**Main thread:**

The server will have the main thread which will initiate the socket and accept all clients. After starting the server, the main thread will initiate all the global variables. Some examples of global variables:

* List of groups. It is linked list of struct called group. The struct will contain all necessarily information about one group and variables that can be used within this group: groupname, maximum number of users, current number of users, all semaphores related to this group, the text of the quiz (possibly as an array or list of characters), the result of the quiz, etc.
* Set of sockets (fd\_set). We have functionality that requires some storage of all connected sockets (users). The example of such functionality is broadcasting the updates in groups to all users.
* Global semaphores. We will need some global semaphores common to all users and groups.
* Etc.

After initiating all the global variables, the main thread runs a loop and wait for the new clients (accept). When the new user has arrived, the server adds its socket to the fd\_set and creates another thread that will service only this user. Then user waits for another user.

**User’s thread:**

Each user’s thread has its own local variables related to only this user: its socket, username, score in quiz, etc. at the beginning of a user’s thread, it will initiate all the necessary variables and will send the list of groups to the client. After that, it runs loop which service the client requests. If the client joins the group, the thread removes it from fd\_set (FD\_CLR), and makes a broadcast about the updates in the group to all users in fd\_set. So, fd\_set contains only sockets that’s client not joined any group yet. Accordingly, when the user leaves the group, the thread adds its socket to fd\_set.

When the user takes a quiz, the thread runs another loop (inside the previous loop) that services the user’s answers and write the results of the quiz. The select function will be used to implement timeout. After the quiz completed, each thread adds its score to the result string (in group struct), and waits until the last user adds its score. After that all users added their results, the result string is written to each client, and the loop ends. The upper loop now waits the commands from the client.

When the user creates group, the thread reads details about the group from the user, initiates all the related variables and semaphores, and accordingly creates the group struct and adds it to the global groups list. Then the thread creates another thread that waits the admin commands from the user, while the current thread will wait for news from the created group.

All critical blocks of codes are done in semaphore\_nutex. All other details will be added later, and some changes may be done during the development.