John F. Lake, Jr.
CSE 30341
Project 4: Virtual Memory

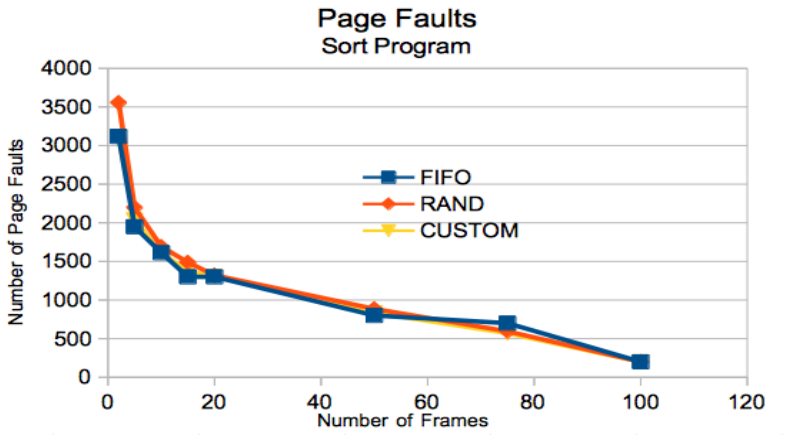
1. The purpose of this project was to determine the most efficient method of swapping out pages in physical memory. With the results of this experiment, I will be able to understand the most efficient way to work with page replacement with varying numbers of pages.

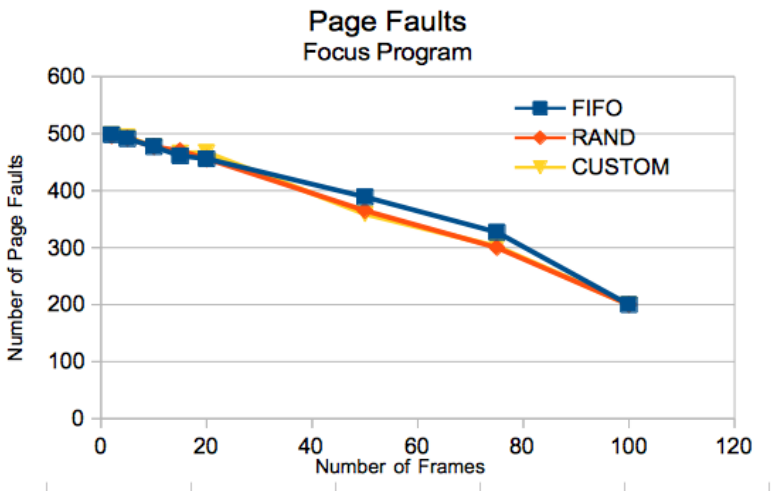
2. For this project, I ran experiments testing the number of disk reads and writes for the three different methods of page replacement required. I ran commands in the following way: "./virtmem 100 X METHOD PROG", where X is the number of pages, METHOD is the page replacement method (fifo,rand,custom), and PROG is the program to run (sort,scan,focus). All experiments were run on the student00 machine.

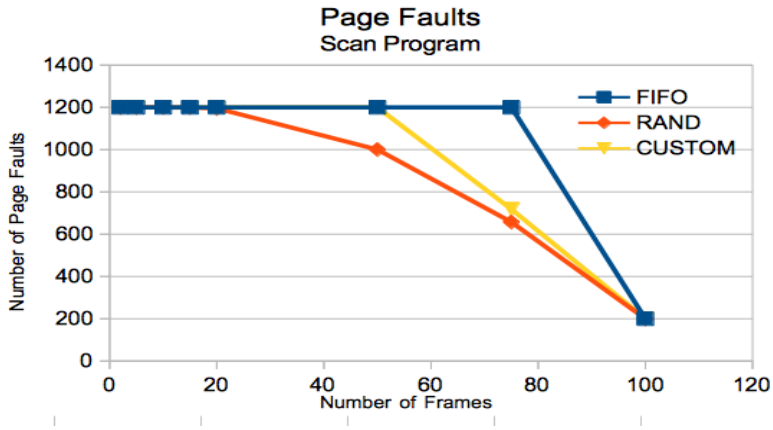
3. The algorithm I use for page replacement is very simple. It's essentially a queue that moves back and forth. The first page I choose is located at frame #0. Each subsequent page I replace is one frame after the previous one (frame #1, then frame #2, and so on) until I reach the last possible frame. Once this frame is hit, I continue the process in reverse (frame #n, frame #n-1, and so on) until I reach frame #0. This continues back and forth until the program ends. This zig zag method is similar to the fifo algorithm, but alternates the end of the "queue". Here is the algorithm illustrated with 10 frames, where X represents the page to replace:

