Assignment Report :

Name - Harsh Vijayvargiya
Roll No. - 22CS60R80

To Run the program use the following command :

- For server : gcc server.c -o filename, for client : gcc client.c -o /filename
- ./filename portno
- At client side commands are given using the same syntax as specified in the Assignment pdf, in the commands client_id specify client unique ID and not the socket ID of the client.

Macros Used :

- max_clients : defining the number of max_clients that can connect to the server.
- buffer_size : defining the maximum size of buffer which is transmitted between client and server at a time.
- max_group_mem : defining the maximum number of members a group can have.
- max_gruops : defining the maximum number of groups the server can hold.

Structures Used :

- client : defines how the client is created and it includes,
    - address : sockaddr_in of the client.
    - sockfd : socket fd of the client.
    - uid : unique id of the client.

- group : defines a group and it includes,
    - admin : array of clients which stores the admins of the group.
    - admin_count : no. of admins of the group
    - gid : group's unique ID.

- members : array of clients which stores the members of the group.
- mem_count : no. of members of the group.
- broadcast_group : defines whether the group is broadcast or not.
- requested_mem : array which stores the unique ID of the clients which are requested to join the group by the admins.
- requested_admin : array which stores the unique ID of the clients which have requested to be an admin of the group
- decline_request_count : array which stores how many of the admins have declined the admin request of the client corresponding to the same index in the requested_admin array.

Global Variables Used :

- connected : stores the number of clients connected to the server.
- client_unique_id : variable used to assign unique ID to a newly connected client.
- group_unique_id : variable used to assign unique ID to a newly connected group.
- clients : array of type client* which stores the clients connected to the server.
- groups : array of type group* which stores the groups connected to the server.
- buf : defines the buffer.
- uid_to_newsockfd_map : array which stores the value of uid of the client which is stored at the i'th index in the client array.

Functions Used :

- main - It creates a server and waits for connection from the clients, it defines various fd sets for master, read, write and exception and sets all of them to zero and sets sockfd of the server in master fd. Then we set all the groups and clients as NULL using the setGroupsNull and setClientsNull function then we wait for any interrupt in the read,

write and except fds using select function. If the readfd is set corresponding to the server socket then some client is trying to connect so we handle it, if the server has reached maximum client limit then we call handleExceededConnections otherwise we add the client to the server using the handleConnection function. Else someone client has given some command which we handle by calling performAction function if the readfd set is set but the number of bytes sent are 0 or less that means the client has left so we remove it using the function removeClient and also remove it from the master fd set using FD_CLR.

- sigCHandler : handles ctrl+c signal from the terminal, broadcasts to all the clients that the server is closing and then closes the server.
- sigZHandler : handles ctrl+z signal from the terminal, broadcasts to all the clients that the server is closing and then closes the server.
- error : prints the error onto the terminal.
- getEmptyClient : scans through the clients array and returns the index which does not hold any client.
- handleConnection : Used to connect a new client to the server, creates a new client* variable assign all the values to it, find the empty index in the clients array and make that index point to the client, increase the connected variable by one.
- handleExceededConnections : Used when a new client tries to connect to a server but the server has reached maximum client limit. Connect with the client and then message it, that server has reached the maximum limit and disconnects with it.
- removeClient : Used to remove a client corresponding to the passed client id from the server. Finds the unique id of the passed client. Broadcasts to all other clients that the client is going to exit the server. Calls the function handleRemoveFromALLGroup to remove the client from all the groups. Then remove the client from the server as well.
- serGroupNull : make all the pointers of the groups array point to NULL.
- setClientNull : make all the pointers of the clients array point to NULL.

- handleRemoveFromAllGroup : used to remove a client from all the groups to which it is connected to. Scans through all the groups finds if the clients is member of the current group by calling the function isGroupMem, if it is then check if the client is one of the admin of the group if yes then remove it as admin reduce the admin count and if admin count becomes zero that means the group has no admins left so it deletes that group by calling handleDeleteGroup else it just removes the client from group members list as well.
- handleDeleteGroup : used to delete the group corresponding to the passed group id. First it finds the index of the group array which points to the group by calling getGroupIndex function than it messages all the members of the group that it is going to be deleted then it deletes the group by making the group array index point to NULL.
- broadcast : Used to broadcast a message to all the clients except for the calling client.
- handleSend : Used to send a message from the specified client. Firstly the receiver's unique id and the message is retrieved from the buffer. Then we find the index at which the receiver is stored using the unique id and the uid_to_newsockfd_map array. It handles the case if the sender provided a Client ID which does not exist, otherwise it sends the message to the client.
- handleActive : It returns the list of active clients to the calling client. It retrieves the unique ID of all the active clients in the buffer and sends this buffer to the calling client.
- handleInvalidCommand : If the client gives a command which is not recognized by the server then we send invalid command message to the calling client.
- getUid : It returns the unique ID corresponding to the client's FD and if no client with specified fd exists then it returns -1.
- getFreeGroupIndex : It returns the first free index in the group array.
- isClientActive : It checks if the specified client is currently active in the system if yes then it returns 1, otherwise it returns -1.
- getClientIndex : It returns the index at which the clients array stores the specified client, if no such client exists then it returns -1.

