Project 2 - Page Swap Algorithms

Implement: Least Frequently (LFU) and Approximate Least Recently Used (ALRU)

Due: Monday March 21st, 11:59 AM (right before class)

Base tag: P2_BASE **Submission tag: P2**

Objectives:

- **Review/Practice the following concepts:**
  - **More modular programming**
  - **Array index mapping, more bytes**
  - **Using the Back Store (block storage file API)**
- **Become familiar with the following concepts:**
  - **Implement two common Page Swap Algorithms**

Specifications:

Prerequisites:

You need to install the the given back store implementation located in your repo, or install your implementation of back store from assignment 2. To install the back store library, you need to **sudo make install** and **sudo ldconfig** inside your build directory of the back store project.

Page Swap Algorithms:

1. LFU (Least Frequently Used)
2. Approximate LRU

Assignment Implementation:

Review and understand the library src/page_swap.c

Each page swap algorithm returns a pointer to a *page_request_result_t* **structure filled with relevant information (**look at the structure given**), <u>returning null if a page fault does not occur</u> at that clock time.**

The TAs will provide a skeleton which contains the required functions to implement.

Specific Limitations and other numerical factors:

- The virtual (logical) memory system will allow 2 MB of memory addressing
- The physical memory of the system is limited to 512 KB of RAM
- Each frame contains 1 KB of memory
- The **access bit** should be shifted onto the **access tracking byte** and reset every 100 iterations.
- Reading/writing to the backing store will require additional address translation, review the operation and valid addressing of **libback_store** for more information.

Functions to implement:

least_frequently_used (const uint16_t page_number, const size_t clock_time)
Look at notes for implementation, note the usage of access bit and access tracking byte.

approx_least_recently_used (const uint16_t page_number, const size_t clock_time)
Look at notes for implementation, note the usage of access bit and access tracking byte.

read_from_back_store(void* data, const unsigned int page) Used when a page fault happens, you will read that from the back_store into a frame that is being replaced.

write_to_back_store(const void* data, const unsigned int page)
Used when a page fault happens, to move contents from the frame table to the back store when a frame is being replaced.

Rubric:

This assignment is out of **150 points**. 60 points per page swap function, and 30 points for the remaining two back store wrapper functions.

Your implementations should be capable of running without bleeding memory or crashing. Use the tool valgrind to help find and correct memory leaks. If your solution does not compile, or has memory leaks, you will receive a zero. If your solution does not do proper error checking and handling, or does not implement the page swap algorithms correctly, you will lose up to 50% of the total points possible.

valgrind --tool=memcheck --leak-check=yes --show-reachable=yes ./page_swap_test