**Assignment 3**

**Implement echo client-server message passing application. Message sent from client should be displayed on server and then program should terminate.**

**1. Write a server (TCP) C Program that opens a listening socket and waits to serve client.**

**2. Write a client (TCP) C Program that connects with the server program knowing IP**

**address and port number.**

**3. Get the input string from console on client and send it to server, server displays the**

**same string.**

**Code for client:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

int main()

{

char \*ip = "127.0.0.1";

int port = 5000;

int sock;

struct sockaddr\_in addr;

socklen\_t addr\_size;

char buffer[1024];

int n;

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

if (n < 0)

{

perror("Socket error......");

printf("#######################################################\n");

printf("\n");

exit(1);

}

printf("TCP server socket created......\n");

memset(&addr, '\0', sizeof(addr));

addr.sin\_family = AF\_INET;

addr.sin\_port = port;

addr.sin\_addr.s\_addr = inet\_addr(ip);

connect(sock, (struct sockaddr \*)&addr, sizeof(addr));

printf("Server connected......\n");

printf("#######################################################\n");

printf("\n");

while (1)

{

bzero(buffer, 1024);

printf("Enter message to send a server (exit for quit) : \n");

scanf("%[^\n]%\*c", buffer);

printf("Your message sended to server is : \n");

printf("%s", buffer);

printf("\n\n");

send(sock, buffer, strlen(buffer), 0);

if ((strncmp(buffer, "exit", 4)) == 0)

{

close(sock);

printf("Server disconnected......\n");

printf("#######################################################\n");

printf("\n");

exit(0);

}

}

return 0;

}

**Code for server:**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

#include <unistd.h>

#include <arpa/inet.h>

int main()

{

char \*ip = "127.0.0.1";

int port = 5000;

int server\_sock, client\_sock;

struct sockaddr\_in server\_addr, client\_addr;

socklen\_t addr\_size;

char buffer[1024];

int n;

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

if (server\_sock < 0)

{

perror("Socket error......\n");

printf("#######################################################\n");

printf("\n");

exit(1);

}

memset(&server\_addr, '\0', sizeof(server\_addr));

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

server\_addr.sin\_port = port;

server\_addr.sin\_addr.s\_addr = inet\_addr(ip);

n = bind(server\_sock, (struct sockaddr \*)&server\_addr, sizeof(server\_addr));

if (n < 0)

{

perror("Bind error......");

printf("#######################################################\n");

printf("\n");

exit(1);

}

listen(server\_sock, 5);

printf("Serever running......\n");

printf("#######################################################\n");

printf("\n");

addr\_size = sizeof(client\_addr);

client\_sock = accept(server\_sock, (struct sockaddr \*)&client\_addr, &addr\_size);

printf("Client Connected......\n");

printf("#######################################################\n");

printf("\n");

while (1)

{

bzero(buffer, 1024);

recv(client\_sock, buffer, sizeof(buffer), 0);

printf("Message from client is : %s\n", buffer);

printf("\n");

if ((strncmp(buffer, "exit", 4)) == 0)

{

// When client send exit request

close(client\_sock);

printf("Client disconnected......\n");

printf("#######################################################\n");

printf("\n");

// For client to enter server again so not stuck here

client\_sock = accept(server\_sock, (struct sockaddr \*)&client\_addr, &addr\_size);

printf("Client Connected......\n");

printf("#######################################################\n");

printf("\n");

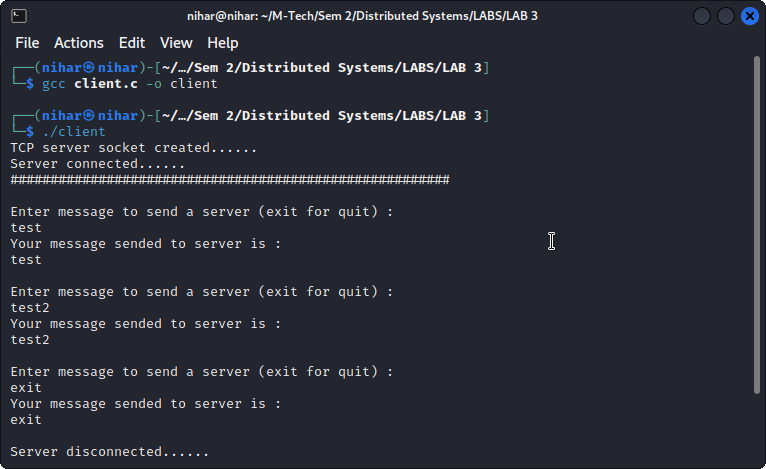
}

}

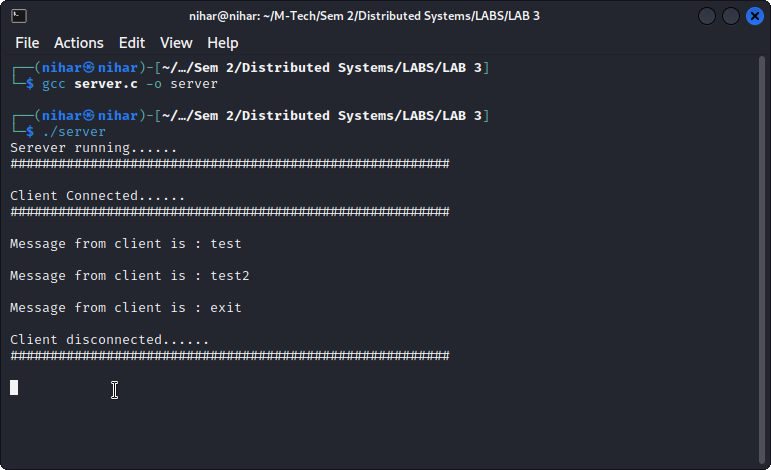
return 0;

}

**Output for client:**

****

**Output for server:**

****