled into the following order in the reference string: `4, 1, 3, 2, 4`.

```
5
4 R
1 w
3 r
2 W
4 r
```

Example Output

After your program parses the command line arguments, your program will display these parameters. This should identify the number of frames being used (or noting `All` for the default case).

After reading the test file, you will need to display the reference string, **line wrapping after every 10 page references**. The page information displayed before running the page replacement algorithms will be the page number, and if the page is written to or read from.

Your program will then run all of the algorithms in exactly the order specified on the previous page while keeping track of the number of page faults for each algorithm. For each number of frames (in order), you will display (prefixed with the `#` symbol at the front of the line) the number of page faults for each of the algorithms in a table format. See the example(s) (below) for information about the ordering of the algorithms and the expected output formatting.

```
shell$ cat given-tests/level1.txt
5
4 R
1 w
3 r
2 W
4 r
shell$ ./pagefault given-test/level1.txt
Num. Frames :  All
```

```
Ref. File   : given-tests/level1.txt
------------------------------
Reference String:
 4:R,  1:W,  3:R,  2:W,  4:R
------------------------------
#####################################################
#     Frames    Opt.    FIFO     LRU      SC     ESC
#          1       5       5       5       5       5
#          2       4       5       5       5       5
#          3       4       5       5       5       5
#          4       4       4       4       4       4
#          5       4       4       4       4       4
#          6       4       4       4       4       4
#          7       4       4       4       4       4
#####################################################
shell$ ./pagefault -f 2 given-test/level1.txt
Num. Frames :  2
Ref. File   : given-tests/level1.txt
------------------------------
Reference String:
 4:R,  1:W,  3:R,  2:W,  4:R
------------------------------
#####################################################
#     Frames    Opt.    FIFO     LRU      SC     ESC
#          2       4       5       5       5       5
#####################################################
```

```
shell$ ./pagefault given-test/level2.txt -f
Error: Must supply an integer argument between 1 and 7 for the -f option
shell$ ./pagefault -f 70 given-test/level2.txt
Error: Must supply an integer argument between 1 and 7 for the -f option
shell$
shell$ ./pagefault given-test/level2.txt -f 7
Num. Frames :  7
Ref. File   : given-tests/level2.txt
------------------------------
Reference String:
 1:R,  2:R,  3:R,  4:R,  1:R,  2:R,  5:R,  1:R,  2:R,  3:R,
 4:R,  5:R
------------------------------
#####################################################
#     Frames    Opt.    FIFO     LRU      SC     ESC
#          7       5       5       5       5       5
#####################################################
shell$
shell$ ./pagefault given-test/level2.txt
Num. Frames :  All
Ref. File   : given-tests/level2.txt
------------------------------
Reference String:
 1:R,  2:R,  3:R,  4:R,  1:R,  2:R,  5:R,  1:R,  2:R,  3:R,
 4:R,  5:R
```

```
------------------------------
################################################
#     Frames    Opt.    FIFO     LRU      SC     ESC
#          1      12      12      12      12      12
#          2       9      12      12      12      12
#          3       7       9      10       9       9
#          4       6      10       8      10      10
#          5       5       5       5       5       5
#          6       5       5       5       5       5
#          7       5       5       5       5       5
################################################
```

```
shell$ ./pagefault given-test/level3.txt
Num. Frames :  All
Ref. File   : given-tests/level3.txt
------------------------------
Reference String:
 1:R,  7:R,  1:R,  3:R,  7:R,  6:R,  4:R,  2:R,  3:R,  7:R,
 1:R,  4:R
------------------------------
################################################
#     Frames    Opt.    FIFO     LRU      SC     ESC
#          1      12      12      12      12      12
#          2       9      10      11      10      10
#          3       8      10      10      10      10
#          4       7       8      10       8       9
#          5       6       7       7       7       7
#          6       6       6       6       6       6
#          7       6       6       6       6       6
################################################
```

Deliverables:

Your project submission is required be written in a **consistent style** according to the **Style Requirements** handout. You will turn in a single compressed archive of your completed project directory to the D2L Dropbox. Be sure to carefully review the entire specification in detail for the list of requirements.

