Experiment - 2: A simple TCP Client-Server program

// serverTCP.c

```
#include<stdio.h>
#include<arpa/inet.h>
#include <unistd.h>
#define port 5000
void main()
{
   struct sockaddr_in serveraddr,newaddr;
   int sersocket,newsocket,s,size;
   char buffer[100];

   sersocket=socket(PF_INET,SOCK_STREAM,0);

   if(sersocket>0)
         printf("\nserver socket is created");

   serveraddr.sin_family= PF_INET;
   serveraddr.sin_port= htons(port);
   serveraddr.sin_addr.s_addr=htonl(INADDR_ANY);
   s=bind(sersocket,(struct sockaddr *)&serveraddr,sizeof(serveraddr));

 if(s==0)
         printf("\nbind success");

   listen(sersocket,1);
   size=sizeof(newaddr);
   printf("\nserver ready");
   newsocket=accept(sersocket,(struct sockaddr *)&newaddr,&size);

if(newsocket>0)
         printf("\naccepted");

recv(newsocket,buffer,1024,0);
printf("\ndata received is %s\n",buffer);
close(sersocket);
}
```

//Execution Steps:

//>> gcc serverChat_TCP.c -o ser

//>> ./ser

// clientTCP.c

```
#include<stdio.h>
#include<arpa/inet.h>
#include <unistd.h>
#define port 5000
```

```
void main()
{
   struct sockaddr_in serveraddr;
   int clisocket;
   char buffer[100];
   clisocket=socket(PF_INET,SOCK_STREAM,0);
   I
f(clisocket>0)
        printf("\nclient socket created");

serveraddr.sin_family= PF_INET;
serveraddr.sin_port= htons(port);
serveraddr.sin_addr.s_addr=inet_addr("127.0.0.1");
connect(clisocket,(struct sockaddr*)&serveraddr,sizeof(serveraddr));

printf("\nEnter string:");
scanf("%s",buffer);

send(clisocket,buffer,sizeof(buffer),0);
close(clisocket);
}
```

//Execution Steps:

//>> gcc clientCP.c -o cli

//>> ./cli

OUTPUT

```
alwin@debian:~$ cd Downloads
alwin@debian:~/Downloads$ gcc serverTCP.c -o ser
alwin@debian:~/Downloads$ ./ser

server socket is created
bind success
server ready
accepted
data received is Hello
alwin@debian:~/Downloads$
```

```
alwin@debian:~$ cd Downloads
alwin@debian:~/Downloads$ gcc clientTCP.c -o cli
alwin@debian:~/Downloads$ ./cli

client socket created
Enter string:Hello World
alwin@debian:~/Downloads$
```

Experiment - 3: A simple UDP Client-Server program

// serverUDP.c

```
#include<stdio.h>
#include<arpa/inet.h>
#include <unistd.h>
#define port 4000
void main()
{
   struct sockaddr_in serveraddr,newaddr;
   int sersocket,s,size;
   char buffer[100];
   sersocket=socket(AF_INET,SOCK_DGRAM,0);
   if(sersocket>0)
         printf("\nServer socket created");
   serveraddr.sin_family=AF_INET;
   serveraddr.sin_port=htons(port);
   serveraddr.sin_addr.s_addr=htonl(INADDR_ANY);
   s=bind(sersocket,(struct sockaddr*)&serveraddr,sizeof(serveraddr));
   if(s==0)
         printf("\nBind success");
   size=sizeof(newaddr);
   recvfrom(sersocket,buffer,1024,0,(struct sockaddr*)&newaddr,&size);
   printf("\nString recieved:%s\n",buffer);
   close(sersocket);
}
```

