Καραγεώργου Ιωάννα 1115201600057

Server

In main.c server is getting ready to establish a connection following the socket, bind,listen and accept prerequisite steps. He manages his connection via select and serving one client at a time.Then he is ready to read and offer his services to the clients.
When he receives a LOG_ON message he uses the functions: server_Log_On, takeClientInfo, searchL and reply_to_clients, in order to store this client in his active clients list and inform the other connected clients about the the presence of the new one.
He follows similarly steps when he received LOG_OFF ,using the functions: takeClientInfo, deleteNode and reply_to_clients.
When he receives a GET_CLIENTS message ,he parses the client list and sends to the asking client a string with all the information about the other connected client to him.
Server terminates when he receives a SIGINT from the user (cleaning up his allocated memory).

Clients

At first a client initializes his ring buffer, makes a new directory to stores what he will be receiving from his peers(only the first client getting in the system), initializes the mutexes, initializes the signals and then he sends a LOG_ON message to the server. Then he creates a listening socket and waits for a peer to try to connect. Same as server he uses select to choose with whom he will communicate. With other words now he is operating as a server for other peers and the other way around. After he sends a GET_CLIENTS message to the server, he stores his peers to a list and he inserting them into the ring buffer, waking up the sleeping worker threads. When he receives a USER_ON message he puts the new client information in the client list, after chcking that this client not already exists in the list,and then he puts him in the ring buffer.
When he receives a USER_OFF message ,he deletes this client from the client list. If this client has sent him already his files, he doesn't bother and lets the worker thread to ingore him if he is waiting in the ring buffer.
When a worker thread wakes up he reads the buffer and follows the predicted steps. He only writes in the ring buffer when he send a GET_FILE_LIST message to a connected peer and he returns a list with his files. After that he continues his work as a reader.
The version of a file is based on checksum. When a byte changes then the checksum is different. With a hash function we have the advantage of finding on the spot the necessary version of a file. When a client receives a file he uses function from the project 2 on Sus.Pro. , these function complete the plysical creating of the file in the right directory and subdirectory. All the received files from each client are underneath the main directory called "PeerDir".
A client terminates when he receives a SIGINT from the user, deallocating his structures.Then all threads are terminating deallocating their space.

Problems:
- Only the port is sending in its binary form,I had very difficult time to convert IP address and the inet_ntop and inet_pton returned different value if the clients had the same IP address.
- I couldn't run valgrind on my computer. It keeps stalling when the worker threads started. Maybe that's because valgrind made them run in single core.

create_infiles.h is a script for making input directories(chmod u+x scriptname to make it executable).