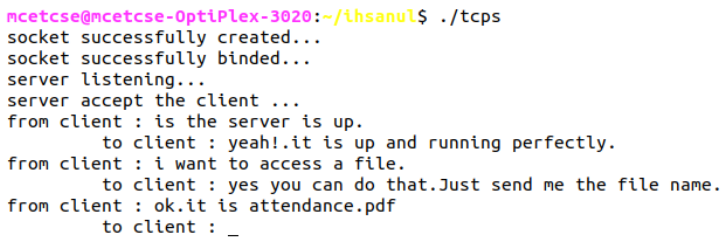
1.Tcps.c

```
#include<stdio.h>
#include<netdb.h>
#include<netinet/in.h>
#include<stdlib.h>
#include<string.h>
#include<sys/socket.h>
#include<sys/types.h>
#include<unistd.h> //read(),write(),close()
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
//define designed for chat between client and server.
void func(int connfd)
{
        char buff[MAX];
        int n;
        //infinite loop for chat
        for(;;)
        {
                bzero(buff,MAX);
                //read the message from client and copy it in buffer
                read(connfd,buff,sizeof(buff));
                //print the buffer which contains the client contents
                printf("from client : %s\t to client : ",buff);
                bzero(buff,MAX);
                n=0;
                //copy server message in the buffer
                while((buff[n++]=getchar())!='\n')
                ;
                //and sent buffer to client
                write(connfd,buff,sizeof(buff));
                //if msg contains "Exit "then server exit and chat ended
                if(strncmp("exit",buff,4)==0)
                {
                        printf("Server Exit....\n");
                        break;
                }
        }
}
//driver function
int main()
{
        int sockfd,connfd,len;
        struct sockaddr_in servaddr,cli;
        //socket create and verification
        sockfd=socket(AF_INET,SOCK_STREAM,0);
        if(sockfd==-1)
        {
                printf("socket creation failed ....\n");
                exit(0);
        }
        else
        {
                printf("socket successfully created...\n");
        }
        bzero(&servaddr,sizeof(servaddr));
        //assign IP,PORT
        servaddr.sin_family=AF_INET;
        servaddr.sin_addr.s_addr=htonl(INADDR_ANY);
        servaddr.sin_port=htons(PORT);
        //binding newly created socket to given ip and verification
        if((bind(sockfd,(SA*)&servaddr,sizeof(servaddr)))!=0)
        {
                printf("socket bind failed....\n");
                exit(0);
        }
        else
                printf("socket successfully binded...\n");
        //now server is ready to listen and verification
        if((listen(sockfd,5))!=0)
        {
                printf("listen failed ...\n");
                exit(0);
        }
        else
                printf("server listening...\n");
        len=sizeof(cli);
        //accept the data packet from client and verification
        connfd=accept(sockfd,(SA*)&cli,&len);
        if(connfd<0)
        {
                printf("server accept failed ...\n");
                exit(0);
        }
        else
                printf("server accept the client ...\n");
        //function for chatting between client and server
        func(connfd);
        //after chatting close the socket
        close(sockfd);
}
```

output – tcps.png



```
mcetcse@mcetcse-OptiPlex-3020:~/ihsanul$ ./tcps
socket successfully created...
socket successfully binded...
server listening...
server accept the client ...
from client : is the server is up.
         to client : yeah!.it is up and running perfectly.
from client : i want to access a file.
         to client : yes you can do that.Just send me the file name.
from client : ok.it is attendance.pdf
         to client : _
```