   a) Step 1: [X][1][2][3][4][5][6][7][8][9]
   b) Step 2: [0][X][2][3][4][5][6][7][8][9]
   c) Step 3: [0][1][X][3][4][5][6][7][8][9]
   d) Step 4: [0][1][2][X][4][5][6][7][8][9]
   e) Step 5: [0][1][2][3][X][5][6][7][8][9]
   f) Step 6: [0][1][2][3][4][X][6][7][8][9]
   g) Step 7: [0][1][2][3][4][5][X][7][8][9]
   h) Step 8: [0][1][2][3][4][5][6][X][8][9]
   i) Step 9: [0][1][2][3][4][5][6][7][X][9]
   j) Step 10: [0][1][2][3][4][5][6][7][8][X]
   k) Step 11: [0][1][2][3][4][5][6][7][X][9]
   l) Step 12: [0][1][2][3][4][5][6][X][8][9]
   m) Step 13: [0][1][2][3][4][5][X][7][8][9]
   n) This continues until the program ends.

4. Experiments:
   a) For the experiments, I ran the program with a varying number of frames, and I tested each algorithm with each program (sort, focus, or scan). My results are listed below.
   b) Page faults for each method and each program:
      - Graphs:



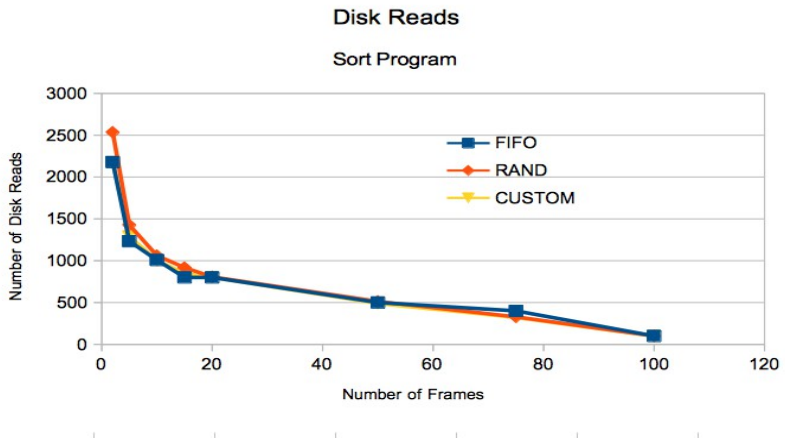


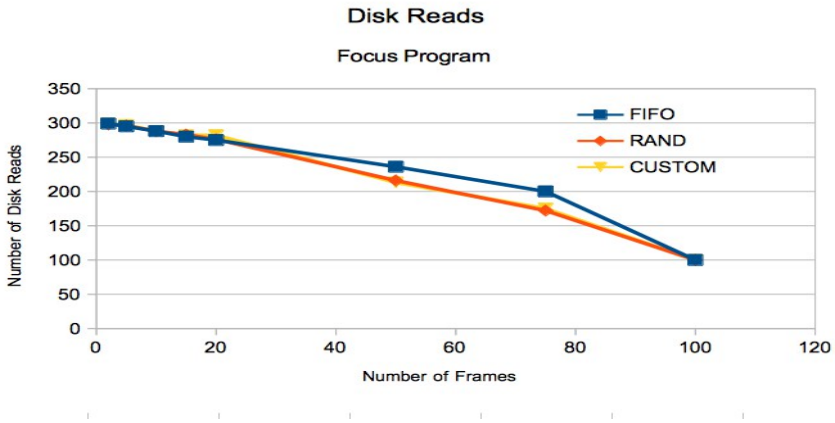


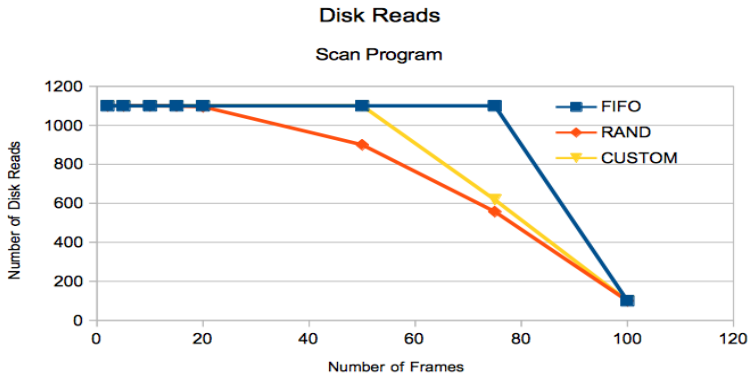
