# Assignment 2 - Socket Programming

**Name -**
Jay Khatri (214101023), Jyotirmoy Malakar (214101024), Keshav Parihar (214101025)

**How to Run:**

- Download zip file and unzip it
- Open terminal in the folder where all the files are
- Write **make** command in terminal
- Now first run server using the command "./**server <port_number>**"
- Then run client using the command "**./client <server_addr> <port_number>**"
- You can send message now
- To close the connection write "**exit**"

## Socket Programming:

Socket programming is a technology that lets two nodes on a network communicate with one another.The server socket listens for a specific port at an IP address, whereas the client socket connects to that socket.

While the client communicates with the server, the server node creates a listening socket. TCP or UDP may be used for communication. TCP is a connection-oriented protocol that is reliable, whereas UDP is not.

**Procedure in Client-Server Communication:**

- **Socket:** Create a new socket.
- **Bind:** Attach a local address to the newly created socket.
- **Listen:**Ready to accept new connections..
- **Accept:** Block caller until a connection request arrives
- **Connect:** Actively attempt to establish a connection
- **Send:** Send data over the connection
- **Receive:** Receive data over the connection
- **Close:** Disconnect the connection

## Server SIde:

int skt = socket( domain,type, protocol )
skt = It is like a descriptor ,an integer.
Domain = it is of integer type such as AF_INET (IPv4 protocol)

Type = SOCK_STREAM is used for TCP SOCK_DGRAM used for UDP
Protocol = Protocol value for Internet Protocol(IP), which is 0.

**Setsockopt:** This supports the manipulation of socket parameters for the file descriptor socket. It allows address and port reuse. Prevents errors like: if the address has been used before.

**Blind**: The bind function links the socket to the address and port number supplied in addr after it has been created (custom data structure). We bind the server to localhost in the sample code, thus we use INADDR ANY to specify the IP address.

**Listen**: TThe listen() method indicates that a connection-mode socket (for example, those of type SOCK STREAM) provided by the socket arguments is accepting connections and sets the backlog argument to the number of outstanding requests in the socket listen queue. Incoming connection requests are accepted and queued for acceptance by the process, and the socket s is put into 'passive' mode.

**Accept:** The **accept**() system call is used with connection-based socket types(**SOCK_STREAM**, **SOCK_SEQPACKET**). It takes the very first connection request for the listening socket sockfd from the queue of pending connections, establishes a new connected socket, and returns a new file descriptor for that socket.

## Stages For Client:

**Socket connection**:Same as on the server side.
**Connect:** TThe connect() system method establishes a socket connection.
Connect() attempts to connect to another socket if the parameter s (a socket) is of type SOCK STREAM.
The other socket is specified by the name argument. The connect() method establishes a link to the provided foreign association.

Send/Receive :- The send() and recv() calls specifies the following:

1.The sockets on which to communicate.

2.The address in the storage of the buffer that contains, or will contain, the data (addr_of_data, addr_of_buffer).

3.The size of this buffer (len_of_data, len_of_buffer).

4.A flag that tells how the data is to be sent.

## Base 64 encoding:

Steps for base 64 encoding:

1.Find the length of the string

2.If it is not multiple of 6 pad with special character i.e., = to make it multiple of 6.

3.Convert the string to ASCII format.

4.Convert the ASCII code  to binary format taking 8-bit each.

5.Divide the binary data taking 6 bits of chunks.

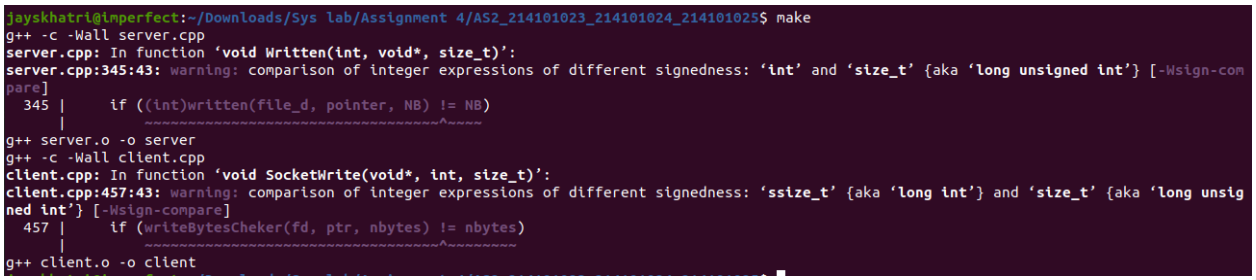
6.Now convert the 6 bit binary chunks of data to decimal format.

