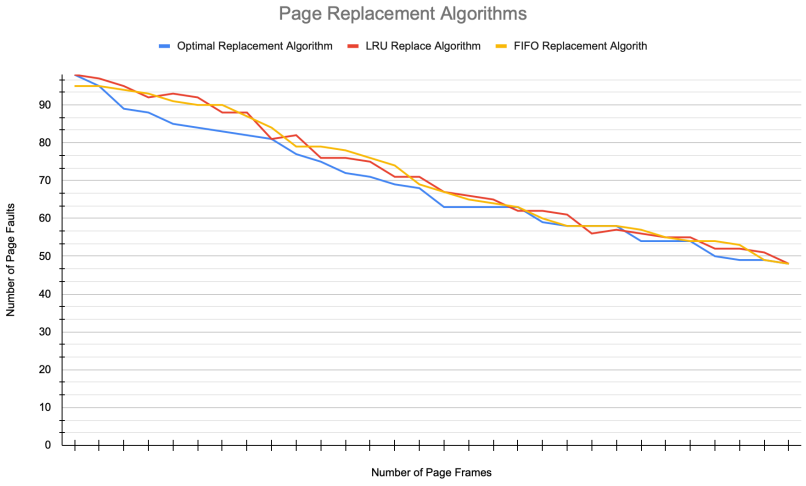
Eduardo Dominguez Arroyo
11/28/2020
OS Project 3: Page Replacement Algorithms

For the following program we were task to make a program that would implement 3 different page replacement algorithms. The goal of the program is to compare and assess the impact of these algorithms on the number of page faults incurred across varying number of physical memory page frames. The algorithms include FIFO, LRU, and OPT.

The program uses the rand() function in order to create the reference string. This string could range form 0 to 49. Furthermore we use the srand() function for the seed and used time in order to have a different string every time the program is ran. The string contains 100 total number of page number references. The program will also allow the user to choose between two option when the program runs. The first options is to run the program in a default mode and the second options is to run the program as a custom mode.

The default mode is simply going through each algorithm with the same reference string 30 times. Each time the number of frames will increase by one so it will run from 1 frame to 30. The results will be output on a text file which can be used to plot data. The second option is to do individual runs at any give number of frames and the results will be shown on the screen. To construct the algorithms vectors are used since they are dynamic. This allowed me not have to worry about allocating a large array and using it entirely. For FIFO, and OPT a vector of string frames was used. The difference was how they were being managed. The vector is restricted but the frame number. For FIFO the page numbers were taken away from the frames base on a first come first out basis. While for the optimal algorithm we looked to replace the page number in frames that would be referenced further into the future.

The third algorithm was also implemented using a vector however it was a bit different. This vector was coupled with the make_pair function. The vector would hold the page number variable from the reference string and a second integer value. The second integer value was to keep track of the “last time used” in other words from the moment the page number was allocated it would increase by 1. If the page number was called again the variable would be reset to 0. This way we would know which of the frames was used the least and we would be able to replace it.



Based on the results we obtained I can assume that the algorithms worked correctly. We see the same trend on all three as the number of frames increase the number of faults decreases. Furthermore, we see that the optimal algorithm shows the least amount of page faults. While LRU and FIFO are fairly similar and it is hard to draw a definite conclusion as to which is better based on the graph. Another interesting point we see is the slight increase we in the optimal algorithm after a period of decrease in page faults. Overall the project was interesting as we were able to see the amount of overhead each algorithm takes when making the program. We are also able to get a sense of the difference in efficiency among the algorithms.