# server

Main server(main thread of the program)

1. inits
2. Open,bind,listen…
3. Inifinite loop for accepting new clients
   1. Accepting all clients
   2. Finding Ind- index of socket&thread of a client in game

If ind >1 -> closing socket and send "NEW\_USER\_DECLINED"

* 1. We have arrays:   
     ThreadInputs[Ind] [socket\_player1 , socket\_player2]  
     ThreadHandles[Ind] [thread\_player1, thread \_player2]   
     (socket passed as param to thread function)

players\_username\_arr[Ind] [username\_player1 ,username \_player2]

1. ServiceThread
   1. Endless loop until- to contact with clients
      1. recieving a message from client- AcceptedStr
      2. get msg type
      3. responds the client according to the msg type
         1. NEW\_USER\_REQUEST – extracts name from msg
            1. If name taken creating NEW\_USER\_DECLINED msg
            2. if username free: filling players\_username\_arr, creating NEW\_USER\_ACCEPTED msg

if 2 players in - notify to start game

* + - 1. PLAY\_REQUEST
         1. Legal move: player's turn and suitable col.

Sending to client PLAY\_ACCEPTED

Play the turn-update server boards and goto Send\_Board\_View

* + - * 1. Illegal move:

Sending to client PLAY\_DECLINED

* + - 1. SEND\_MESSAGE
         1. create RECEIVE\_MESSAGE and send only if there is another player connected.
      2. other- don’t understand
      3. checks if notify to start game
         1. sending GAME\_STARTED to the 1st,2nd player (with buffer to tell UI that we're in a game mode)
         2. creating and sending BOARD\_VIEW message to 1st,2nd player  
            (straight to the recv thread without buffer)
         3. checking if game ended- if so sending GAME\_ENDED
         4. creating and sending TURN\_SWITCHmessage to 1st,2nd player  
            (straight to the recv thread without buffer)

# client

UI thread

1. Connect
2. Open mutex
3. Open threads
   1. Open **send thread**
      1. Polling the linked list mutex-
         1. if mutex open: CRITICAL SECTION
            1. checks if there's a play/chat message in the list (that UI thread entered)
            2. removes the first message from the LL and saves it in buffer\_send\_message -
            3. Release mutex
      2. If - buffer\_send\_message was filled- sends it to server sends SEND\_MESSAGE to the server
      3. Waits until gets NEW\_USER\_REQUEST message- bufer\_new\_user\_request != \0
         1. Send through buffer\_new\_user\_request\_message   
            sends NEW\_USER\_REQUEST to the server
   2. Open **recv thread**
      1. Checks in a loop for server messages & prints them
      2. If TRNS\_FAILED or TRNS\_DISCONNECTED – closing program resources.
      3. Else getting message type-
         1. If BOARD\_VIEW
            1. Print board view
         2. If TURN\_SWITCH
            1. Prints turn switch message
            2. Update who's turn is it.
         3. If "GAME\_ENDED"
            1. Print who won
         4. If NEW\_USER\_DECLINED or NEW\_USER\_ACCEPTED
            1. Updates buffer so UI can access the message
         5. If RECEIVE\_MESSAGE
            1. Prints it in the right format
         6. PLAY\_ACCEPTED
            1. Prints "well played"
         7. PLAY\_DECLINED- if you played not in your turn or played illegal move (in your turn) or game not started
            1. if played illegal move in your turn- giving the turn back to you.(didnt really play the last turn)
         8. GAME\_STARTED
            1. prints game on
   3. Waits for input- username
      1. CreateNewUserRequestMessage updates buffer\_new\_user\_request\_message with the msg
   4. Infinite loop:   
      first part- if the user can send commands- after getting a valid user name  
      second part- user can't send commands- before getting a valid user name- waiting till user is accepted or refused. (client checks if he got msg from the server. if so- respond according to msg type. )
      1. first part - user can send msg to server- RecognizeGameMessageType
         1. if play
            1. send to server PLAY\_REQUEST through the LL, protected by mutex
         2. if message
            1. send to server SEND\_MESSAGE\_REQUEST through the LL, protected by mutex
         3. if exit
            1. disconnent client and server has to close connection to other cleint
         4. else- Illegal command
      2. second part- getting messages from server through Recv thread
         1. NEW\_USER\_ACCEPTED
         2. NEW\_USER\_DECLINED
   5. close recv&send threads.
4. Open recv thread