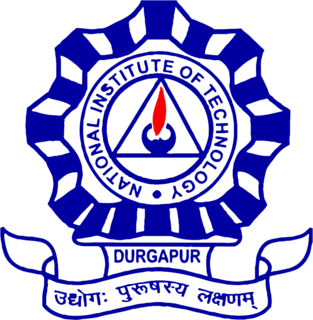
# Assignment **1** Of

**Network & Distributed System Lab (CS2051)**

**Masters of Technology in Computer Science And Engineering**

**submitted to**

**Dr Suvrojit Das Associate Professor Dept. of CSE**

****

**National Institute of Technology, Durgapur**

submitted by Arghya Bandyopadhyay RollNo. 20CS4103

29 June 2021

# Objective: Write a TCP Chat Server-Client Program using Sockets

# **Solution:**

//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;

}

//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;

server\_addr.sin\_family = AF\_INET;

server\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1");

server\_addr.sin\_port = htons(9000);

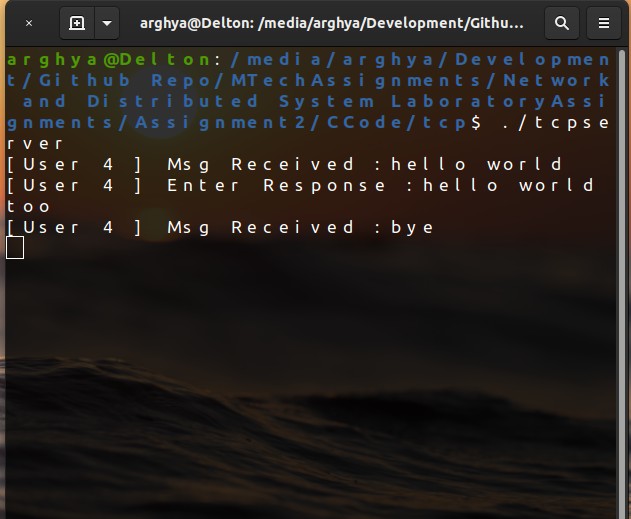
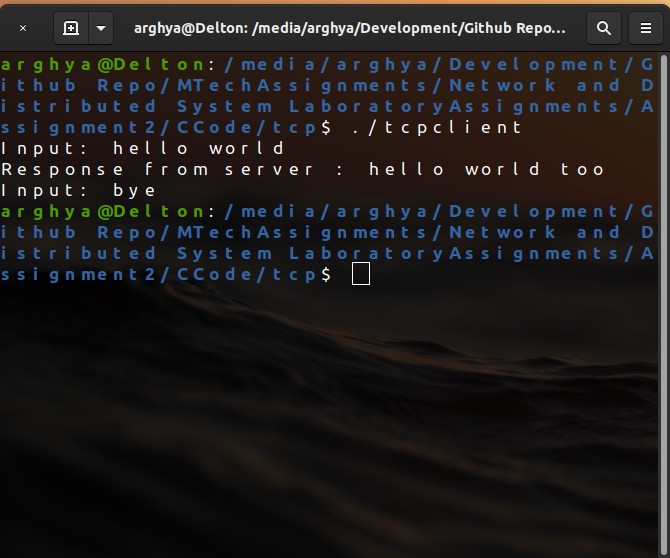
int result = connect(sock, (struct sockaddr\*)&server\_addr, sizeof(server\_addr));

TCP\_ChatClient(sock);

close(sock);

return 0;

}

(a) TCPServer (b) TCPClient

## Figure 1: Output:TCP