HYBRID INSTANT MESSAGING

Client Side:

Client side is implemented using select; following are some of the descriptors which are checked by select.

1: TCP socket:

Connected with server.

2(a): Listening TCP socket:

This socket is used for creating P2P session.

2(b): Connected TCP socket:

This socket is created through listening TCP socket for direct chat. When direct session established successfully then listening socket will be replaced by this socket meaning that single direct chat is possible, when direct session terminate listening socket again gets into the read set of select.

3: stdin:

This descriptor is used for taking input from user.

4: UDP bindedsocket:

This socket will be used for interaction with other users via server.

Following are some of the functions used at client side.

1: int check_input (char []);

This function the check the input arguments using number of spaces, and return the number of spaces in input.

2: int send_data (int conn_socket, struct sockaddr_in * server, char buffer [])

Used to send data to the server where

- conn_socket: connected socket.
- server: protocol adress of the server/host.
- buffer: data to send.

Return Value:

- 1 on success.
- -1 on error.

3: int help (void)

Simply open the help_file file and show it to the user.

4: int process_input (int conn_socket, char input [], struct sockaddr_in * server)

This function process the input.

First it check the input arguments for particular command using check_input() function, and then send it to the server using send_data () function.

5: int establish_connection (struct sockaddr_in * server, int length)

Establish the connection with server and return connected socket.

6: int create_session (char address [], int flag, int listen_socket)

This function is used to create the direct session.

If flag==1 then it means this user wants to create direct session and adress field isvalid. (This adress is of peer taken from server).

If flag==0 means that request arrives on listening socket and adress field is not valid only accept will be called and connected socket will be returned.

Return: the connected socket.

7: int receive_data (int socket);

This function is used to receive data from other host (server/direct chat/UDP socket). On the basis of received string this function returns the specified value.

Return Values:

- +ve file descriptor for direct chat
- -2 on successfully login.
- -3 when channel created successfully
- -4 if channel moved
- -5 channel quit success
- -6 if user is admin
- -7 if logout success

Flags

Following are some of the flags which are used at the client side to set the status

- *Login_flag* : user login or not, on the basis of this flag user will be restricted or allowed to issue any command
- *Dc_flag*: User is connected to any peer or not.
- *Room_admin_flag*: User is admin of room or not.
- *Server_admin_flag*: User is admin of server or not.
- *Connected_flag*: User is connected to server or not.

How Client works:

When client process starts it first read its configuration file for listening port (Direct chat), udp port, server listening port and server IP. At this step client end can only issue register command since it is not logged in.

After successful login client end send two commands, without user intervention which contain UDP port and Listening port, to the server. After that user will be able to send commands to the server using above mentioned functions.

All the commands are ANDed with specific flags to restrict the user to issue unnecessary commands to the server e.g al the admin rights are ANDed with server_admin_flag, similarly channel commands are ANDed with Room_flag, and all the commands except register command are ANDed with Login_flag.

SERVER SIDE

Server Side is implemented using fork () where each client is handled by separate process/child.

Following are some of the files which are maintained by server.

- Login/logged_in.txt
  Contain all Logged In usernames.

- Reg_file.txt
  All the registered user.

- Info_file.txt
  Contain listening port and IP of users for direct chat.

- Udp_connected.txt:
  Contain the UDP port and IP of all users.

- Admin.txt
  Contain the entries' of all admin users.

- Inbox folder :
  Used for inbox for each user. Two files are maintained against each user. i.e. username.inbox and username.unread

  - *Username.inbox*: contain the bbs messages of each user
  - *Username.unread*: contain total messages and unread messages.

- Rooms folder:
  Contain all rooms' files.

- Log folder:
  Contain the log of each user and the main log of server.

Following are some of the functions which are used through out the program

1: int send_dat(int conn_socket,struct sockaddr_in client ,char buffer[])

Used to send data to the client, normally data is the response of any command.

2: int checkit (char name [],char path[])

Check the presence of username in specified path (file).

Return:

- 1 if user exist
- -1 if user not exist

3: int register_user (char command [])

Used to register the user, this function first checks the availability of username and then register the user.

