Assignment 2
Of
Network & Distributed System Lab (CS2051)
Masters of Technology in Computer Science And Engineering

submitted to
Dr Sujoy Saha
Assistant Professor
&
Dr Suvrojit Das
Associate Professor
Dept. of CSE

National Institute of Technology, Durgapur

submitted by
Arghya Bandyopadhyay
RollNo. 20CS4103

19 June 2021

Write TCP Chat Program.

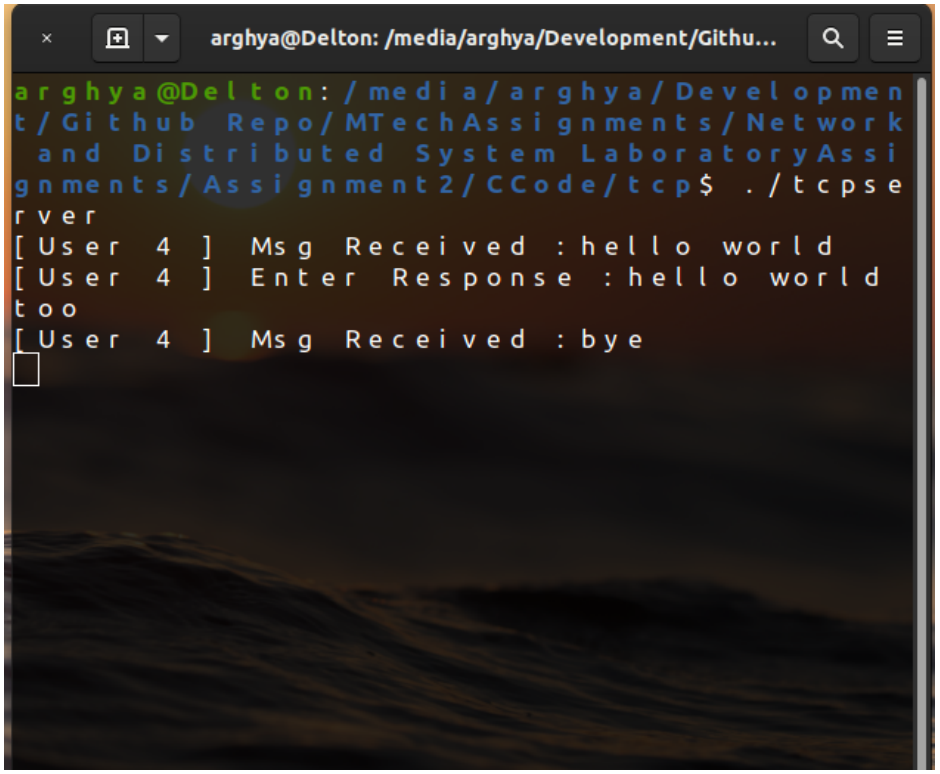
Answer.

```c
//This is the Server side implementation of TCP Chat

#include<string.h>
#include<sys/socket.h>
#include<netinet/in.h>
#include<unistd.h>
#include<stdio.h>
#include <stdbool.h>


int TCP_ChatServer(int client_desc)
{
  const int BUFFER_SIZE = 4096;
  char msg[BUFFER_SIZE];
  int  msg_len = 0;

  while((msg_len = read(client_desc, msg, BUFFER_SIZE)) != 0)
  {
    printf("%s %d %s","[User",client_desc,"] Msg Received :");
    fflush(stdout);

    write(fileno(stdout), msg, msg_len);

    if(msg[0] == 'b' && msg[1] == 'y' && msg[2] == 'e')
      return 0;

    printf("%s %d %s","[User",client_desc,"] Enter Response :");

    fflush(stdout);


    msg_len = read(fileno(stdin), msg, BUFFER_SIZE);

    write(client_desc, msg, msg_len);

    if(msg[0] == 'b' && msg[1] == 'y' && msg[2] == 'e')
      return 0;
  }

  return 0;
}

int main()
{
    //Create Socket
  int  server_desc = socket(AF_INET,SOCK_STREAM,0);

    //Create and Fill Address Structure for this Server
  struct  sockaddr_in server_addr;
```

