# SSN College of Engineering
# UCS1511 - Network Lab
# Exercise 2A - Simple Client Server using TCP
# Exercise 2B - Echo Server Using TCP

Prasanna Kumaran D
185001110

September 6, 2020

---

## 1 Simple Client server using TCP

### 1.1 Aim

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

### 1.2 Algorithm

#### 1.2.1 Server

1. Create a socket descriptor with socket() system call and use AF_INET as domain and SOCK_STREAM for domain and communication type, store the socket descriptor in sockfd.
2. 2. If sockfd is a negative number,
   (a) print socket creation failed, terminate program.
3. Assign family, address and port to the server socketadd_in object. Set the family as AF_INET to access IPv4 protocols, and INADDR_ANY for address to accept connections from any client.
4. bind the socket to the server sockadd_in object
5. If bind is non zero,
   (a) Print bind creation failed 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 client using accept() system call
8. Read the buffer contents using read()
9. Write the message from the client into the buffer using write() system call.
10. Close socket connections using close() and terminate program.

### 1.2.2 Client

1. Create a socket descriptor using socket()
   with AF_INET(IPv4 domain), SOCK_STREAM(connection type)
2. if socket ¡ 0
   (a) Print socket creation failed and terminate program
3. Create a sockaddr_in object for the client and set up family, address and port number.
4. Call connect() to establish connection between client and server
5. if connect() returns -1
   (a) Print connection failed and terminate program
6. Read (read()) the user message and write (write()) into the buffer
7. Display the message received from the server
8. Close the socket connection using close() and terminate program

## 1.3 Program

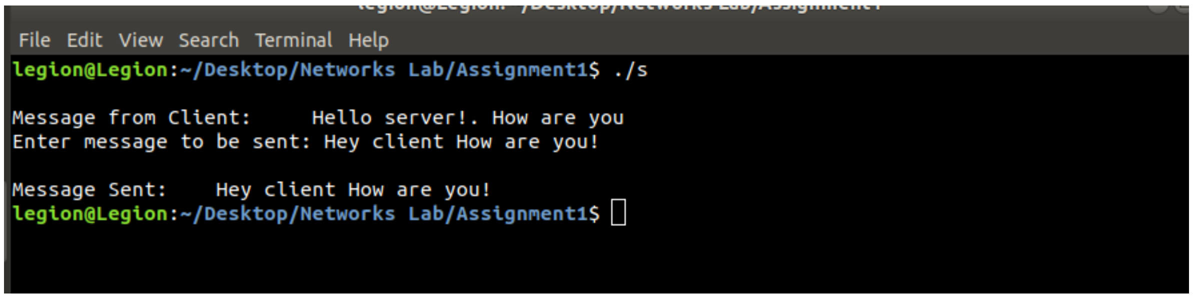
### 1.3.1 Server

```
#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(8083);

        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 the message
        n=read(newfd,buff,sizeof(buff));
        printf("\nMessage from Client:\t%s",buff);
        printf("\nEnter message to be sent:");
        scanf("%[^\n]s",str);
        strcpy(buff,str);
        n=write(newfd,buff,sizeof(buff));
        printf("\nMessage Sent:\t%s\n",buff);
        close(sockfd);
        close(newfd);
        return 0;
}
```

#### 1.3.2 Client

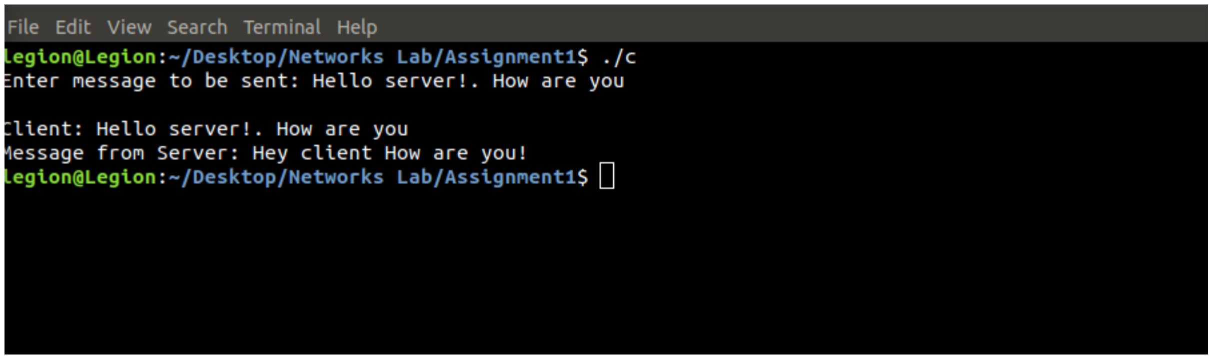
```
#include<stdio.h>
#include<sys/types.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<string.h>

int main(int argc,char **argv)
{
        int len;
        int sockfd,n;
        struct sockaddr_in servaddr,cliaddr;

        char str[1000]; char buff[1024];

        sockfd=socket(AF_INET,SOCK_STREAM,0);
        if(sockfd<0)
                perror("cannot create socket");

        bzero(&servaddr,sizeof(servaddr));

        cliaddr.sin_family=AF_INET;
        cliaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
        cliaddr.sin_port=htonl(8083);

        connect(sockfd,(struct sockaddr*)&cliaddr,sizeof(cliaddr));

        //Sending Message
        printf("Enter message to be sent:");
        scanf("%[^\n]s",buff);
        printf("\nClient:%s",buff);
        n=write(sockfd,buff,sizeof(buff));
        n=read(sockfd,buff,sizeof(buff));
        printf("\nMessage from Server:%s\n",buff);
        close(sockfd);
        return 0;
}
```

