MTL 458: Operating Systems, Monsoon 2022, Homework 3

In this assignment, you are supposed to simulate a page table and given a sequence of memory accesses and whether each access was read or write, you are supposed to output the number of page faults, the number of reads from the disk and the number of writes to the disk. The specifications are given below.

Details:

(i) You need to implement 5 strategies, OPT, FIFO, CLOCK, LRU, and RANDOM, correct output to each would carry 2 points.

(ii) Your program will take input a *trace file* named `trace.in`, an integer denoting the number of frames available, and a string which would denote the strategy. For instance, if the executable is named `foo`, then an example command that should work is `./foo trace.in 100 OPT`

(iii) The trace file is a list of virtual memory addresses followed by letter `"R"` or `"W"`, which would indicate whether the memory access was a read or write. Each line of the trace file would have one such memory access. A part of the file might look like the following.

```
0x4000a340  R
0x4000a345  R
0x4000a34b  R
0x4000a353  W
0x4000a355  R
```

(iv) The string for the strategies would be named as following: OPT, FIFO, CLOCK, LRU, and RANDOM.

(v) The size of one frame is 4KB, and assume that the physical memory can hold number of frames that is given as an integer input (maximum 1000).

(vi) All the memory references are generated by the same process, so you need to maintain only one page table. Also, since the process never ends, once all the frames in physical memory are full, they will not be freed up again, except to bring a page from the memory.

(vii) Number the frames from 0 to $n-1$ where n was the input integer. Initially, you can assume all the frames to be free and can start assigning from the first. Page replacements will be needed after all the frames will be full.

(viii) You are free to choose the page table to have any format you would like (linear or inverted). Remember that the should at least have a valid bit and a dirty bit to have the correct output.

(ix) If the page to be replaced is not dirty, it does not need to get written, it is simply "dropped" from the physical memory.

(x) You do not need to simulate the disk, just keep track of number of reads from the disk and writes to the disk. Assume that a 4KB page can be written to/read from the disk in a single write/read operation.

(xi) You do *not* have to optimize the number of writes to the disk by taking into account the dirty bit for page replacement.

(xii) For RAND, use the `rand()` and `srand()` functions in C, and set the seed to 5635. You would need to find the generated random number modulo n (the input integer) to get the random frame.

(xiii) For each of the strategies, your output should look like the following.

```
Number of memory accesses:
Number of misses:
Number of writes:
Number of drops:
```

(xiv) In addition to the above, if the `-verbose` option is used then after each page replacement, the program should print which page (page number in hex) was brought in, and which page was dropped/written on disk. The format would be like one of the lines below.

```
Page 0x40345 was read from disk, page 0x40310 was written to the disk.
Page 0x40345 was read from disk, page 0x40310 was dropped (it was not dirty).
```

(xv) An example trace file is sent along with the assignment.

Guidelines: While doing the assignment, please be careful about the following.

- You will submit a `.zip` file `<Entry_No>_A3.zip` which would contain the file `frames.c`.
- It will be better to start early, as the time given for this assignment is a bit less.
- The assignment would be evaluated in an objective way, with 2 marks being awarded for each right part and 0 for the wrong ones. Please make sure your answers are in the right format and are correct. No partial marks will be awarded if the output doesn't match the right answer.