7.Convert the decimal data to string using the base64 table.

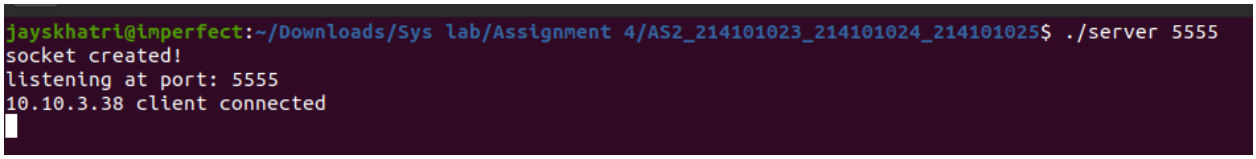
8.The output string of this will be the final encoded data.

Output:

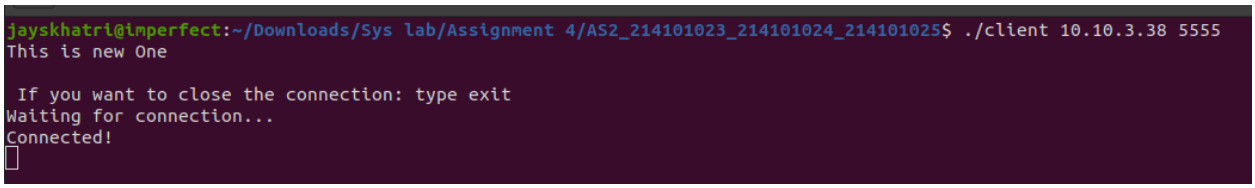
1) Compiling both files




2) Running server




3) Running client




4) sending message

**Server side -**

```
jayskhatri@imperfect:~/Downloads/Sys lab/Assignment 4/AS2_214101023_214101024_214101025$ ./server 5555
socket created!
listening at port: 5555
10.10.3.38 client connected
Msg received from client 10.10.3.38
Type id:1 || message: SGkgaSBhbSBzb2NrZXQ=

Encoded message: SGkgaSBhbSBzb2NrZXQ=

Decoded message: Hi i am socket
```

**Client side -**

```
jayskhatri@imperfect:~/Downloads/Sys lab/Assignment 4/AS2_214101023_214101024_214101025$ ./client 10.10.3.38 5555
This is new One

 If you want to close the connection: type exit
Waiting for connection...
Connected!
Hi i am socket
Msg received from server:
Type_id: 2 || message: W[TR
```

5) Sending exit

**Client side**

```
jayskhatri@imperfect:~/Downloads/Sys lab/Assignment 4/AS2_214101023_214101024_214101025$ ./client 10.10.3.38 5555
This is new One

 If you want to close the connection: type exit
Waiting for connection...
Connected!
Hi i am socket
Msg received from server:
Type_id: 2 || message: W[TR


exit
Msg received from server:
Type_id: 2 || message: W[TR


closing socket....
exiting client.....
jayskhatri@imperfect:~/Downloads/Sys lab/Assignment 4/AS2_214101023_214101024_214101025$
```

**Server side:**

```
jayskhatri@imperfect:~/Downloads/Sys lab/Assignment 4/AS2_214101023_214101024_214101025$ ./server 5555
socket created!
listening at port: 5555
10.10.3.38 client connected
Msg received from client 10.10.3.38
Type id:1 || message: SGkgaSBhbSBzb2NrZXQ=

Encoded message: SGkgaSBhbSBzb2NrZXQ=

Decoded message: Hi i am socket

Msg received from client 10.10.3.38
Type id:3 || message: ZXhpdA==

Encoded message: ZXhpdA==

Decoded message: exit

-client wants to close.
-closing the client socket
```