## 1.4 Output



```
legion@Legion:~/Desktop/Networks Lab/Assignment1$ ./s

Message from Client:     Hello server!. How are you
Enter message to be sent: Hey client How are you!

Message Sent:    Hey client How are you!
legion@Legion:~/Desktop/Networks Lab/Assignment1$ 
```

Figure 1: Server side



```
legion@Legion:~/Desktop/Networks Lab/Assignment1$ ./c
Enter message to be sent: Hello server!. How are you

Client: Hello server!. How are you
Message from Server: Hey client How are you!
legion@Legion:~/Desktop/Networks Lab/Assignment1$ 
```

Figure 2: Client side

### 1.5 Learning Outcomes

- Learnt how to establish a simple client server connection using TCP
- Learnt about the basic syntax and system calls used in socket programming
- Learnt how to handle errors in socket programming

## 2 Echo server using TCP

### 2.1 Aim

To 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.

### 2.2 Algorithm

#### 2.2.1 Server

1. Create a socket descriptor with socket() system call and use AF_INET as domain and SOCK_STREAM for domain and communication type, store the socket descriptor in sockfd.

2. 2. If sockfd is a negative number,
    (a) print socket creation failed, terminate program.
3. Assign family, address and port to the server socketadd_in object. Set the family as AF_INET to access IPv4 protocols, and INADDR_ANY for address to accept connections from any client.
4. bind the socket to the server sockadd_in object
5. If bind is non zero,
    (a) Print bind creation failed 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 client using accept() system call
8. Read the buffer contents using read()
9. Write the message from the client into the buffer using write() system call.
10. Close socket connections using close() and terminate program.

#### 2.2.2 Client

1. Create a socket descriptor using socket()
   with AF_INET (IPv4 domain) SOCK_STREAM(connection type)
2. if socket ¡ 0
    (a) Print socket creation failed and terminate program
3. Create a sockaddr_in object for the client and set up family, address and port number.
4. Call connect() to establish connection between client and server
5. if connect() returns -1
    (a) Print connection failed and terminate program
6. Read (read()) the user message until "!" is encountered and write the contents (write()) into the buffer
7. Read the echoed response from the server into the buffer (read()) and display the message.
8. Close the socket connection using close() and terminate program

## 2.3 Program

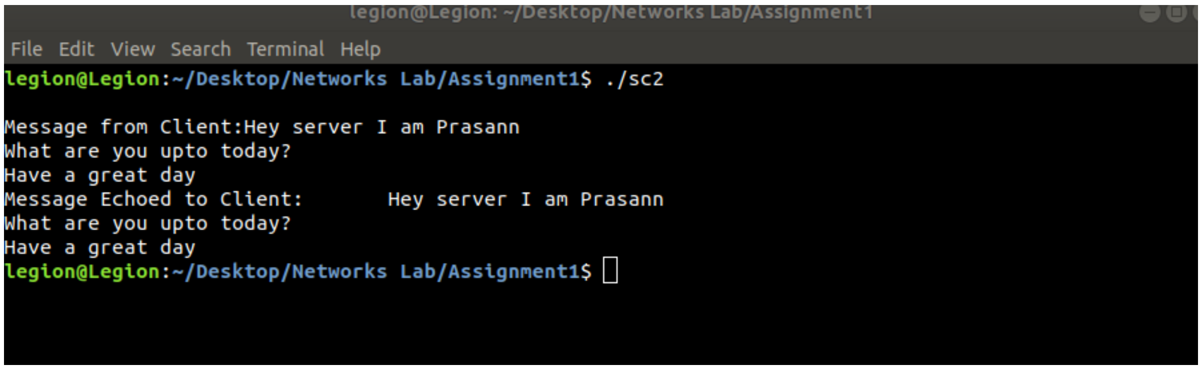
### 2.3.1 Server

```
#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(8081);

        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 the message
        n=read(newfd,buff,sizeof(buff));
        printf("\nMessage from Client:%s",buff);
        n=write(newfd,buff,sizeof(buff));
        printf("\nMessage Echoed to Client:\t%s\n",buff);
        close(sockfd);
        close(newfd);
        return 0;
}
```

