UCS 1511 - Network Lab

## Exercise 02A - Simple Client Server using TCP Excersie 02B - Echo Server Using TCP

**Name: Kshitij Sharma Reg No: 185001080 Date: 25-08-2020**

1. **Simple Client Server using TCP**

**Aim:**

To develop a socket program to establish a client server communication. The client sends data to server. The server replies to the client.

## Algorithm

### Server

* 1. Create a socket descriptor with **socket()** system call with AF INET (IPV4 domain), SOCK STREAM, default protocol and store as sockfd.
  2. If sockfd is a negative number, socket creation failed, end program.
  3. Create sockaddr in object to assign IP address and Port number for socket. Set family to AF INET, IP address to INADDR ANY to accept connections from any client and port number required.
  4. Bind newly created socket to addresss given in sockaddr in.
  5. If bind is non zero, bind failed, print error message and terminate.
  6. Listen on the socked defined for as many clients as required. If **listen()** returns non zero value, print error message and terminate.
  7. Accept connections from socket using **accept()** system call and store client socket descriptor in a separate variable.
  8. Read message into buffer using **read()** system call.
  9. Read the message to send to client into the buffer from the user.
  10. Write the message from buffer onto client using **write()** system call.
  11. Close connections on socket using **close()** and terminate program.

### Client

* 1. Create a socket descriptor with **socket()** system call with AF INET (IPV4 domain), SOCK STREAM, default protocol and store as sockfd.
  2. If sockfd is a negative number, socket creation failed, end program.
  3. Create sockaddr in object to assign IP address and Port number for socket. Set family to AF INET, IP address to localhost(127.0.0.1) to connect to server and port number required.
  4. Connect the client to server at address given in socket descriptor using **connect()** system call.
  5. If connect() returns -1, connection failed; Print error message and terminate the program.
  6. Read message from user into buffer variable and write into server socket using **write()** system call.
  7. Read the response from server into buffer variable using **read()** system call and display received message from the user.
  8. Close the connections on socket using **close()** and terminate program.

## Program

### Server Side

### #include<stdio.h>

### #include<sys/types.h>

### #include<sys/socket.h>

### #include<netinet/in.h>

### #include<string.h>

### int main(int argc, char \*\*argv)

### {

### int len;

### int sockfd, newfd, n;

### struct sockaddr\_in servaddr, cliaddr;

### char buff[1024];

### char str[1000];

### sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

### if(sockfd<0)

### perror("Cannot create socket");

### bzero(&servaddr, sizeof(servaddr));

### servaddr.sin\_family = AF\_INET;

### servaddr.sin\_addr.s\_addr = INADDR\_ANY;

### servaddr.sin\_port = htons(7229);

### if(bind(sockfd,(struct sockaddr\*) &servaddr, sizeof(servaddr))<0)

### perror("Bind error");

### listen(sockfd, 2);

### len = sizeof(cliaddr);

### newfd = accept(sockfd, (struct sockaddr\*) &cliaddr, &len);

### //Receiving msg

### n = read(newfd, buff, sizeof(buff));

### printf("Message from client: %s\n", buff);

### bzero(buff, sizeof(buff));

### printf("Message sent: ");

### scanf("%[^\n]", buff);

### n = write(newfd, buff, sizeof(buff));

### close(sockfd);

### close(newfd);

### return 0;

### }

### Client Side

### #include<stdio.h>

### #include<sys/types.h>

### #include<sys/socket.h>

### #include<netinet/in.h>

### #include<string.h>

### int main(int argc, char \*\*argv)

### {

### int len;

### int sockfd, n;

### struct sockaddr\_in servaddr, cliaddr;

### char buff[1024];

### char str[1000];

### sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

### if(sockfd<0)

### perror("Cannot create socket");

### bzero(&servaddr, sizeof(servaddr));

### servaddr.sin\_family = AF\_INET;

### servaddr.sin\_addr.s\_addr = inet\_addr(argv[1]);

### servaddr.sin\_port = htons(7229);

### connect(sockfd,(struct sockaddr\*) &servaddr, sizeof(servaddr));

### //sending msg

### printf("Client: ");

### scanf("%[^\n]", buff);

### n = write(sockfd, buff, sizeof(buff));

### n = read(sockfd, buff, sizeof(buff));

### printf("Message from server: %s\n", buff);

### close(sockfd);

### return 0;

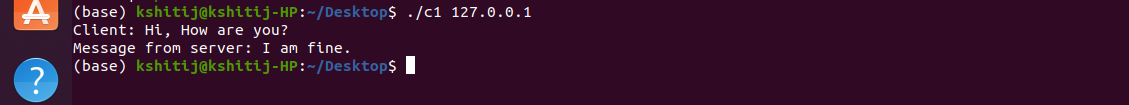
}

## Output

Figure 1: Server Program Output



Figure 2: Client Program Output



# Echo server using TCP

## Aim:

Develop a socket program to establish a client server communication. The client sends data to server. The server in turn sends the same message back to the client. Transmit multiple lines of text. In client side display the echoed message. In server side display the message which is echoed to client.