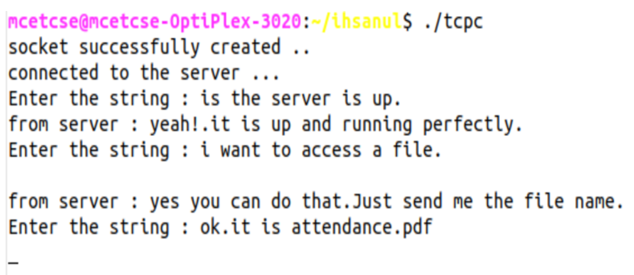
tcpc.c

```
#include<arpa/inet.h>//inet_addr()
#include<stdio.h>
#include<netdb.h>
#include<strings.h>//bzero()
#include<stdlib.h>
#include<string.h>
#include<sys/socket.h>
#include<unistd.h> //read(),write(),close()
#define MAX 80
#define PORT 8080
#define SA struct sockaddr
void func(int sockfd)
{
        char buff[MAX];
        int n;
        for(;;)
        {
                bzero(buff,sizeof(buff));
                printf("Enter the string : ");
                n=0;
                while((buff[n++]=getchar())!='\n')
                ;
                write(sockfd,buff,sizeof(buff));
                bzero(buff,sizeof(buff));
                read(sockfd,buff,sizeof(buff));
                printf("from server : %s",buff);
                if((strncmp(buff,"exit",4))==0)
                {
                        printf("client Exit...\n");
                        break;
                }
        }
}
int main()
{
        int sockfd,connfd;
        struct sockaddr_in servaddr,cli;
        //socket create & verification
        sockfd=socket(AF_INET,SOCK_STREAM,0);
        if(sockfd==-1)
        {
                printf("socket successfully created ...\n");
                exit(0);
        }
        else
                printf("socket successfully created ..\n");
        bzero(&servaddr,sizeof(servaddr));
        //assign IP,PORT
        servaddr.sin_family=AF_INET;
        servaddr.sin_addr.s_addr=inet_addr("127.0.0.1");
        servaddr.sin_port=htons(PORT);
        //connect the client socket to server socket
        if(connect(sockfd,(SA*)&servaddr,sizeof(servaddr))!=0)
```

```
	{
		printf("connection with the server failed ...\n");
		exit(0);
	}
	else
		printf("connected to the server ...\n");
	//function for chat
	func(sockfd);
	//close the socket
	close(sockfd);
}
```

---

output- tcpc.c



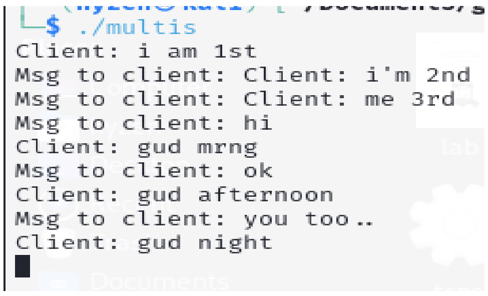
```
mcetcse@mcetcse-OptiPlex-3020:~/ihsanul$ ./tcpc
socket successfully created ..
connected to the server ...
Enter the string : is the server is up.
from server : yeah!.it is up and running perfectly.
Enter the string : i want to access a file.

from server : yes you can do that.Just send me the file name.
Enter the string : ok.it is attendance.pdf

_
```

2.multis.c

```
#include <netinet/in.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/socket.h>
#include <string.h>
#include <unistd.h>
#include <asm-generic/socket.h>
#define PORT 8080

int main(int argc, char const* argv[])
{
   int server_fd, new_socket, valread;
   struct sockaddr_in address;
   int opt = 1;
   int addrlen = sizeof(address);
   char buffer[1024] = {0};
   char hello[1024];
   pid_t childpid;

   // Create socket
   if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == 0)
   {
      perror("socket failed.");
      exit(EXIT_FAILURE);
   }

   // Set socket options
   if (setsockopt(server_fd, SOL_SOCKET, SO_REUSEADDR | SO_REUSEPORT,
&opt, sizeof(opt)))
   {
      perror("setsockopt");
      exit(EXIT_FAILURE);
   }

   // Configure server address
   address.sin_family = AF_INET;
   address.sin_addr.s_addr = INADDR_ANY;
   address.sin_port = htons(PORT);

   // Bind socket to address
   if (bind(server_fd, (struct sockaddr*)&address, sizeof(address)) < 0)
   {
      perror("bind failed");
      exit(EXIT_FAILURE);
   }
   // Listen for connections
   if (listen(server_fd, 3) < 0)
   {
      perror("listen");
      exit(EXIT_FAILURE);
   }
   // Accept connections
   for (;;)
   {
      if ((new_socket = accept(server_fd, (struct sockaddr*)&address,
(socklen_t*)&addrlen)) < 0)
      {
         perror("accept");
         exit(EXIT_FAILURE);
      }
      // Fork a child process
      if ((childpid = fork()) == 0)
      {
         close(server_fd); // Child doesn't need the listener
         // Handle client communication
         while ((valread = read(new_socket, buffer, 1024)) > 0)
         {
            printf("Client: %s\n", buffer);
            printf("Msg to client: ");
            scanf("%s", hello);
            send(new_socket, hello, strlen(hello), 0);
            memset(buffer, 0, sizeof(buffer)); // Clear buffer
         }
         if (valread == 0)
         {
            printf("Client disconnected.\n");
         }
         else
         {
            perror("read");
         }
         close(new_socket);
         exit(EXIT_SUCCESS); // Exit child process
      }
      close(new_socket); // Parent closes client socket
   }
   return 0;
}
```

