Data Sorter server/client

Sorts a large amount of data, by using a multithreaded C server and C client program to sort a list of records by the data in a given column. Uses the concepts of file/directory access, pthreads, and synchronization constructs (locks/semaphores), and sockets. Uses the merge sort algorithm. The client will read in CSV files from multiple directories and instead of sorting them itself, it will connect to a server, send the files to it, and the server will sort them and send the files back. Imagine a situation where there is one machine that is very powerful and flushed with resources. Then imagine a tiny, embedded system that must accomplish CPU-intensive tasks. That is what this project is trying to implement, the tiny system sending all of the work to the powerful system and get the results back much faster.

Execution

Code will read in a set of parameters from the command line. Records will be stored in CSV files in the provided directory. Directories may have multiple levels and you must find all CSV files. Code ignores non-CSV files and CSV files that do not have the correct format. The first parameter is '-c' to indicate sorting by column name, the second will '-h' to indicate the hostname of the server, and the third will be '-p' to indicate the port number the server is listening on. The fourth parameter is '-d' indicating the directory the program should search for .csv files. This parameter is optional. The default behavior will search the current directory. The fifth parameter is '-o' indicating the output where the sorted CSV should be written to. This parameter is optional. The default behavior will be to output in the same directory as the source directory:

./sorter_client -c food -h hostname.com -p 9091 -d thisdir/thatdir -o anotherdir

The server has only one parameter: '-p':

./sorter_server -p 9091

Functioning

Client traverses all subdirectories under the directory it is given. For each file it finds, it spawns a thread. Each thread it spawns should read in the file, construct a search request, connect to the server, send the request to the server and then wait for and read back the server's response to be sure the file was sorted. Once the client has sent all files under the search directory to the server, it should send the server another request to give it the sorted version of all the files it was sent. When the server responds and sends back the sorted version of all the files, the client outputs a single CSV file named, AllFiles-sorted-<columnName>.csv.

Server will open a port and wait for connection requests. On connection, it will spawn a service thread to handle that connection and go back to waiting for requests. Each service thread should read in a client request, if it is a sort request, it performs the sort and store the results at the server. If it is a dump request, it merges the current collection of sorted results into one sorted list and send the result back to the client. The server will run until stopped by a SIGKILL. To STDOUT, outputs a list of the ip addresses of all the clients that have connected.