Hayden Migliore

CMSC 412 6380

Final Project Report

10/7/19

Introduction:

This project simulates four different page replacement algorithm: First In First Out, Optimal, Least Recently Used, and Least Frequently Used. First In First Out will always replace the oldest page when there is a page fault. Optimal relies on knowing the entire reference string. It will pick the optimal page to replace by looking ahead and choosing to replace the page that will not be required the longest. Least Recently Used keeps track of each page being used. The one that has not been used the longest will be replaced. Least Frequently Used tracks the amount of times that each page has been called. It will then replace the page that has been called the least.

Design:

The base of the loop is a while loop that prints the menu and the Options method that contains a switch statement. The user inputs a choice from the menu, then the code goes to the correct method. Options 8 and 9 are not valid, but the code prints an error and then the while loop repeats. 0 will exit the system using System.Exit(0). 1 will allow the user to enter the length of the reference string and then the reference string. 2 will allow the user to enter the length of the reference string and then will generate a random reference string that long. 3 will check to see if a reference string has been generated. If it has, 3 will print the reference string.

First In First Out starts by checking for a reference string. Then it begins by looping through the reference string and checking for a page fault by comparing the fifoArray to the reference string. If it finds a page fault, it will push each page in the fifoArray back one unit and will put the new page in the front. This way the oldest page is always the one replaced. It prints it’s results while it goes through each step, and will then print the final results in a table using Print(printArray).

Optimal also begins by checking if a reference string has been created. Then it loops through the reference string and checks for page faults with the optArray. For the first n page faults, it will fill the optArray. After that, it will begin find the optimal page to replace with each page fault. It will first check that each page is seen again in the reference string. If a page is never required again, that page is set as the optimal. Then, it goes down the length of the reference string starting at the location of the page fault. When it finds a page that is already in the optArray, it records how far it is by adding an integer to the same index in the holder array. After the holder array is filled, it compares each index and finds the optimal page. It prints it’s results while it goes through each step, and will then print the final results in a table using Print(printArray).

Least Recently Used works very similarly to Optimal. It starts by checking for a reference string, then, it begins looping through the reference string. The first n page faults are used to fill the lruArray. Once the lruArray is filled, it begins fixing page faults by finding the lru (least recently used page). It does this by comparing each page in the lruArray to the reference string thus far. It goes back through the reference string and when it finds the first instance of a page, it records the count in holder. Then holder is compared to find the lru. Finally, it prints it’s results while it goes through each step, and will then print the final results in a table using Print(printArray).

Least Frequently Used starts by checking to see if a reference string has been created. Then it loops through the reference string and looks for page faults in the lfuArray. If it does not find a page fault, it will add a count to the holder string to represent that one of the pages in the memory has been used. The first n page faults are used to fill the lfuArray. Afterwards it begins to replace pages by using the holder array. The page will the lowest count will be replaced. It prints it’s results while it goes through each step, and will then print the final results in a table using Print(printArray).