/\*

TCP\_Server. This Program will will create the Server side for TCP\_Socket Programming.

It will receive the data from the client and then send the same data back to client.

\*/

#include <stdio.h>

#include <string.h>

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

#include <arpa/inet.h> //inet\_addr

#include <pthread.h>

struct thread\_struct

{

int client\_sock;

};

int connected = 0;

void respond\_client(void \*thread\_struct) {

char client\_message[2000];

char server\_message[2000];

connected++;

struct thread\_struct \*data = (struct thread\_struct\*)thread\_struct;

while(1) {

if (recv(data->client\_sock, client\_message, sizeof(client\_message),0) < 0)

{

printf("Receive Failed. Error!!!!!\n");

return -1;

}

if (!strcmp(client\_message, "DISCONNECT")) {

strcpy(server\_message, "DISCONNECTED");

if (send(data->client\_sock, server\_message, strlen(server\_message),0)<0)

{

printf("Send Failed. Error!!!!!\n");

return -1;

}

connected--;

memset(server\_message,'\0',sizeof(server\_message));

memset(client\_message,'\0',sizeof(client\_message));

pthread\_exit(NULL);

break;

}

printf("Client Message: %s\n",client\_message);

strcpy(server\_message, client\_message);

if (send(data->client\_sock, server\_message, strlen(client\_message),0)<0)

{

printf("Send Failed. Error!!!!!\n");

return -1;

}

}

}

int main(void)

{

int socket\_desc, client\_sock, client\_size;

struct sockaddr\_in server\_addr, client\_addr; //SERVER ADDR will have all the server address

char server\_message[2000], client\_message[2000]; // Sending values from the server and receive from the server we need this

//Cleaning the Buffers

memset(server\_message,'\0',sizeof(server\_message));

memset(client\_message,'\0',sizeof(client\_message)); // Set all bits of the padding field//

//Creating Socket

socket\_desc = socket(AF\_INET, SOCK\_STREAM, 0);

if(socket\_desc < 0)

{

printf("Could Not Create Socket. Error!!!!!\n");

return -1;

}

printf("Socket Created\n");

//Binding IP and Port to socket

server\_addr.sin\_family = AF\_INET; /\* Address family = Internet \*/

server\_addr.sin\_port = htons(2002); // Set port number, using htons function to use proper byte order \*/

server\_addr.sin\_addr.s\_addr = inet\_addr("127.0.0.1"); /\* Set IP address to localhost \*/

// BINDING FUNCTION

if(bind(socket\_desc, (struct sockaddr\*)&server\_addr, sizeof(server\_addr))<0) // Bind the address struct to the socket. /

//bind() passes file descriptor, the address structure,and the length of the address structure

{

printf("Bind Failed. Error!!!!!\n");

return -1;

}

printf("Bind Done\n");

//Put the socket into Listening State

if(listen(socket\_desc, 1) < 0) //This listen() call tells the socket to listen to the incoming connections.

// The listen() function places all incoming connection into a "backlog queue" until accept() call accepts the connection.

{

printf("Listening Failed. Error!!!!!\n");

return -1;

}

printf("Listening for Incoming Connections.....\n");

//Accept the incoming Connections

client\_size = sizeof(client\_addr);

pthread\_t thread[4];

struct thread\_struct thread\_st[4];

while(connected < 4) {

client\_sock = accept(socket\_desc, (struct sockaddr\*)&client\_addr, &client\_size);

thread\_st[connected].client\_sock= client\_sock;

pthread\_create(&thread[connected],NULL,respond\_client,(void\*)&thread\_st[connected]);

if (client\_sock < 0)

{

printf("Accept Failed. Error!!!!!!\n");

return -1;

}

printf("Client Connected with IP: %s and Port No: %i\n",inet\_ntoa(client\_addr.sin\_addr),ntohs(client\_addr.sin\_port));

}

//Closing the Socket

pthread\_join(thread[0],NULL);

pthread\_join(thread[1],NULL);

pthread\_join(thread[2],NULL);

pthread\_join(thread[3],NULL);

close(client\_sock);

close(socket\_desc);

return 0;

}