Project 3 - Server Client sorter

Server: Neel Patel (ruid: 163007037 netid: nap166)
Fareen Pourmoussavian (ruid:165007750 netid:fp183)

Client: Antonio Diaz (ruid:159006867 netid:avd41)
Justin Scro (ruid:164005035 netid:jss381)

SERVER

- Design
  - Multithread for each client connecting to the server
  - Each client sends over all their csvs to server, Server adds all of them into a Queue, then mergesorts and joins them together resulting in one single large struct
  - Server sends the single struct back to client
  - The entire process is mutex locked so ONLY ONE CLIENT'S data is being manipulated at any given time.
    - I.e. first client1's data will be stored and returned, then client2's data. It is in order of connection.
  - Used Queue implementation from Project 2
  - Used Multithread implementation from Project 2
  - Modified main method from Project 0 to allow storing into structs
- Compilation and Execution
  - Compile with gcc server.c -lpthread -o server
  - ./server -p <port number>
- Assumptions
  - Each client must have csvs of movie_metadata.csv format ONLY. Lines can be removed, but with proper care.
  - For time's sake, each client should send no more than 2 csvs of movie_metadata.csv.
  - Server allows client connections from same computer, physically different computer but same local network, and any two computers across a wide network. However Server does not perform well the further the two computers are.
  - >>> According to the professor, there is No Penalization for things we have been graded on already: Mergesort accuracy, printing out csv, joining csv in one large struct accuracy (project 2).
- Difficulties
  - Storing csvs into global Queue data structure
  - Resetting variables so next client does not have issues
  - Learning about client and server sockets and the various functions needed to implement the server and clients.
  - Tracking Segmentation Faults and fixing them

CLIENT

- Design

    - Checks parameters and assign proper variables
    - Creates socket and connects to the host using hostname and port
    - Counts number of lines to send to server and sends that number to the server
    - Traverses path and send each line of each found valid csv to the server
    - Sends dump request via column to sort
    - Opens output file and print lines as they come back from the server
- Compilation Execution
    - Compile with gcc client.c -lpthread -o client
    - ./client -c <category> -h <hostname> -p <port number> -d <search dir> -o <output dir>
        - (-o and -d can be switched and are optional)
- Assumptions
    - Server knows the order in which the client will send data and is following the same protocol for encoding and decoding
    - After a client connects to the server, a sort request is implied and is immediately followed by a dump request
    - Dump request equal to asking for a large sorted csv.
- Difficulties
    - Synchronizing server and client
    - Implementing socket connection with write and recv
    - Encoding data properly so that it may be understood by the server

SERVER AND CLIENT

- Testing Procedure

# of Server	# of client	# of csvs per client	Result
1	1	1 (5044 lines)	Success
1	1	2, 3, and 4 (all 5044 lines)	Success
1	2	1 (5044 lines)	Success
1	2	3 (5044 lines)	Success

Diagram of how this project works