



---

# COMPUTER NETWORKS

---

EXP 7



## FULL DUPLEX

SEPTEMBER 6, 2021

ROEHIT RANGANATHAN  
RA1911033010017 | L2

### Aim:

To implement a full duplex application, where the Client establishes a connection with the Server. The Client and Server can send as well as receive messages at the same time. Both the Client and Server exchange messages.

### Procedure:

#### Server:

- Include the necessary header files.
- Create a socket using socket function with family AF\_INET, type as SOCK\_STREAM.
- Initialize server address to 0 using the bzero function.
- Assign the sin\_family to AF\_INET, sin\_addr to INADDR\_ANY, sin\_port to dynamically assigned port number.
- Bind the local host address to socket using the bind function.
- Listen on the socket for connection request from the client.
- Accept connection request from the Client using accept function.
- Fork the process to receive message from the client and print it on the console.
- Read message from the console and send it to the client.

#### Client:

- Include the necessary header files.
- Create a socket using socket function with family AF\_INET, type as SOCK\_STREAM.
- Initialize server address to 0 using the bzero function.
- Assign the sin\_family to AF\_INET.
- Get the server IP address and the Port number from the console.
- Using gethostbyname function assign it to a hostent structure, and assign it to sin\_addr of the server address structure.
- Request a connection from the server using the connect function.
- Fork the process to receive message from the server and print it on the console.
- Read message from the console and send it to the server.

### Code:

SERVER.C

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

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

```

#include<stdio.h>
#include<unistd.h>
#include<netdb.h>
#include<arpa/inet.h>
#include<netinet/in.h>
#include<string.h>

int main(int argc,char *argv[])
{
    int ad,sd;
    struct sockaddr_in servaddr,cliaddr;
    socklen_t servlen,clilen;
    char buff[1000],buff1[1000];
    pid_t cpid;
    bzero(&servaddr,sizeof(servaddr));
    /*Socket address structure*/
    servaddr.sin_family=AF_INET;
    servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
    servaddr.sin_port=htons(5500);
    /*TCP socket is created, an Internet socket address structure is filled with
    wildcard address & server's well known port*/
    sd=socket(AF_INET,SOCK_STREAM,0);
    /*Bind function assigns a local protocol address to the socket*/
    bind(sd,(struct sockaddr*)&servaddr,sizeof(servaddr));
    /*Listen function specifies the maximum number of connections that kernel should queue
    for this socket*/
    listen(sd,5);
    printf("%s\n","Server is running.....");
    /*The server to return the next completed connection from the front of the
    completed connection Queue calls it*/
    ad=accept(sd,(struct sockaddr*)&cliaddr,&clilen);
    /*Fork system call is used to create a new process*/

```

```

cpid=fork();

if(cpid==0)
{
while(1)
{
bzero(&buff,sizeof(buff));
/*Receiving the request from client*/
recv(ad,buff,sizeof(buff),0);
printf("Received message from the client:%s\n",buff);
}
}
else
{
while(1)
{

bzero(&buff1,sizeof(buff1));

printf("%s\n","Enter the input data:");
/*Read the message from client*/
fgets(buff1,10000,stdin);
/*Sends the message to client*/
send(ad,buff1,strlen(buff1)+1,0);
printf("%s\n","Data sent...");

}
}
return 0;
}

```

CLIENT.C

```

#include<sys/socket.h>
#include<sys/types.h>
#include<stdio.h>
#include<arpa/inet.h>
#include<unistd.h>
#include<netdb.h>
#include<string.h>
#include<netinet/in.h>

int main(int argc,char *argv[])
{
    int sd,cd;
    struct sockaddr_in servaddr,cliaddr;
    socklen_t servlen,clilen;
    char buff[1000],buff1[1000];
    pid_t cpid;
    bzero(&servaddr,sizeof(servaddr));
    servaddr.sin_family=AF_INET;
    servaddr.sin_addr.s_addr=inet_addr(argv[1]);
    servaddr.sin_port=htons(5500);
    /*Creating a socket, assigning IP address and port number for that socket*/
    sd=socket(AF_INET,SOCK_STREAM,0);
    /*Connect establishes connection with the server using server IP address*/
    cd=connect(sd,(struct sockaddr*)&servaddr,sizeof(servaddr));
    /*Fork is used to create a new process*/
    cpid=fork();
    if(cpid==0)
    {
        while(1)
        {
            bzero(&buff,sizeof(buff));
            printf("%s\n","Enter the input data:");

```

```

/*This function is used to read from server*/
fgets(buff,10000,stdin);

/*Send the message to server*/
send(sd,buff,strlen(buff)+1,0);

printf("%s\n","Data sent...");
}

}

else
{
while(1)
{
bzero(&buff1,sizeof(buff1));

/*Receive the message from server*/
recv(sd,buff1,sizeof(buff1),0);

printf("Received message from the server:%s\n",buff1);
}
}

return 0;
}

```

## OUTPUT:

```

server.c
1 #include<sys/types.h>
2 #include<sys/socket.h>
3 #include<stdio.h>
4 #include<unistd.h>
5 #include<netdb.h>
6 #include<arpa/inet.h>
7 #include<netinet/in.h>
8 #include<string.h>
9
10 int main(int argc,char *argv[])
11 {
12     int ad,sd;
13     struct sockaddr_in servaddr,cliaddr;
14     socklen_t servlen,clilen;
15     char buff[1000],buff1[1000];
16     pid_t cpid;
17     bzero(&servaddr,sizeof(servaddr));
18     /*Socket address structure*/
19     servaddr.sin_family=AF_INET;

```

```

client.c
1 #include<sys/socket.h>
2 #include<sys/types.h>
3 #include<stdio.h>
4 #include<arpa/inet.h>
5 #include<unistd.h>
6 #include<netdb.h>
7 #include<string.h>
8 #include<netinet/in.h>
9 int main(int argc,char *argv[])
10 {
11     int sd,cd;
12     struct sockaddr_in servaddr,cliaddr;
13     socklen_t servlen,clilen;
14     char buff[1000],buff1[1000];
15     pid_t cpid;
16     bzero(&servaddr,sizeof(servaddr));
17     servaddr.sin_family=AF_INET;
18     servaddr.sin_addr.s_addr=inet_addr(argv[1]);
19     servaddr.sin_port=htons(5500);

```

```

./a.out - "ip-172-31-9-200" x
RA1911033010021:~/environment $ cd RA1911033010017 $ cd full_duplex
RA1911033010021:~/environment/RA1911033010017/full_duplex $ cc server.c
RA1911033010021:~/environment/RA1911033010017/full_duplex $ ./a.out
Server is running.....
Enter the input data:
Received message from the client:hi
hi
Data sent...
Enter the input data:
Received message from the client:i m client
i m server
Data sent...
Enter the input data:

```

```

./a.out - "ip-172-31-9-200" x
RA1911033010021:~/environment $ cd RA1911033010017
RA1911033010021:~/environment/RA1911033010017 $ cd full_duplex
RA1911033010021:~/environment/RA1911033010017/full_duplex $ cc client.c
RA1911033010021:~/environment/RA1911033010017/full_duplex $ ./a.out 127.0.0.1
Enter the input data:
hi
Data sent...
Enter the input data:
Received message from the server:hi
i m client
Data sent...
Enter the input data:
Received message from the server:i m server

```