4: int check_limit (char path [])

Used to check the number of rooms in “rooms” folder, if limit exceed than the mentioned limit then it return 0.

5:create_channel (char command [])

Used to create a room, first it checks the limit if limit exceed then return -2.

If room exists then move the user to that room and return 3

If room does not exist then create the room, make that user owner of that room and return 2.

6:direct_chat (char command [], char information []);

Used to facilitate direct session, where command contain user to whom direct session is to be created, this function search the info_file.txt and return the information by placing it in the information buffer. In case of any error it returns -1;

7:int send_list(int conn_socket,struct sockaddr client, char command[])

This function sends the list of logged in users to client by simply opening the login/loged_in.txt file. In case of any error it returns -1;

8: int single_user (char message [],char receiver[])

Send the message to receiver on udp socket, with the help of udp_connected.txt file.

9: int show_rooms (char path [], int conn_socket);

This function opens the path (directory) and sends the list of files on conn_socket. In case of any Error it returns -1;

10: int login_user (char command [],char ip [])

This function check following files for satisfaction using checkit() function.

a. reg_file.txt : user is registered or not.
b. login/loged_in.txt : user already login or not
c. banuser.txt: banuser or not
d. banip.txt: IP is banned or not

If user exist in files b c, d and not found file a then he/she will not be allowed to login, and this function return with corresponding error.

Else the username will be added to login/logged_in.txt file and function return with success.

11: int save_info (char command [],char path[])

This function is used to write the username or any information in file given by path buffer. In case of any error this function returns -1;

12: int remove_it(char path [],char user[])

Remove a user from given path (file), this function opens two file, desired file(path) for reading and temporary file(in trash folder) for writing, whenever match found in desired file this function doesn't copy that data to temporary file and continue on copying all the remaining entries. When copying of file finishes this function replace the desired file with temporary file. In case of deleting entries' of any room this function will also delete room file if that user is the only one in that room.

In case of any error it returns -1;

13: Int send_message (char input [])

This function is used to send message to all users of the room, input string contain the user who send the message, room name and message to send. This function opens the udp_connected.txt file and room file. By simply searching the users of room in udp_connected.txt file send the message to users on udp sockets.

In case of any error it returns -1.

14: int bbs_delete(char command[])

This function deletes the specific message from inbox of user. Deletion take place in the same way as done in remove_it () function.

15: int bbs_read (char command [])

This function opens the inbox of specific user and search for desired message for the user. Command buffer contain the message number to be read and username.

If message not found it send the error message to the user other wise send the message to the user.

16: Int bbs_list (char receiver[])

Simply open the inbox file of receiver and send the message to the user on UDP socket.

17: Int check_bbs (char receiver [])

This function is used at the login time to check the unread messages of the user. In case if there are unread messages it returns the number of unread message.

18: int send_bbs (char sender [], char receiver [], char message)

This function first check the login file for availability of user, if user exist then send that message to the receiver and store it in the inbox after marking it as read. If user is not available store that message to the inbox and mark it as unread. In case of any error it returns -1;

19: Int bbs_all (char sender [], char message [])

This function open the reg_file.txt file and send the message to all user using send_bbs () function. In case of any error it returns -1;

20: Int bbs (char command [])

Used for BBS if receiver is “all” then call bbs_all function else it call send_bbs function. In case of any error it returns -1;

21:Int process command(char command[])

This function compares the command received by user and calls the corresponding function. In case of any error write the error message to the log file of specific user.

HOW Server Works

When server starts it reads its configuration file. And open main server log file which is used to dump any error messages. When user try to login server check the following files.

- Reg_file.txt
- Banuser.txt
- Banip.txt

If user pass the above mentioned files he/she is allowed to login and server place the user name in login/logged_in.txt file and the successfully login message to the User. After that server check the admin.txt file for to check whether it is admin user or not, if he/she is admin server send the admin user message to the client. After that server check the BBS messages by using function check_bbs() function , if there are unread messages in the inbox then server notify the client of unread messages. On the basis of each notification client side make its corresponding flags on. If the client side goes down without login the server will remove all of its entries from corresponding files.