---

output-multis.png



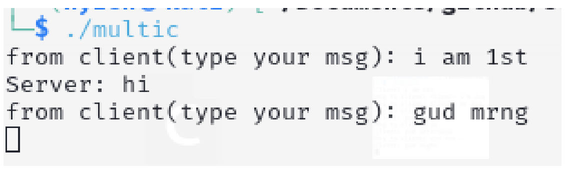
```
└─$ ./multis
Client: i am 1st
Msg to client: Client: i'm 2nd
Msg to client: Client: me 3rd
Msg to client: hi
Client: gud mrng
Msg to client: ok
Client: gud afternoon
Msg to client: you too..
Client: gud night
█
```

multic.c

```
#include <arpa/inet.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <unistd.h>
#define PORT 8080

int main(int argc, char const* argv[]) {
   int sock = 0, valread;
   struct sockaddr_in serv_addr;
   char hello[1024] = {0};
   char buffer[1024] = {0};

   // Create socket
   if ((sock = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
      printf("\nSocket creation error\n");
      return -1;
   }

   // Configure server address
   serv_addr.sin_family = AF_INET;
   serv_addr.sin_port = htons(PORT);

   // Convert IP address from text to binary
   if (inet_pton(AF_INET, "127.0.0.1", &serv_addr.sin_addr) <= 0) {
      printf("\nInvalid address / Address not supported.\n");
      return -1;
   }
```

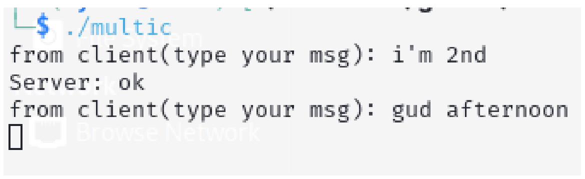
```
    // Connect to server
    if (connect(sock, (struct sockaddr*)&serv_addr, sizeof(serv_addr)) < 0) {
        printf("\nConnection failed\n");
        return -1;
    }

    // Send and receive messages
    while (1) {
        printf("from client(type your msg): ");
        fgets(hello, 1024, stdin);
        hello[strcspn(hello, "\n")] = '\0'; // Remove newline character

        // Exit if user types "exit"
        if (strcmp(hello, "exit") == 0) {
            printf("Exiting...\n");
            break;
        }

        // Send message to server
        if (send(sock, hello, strlen(hello), 0) < 0) {
            perror("send failed");
            break;
        }

        // Receive response from server
        valread = read(sock, buffer, 1024);
        if (valread < 0) {
            perror("read failed");
            break;
        } else if (valread == 0) {
            printf("Server disconnected.\n");
            break;
        }

        // Print server's response
        buffer[valread] = '\0'; // Null-terminate the buffer
        printf("Server: %s\n", buffer);
    }

    close(sock);
    return 0;
}
```

<u>output-multic1.png</u>



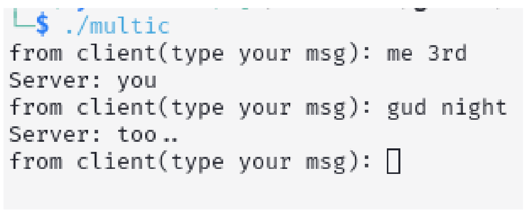
```
└─$ ./multic
from client(type your msg): i am 1st
Server: hi
from client(type your msg): gud mrng
▯
```

<u>multic2.png</u>



```
└─$ ./multic
from client(type your msg): i'm 2nd
Server: ok
from client(type your msg): gud afternoon
▯
```

<u>multic3.png</u>

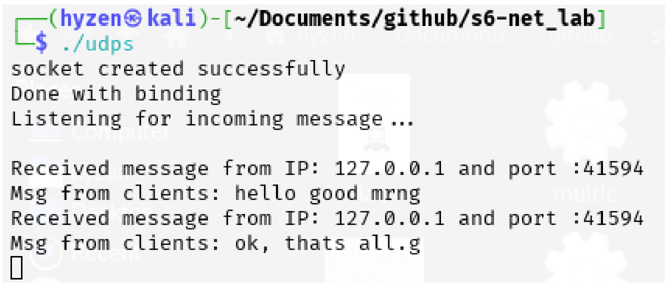


```
└─$ ./multic
from client(type your msg): me 3rd
Server: you
from client(type your msg): gud night
Server: too..
from client(type your msg): ▯
```

