Gebze Technical University
Department of Computer Engineering
CSE 312 /CSE 504
Operating Systems Spring 2020

Final Exam Project
Due Date: July 10th 2020
No Late Submissions!
Virtual Memory Systems and Paging

In this project, you will design and implement a simulated and simplified virtual memory management system and a number of page replacement algorithms in C/C++. This project does not use MIPS or SPIM, so it is completely independent of your other homework/exam assignments. This will count as final exam for this course.

Since this is a simulation, we will use an integer C array for your physical memory as in Fig 3-9 in your textbook. As expected, your virtual memory may be much larger as shown in the same figure. You will keep your data that does not fit in your physical memory on a disk file.

Your simplified system will do simple integer array sorting with 4 different sorting algorithms (Bubble sort, quick sort, merge sort, index sort). The main steps of your program will be as follows

1. Fill the entire virtual memory with random integers using C/C++ function **`rand`**. Before you start filling, call **`srand(1000)`** to use the same random numbers for all runs of your program.

2. Create 4 C/C++ threads, one for each sorting algorithm. Each thread will sort 1/4 of the virtual memory in increasing order. Each thread will work on a different part of the virtual array (Bubble sort will work on first quarter, quick sort will work on second quarter, etc.)

3. Wait until all threads are finished and then scan the array to make sure that the array is sorted for each part.

4. Print your page table statistics at the end such as number of page misses, number of accesses for each algorithm as described below.

The above steps will access your virtual integer memory using only the following C/C++ functions

```
void set(unsigned int index, int value, char * tName);

int get(unsigned int index, char * tName);
```

where **`index`** is the virtual address of the integer, **`value`** is the value of the integer to put at that index, **`tName`** is the String that defines the thread (“fill”, “quick”, “bubble”, “merge”, “index”, “check”, etc.) The above functions will be implemented by you to get/set the desired data. If the data is available in the physical memory, it is handled immediately. If the data is not available, then the page replacement algorithm has to be executed. The page replacement statistics can be easily calculated in these functions.

Part 1

Design a page table structure that makes it possible to implement Not-Recently-Used (NRU), FIFO, Second-Chance (SC), Least-Recently-Used (LRU), Working set clock (WSClock) algorithms. Note that since we are simulating the hardware, we can add any features as we like to the page table including updating the table at every array reference for LRU.

In your project report, draw figures and explain each field of the page table entries like Fig 3-11. Your report should include references to your implementation code (line numbers, C files, etc.) in your report.

Part 2

Write a C/C++ program that follows above 4 steps. Your program will be named **sortArrays** and it will have the following command line structure

sortArrays frameSize numPhysical numVirtual pageReplacement allocPolicy pageTablePrintInt diskFileName.dat

frameSize is the size of the page frames, **numPhysical** is the total number of page frames in the physical memory, **numVirtual** is the total number of frames in the virtual address space, **pageReplacement** is the page replacement algorithm (NRU, FIFO, SC, LRU, WSClock), **allocPolicy** is the allocation policy (global or local), **pageTablePrintInt** is the interval of memory accesses after which the page table is printed on screen, **diskFileName.dat** is the name of the file that keeps the out of memory frames. For example,

sortArrays 12 5 10 LRU local 10000 diskFileName.dat

defines a frame size of 2^{12} (4096) integers, 2^5 (32) physical frames, 2^{10} (1024) virtual frames, uses LRU as page replacement algorithm, local allocation policy, and **diskFileName.dat** as the disk file name. In other words, this system has a physical memory that can hold 4096*32=131.072 integers and has a virtual memory that can hold 4096*1024= 4.194.304 integers. This command prints the page table on the screen at every 10000 memory accesses.

At the end of the program run, your statistics will include the following for each phase of the program (fill, quick, bubble, merge, index, check)

- Number of reads
- Number of writes
- Number of page misses
- Number of page replacements
- Number of disk page writes
- Number of disk page reads

In this part, you will implement all necessary functions including get/set, page replacement, and all program phases. Note that you will use the virtual memory only for array storage. For other temporary sorting data (such as indexes), you will use C variables.

Write your report for this part that discusses how you implemented the get/set methods, your page replacement algorithms, and allocation policy issues. Do not forget to include a discussion about your backing store (Section 3.6.5 of the textbook)

Part 3

For this part, physical memory can hold 16K integers and the virtual memory can hold 128K integers. Include a detailed discussion about each of these tasks in your report and list all the data for the graphs in your report. You may use tools like MS Excel to draw the graphs.

- Write a program that changes the page frame size in a loop for the given system to find the optimal page size for each sorting algorithm. The best frame size is the one that causes the smallest number of page replacements.
- (Bonus) Write another program that finds the best page replacement algorithm for each sort method. You should run many different configurations to make a decision.
- (Bonus)Draw the working set graph (Fig 3-18) for each sorting method for LRU. You need to run many experiments to get the data for this graph.

Notes

1. Always be careful about the errors, such as inconsistent memory configurations
2. Run experiments that tests end cases such as same size virtual and physical memory
3. Do not use any code from any other source even a single line!

General Homework Guidelines

1. No cheating, No copying, No peaking to other people homework

2. Follow the instructions very carefully.
3. Send required files only. Do not share your whole file system with us.
4. If you fail to implement one of the requirements, leave it be. Do not send an emptyfile
5. Respect the file names! Our HW grading is case-sensitive.
6. Failing to comply any of the warnings above will result in getting a **0** for your current homework.

Homework Instructions

1. Download and Install Vmware Player from Official site.
2. Download and install our virtual machine fromhttps://drive.google.com/open?id=1YppX3lNkyTsHV_lvA4w9TomNCUkpLeEg