Operating Systems, Program 3
Page Replacement Simulation

Daniel Nix

April 30, 2015

Program Description The purpose of the page replacement simulation is to demonstrate different page replacement algorithms to a new user. A user may select from a list of page replacement algorithms consisting of:

- First In First Out
- Optimal
- Least Recently Used
- Least Frequently Used
- Second Chance/Clock

The output of each algorithm is a table representing frames filled with their respective pages in memory along with a note of what happens with each page request.

1 Algorithm

1.1 Setup

Each simulation begins by prompting the user for a number of physical memory frames (F), number of different frames the simulated program may request(P), and the number of page requests the simulated program will make(R). A table simulating empty physical memory with F frames is created first. Then a list of randomly generated pages from numbers 0 to P is created.

The program then simulates a page request for each request in the list. If there is an empty entry in physical memory it is used. If the page is already in memory it is not swapped out, but may be updated according to the appropriate algorithm. If there is a page miss the selected algorithm is invoked to determine the victim frame which is swapped for the requested frame.

1.2 First In First Out

The FIFO algorithm tracks the order the pages were put into physical memory and selects the first one to arrive as the victim frame. A deque is used to track frames as they are brought into and taken out of memory. When a page miss occurs the first item in the queue is dequed and its frame number is used as the index to store the requested page in physical memory.

1.3 Optimal

Optimal simulates a perfect situation when we know the minimum time each frame in physical memory has before being used again. Using a greedy algorithm to select the frame with the "longest-to-next-use" time as the victim frame guarantees the minimum number of page misses.

While a real machine cannot see into the future to implement this algorithm this simulation has the full list of page requests to simulate what would happen if this were possible. This provides a goal for the minimum number of page misses for a given physical memory size and page request list.

1.4 Least Recently Used

The least recently used (LRU) algorithm selects the frame that has sat idle for the longest amount of time as the victim. This comes from the assumption that if a frame has not been used in a while it is not likely to be required soon.

The LRU algorithm is implemented by setting a counter in each page to the current clock time each time a page request is made. This makes it easy to see which frame in physical memroy has not been used for the greatest amount of time.

1.5 Least Frequently Used

The Least Frequently Used (LFU) algorithm selects the frame in physical memory with the smallest number of previous page requests as the victim. This comes from the assumption that if a page has not been used many times in the past it is not as likely to be used again in the future.

The LFU algorithm is implemented by creating a counter for each possible page that might be requested. Each time a page is requested its counter is incremented. When a page miss occurs the simulation looks at the counters for each page in memory and selects the page with the smallest counter as the victim.

1.6 Second Chance/Clock

Second Chance

2 Simulation Testing

Testing for each algorithm was done by printing a verbose output for each step in the page replacement algorithm and following it by hand. When each algorithm was confirmed to be correct it was pushed to the GitHub repository. This method for testing ensures correctness for most cases but, in some extraneous circumstance, may have missed an unknown bug that could occur in a larger simulation. However, all simulations run in testing produced correct output.

3 Submission Description

3.1 Compiling Instructions

To compile the source code, unzip the prog3.tar file. From the prog3 directory, type “make" and the source code will be compiled. The page replacement executable will be placed in the root prog3 directory. Run the simulation with the command "./page_repl_sim". All output from the dash shell is directed to the terminal.

3.2 External Functions

For portability, functions were separated into paging.cpp and paging_algorithms.cpp. Header files for each cpp file were also created was also created to hold constants, structures, and function prototypes.