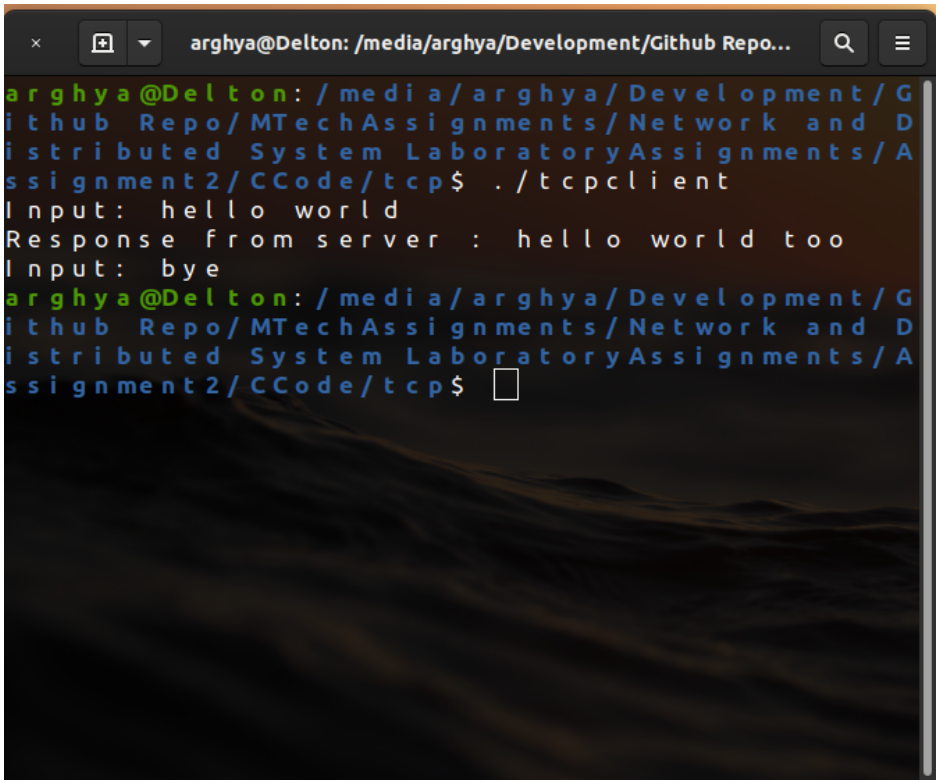
```
  server_addr.sin_family      = AF_INET;    //Address Family (AF_INET, AF_INET6, AF_LOCAL, ...)
    server_addr.sin_addr.s_addr = INADDR_ANY; //Internet Address (INADDR_ANY-> Accept connection at any IP Address)
    server_addr.sin_port        = htons(9000);//Port Number (htons -> h.HOST t.TO n.NETWORK s.SHORT , Ensures proper byte ordering)

  //Bind Socket Descriptor and Address Structure together
  int result = bind(server_desc,  (struct sockaddr*) &server_addr, sizeof(server_addr));

  //Start Listioning (Tell kernel to accept connections directed towards this socket) (Puts socket into passive mode)
  listen(server_desc,4);




  //Server Loop
  bool RunServer = true;
  while(RunServer)
  {
    //Accept a Connection (Puts process in sleep mode if Connection Queue is Empty)
    int client_desc;
    client_desc = accept(server_desc,NULL,NULL);        //Listening Socket

    //Create Child Process to handle connection
    int pid = fork();

    if(pid > 0)                               //Parent Process
    {
      //Close Client Socket
      close(client_desc);
      continue;
    }
    else
    if(pid == 0)                              //Child Process
    {
      //Close Listening Socket
      close(server_desc);

      TCP_ChatServer(client_desc);

      //Close Connection
      close(client_desc);
      break;                                //Work Done! Exit Child Process.
    }
    else
    {
      printf("%s ","fork() Error!!!");
      break;
    }
  }


  return 0;
}
```

```c
//This is the client side implementation of TCP Chat

#include<arpa/inet.h>
#include<sys/socket.h>
#include<sys/wait.h>
#include<netinet/in.h>
#include <stdbool.h>
#include<string.h>
#include<time.h>
#include<stdio.h>
#include<unistd.h>
#include<stdlib.h>
#include<string.h>
#include<signal.h>
#include<errno.h>


int TCP_ChatClient(int server_desc)
{
  const int BUFFER_SIZE = 4096;
  char msg[BUFFER_SIZE];
  int  len = 0;

  while(true)
  {
    printf("%s ","Input:"); fflush(stdout);
    len = read(fileno(stdin), msg, BUFFER_SIZE);

    if(len == 0) //EOF
      return 0;

    len = write(server_desc, msg, len);

    if(msg[0] == 'b' && msg[1] == 'y' && msg[2] == 'e')
      return 0;

    len = read(server_desc, msg, BUFFER_SIZE);

    printf("%s ","Response from server :"); fflush(stdout);
    len = write(fileno(stdout), msg, len);

    if(msg[0] == 'b' && msg[1] == 'y' && msg[2] == 'e')
      return 0;

  }
  return 0;
}

int main()
{
  int sock = socket(AF_INET, SOCK_STREAM, 0);

  struct sockaddr_in  server_addr;
```

```
54    server_addr.sin_family         = AF_INET;
55    server_addr.sin_addr.s_addr    = inet_addr("127.0.0.1");
56    server_addr.sin_port           = htons(9000);
57
58    int result = connect(sock, (struct sockaddr*)&server_addr, sizeof(server_addr));
59
60    TCP_ChatClient(sock);
61
62
63    close(sock);
64    return 0;
65 }
```



(a) TCPServer



(b) TCPClient

Figure 1: Output:TCP