Distributed Systems (CS304)

Assignment - 4

U19CS012

1. 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

[server.c]

```c
// TCP SERVER {Opens a Listening Socket and Waits for Client}
#include <stdio.h>
// For strlen
#include <string.h>
// For sockets
#include <sys/socket.h>
// For inet_addr
#include <arpa/inet.h>
// For write
#include <unistd.h>

#define MAX_SIZE 2000
// [U19CS012] BHAGYA VINOD RANA

int main(int argc, char *argv[])
{
    int socket_desc, client_sock, c, read_size;
    struct sockaddr_in server, client;
    char client_message[MAX_SIZE];

    // Create socket
    socket_desc = socket(AF_INET, SOCK_STREAM, 0);
    if (socket_desc == -1)
    {
        printf("Could Not Create Socket!\n");
    }
    printf("Socket Created!\n");
```

```c
    // Prepare the sockaddr_in structure
    server.sin_family = AF_INET;
    server.sin_addr.s_addr = INADDR_ANY;
    server.sin_port = htons(8888);

    // Bind
    if (bind(socket_desc, (struct sockaddr *)&server, sizeof(server)) < 0)
    {
        // print the error message
        perror("Bind Failed! Error Occured!");
        return 1;
    }
    printf("Bind Done!\n");

    // Listen
    listen(socket_desc, 3);

    // Accept and incoming connection
    puts("Waiting for Incoming Clients Connections ...");
    c = sizeof(struct sockaddr_in);

    // Accept connection from an incoming client
    client_sock = accept(socket_desc, (struct sockaddr *)&client, (socklen_t *)&c);

    if (client_sock < 0)
    {
        perror("Accept Failed");
        return 1;
    }
    printf("Connection Accepted!\n");

    // Receive a Message from client
    while ((read_size = recv(client_sock, client_message, MAX_SIZE, 0)) > 0)
    {
        // Send the message Back to client [Echo]
        write(client_sock, client_message, strlen(client_message));
    }

    if (read_size == 0)
    {
        printf("Client Disconnected!\n");
        fflush(stdout);
    }
    else if (read_size == -1)
    {
        perror("recv() failed");
    }
    return 0;
}
```

[client.c]

