CS4420/5420
Virtual Memory Simulator
Due: Monday, November 15th - 11:55pm

version 1.1 - Wed Nov 10, 2021

Overview

This programming assignment is to simulate a simple virtual memory system using the three page replacement policies studied in class: First-In First-Out (FIFO), Optimal (extra credit), and Least Recently Used (LRU). Your program reads an input file containing a sequence of page requests (page numbers) and generates an output that shows how the requested pages are mapped into physical frames.

Program

The program is called as:

```
./vm FIFO|LRU|OPTIMAL input.handout
```

Input File

The input file consists of 3 parts:

- The first (non-comment) line is an integer specifying how many frames to simulate
- The second (non-comment) line is an integer specifying how many pages to simulate
- The rest of the file contains page references that consist of lines of: r/w pagenumber

The input file can also contain the following lines:

- Any line that starts with a pound sign is a comment and should be ignored
- Any line that contains only the command ”debug” should enable debugging
- Any line that contains only the command ”nodebug” should disable debugging
- Any line that contains only the command ”print” should print the defined output format as shown below

Requirements

1. Your program must be writted in C, C++, or Python
2. Your program must check the command line arguments and report errors appropriately
3. Your program must carefully check the input file format and report errors appropriately (any error should terminate the simulation)
4. Your program must work for *any* specified value for the number of frames and the number of pages

Required Output

When the program starts, it should print the following output: (this is from "input,handout")

```
Num frames: 4
Num pages: 20
Reclaim algorithm: FIFO
```

After processing the entire input file (and any time the input file contains a command to "print"), you should print output as follows:

```
Frames
    0 inuse:1  dirty:0  first_use:    13  last_use:    13
    1 inuse:1  dirty:0  first_use:    14  last_use:    14
    2 inuse:1  dirty:0  first_use:    15  last_use:    15
    3 inuse:1  dirty:0  first_use:    12  last_use:    12
Pages
    0 type:   Taken  ondisk:1  framenum:(unassigned)
    1 type:  MAPPED  ondisk:1  framenum:3
    2 type:  MAPPED  ondisk:1  framenum:0
    3 type:  MAPPED  ondisk:1  framenum:1
    4 type:  MAPPED  ondisk:1  framenum:2
    5 type:   Taken  ondisk:1  framenum:(unassigned)
    6 type:   Taken  ondisk:1  framenum:(unassigned)
    7 type:   Taken  ondisk:1  framenum:(unassigned)
    8 type:   Taken  ondisk:1  framenum:(unassigned)
    9 type:   Taken  ondisk:1  framenum:(unassigned)
   10 type:UnMapped  ondisk:0  framenum:(unassigned)
   11 type:UnMapped  ondisk:0  framenum:(unassigned)
   12 type:UnMapped  ondisk:0  framenum:(unassigned)
   13 type:UnMapped  ondisk:0  framenum:(unassigned)
   14 type:UnMapped  ondisk:0  framenum:(unassigned)
   15 type:UnMapped  ondisk:0  framenum:(unassigned)
   16 type:UnMapped  ondisk:0  framenum:(unassigned)
   17 type:UnMapped  ondisk:0  framenum:(unassigned)
   18 type:UnMapped  ondisk:0  framenum:(unassigned)
   19 type:UnMapped  ondisk:0  framenum:(unassigned)
Pages referenced: 15
Pages mapped: 10
Page misses: 15
Frames taken: 11
Frames written to disk: 10
Frames recovered from disk: 5
```

Debugging

You are required to provide helpful debugging when ”debug” is requested in the input file. When disabled, the program should output nothing except for the final counters, page table, and frame table.

Hints

1. **You should budget you time and priorities carefully. Although the basic program can be written in a single evening, it may take a couple of days to debug**
2. **The first thing you should write is the data s tructures for the pagetable and frametable**
3. **The second thing you should write is the printing routine**
4. **The third thing you should write is the ”debug” and ”nodebug” input line recognition**
5. **Add in debugging code as you go**

Extra Credit Ideas

For those of you who want to further explore the project, and want to earn **at most** an additional 20% on the project, I suggest that you try to make the program more fully-functional, perhaps using the following ideas:

Implement OPTIMAL

The ”OPTIMAL” (or ”MIN” in the textbook) is fun to implement, but a little difficult to debug. To receive full credit, your implementation needs to be able process the input file ”input.bigrandom” in a few seconds at most.

Implement LFU - Least Frequently Used

LFU is sometimes useful and isn’t difficult to simulate.

Duplicate the charts in the book

The textbook has a graphical representation that can be handy, but is difficult to draw. Implement it for extra credit and show it after the input file has been completely processed.

Example Input File

(this is from ”input,handout”)

```
4
20
w 0
w 1
w 2
w 3
w 4
w 5
w 6
w 7
w 8
w 9
r 0
r 1
r 2
r 3
r 4
```