CS214 Project 2: Basic Data Sorter - multithreaded

Section: 2, Charmian Goh, Nikita Kolotov

Compilation:

We used the -pthread flag during compilation to help support program functionality. Our compilation line went: gcc -Wall -pthread -o multiThreadSorter_thread multiThreadSorter_thread.c multiThreadSorter_thread.h mergesort.c

Design:

- **multiThreadSorter_thread.h-** This file contains a typedef for a struct movie_data that is used to store each tuple from the csv file. It also contains a define initial_size that is 1 to be used during reallocation of memory when we need to expand the memory for more tuples. It also contains method declarations to helper methods to be used in multiThreadSorter_thread.c .

- **mergesort.c -** Mergesort.c takes in an array of structs and contains 2 functions that will modify them. Mergesort will take the array and find the midpoint, then it will recursively split them until they are singular. Then it will throw those array elements into mergeArr and it will then merge the elements in ascending order into the original array. This function uses temp arrays that save all the data until no longer needed. At the end of mergesort, a sorted array of structs is released back into the main program.

- **multiThreadSorter_thread.c -** There are 5 helper functions and a main function. The print_element function is used to print tuples to a file. The struct_it function is used to put valid tuples from csvs the program finds into structs to then print into a collection file in the proper format. The sort function takes in the collection file, sorts it and prints it out to an AllFiles-sorted-<column>.csv. List_dir traverses through the directory and spawns a new thread each time it finds a new directory and a valid csv.

Assumptions:

- Each row should contain under 1000 characters.

Difficulties:

- Figuring out where to put the mutex locks was difficult. In many instances, even with the locks we found that variables were being changed and it was saving files to different directories. Our main issue was figuring out how to synchronise the threads so that they did not overwrite variables the other threads needed.

Testing:

We executed the code using various combinations of types of columns and different source and output locations. We also executed it without a source location and without an output location. We used the time command to determine the real time execution time in comparison with project 1.