#### 2.3.2 Client

```
#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 str[1000]; char buff[1024];

        sockfd=socket(AF_INET,SOCK_STREAM,0);
        if(sockfd<0)
                perror("cannot create socket");

        bzero(&servaddr,sizeof(servaddr));

        cliaddr.sin_family=AF_INET;
        cliaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
        cliaddr.sin_port=htons(8081);

        connect(sockfd,(struct sockaddr*)&cliaddr,sizeof(cliaddr));

        //Sending Message
        printf("Enter message to be sent:(End with !)");
        scanf("%[^!]s",buff);
        printf("\nClient:%s",buff);
        n=write(sockfd,buff,sizeof(buff));
        n=read(sockfd,buff,sizeof(buff));
        printf("\nMessage Echoed from Server:%s\n",buff);
        close(sockfd);
        return 0;
}
```

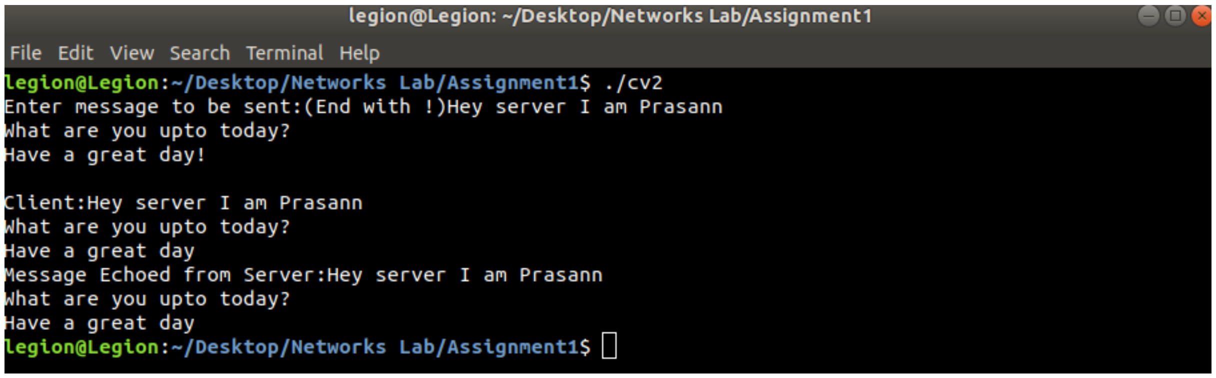
### 2.4 Output



legion@Legion: ~/Desktop/Networks Lab/Assignment1
File Edit View Search Terminal Help

```
legion@Legion:~/Desktop/Networks Lab/Assignment1$ ./sc2

Message from Client:Hey server I am Prasann
What are you upto today?
Have a great day
Message Echoed to Client:       Hey server I am Prasann
What are you upto today?
Have a great day
legion@Legion:~/Desktop/Networks Lab/Assignment1$ 
```

Figure 3: Server side



legion@Legion: ~/Desktop/Networks Lab/Assignment1
File Edit View Search Terminal Help

```
legion@Legion:~/Desktop/Networks Lab/Assignment1$ ./cv2
Enter message to be sent:(End with !)Hey server I am Prasann
What are you upto today?
Have a great day!

Client:Hey server I am Prasann
What are you upto today?
Have a great day
Message Echoed from Server:Hey server I am Prasann
What are you upto today?
Have a great day
legion@Legion:~/Desktop/Networks Lab/Assignment1$ 
```

Figure 4: Client Side

### 2.5 Learning Outcomes

- Learnt how to establish a simple client server connection using TCP
- Learnt about the basic syntax and system calls used in socket programming
- Learnt how to handle errors in socket programming

---