//Execution Steps:

```
//>> gcc serverChat_TCP.c -o ser
//>> ./ser
```

// clientUDP.c

```
#include<stdio.h>
#include<arpa/inet.h>
#include <unistd.h>
#define port 4000
void main()
{
   struct sockaddr_in serveraddr;
   int clisocket;
   char buffer[100];
   clisocket=socket(AF_INET,SOCK_DGRAM,0);
   if(clisocket>0)
         printf("\nClient socket created");
   serveraddr.sin_family=AF_INET;
   serveraddr.sin_port=htons(port);
   serveraddr.sin_addr.s_addr=inet_addr("127.0.0.1");
   printf("\nEnter string:");
   scanf("%s",buffer);
   sendto(clisocket,buffer,sizeof(buffer),0,(struct sockaddr*)&serveraddr,sizeof(serveraddr));
```

```
    close(clisocket);
}
```

//Execution Steps:

//>> gcc clientUDP.c -o cli

//>> ./cli

OUTPUT

```
alwin@debian:~$ cd Downloads
alwin@debian:~/Downloads$ gcc serverUDP.c -o ser
alwin@debian:~/Downloads$ ./ser

Server socket created
Bind success
String recieved:Hello
alwin@debian:~/Downloads$
```

```
alwin@debian:~$ cd Downloads
alwin@debian:~/Downloads$ gcc clientUDP.c -o cli
alwin@debian:~/Downloads$ ./cli

Client socket created
Enter string:Hello
alwin@debian:~/Downloads$
```

Experiment - 4: Client-Server Communication using TCP

// serverChat_TCP.c

```
#include<stdio.h>
#include<arpa/inet.h>
#include<string.h>
#include<unistd.h>

#define PORT 5000

void chat(int newsocket) {
        char buffer[100];
        while(1) {
        recv(newsocket, buffer, sizeof(buffer), 0);
        printf("\nFrom client: %s", buffer);
        if (strcmp(buffer, "bye\n") == 0)
        break;
        printf("\tTo client: ");
        fgets(buffer, sizeof(buffer), stdin);
        send(newsocket, buffer, sizeof(buffer), 0);
        }
}

int main() {
        struct sockaddr_in serveraddr, newaddr;
        int sersocket, newsocket, s, size;

        sersocket = socket(PF_INET, SOCK_STREAM, 0);
        if (sersocket > 0)
        printf("\nServer socket is created");

        serveraddr.sin_family = PF_INET;
        serveraddr.sin_port = htons(PORT);
        serveraddr.sin_addr.s_addr = htonl(INADDR_ANY);

        s = bind(sersocket, (struct sockaddr *)&serveraddr, sizeof(serveraddr));
        if (s == 0)
        printf("\nBind success");

        listen(sersocket, 1);
        size = sizeof(newaddr);
        printf("\nServer ready");

        newsocket = accept(sersocket, (struct sockaddr *)&newaddr, &size);
        if (newsocket > 0)
        printf("\nAccepted\n");

        chat(newsocket);
        close(newsocket);

        return 0;
}
```

//Execution Steps:

//>> gcc serverChat_TCP.c -o ser

//>> ./ser

// clientChat_TCP.c

```
#include <stdio.h>
#include <arpa/inet.h>
#include <string.h>
#include <unistd.h>

#define port 5000

void chat(int clisocket) {
        char buffer[100];
        while (1) {
        printf("\nTo server:");
        fgets(buffer, sizeof(buffer), stdin);
        send(clisocket, buffer, sizeof(buffer), 0);
        recv(clisocket, buffer, sizeof(buffer), 0);
        printf("\tFrom server:%s", buffer);
        if (strcmp(buffer, "bye\n") == 0)
        break;
        }
}

int main() {
        int clisocket;
        struct sockaddr_in serveraddr;
        clisocket = socket(PF_INET, SOCK_STREAM, 0);
        if (clisocket > 0)
        printf("client socket created\n");

        serveraddr.sin_family = PF_INET;
        serveraddr.sin_port = htons(port);
        serveraddr.sin_addr.s_addr = inet_addr("127.0.0.1");

        connect(clisocket, (struct sockaddr*)&serveraddr, sizeof(serveraddr));
        chat(clisocket);
        close(clisocket);

        return 0;
}
```

//Execution Steps:

//>> gcc clientChat_TCP.c -o cli

//>> ./cli

OUTPUT

```
alwin@debian:~/Downloads$ gcc serverChat_TCP.c -o ser
alwin@debian:~/Downloads$ ./ser

server socket is created
bind success
server ready
accepted

From client:Hai
        To client:I am fine

From client:Bye
        To client:
```

```
alwin@debian:~$ cd Downloads
alwin@debian:~/Downloads$ gcc clientChat_TCP.c -o cli
alwin@debian:~/Downloads$ ./cli
client socket created

To server:Hai
        From server:I am fine

To server:Bye
```

Experiment - 5: Implement a simple UDP client-server chatting programming

//serverChat_UDP.c

```
#include <stdio.h>
#include <arpa/inet.h>
#include <string.h>
#include <unistd.h>

#define port 4000

void main()
{
	struct sockaddr_in serveraddr, newaddr;
	int sersocket, size, s;
	char buffer[100];

	sersocket = socket(AF_INET, SOCK_DGRAM, 0);
	if (sersocket > 0)
	printf("\nServer socket created");

	serveraddr.sin_family = AF_INET;
	serveraddr.sin_port = htons(port);
	serveraddr.sin_addr.s_addr = htonl(INADDR_ANY);

	s = bind(sersocket, (struct sockaddr *)&serveraddr, sizeof(serveraddr));
	if (s == 0)
	printf("\nBind success");

	size = sizeof(newaddr);

	while (1)
	{
	recvfrom(sersocket, buffer, sizeof(buffer), 0, (struct sockaddr *)&newaddr, &size);
	printf("\nFrom client: %s", buffer);

	if (strcmp(buffer, "bye\n") == 0)
	break;
	else
	{
	printf("\tTo client: ");
	fgets(buffer, sizeof(buffer), stdin);
	sendto(sersocket, buffer, sizeof(buffer), 0, (struct sockaddr *)&newaddr,
sizeof(newaddr));
	}
	}

	close(sersocket);
}
```

//Execution Steps:

//>> gcc serverChat_UDP.c -o ser
//>> ./ser

// clientChat_TCP.c

```
#include <stdio.h>
#include <arpa/inet.h>
#include <string.h>
#include <unistd.h>

#define port 4000

void main()
{
	struct sockaddr_in serveraddr;
	int clisocket, size;
	char buffer[100];

	clisocket = socket(AF_INET, SOCK_DGRAM, 0);
	if (clisocket > 0)
	printf("\nClient socket created");

	serveraddr.sin_family = AF_INET;
	serveraddr.sin_port = htons(port);
	serveraddr.sin_addr.s_addr = inet_addr("127.0.0.1");

	while (1)
	{
	printf("\nTo server:");
	fgets(buffer, sizeof(buffer), stdin);
	sendto(clisocket, buffer, sizeof(buffer), 0, (struct sockaddr *)&serveraddr, sizeof(serveraddr));

	if (strcmp(buffer, "bye\n") == 0)
	break;

	size = sizeof(serveraddr);
	recvfrom(clisocket, buffer, sizeof(buffer), 0, (struct sockaddr *)&serveraddr, &size);
	printf("\tFrom server:%s", buffer);
	}

	close(clisocket);
}
```

//Execution Steps:
//>> gcc clientChat_UDP.c -o cli
//>> ./cli

OUTPUT

```
alwin@debian:~$ cd Downloads
alwin@debian:~/Downloads$ gcc serverChat_UDP.c -o ser
alwin@debian:~/Downloads$ ./ser

Server socket created
Bind success
From client:Hai
        To client:I am fine

From client:Okay Thank you
        To client:bye

From client:bye
alwin@debian:~/Downloads$
```

```
alwin@debian:~$ cd Downloads
alwin@debian:~/Downloads$ gcc clientChat_UDP.c -o cli
alwin@debian:~/Downloads$ ./cli

Client socket created
To server:Hai
        From server:I am fine

To server:Okay Thank you
        From server:bye

To server:bye
alwin@debian:~/Downloads$
```