Dalton Meny

CS 470

Memory Management Project

There is a makefile included. To run use ./memory followed by the input file and number of frames. (./memory 10p9898.dat 50)

My program consists of 4 auxiliary functions and the main function. In main, the input from the file is parsed and read into a vector so it can be easily accesses and so it can look ahead when needed. The main then calls the other functions and prints out the results. A struct called mem holds the PID and the page number from the input and is the type that is stored in the vectors. The first function is a helper function just to easily compare the PID and page number to anther PID and page number. All three management functions take an input vector of type mem and the number of frames and return an integer which is the number of page faults. In each function, a deque is used as the table so when an element is deleted the others fall in place, but with a deque every element can be read and deleted. The aspect of a queue is very important for FIFO and LRU. In FIFO, if the page table is full it simply pops the first element of the table and the new element is pushed on the back. In LRU, it is almost exactly the same expect that if an element is already in the table it gets deleted and pushed onto the back. This ensures that the front of the table is the least recently used. The OPT function is more complicated. If the table is full, a new temporary table (deque) is created. It reads through the rest of the input and if it matches an element in the table it is added to the new table and deleted from the old table to prevent duplicates. This happens until the new table is equal to the number of frames minus 1 or the input ends. Then the table is set to the new table and the element that needed to be added to the table is pushed on.

Questions

1. I would choose to use LRU. OPT is definitely the hardest to implement and in my program, it seems to take the longest to run. Even though OPT has less page faults than FIFO and LRU, it was never by a huge amount. Usually FIFO and LRU were very close if not the same, but LRU did run marginally better than FIFO most of the time and it was never worse. This is why I pick LRU over FIFO and OPT.
2. I found OPT the hardest part to implement. It's not that it required much more code to do, but it definitely required more brainpower and testing than the other two methods.
3. I found FIFO the easiest part to implement. It was very straightforward to do, and it did not require as much effort as the other function. But it seems the easier to implement, the more page faults.
4. There's not much I would change in my project, but if I had to do something I would maybe like to try to make OPT more efficient (if it's even possible). Reading through the input so much definitely makes it run slower.
5. I found it interesting how easy it was to implement FIFO and LRU. I also found it interesting how close their page faults where to each other when running all those files.