<u>3.upds.c</u>

```
#include<stdio.h>
#include<string.h>
#include<unistd.h>
#include<sys/socket.h>
#include<arpa/inet.h>
int main(void)
{
        int socket_desc;
        struct sockaddr_in server_addr,client_addr;
        char server_message[2000],client_message[2000];
        int client_struct_length=sizeof(client_addr);
        //clean buffers
        memset(server_message,'\0',sizeof(server_message));
        memset(client_message,'\0',sizeof(client_message));
        //create udp socket
        socket_desc=socket(AF_INET,SOCK_DGRAM,IPPROTO_UDP);
        if(socket_desc<0)
        {
                printf("Error while creating socket\n");
                return -1;
        }
        printf("socket created successfully\n");
        //set port and ip
        server_addr.sin_family=AF_INET;
        server_addr.sin_port=htons(4000);
        server_addr.sin_addr.s_addr=inet_addr("127.0.0.1");
        //bind to the set port and ip
        if(bind(socket_desc,(struct
sockaddr*)&server_addr,sizeof(server_addr))<0)
        {
                printf("couldn't bind to the port\n");
                return -1;
        }
        printf("Done with binding\n");
        printf("Listening for incoming message...\n\n");

        while(1)
        {
                //Receive client's message

        if(recvfrom(socket_desc,client_message,sizeof(client_message),0,
(struct sockaddr*)&client_addr,&client_struct_length)<0)
                {
                        printf("couldn't receive \n");
                        return -1;
                }
                printf("Received message from IP: %s and port :
%i\n",inet_ntoa(client_addr.sin_addr), ntohs(client_addr.sin_port));
                printf("Msg from clients: %s\n",client_message);
                //respond to client:
                strcpy(server_message,client_message);

        if(sendto(socket_desc,server_message,strlen(server_message),0,
(struct sockaddr*)&client_addr,client_struct_length)<0)
                {
                        printf("can't send\n");
                        return -1;
                }
        }
        //close the socket:
        close(socket_desc);
        return 0;
}
```

<u>output-udps.png</u>



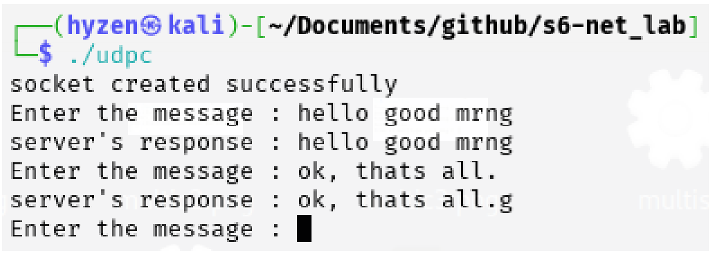
```
┌──(hyzen㉿kali)-[~/Documents/github/s6-net_lab]
└─$ ./udps
socket created successfully
Done with binding
Listening for incoming message...

Received message from IP: 127.0.0.1 and port :41594
Msg from clients: hello good mrng
Received message from IP: 127.0.0.1 and port :41594
Msg from clients: ok, thats all.g
▯
```

udpc.c

```
#include<stdio.h>
#include<unistd.h>
#include<string.h>
#include<sys/socket.h>
#include<arpa/inet.h>
int main(void)
{
		int socket_desc;
		struct sockaddr_in server_addr;
		char server_message[2000],client_message[2000];
		int server_struct_length=sizeof(server_addr);

		//clean buffers:
		memset(server_message,'\0',sizeof(server_message));
		memset(client_message,'\0',sizeof(client_message));

		//create socket:
		socket_desc=socket(AF_INET,SOCK_DGRAM,IPPROTO_UDP);

		if(socket_desc<0)
		{
				printf("Error while creating socket\n");
				return -1;
		}
		printf("socket created successfully\n");

		//set port and ip:
		server_addr.sin_family=AF_INET;
		server_addr.sin_port=htons(4000);
		server_addr.sin_addr.s_addr=inet_addr("127.0.0.1");

		while(1)
		{
				//get input from the user:
				printf("Enter the message : ");
				fgets(client_message, sizeof(client_message), stdin);
				client_message[strcspn(client_message,"\n")]='\0';

				//send the message to server:

		if(sendto(socket_desc,client_message,strlen(client_message),0,
(struct sockaddr*)&server_addr,server_struct_length)<0)
				{
						printf("unable to send message\n");
						return -1;
				}

				//receive the server's  response:

		if(recvfrom(socket_desc,server_message,sizeof(server_message),0,
(struct sockaddr*)&server_addr,&server_struct_length)<0)
				{
						printf("Error while receiving server's msg\n");
						return -1;
				}
				printf("server's response : %s\n",server_message);
		}

		//close the socket:
		close(socket_desc);

		return 0;
}
```

---

output-udpc.png



```
┌──(hyzen㉿kali)-[~/Documents/github/s6-net_lab]
└─$ ./udpc
socket created successfully
Enter the message : hello good mrng
server's response : hello good mrng
Enter the message : ok, thats all.
server's response : ok, thats all.g
Enter the message : 
```