## Algorithm

### Server

* 1. Create a socket descriptor with **socket()** system call with AF INET (IPV4 domain), SOCK STREAM, default protocol and store as sockfd.
  2. If sockfd is a negative number, socket creation failed, end program.
  3. Create sockaddr in object to assign IP address and Port number for socket. Set family to AF INET, IP address to INADDR ANY to accept connections from any client and port number required.
  4. Bind newly created socket to addresss given in sockaddr in.
  5. If bind is non zero, bind failed, print error message and terminate.
  6. Listen on the socked defined for as many clients as required. If **listen()** returns non zero value, print error message and terminate.
  7. Accept connections from socket using **accept()** system call and store client socket descriptor in a separate variable.
  8. Read message into buffer using **read()** system call.
  9. Write the back the message from buffer onto client using **write()** system call.
  10. Close connections on socket using **close()** and terminate program.

### Client

* 1. Create a socket descriptor with **socket()** system call with AF INET (IPV4 domain), SOCK STREAM, default protocol and store as sockfd.
  2. If sockfd is a negative number, socket creation failed, end program.
  3. Create sockaddr in object to assign IP address and Port number for socket. Set family to AF INET, IP address to localhost(127.0.0.1) to connect to server and port number required.
  4. Connect the client to server at address given in socket descriptor using **connect()** system call.
  5. If connect() returns -1, connection failed; Print error message and terminate the program.
  6. Read message from user into buffer variable and write into server socket using **write()** system call.
  7. Read the echoed response from server into buffer variable using **read()** system call and display received message from the user.
  8. Close the connections on socket using **close()** and terminate program.

## Program

### Server Side

### #include<stdio.h>

### #include<sys/types.h>

### #include<sys/socket.h>

### #include<netinet/in.h>

### #include<string.h>

### int main(int argc, char \*\*argv)

### {

### int len;

### int sockfd, newfd, n;

### struct sockaddr\_in servaddr, cliaddr;

### char buff[1024];

### char str[1000];

### sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

### if(sockfd<0)

### perror("Cannot create socket");

### bzero(&servaddr, sizeof(servaddr));

### servaddr.sin\_family = AF\_INET;

### servaddr.sin\_addr.s\_addr = INADDR\_ANY;

### servaddr.sin\_port = htons(7226);

### if(bind(sockfd,(struct sockaddr\*) &servaddr, sizeof(servaddr))<0)

### perror("Bind error");

### listen(sockfd, 2);

### len = sizeof(cliaddr);

### newfd = accept(sockfd, (struct sockaddr\*) &cliaddr, &len);

### //Receiving msg

### read(newfd, buff, sizeof(buff));

### printf("Message from client: %s\n", buff);

### printf("Echoed message to client: %s", buff);

### write(newfd, buff, sizeof(buff));

### close(sockfd);

### close(newfd);

### return 0;

}

### Client Side

#include<stdio.h>

#include<sys/types.h>

#include<sys/socket.h>

#include<netinet/in.h>

#include<string.h>

int main(int argc, char \*\*argv)

{

int len;

int sockfd, n;

struct sockaddr\_in servaddr, cliaddr;

char buff[1024];

char str[1000];

sockfd = socket(AF\_INET, SOCK\_STREAM, 0);

if(sockfd<0)

perror("Cannot create socket");

bzero(&servaddr, sizeof(servaddr));

servaddr.sin\_family = AF\_INET;

servaddr.sin\_addr.s\_addr = inet\_addr(argv[1]);

servaddr.sin\_port = htons(7226);

connect(sockfd,(struct sockaddr\*) &servaddr, sizeof(servaddr));

//sending msg

printf("Client: ");

n=0;

char x;

bzero(buff, sizeof(buff));

while((x=getchar()) != '\*')

{

buff[n++] = x;

}

write(sockfd, buff, sizeof(buff));

read(sockfd, buff, sizeof(buff));

printf("\nEchoed message: %s\n", buff);

close(sockfd);

return 0;

}

## Output:

Figure 3: Server Program Output



Figure 4: Client Program Output



Learning Outcomes:

* From this assignment I learnt how to communicate between a client and server.
* How to send a single line and multiple line messages.
* Most importantly I learnt the basics of socket programming.