Assignment 2 - Part 2
March 15 and 17, 2023
Multi-Client Chat server

In this assignment you need to implement a client-server chat application using TCP sockets. There will be a single server and multiple clients communicating with the server. The server process can handle at most 10 concurrent connections. Each client process will open a new connection with the server and add the client socket id to its fd_set(). **Use *select* to handle multiple client requests. The functionalities to be added in this assignment will be in continuation to the last week's assignment.**

Task : Simple Chat with Group Functionality.

1. **Group Formation** : In this part, a client should be able to make a group having a unique id with available active members (max 5 in a group). The person **who shoots the group formation request is the admin**. Two basic paradigms that have to be implemented are:
    a. ***Without permission of other members***: The client should be able to make a simple group by typing "/makegroup" followed by the client ids of all the members that the admin want to include and the members will be automatically joined (making a group in WhatsApp). A notification should be sent to those specific clients that they are added to the group. [20]

    b. ***With permission of other members:*** The client should be able to make a request of forming a simple group by typing "/makegroupreq" followed by the client ids of all the members that the admin want to include. A notification should be sent to those specific clients asking for permission to join the group. If a particular client wants to join, then it may send "/joingroup" followed by group id as the response. Clients can also decline the joining request using "/declinegroup" call as the response. [20+10+10]

    ***NOTE**: Group will be created only after receiving replies from all the clients to whom join request has been sent before by the admin(or creator) of the group.*

    c. ***Admin privileges***: After formulation of group:
        i. ***Making new admins:*** the primary admin can elevate the privilege of the group members to be that of admin using "/makeadmin" command. [10]
        ii. ***Adding new member(s) to the group:*** Both the primary admin and the newly added admin can add more member(s) to the group by using the "/addtogroup" command specifying the client id(s). At any point in time the number of members in the group should not exceed 5. A notification should be sent to the specific clients that they are added to the group. [10]
        iii. ***Removing member(s) from the group:*** Both the primary admin and the newly added admin can remove member(s) from the group by using the

"/removefromgroup" command specifying the client id(s). A notification should be sent to the specific clients that they are removed from the group. [10]

2. **Making Broadcast-only group:** This is a concept you must be familiar with in Whatsapp. It is a special category of group in which only admins are allowed to message. Any admin (*of a group that (s)he is part of*) should be able to initiate a request to make the group broadcast-only using "/makegroupbroadcast" request. [20]

3. **Group Chatting and Status:** Once the group is formed, a client should be able to send a message to the group that (s)he is part of using "/sendgroup" call. Additionally, a client should be able to retrieve the details of active groups (s)he is part of using the "/activegroup" command. [20+20]

   [**Bonus-30**]: Advantage of attempting the bonus is as follows. At the end of the semester if you are at a grade boundary, we shall check if you attempted the bonus problem correctly. The additional marks (if any) that you obtain in this part will be used to lift your overall marks and this might help you to move to the next higher grade.

4. **Initiating request to become admin:** Once the group is formed, a client should be able to send a request to the group admin(s) to become an admin using "/makeadminreq" followed by the group id. All the admin(s) of the group must be notified of the request. They can send a response of approval using "/approveadminreq" or decline using "/declineadminreq". In case more than one admin replies, you should be able to resolve the conflicts via majority voting. Consider acceptance as the default in case of ties.

Some Scenarios to consider

1. If all the admins of the group have gone offline the groups should be deleted automatically and all the clients should be notified of the group deletion via a message from the server.
2. If a client that is part of some group is going offline, you should appropriately handle scenarios where the admin can pass this client's id in one of its requests.
3. For all those commands that require admin privileges, you have to ensure that the request is indeed sent by the group admin. If not, you have to show an error message stating: "!!You are not an admin!!".

Functionalities to be included:

1. **/makegroup <client id1> <client id2> ... <client idn>**: A group with unique id will be made including all the mentioned clients along with the admin client.
2. **/makegroupreq <client id1> <client id2> ... <client idn>** : A group having unique id should be made with currently only the admin client. The request message for joining the group should be notified to all the specified clients. Clients can respond to join that group.
3. **/joingroup <group id>** : If this message is sent by a client having the request for joining the group, then he will be added to the group immediately.

4. **/declinegroup <group id>** : If this message is sent by a client having the request for joining the group, then the client will not be added to the group.
5. **/sendgroup <group id> <Message>**: The sender should be in the group to transfer the message to all his peers of that group. The message should be sent to all the peers along with group info.
6. **/makeadmin <group id> <client id>:** The admin should be able to alleviate the privileges of *client id* to that of admin.
7. **/addtogroup <group id> <client id1> <client id2> ... <client idn> :** The admin should be able to add member(s) to the group.
8. **/removefromgroup <group id> <client id1> <client id2> ... <client idn> :** The admin should be able to remove member(s) from the group.
9. **/makegroupbroadcast <group id>:** Any admin of the group should be able to modify the type of group as broadcast-only in which only admins are allowed to message.
10. **/activegroups** : To display all the groups that are currently active on the server and the sender is a part of. Here you have to display at the client side the group ids followed by the group admin's client id, and the ids of all the clients that are part of this group.
11. **/quit** : The client will be removed from the server. This client will be removed from all the active groups.
12. **/makeadminreq <group id>**: A member of a group should use this functionality to make a request to become an admin. All the admin should be notified of this request and they can respond appropriately.
13. **/approveadminreq <group id> <client id>**: An admin of a group can approve the /makeadminreq using this functionality.
14. **/declieadminreq <group id> <client id>**: An admin of a group can decline the /makeadminreq using this functionality.

12-14: Bonus queries.

Things to remember:

Server Side

1. Use select() for the implementation.
2. Use send() and recv() system calls. It will make your life easier.
3. If client B wants to send a message to client A, B won't be able to send to A's socket directly. Instead use a message details table at the server side. Client B will send a message to server and then server will pass on the message to client A.
4. Server will check the message details table. If it finds an entry of a particular client with dest_id as its sock_id, then send the message to the client. Remove that entry.
5. Log messages on server (server's terminal).
6. Handle cases when a client is no longer a part of a group, and similar cases.

Client Side

1. Reading from the standard input and writing to the server (send() system call) and reading from the server (recv() system call) will be handled by different processes (use *fork* here, we believe you know the reason).