K

New York University
Tandon School of Engineering
Department of Computer Science and Engineering

Introduction to Operating Systems
Spring 2025

Assignment 10
(10 points)

A) (9 points) Create a simulation program using C++ that simulates the **second chance page replacement algorithm** in a 32-bit computer system. Your program shall include the following routines/subroutines:

The `main()` routine accepts two parameters n and p, i.e. when you invoke your program from the shell, you pass it two parameters, n and p, where n >=16 and p >=8.

It shall then invoke the `PageTraceGenerator()` to create a random page trace (only one page trace) and then it invokes the `Simulate()` subroutine multiple time, passing the generated random trace and a parameter f (number of memory frames) that ranges from 4 to p. Thus, your `Simulate()` subroutine will be run (p-4+1) times. Your main routine shall then record the number of page faults for each run.

`PageTraceGenerator()`: This subroutine shall generate a random page trace of length n, where the page numbers have values ranging from 0 to $p-1$.

`Simulate()`: Accepts a page trace and a parameter f for the number of memory frames available and returns the number of page faults. It has the following variables:

- `Frames_list`: keeps track of the frames available to the process. Logically, it's a list of available frames, initialized with values from 0 to *f-1*. Alternatively, you may just keep track of the number of frames available (`num_frames_avail`) and allocating frames starting from 0 and ending at `num_frames_avail-1`.
- A `page_table`: of length *p* entries (properly initialized), with each entry being an unsigned long integer, containing three types of information;
  - a `frame_number`,
  - a `present` bit (indicating whether the page is memory-resident or not) and
  - a `reference` bit, indicating whether a page has been referenced or not
- `num_page_faults`: keeps track total number of page faults, initialized to 0. This is what gets returned at the end of a simulation run.

The Simulate subroutine implements/simulates the page replacement algorithm by looping over the page trace, where each iteration represents an entry in the page trace. In each iteration:

- It determines if a page faults occurred and updates all the variables. If a page fault occurred, then a page needs to be loaded into a memory frame:
  - If there is an available frame, it's allocated,
  - If there are no available frames, it invokes the `FindVictim()` routine to evict a page that is already occupying a memory frame.

`FindVictim()`: This subroutine is a given a page table as a parameter, and it chooses and returns a victim page to evict based on the aforementioned page replacement algorithm.

Run your program using a page trace of length n=64, p=16 (thus f ranges from 4 to 16). Plot and submit a graph displaying the number of faults vs the number of frames allocated. If you are doing this in C/C++, you may store the results into a csv file (comma separated values) and plot using Excel.

B) (1 point) In a single paragraph (no more than 3 sentences), please state your observations, including whether it's possible to encounter the Belady anomaly or not.

What to submit to gradescope:

Please submit the following files individually:

1) Source file(s) with appropriate comments.
   The naming should be similar to "**lab#_$.c**" (**#** is replaced with the assignment number and **$** with the question number within the assignment, e.g. lab4_b.c, for lab 4, question b OR lab5_1a for lab 5, question 1a).
2) A single pdf file (for images + report/answers to questions), named **"lab#.pdf"** (**#** is replaced by the assignment number), containing:
   - Screen shot(**s**) of your terminal window showing the current directory, the command used to compile your program, the command used to run your program and the output of your program.
3) Your Makefile, if any. This is applicable only to kernel modules.

RULES:

- You shall **use kernel version 4.x.x or above**. You shall not use kernel version 3.x.x.
- You may consult with other students about GENERAL concepts or methods but copying code (or code fragments) or algorithms is NOT ALLOWED and is considered cheating (whether copied form other students, the internet or any other source).
- If you are having trouble, please ask your teaching assistant for help.
- You must submit your assignment prior to the deadline.
-