CS342 Fall 2016 – Project 4
Paging Simulator

Assigned: Nov 30, 2016, Wed
Due date: Dec 13, 2016, Tue, 23:55

Objectives: *Practicing memory management, address translation, page replacement.*

In this project, you will develop a paging simulator that will simulate the paging related activities in a computer for a process. The virtual memory layout of the process will be specified in an input file. Another input file will include a set of virtual addresses that the program is referencing. The process is given a number of frames for its use. When all the frames are filled up, page replacement will take place. For that you will use an algorithm (1-LRU or 2-FIFO) . You will simulate two level paging with address split scheme: [10, 10, 12]. Virtual addresses and physical addresses are 32 bits long. As output, which will go to an output file, you will produce a list of physical addresses corresponding to the virtual addresses in the input file. For each virtual/physical address you will also indicate if that reference caused a page fault or not with an "x" sign. When a page fault occurs, if there is free frame, you will select the frame with smallest number; otherwise you will choose a victim page/frame.

The program will be called pagesim and will take the following parameters.

pagesim <in1> <in2> <M> <out> -a <alg> [-r <vmssize>]

Here, <in1> is the input file that will indicate the used virtual regions of the process. Each line in the file specifies a range X Y, where X is the start-address and Y is end-address+1. <in2> is the input file containing virtual addresses that process is referencing. <out> is the output file you will produce. The –a <alg> parameter indicates the algorithm to use for page replacement: -a 1 indicated LRU, -a 2 indicates FIFO. The -r parameter is optional. When –r is specified, the program will not use <in1> input file but will use a single virtual memory region of size <vmsize> starting at virtual address 0 and virtual addresses will be generated randomly according to a random distribution instead of taking them from input file <in2>.

All numbers (in input files and output file) are in hexadecimal starting with 0x. For example, 0x0000a40c.

Example:

pagesim in1.txt in2.txt 10 out.txt -a 1

in.txt content:
0x00000000 0x00010000
0x00100000 0x001a0000
0x10000000 0x10c00000

in2.txt content:

0x00000000
0x00000010
0x00000011
0x10000d00
0x10000d00
0x10000c20

out.txt content:
0x00000010 x
….

Report: Run your program with different number of frames and count the number of page faults for the same input files. Draw the frame count versus page fault count curve and try to fit a curve to it. Do this for each algorithm. Compare the performance of the algorithms. Apply your Probability and Statistics knowledge. Generate random input (random virtual addresses) as well. Try different distributions while generating random input; at least the following: uniform, exponential, Gaussian. Apply your Probability and Statistics knowledge.

More information and clarifications about the project can be posted to the course website and piazza.

Submission: Submit through Moodle.

Clarifications:
--- Minimum frame count is 10, maximum is 1000.
--- You will implement two level paging. You can not implement one level paging and produce results accordingly. If you do that, you will not get any points from the project (you may get negative points).
--- Assume the frames that the process can use are numbered 0 through <M>-1.