      - Notice that my custom method is the most efficient for the sort and focus program from roughly 60 to 100 frames. For the scan program, my method was in between the performance of the FIFO and RAND methods.

c) Secondly, the disk reads:

- Graphs:



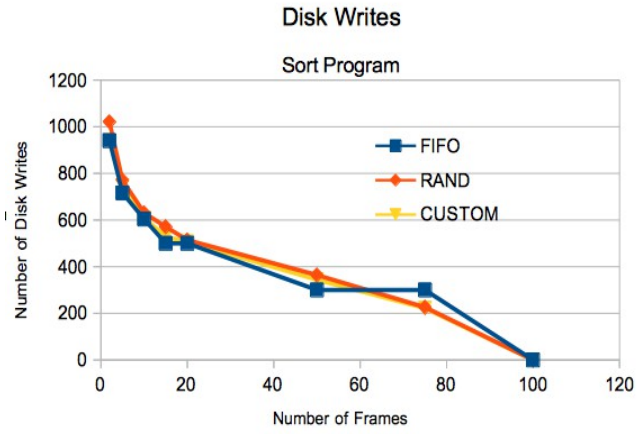


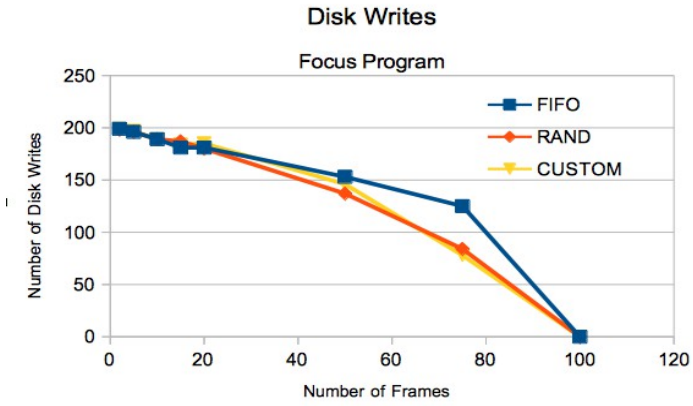


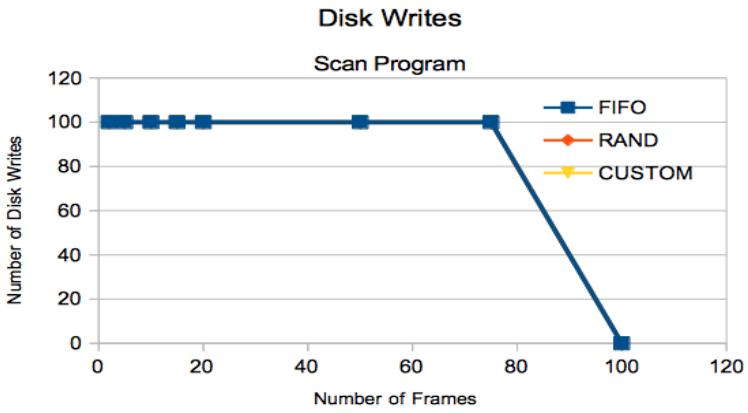
- Explanation: for disk reads, the custom algorithm I made was about the same as the RAND algorithm, and was definitely better than the FIFO algorithm. For some setups (with particular numbers of frames), my algorithm was the best by a slight amount. At 50 frames, for example, my custom algorithm was better than both of the others for the sort and focus programs, but not for the scan program.

d) Thirdly, the disk writes:

- Graphs:







- Explanation: my custom algorithm is the best from around 60 frames to 100 frames for the sort and focus program, but for the scan program every method has the same results.

5. Overall Explanation of Results:
    a) In general, my algorithm performed on par or better than the other two algorithms for the sort and focus programs, but for the scan program the random algorithm was by far the best.