Louis Rozencwajg-Hays, Nicholas Gattuso, Kexian Wu

I pledge my honor that I have abided by the Stevens Honor System.

Analysis

Before we ran the tests, we expected LRU to be the algorithm that would result in the least swaps, followed by Clock, and then FIFO. LRU is the most efficient because only the most rarely used pages are swapped, leading to less faults. FIFO is the least efficient because a frequently used page could be swapped simply because it was entered early. Nevertheless, FIFO and Clock would likely have similar performance, because they have very similar implementations. We also predicted that the demand and pre-paging policies would produce very similar results. Additionally, we theorized that with a bigger page size, we would see more page faults. Because of proportional allocation, a larger page size will result in less pages per process. Therefore, having less space will result in more swapping.

After we ran the tests, we found that while our deductions about the efficiencies of the algorithms were correct, our other conclusions were wrong. For demand paging, as the page size increases, the number of faults decreases until size 8. Afterwards, for FIFO and Clock, it begins to increase again. With pre-paging, LRU and Clock slowly decline until page size 8, but then suddenly spike when the page size is 16. FIFO, however, declines until 2, then slowly rises until 8, before spiking like the others. Additionally, demand starts at a high page fault rate, while pre-paging starts at a moderately low rate. Nevertheless, all algorithms start at almost the exact same value (demand and pre-paging have different starting values).

If a random memory access trace had been supplied, it is possible that the results would change by a considerable margin. This is because by having random input, the program may be lucky or unlucky with regards to the order in which the pages are inserted, leading to noticeably more or less page faults.

Expected:

(For both demand and pre)

-LRU is most efficient, then Clock, then FIFO

-LRU is most efficient because only the most rarely used pages are swapped

-FIFO is least efficient because a frequently used page can be swapped simply because it was entered early

-FIFO and Clock should be very similar because they have a similar implementation

(Demand vs Pre)

-Demand should be more efficient with more space b/c there is more room to swap pages

-Pre should be more efficient with less space because due to proportional allocation, a larger page size will result in less pages per process. Because pre-paging requires a working set of pages, not having enough space for a working set will cause thrashing. Thrashing is the act of busy swapping pages due to a lack of space, in an attempt to conform to the pre-paging policy.

FIFO: O(n)

LRU: O(n)

Clock: O(n)