- ◯ Program must compile with the Makefile provided without warnings or errors.
- ◯ Command line arguments fully supported, and proper error checking in place (e.g., if the user forgets to supply an integer value to the `-f` parameter).
- ◯ Correct implementation of OPT
- ◯ Correct implementation of FIFO
- ◯ Correct implementation of LRU
- ◯ Correct implementation of LRU SC
- ◯ Correct implementation of LRU ESC
- ◯ Complete documentation & code styled according to the **Style Requirements** handout.
- ◯ Set of 5 unique tests

1 `level4.txt` Enhanced Second-Chance LRU

4 Frames: How many page faults?

0 R	1 W	3 R	6 R	2 W	4 R	5 R	2 W	5 R	0 R	3 W	1 R	2 R	5 W	4 R	1 R	0 W

2 `level7.txt` Enhanced Second-Chance LRU

4 Frames: How many page faults?

0 R	1 W	3 R	6 R	2 W	4 R	5 R	2 R	5 R	0 R	3 W	1 R	2 R	5 W	4 R	1 R	0 W

3 `level4.txt` Enhanced Second-Chance LRU

4 Frames: How many page faults? 15 page faults

0 R	1 W	3 R	6 R	2 W	4 R	5 R	2 W	5 R	0 R	3 W	1 R	2 R	5 W	4 R	1 R	0 W
$0^{(1,0)}$	$0^{(1,0)}$	$0^{(1,0)}$	$0^{(1,0)}$	$2^{(1,1)}$	$2^{(1,1)}$	$2^{(1,1)}$	$2^{(1,1)}$	$2^{(1,1)}$	$2^{(0,1)}$	$3^{(1,1)}$	$3^{(1,1)}$	$3^{(1,1)}$	$3^{(0,1)}$	$3^{(0,1)}$	$3^{(0,1)}$	$0^{(1,1)}$
	$1^{(1,1)}$	$1^{(1,1)}$	$1^{(1,1)}$	$1^{(0,1)}$	$1^{(0,1)}$	$1^{(0,1)}$	$1^{(0,1)}$	$1^{(0,1)}$	$0^{(1,0)}$	$0^{(1,0)}$	$0^{(1,0)}$	$0^{(1,0)}$	$5^{(1,1)}$	$5^{(1,1)}$	$5^{(1,1)}$	$5^{(1,1)}$
		$3^{(1,0)}$	$3^{(1,0)}$	$3^{(0,0)}$	$4^{(1,0)}$	$4^{(1,0)}$	$4^{(1,0)}$	$4^{(1,0)}$	$4^{(1,0)}$	$4^{(0,0)}$	$1^{(1,0)}$	$1^{(1,0)}$	$1^{(0,0)}$	$4^{(1,0)}$	$4^{(1,0)}$	$4^{(1,0)}$
			$6^{(1,0)}$	$6^{(0,0)}$	$6^{(0,0)}$	$5^{(1,0)}$	$5^{(1,0)}$	$5^{(1,0)}$	$5^{(1,0)}$	$5^{(0,0)}$	$5^{(0,0)}$	$2^{(1,0)}$	$2^{(0,0)}$	$2^{(0,0)}$	$1^{(1,0)}$	$1^{(1,0)}$

4 `level7.txt` Enhanced Second-Chance LRU

4 Frames: How many page faults? 15 page faults

0 R	1 W	3 R	6 R	2 W	4 R	5 R	2 R	5 R	0 R	3 W	1 R	2 R	5 W	4 R	1 R	0 W
$0^{(1,0)}$	$0^{(1,0)}$	$0^{(1,0)}$	$0^{(1,0)}$	$2^{(1,1)}$	$2^{(1,1)}$	$2^{(1,1)}$	$2^{(1,1)}$	$2^{(1,1)}$	$2^{(0,1)}$	$3^{(1,1)}$	$3^{(1,1)}$	$3^{(1,1)}$	$3^{(0,1)}$	$3^{(0,1)}$	$3^{(0,1)}$	$0^{(1,1)}$
	$1^{(1,1)}$	$1^{(1,1)}$	$1^{(1,1)}$	$1^{(0,1)}$	$1^{(0,1)}$	$1^{(0,1)}$	$1^{(0,1)}$	$1^{(0,1)}$	$0^{(1,0)}$	$0^{(1,0)}$	$0^{(1,0)}$	$0^{(1,0)}$	$5^{(1,1)}$	$5^{(1,1)}$	$5^{(1,1)}$	$5^{(1,1)}$
		$3^{(1,0)}$	$3^{(1,0)}$	$3^{(0,0)}$	$4^{(1,0)}$	$4^{(1,0)}$	$4^{(1,0)}$	$4^{(1,0)}$	$4^{(1,0)}$	$4^{(0,0)}$	$1^{(1,0)}$	$1^{(1,0)}$	$1^{(0,0)}$	$4^{(1,0)}$	$4^{(1,0)}$	$4^{(1,0)}$
			$6^{(1,0)}$	$6^{(0,0)}$	$6^{(0,0)}$	$5^{(1,0)}$	$5^{(1,0)}$	$5^{(1,0)}$	$5^{(1,0)}$	$5^{(0,0)}$	$5^{(0,0)}$	$2^{(1,0)}$	$2^{(0,0)}$	$2^{(0,0)}$	$1^{(1,0)}$	$1^{(1,0)}$