- isGroupMem : It checks if the specified client is a member of the specified group if yes then it returns 1, otherwise it returns -1.
- isRequestedMem : It checks if the specified client is among the requested members to join the group or not if yes then it returns 1, otherwise it returns -1.
- allotGroupMem : It is used to allot group members to the specified group from the specified admin. It retrieves the unique ID passed by the admin one by one from the buffer and do the following,
  Firstly if the specified unique ID is same as admin unique ID then no action is taken as admin is already a member of the group,
  If the member count of the group is more than or equal to the maximum possible size than admin is informed that no more clients can be added,
  It also checks if the specified client is active or not if not then the admin is informed that the client is inactive and can't be added to the group,
  It also checks if the specified client is already a group member, if yes then the admin is informed that the client is already a member and can't add it again to the group,
  Otherwise, the client is added to the group and the client is informed that it has been added to the group with group ID given.
- allotGroupMemReq : It is used to request the clients to join the group with specified group ID from the admin given by admin ID. Client IDs are retrieved from the buffer one by one and following things are done,
  Firstly if the specified unique ID is same as admin unique ID then no action is taken as admin is already a member of the group,
  It also checks if the specified client is active or not if not then the admin is informed that the client is inactive and can't be added to the group,
  It also checks if the specified client is already a group member, if yes then the admin is informed that the client is already a member and can't add it again to the group,

Otherwise, a message is sent to the specified client that it is requested to join the group with a given group ID and the Client unique ID is added to the requested_mem array of the group.

- handleMakeGroup : It is used to create a Group with a specified client as the admin of the group. It first creates a group and finds the free index in the group array using getFreeGroupIndex function, if no index is free then maximum possible groups are created and admin is informed that group can't be created otherwise, we set all the admins and the members of the group as NULL by calling the function setNull and then we assign values to the group and assign the calling client as its first member and its first admin and then we call allotGroupMem function to add the specified clients to the group.
- setNull : It is used to set all the admins and all the members of a group to NULL.
- handleMakeGroupReq : It is used to create a Group with a specified client as the admin and to request all the other specified clients to join the group. It first creates a group and finds the free index in the group array using getFreeGroupIndex function, if no index is free then maximum possible groups are created and admin is informed that group can't be created otherwise, we set all the admins and the members of the group as NULL by calling the function setNull and then we assign values to the group and assign the calling client as its first member and its first admin and then we call allotGroupMemReq function to request the specified clients to join the group.
- getGroupIndex : It returns the index of the groups array which points to the specified group, if no such group exists then it returns -1.
- handleJoinGroup : It is used to handle the case when a client wants to join a particular group. We first retrieve the group ID from the buffer, check if the group exists, if not then we inform the client that no such group exists and it can't join the group. Otherwise we check if the group has space to include more clients, if not then we inform the calling client that the group has reached maximum limit and it can't join. Else we check if the client is among the requested members of the group or not, if not then we inform the client that it is not requested to join the group hence, it can't join the group.

Otherwise we add the client to the group, increase the member count of the group and inform the client that it has been added to the group.

- handleDeclineGroup : It is used to decline a request to join the group. We first retrieve the group ID from the buffer, check if the group exists, if not then we inform the client that no such group exists and it can't decline the group. Else we check if the client is among the requested members of the group or not, if not then we inform the client that it is not requested to join the group hence, it can't decline the request. Otherwise we decline the group join request of the client and remove it from the requested member list of the group.
- handleSendGroup : It is used to send a message to a particular group. Firstly we retrieve the group ID and the message from the buffer. Then we get the group index of the group using the function getGroupIndex if no such group exists then we inform the calling client that no such group exists, otherwise we check if the client is a member of the group or not using the function isGroupMem, if not then we inform the client that it can't send the message to the group as it is not a member of the group, otherwise we check if the client is admin or not and also if the group is a broadcast group if the client is an admin and if the group is broadcast then we send the message to all the members of the group and if it is not the admin and the group is broadcast then we inform the client that it is not the admin and only admin can send the messages in the group and if the group is not a broadcast group then we simply send the message to all the members of the group.
- isAdmin : It checks whether the specified client is an admin of the specified group or not.
- handleMakeAdmin : It is used to appoint a client as an admin of the group. Firstly we retrieve the group ID and specified client unique ID from the buffer and then we check if the calling client is admin of the group or not using the function isAdmin if not then we inform the client that it can't make anyone admin as it is not the admin of the group. Then we check if the specified client is a member of the group or not. If not then we inform the calling client that the specified client is not the member of the group so can't appoint it as admin. Then we

check if the specified client is already an admin of the group or not if yes, then we inform the calling client that the specified client is already an admin so can't appoint it as admin. Otherwise, we appoint the client as an admin of the group and increase the admin count of the group and also place the client in the admin array of the group.

