Name: Erin Smajdek
Date: 2023-07-11
Title: Lab4

How I worked:

My program fifo.c implements the FIFO page replacement algorithm. It uses helper files that we received from the professor that can manipulate a queue of nodes.

To get started on this script, I ran the given queue_test file with the node and queue files. This helped give me an understanding of how the helper files work.

From there, I copied the skeleton file and made my edits.

To implement the FIFO page replacement algorithm, I included the node.h and queue.h files so I can use those.

I left all of the parameters from the skeleton file and included a queue (q) so I could perform the FIFO operations.

To get into the meat of the code, the actual page replacement algorithm, I start reading in lines from the file given by the user. Then, I check to see if the queue, or cache, is completely full.

If it is not full, then I check if the page number already exists (in case it is partially full), if it does, then I move on to the next page number. If the page number is not found, then that is a miss so the page number is added to the queue and the total faults increase.

If the queue is full, then I check if the page number exists in the cache. If it does not exist, then that is a miss. So I have to remove the first-in value, replace it with the current page number and increase the total faults.

If it does, then that's a hit and we can move on to the next page.

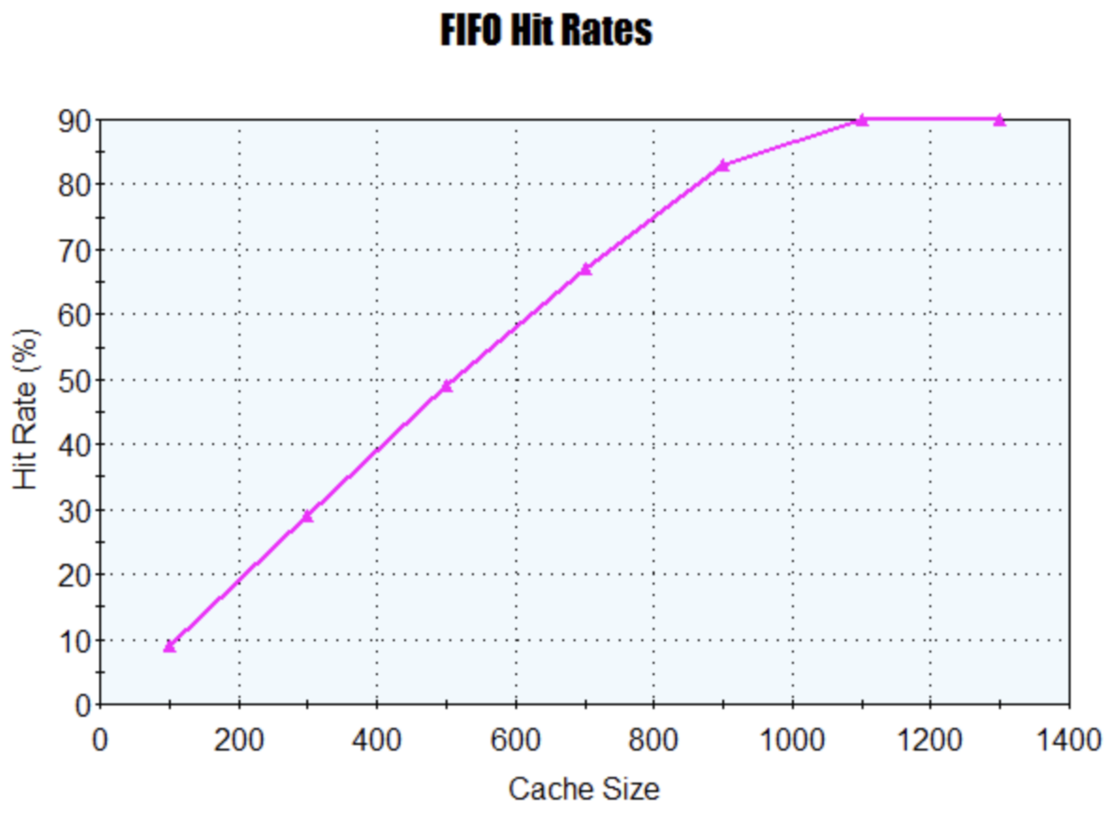
At each time I have to increase the page fault, I print out the corresponding page number that missed. At the very end of the output, I list the total number of faults.

Outcome:

As I increase the cache size, it is clear that the number of misses decreases. This makes sense because when there is a larger queue, there is a higher probability that I will come across a page number that is already listed.

For example, when I set the cache size to 100, I see ~9007 misses. But when I set the cache size to 1000, I see 999 misses. In the results file in my submission, this is the result file when I set the cache size to 1300.

As you can see from the attached graph, the hit rate absolutely increases as the cache is increased.



Overall, I greatly enjoyed creating this FIFO page replacement algorithm. This makes me excited to learn more C programming and how I can optimize running processes in the future.