CSE 410 **Summer 2018**

Extra Credit

Assignment Overview

This optional assignment focuses on memory management within an operating system. You will design and implement a C/C++ program which simulates a virtual memory system, as described below.

It is worth 20 points.

Assignment Deliverables

The deliverables for this assignment are the following files:

`extracredit04.makefile` – the makefile which produces "extracredit04"
`extracredit04.student.c` – the source code file for your solution

Be sure to use the specified file names and to submit your files for grading via the CSE Handin system before the project deadline.

Assignment Specifications

The program will simulate some of the steps that an operating system takes to manage a virtual memory system which uses demand paging and a fixed allocation of page frames. The simulator will gather statistics about the behavior of a single user process under a given page replacement algorithm.

The system to be simulated contains 1,048,576 bytes of RAM which is divided into 256 page frames. Logical addresses are 16 bits in length.

1. Your program will process a file which contains the simulation parameters, as well as zero or more memory references. The first line of the file will contain the page replacement algorithm to be studied (the character string "FIFO", "LRU" or "Clock"), and the second line will contain the number of page frames allocated to the process.

Each additional line of the file (if any) will contain the following information about a memory reference:

a) logical address being referenced (4 hexadecimal digits, with leading zeroes)
b) operation being performed (one character; R for read and W for write)

Items within a line will be separated by exactly one space, and each line will terminate with a newline.

2. For each memory reference in the file, your program will display one line with the following information:

a) logical address being referenced (4 hexadecimal digits, with leading zeroes)
b) page number (1 hexadecimal digit)
c) page offset (3 hexadecimal digits, with leading zeroes)
d) operation being performed (one character; R for read and W for write)
e) page fault flag (one character; F for page fault, blank otherwise)
f) write back flag (two characters; WB for write back, blanks otherwise)

g) physical address (5 hexadecimal digits, with leading zeroes)

Items within a line will be separated by exactly one space, and each line will terminate with a newline.

3. After the simulation is completed, your program will display the following:

   a) simulation parameters (page replacement algorithm and number of frames allocated)
   b) total number of memory references
   c) total number of read operations
   d) total number of write operations

The summary information will be appropriately labeled and formatted.

4. Your program will display the contents of the page table at the start of the simulation, after every N memory references, and at the end of the simulation. The page table will not be displayed twice in succession (there must be some intervening output related to memory references).

The display will contain one line for each page table entry:

   a) index of the page table entry (one hexadecimal digit)
   b) V bit (one character; 0 for not valid, 1 for valid)
   c) R bit (one character; 0 for not referenced, 1 for referenced)
   d) M bit (one character; 0 for not modified, 1 for modified)
   e) frame number stored in that page table entry (2 hexadecimal digits, with leading zeroes)

Items within a line will be separated by exactly one space, and each line will terminate with a newline. The page table display will begin and end with a blank line and will include appropriate column headers.

The value of N will be a positive integer value; any other value (such as 0) will prevent the program from displaying the contents of the page table.

5. Your program will accept two command-line arguments: an integer representing the value of N and the name of the input file.

6. The three-page replacement algorithms which your program will support are FIFO, LRU and Clock.

For FIFO page replacement, the page which has been in primary storage for the longest time will be selected as the victim.

For LRU page replacement, the page which has been referenced least recently will be selected as the victim.

For Clock page replacement, your program will use the strategy described in the Stallings textbook.

For all the page replacement algorithms, your program will use “less than” to break any ties. For example, if frame 42 and frame 43 are both empty when a page fault occurs, your program will select frame 42.

7. The program will include appropriate error-handling.

Assignment Notes

1. You must use "g++" to translate your source code file in the CSE Linux environment.

2. The page frames allocated to the process being simulated will be consecutive and will begin with frame 0x20 hexadecimal.