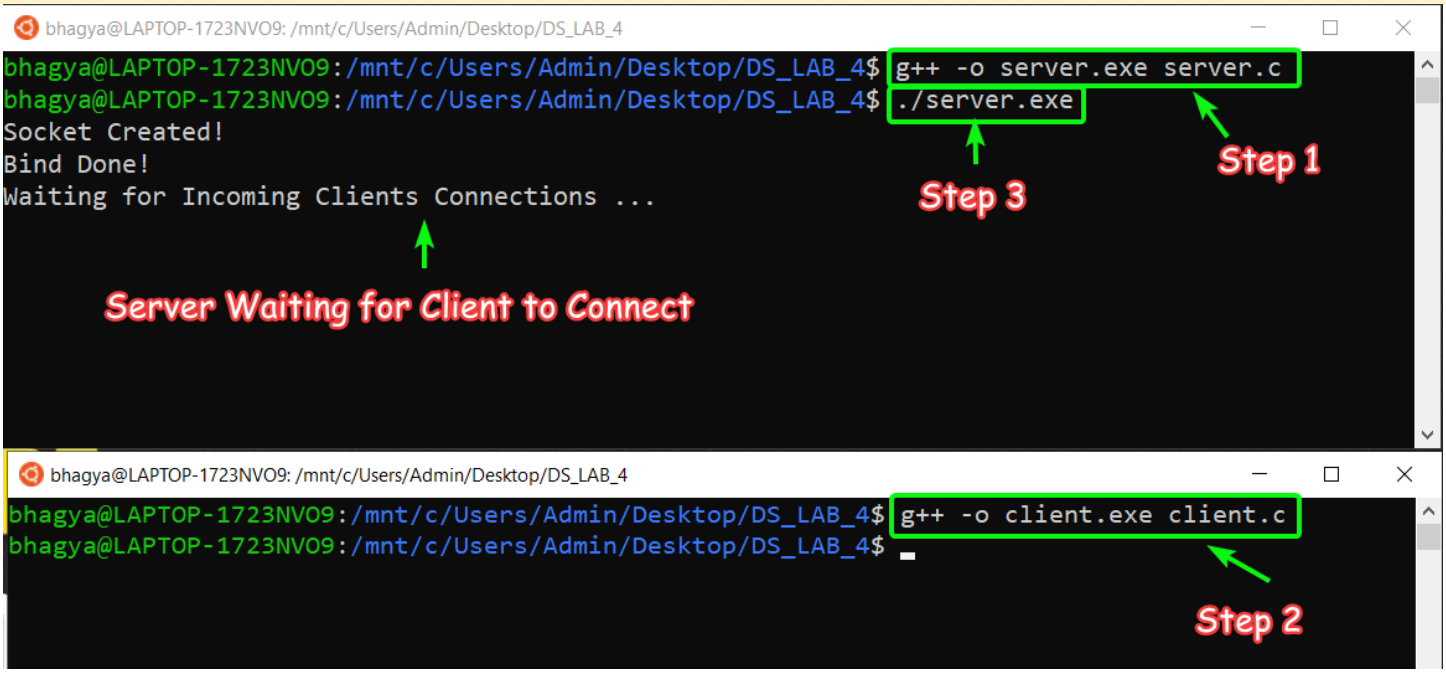
```c
// TCP CLIENT {Connects with the server program knowing IP address and port number.}

// For printf
#include <stdio.h>
// For strlen
#include <string.h>
// For socket
#include <sys/socket.h>
// For inet_addr
#include <arpa/inet.h>
// For write
#include <unistd.h>

#define MAX_SIZE 2000

// [U19CS012] BHAGYA VINOD RANA

int main(int argc, char *argv[])
{
    int sock;
    struct sockaddr_in server;

    // Create socket
    sock = socket(AF_INET, SOCK_STREAM, 0);
    if (sock == -1)
    {
        printf("Could Not Create Socket!\n");
    }
    printf("Socket Created Successfully!\n");

    // IP Address
    server.sin_addr.s_addr = inet_addr("127.0.0.1");
    server.sin_family = AF_INET;
    // Port Number
    server.sin_port = htons(8888);

    // Connect to remote server
    if (connect(sock, (struct sockaddr *)&server, sizeof(server)) < 0)
    {
        perror("Connection Failed! Error Occured!");
        return 1;
    }

    printf("Client Connected!\n");

    // keep communicating with server
    while (1)
    {
```

```c
        char message[MAX_SIZE], server_reply[MAX_SIZE];
        printf("Enter Message [to be Echoed by Server] : ");
        scanf("%s", message);

        // Send Message to the Server
        if (send(sock, message, strlen(message), 0) < 0)
        {
            printf("Send Failed\n");
            return 1;
        }

        // Receive Reply from Server
        if (recv(sock, server_reply, MAX_SIZE, 0) < 0)
        {
            printf("recv() Failed!\n");
            break;
        }
        printf("Server Reply [Echo] : %s \n\n", server_reply);
    }

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

Output

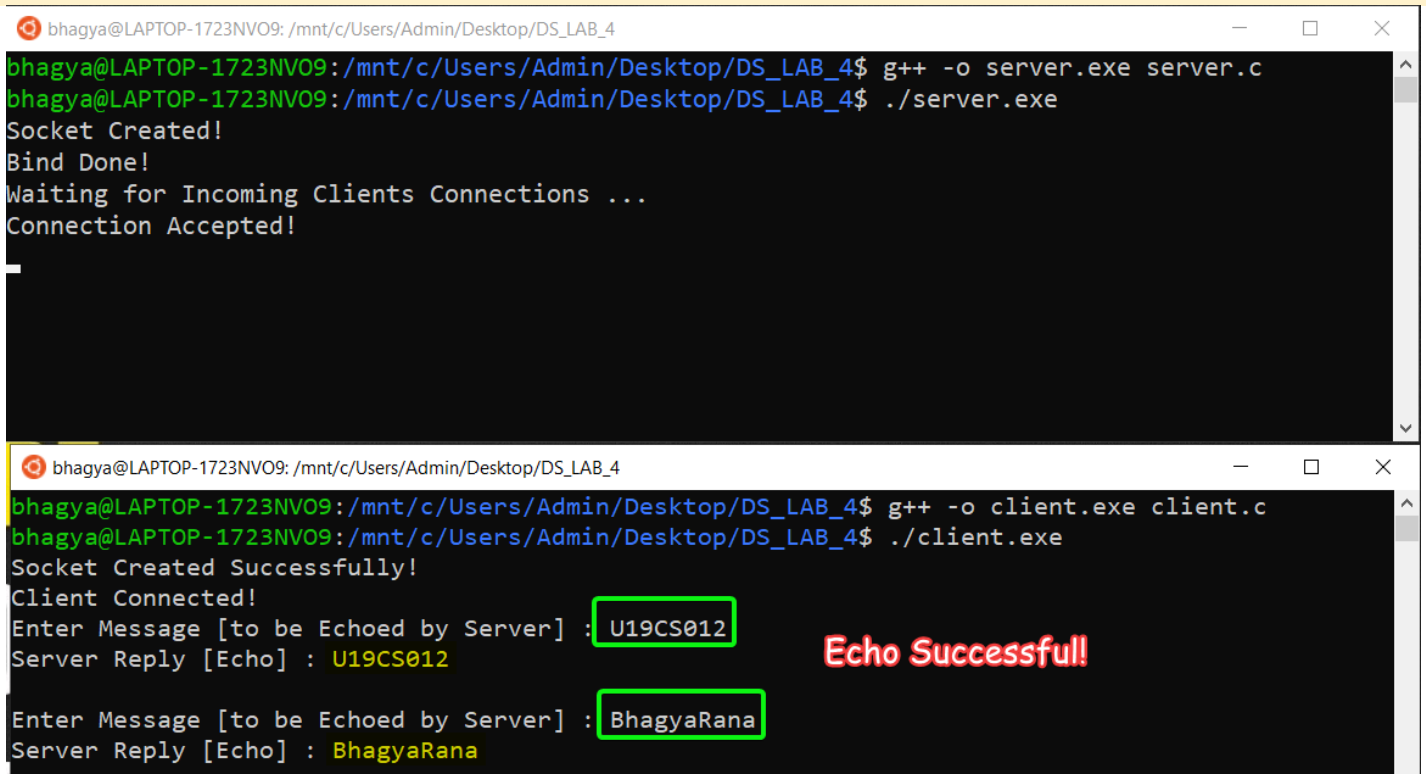
Step 1-2-3: Compile both `server.c` and `client.c` to generate the executable Files.

Start the **Server by executing** the `server.exe`



Step 4: Run the Client, So Server gets the Client Connected and Ready to **Echo the Message** from the Client.

Step 5: Enter "String" and Server will Echo it!



SUBMITTED BY: U19CS012

BHAGYA VINOD RANA