- handleAddToGroup : It is used to add clients to a group. Firstly, we retrieve the group ID from the buffer than we retrieve the client ID's one by one and do the following,
  Check if the calling function is an admin of the group or not, if not then we inform it that it is not the admin of the group so it can't add anyone to the group.
  Otherwise, we check if the specified client unique ID is already a member of the group or not if yes, then we inform the calling client that the specified client is already a member of the group so can't add it again.
  If not, then we check the number of members in the group if it has reached its maximum limit then we inform the calling client that the group member limit reached can't add any new members.
  Otherwise, we add the client to the group and inform it that it has been added to the group.
- handleRemoveFromGroup : It is used to remove a member from the group. Firstly, we retrieve the group ID from the buffer than we retrieve the client ID's one by one and do the following,
  Check if the calling function is an admin of the group or not, if not then we inform it that it is not the admin of the group so it can't remove anyone from the group.
  Otherwise, we check if the specified client unique ID is already a member of the group or not if not, then we inform the calling client that the specified client is not a member of the group so can't remove it from the group.
  Else, we check if the specified client is an admin of the group or not using the function isAdmin if yes, then we remove it from the admin list of the group and decrease the admin count of the group if the admin count of the group becomes zero then we delete the group this can happen when the calling client is the only admin of the group and

it tries to remove itself from the group. Else we just remove the client from the group and decrease the member count of the group.

- handleMakeGroupBroadcast : It is used to make a group as a broadcast group in which only admins can send the messages.
  Firstly we retrieve the group ID from the buffer than we do the following,
  Check if the group exists or not using the function getGroupIndex, if it does not exist then we inform the calling client that no such group exists.
  Otherwise, we check if the calling client is a member of the group or not using the function isGroupMem, if not then we inform the calling client that it is not the member of the group so it can't make it broadcast.
  Else, we check if the calling client is an admin of the group or not using the function isAdmin, if not then we inform the client that it is not the admin of the group so it can't make it as broadcast.
  Otherwise, we make the group as a broadcast group and inform all the clients that the group is made as a broadcast group using the function handleSendGroup.
- handleActiveGroups : It returns all the group ID of which the calling client is a member of. It first traverses through the groups array and checks if the calling client is one of the members of the groups using the function isGroupMem is yes then we add it to the list and at last we give the calling client the complete list of all the groups of which it is part of.
- performAction : It is used to perform the action as specified by the calling client, it matches the command using strncmp function and calls the respective functions if no command matches then it calls handleInvalidCommand function.
- handleMakeAdminReq : It is used to make requests to make a client as admin of the group.
  Firstly we check if the requesting client is already an admin of the group if yes then we message it that it is already an admin.

Else, we check if the requesting client is a member of the group or not, if not then we message it that it can't request to be an admin as you are not the member of the group.
Else, we find an index in the requested_admin array of the group which is not used already and assign it to the requesting client and also message all the admins of the group that this client wants to be the admin of the group.

- handleApproveAdminReq - This is used to approve any client request which has been requested to be an admin of the group.
  Firstly, we check if the approving client is an admin or not. If not then it is informed that it is not an admin so can’t approve any admin request.
  Then, we check if the specified client is a member of the group or not, if not then the calling client is informed that it can't make the client as admin as it is not the member of the group.
  Then, we check if the specified client is already an admin of the group, if yes then the calling client is informed that the client is already an admin so can’t approve its request.
  Then, we check if at all the specified client has even requested to be an admin of the group, if not, then we inform the calling client that the client has not requested to be admin of the group.
  Otherwise, we appoint the specified client as an admin and message it that it has been appointed as one of the admins of the group.
- handleDeclineAdminReq : It is used to decline any client request which has been requested to be an admin of the group.
  Firstly, we check if the approving client is an admin or not. If not then it is informed that it is not an admin so can’t decline any admin request.
  Then, we check if the specified client is a member of the group or not, if not then the calling client is informed that it can't decline the client as admin as it is not the member of the group.
  Then, we check if the specified client is already an admin of the group, if yes then the calling client is informed that the client is already an admin so can’t decline its request.

Then, we check if at all the specified client has even requested to be an admin of the group, if not, then we inform the calling client that the client has not requested to be admin of the group.
Otherwise, we decline the request of the specified client to be an admin and increase the decline_request_count corresponding to the client and if equals to the admin_count of the group then it means that all the admin of the group have declined the request so, the requesting client can’t be appointed as an admin and it is informed so.