Jon Cobi Delfin and Jeffrey Pham
Systems Programming
Project 2: MultiThreaded CSV sorter

Design, Assumptions, and Difficulties

Starting the project, we had a lot of difficulties understanding how threads work with one another and how to integrate that into the code we already have. Multithreading has a good amount of differences from multiprocessing, as multithreading can all interact with the same data whereas processes all act independently.

The program takes in the same parameters as Asst1 and then creates a new csv file called AllFiles-sorted-<columnNameToSortOn>. It works similar to Asst1 in that it navigates all directories/files starting from the cwd directory or a specified directory if a user inputted one. If it's a file, it will check if it is a csv and if it is, it will create a thread for it and write the contents of the csv into the new AllFiles-sorted-<columnNameToSortOn> file. The threads get locked and unlocked to make sure that multiple threads don't modify the same thing at the same time. After the AllFiles-sorted csv is populated with all the lines of the csv found, it will then call mergesort on it to make sure it is sorted.

The threads work on a globally declared linked list that holds all the information so that it could all be joined together correctly. Additionally, while the threads are running, they will print their TIDs and increment a thread counter that keeps track of the number of threads that get created.

At the moment, our program only works with csv files that have all the headings and in the correct order. We tried to implementing a binary array that includes all the headings and would insert a 0 and 1 if the heading was present and there was an entry, and 0 otherwise. Then, we would repopulate the array with the values once it was established. However, it is not fully working and does not work with randomized headings and csv files that are lacking headings.

Testing

We tested our code using the test cases from Asst1. We found that if a csv file does not include an empty line at the end, the AllFiles-sorted csv file will be slightly off because the last line in each csv file will not have a new line character. Therefore, there will be certain lines in the final csv file that will be 2 lines in one. Therefore, if possible, it would be ideal to have csv files with an empty line at the end to ensure that each lines includes a new line character so that our final csv file will be correctly formatted. It does not affect the actual sorting as it sorts before printing it to the csv file, so it is just the format of the csv file that is incorrect.

Using Our Program

We have included a makefile, so please call make in order to create the executable named multiThreadedSorter. From there, you may run it in the following fashion:

No specified starting directory or output directory, sorted csv files will be placed in the directory the original files were found in:

./multiThreadedSorter –c <columnName>

Only starting directory specified, sorted csv files will still be placed in the directory the original files were found in:

./multiThreadedSorter –c <columnName> -d <startingDirectoryPath>

Only output directory specified, sorted csv files will be placed in the output directory:

./multiThreadedSorter –c <columnName> -o <outputDirectoryPath>

Starting directory and output directory specified, sorted csv files will be placed in the output directory:

./multiThreadedSorter –c <columnName> -d <startingDirectoryPath> -o <outputDirectoryPath>

Order of the flags may be in any order.

If you are expecting an error in any of the csv files, PLEASE include 2>nameOfErrorLog.txt at the end of the input so that the error statements are not printed in the terminal itself. The reasoning for this is that we made it so that the error statements print out the name of the csv file that the error occurred on, so we could not use write() as the name of the csv file changes from one to the next and we instead used fprintf() to stderr, which will also print to the terminal if not redirected. The input would look like the following:

./multiThreadedSorter –c <columnName> 2>errorLog.txt