29. Creating the applications using TCP echo server and client in java/C.

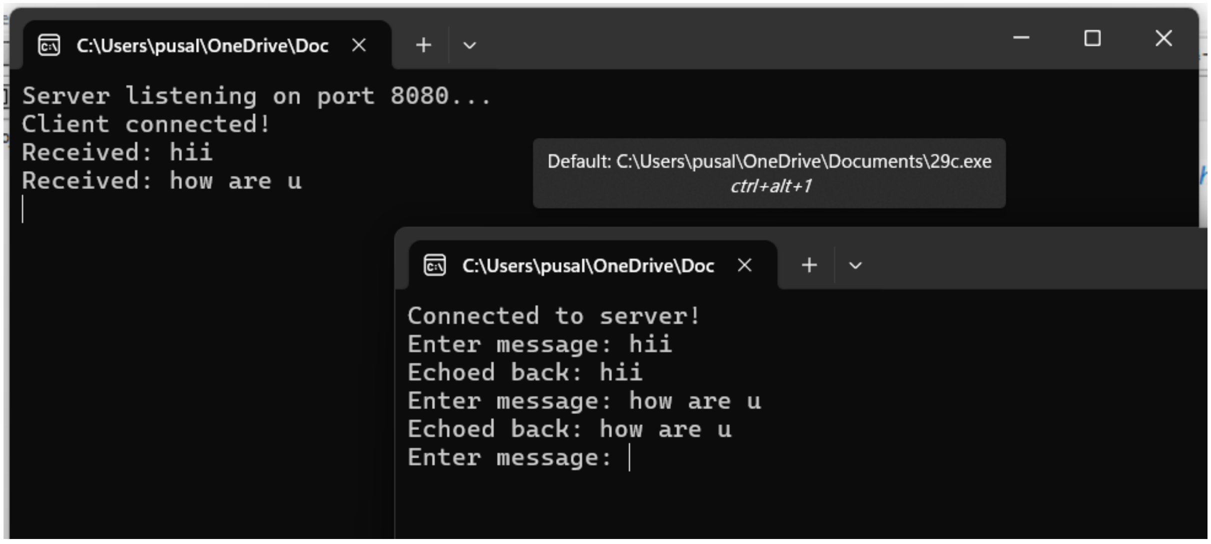
```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <winsock2.h>

#define SERVER_IP "127.0.0.1"
#define PORT 8080
#define BUFFER_SIZE 1024

#pragma comment(lib, "ws2_32.lib") // Link Winsock library

void error_exit(const char *msg) {
   perror(msg);
   exit(EXIT_FAILURE);
}

int main() {
   WSADATA wsa;
   SOCKET client_fd;
   struct sockaddr_in server_addr;
   char buffer[BUFFER_SIZE];

   // Initialize Winsock
   if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0) {
      printf("WSAStartup failed. Error Code: %d\n", WSAGetLastError());
      return 1;
   }

   // Create socket
   if ((client_fd = socket(AF_INET, SOCK_STREAM, 0)) == INVALID_SOCKET) {
    printf("Socket creation failed. Error Code: %d\n", WSAGetLastError());
    return 1;
}

// Configure server address
server_addr.sin_family = AF_INET;
server_addr.sin_port = htons(PORT);
server_addr.sin_addr.s_addr = inet_addr(SERVER_IP);

// Connect to server
if (connect(client_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) == SOCKET_ERROR) {
    printf("Connection failed. Error Code: %d\n", WSAGetLastError());
    closesocket(client_fd);
    WSACleanup();
    return 1;
}

printf("Connected to server!\n");

// Sending loop
while (1) {
    printf("Enter message: ");
    fgets(buffer, BUFFER_SIZE, stdin);
    buffer[strcspn(buffer, "\n")] = '\0'; // Remove newline character

    if (strcmp(buffer, "exit") == 0) {
        break;
    }

    send(client_fd, buffer, strlen(buffer), 0);
```

```
        int bytes_received = recv(client_fd, buffer, BUFFER_SIZE, 0);
        if (bytes_received <= 0) {
            printf("Connection closed or error occurred.\n");
            break;
        }

        buffer[bytes_received] = '\0';
        printf("Echoed back: %s\n", buffer);
    }

    // Cleanup
    closesocket(client_fd);
    WSACleanup();

    return 0;
}
```

Server:

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <winsock2.h>

#define PORT 8080
#define BUFFER_SIZE 1024

#pragma comment(lib, "ws2_32.lib") // Link Winsock library

void error_exit(const char *msg) {
    perror(msg);
    exit(EXIT_FAILURE);
}
```

```
int main() {
    WSADATA wsa;
    SOCKET server_fd, client_fd;
    struct sockaddr_in server_addr, client_addr;
    int addr_len = sizeof(client_addr);
    char buffer[BUFFER_SIZE];

    // Initialize Winsock
    if (WSAStartup(MAKEWORD(2, 2), &wsa) != 0) {
        printf("WSAStartup failed. Error Code: %d\n", WSAGetLastError());
        return 1;
    }

    // Create socket
    if ((server_fd = socket(AF_INET, SOCK_STREAM, 0)) == INVALID_SOCKET) {
        printf("Socket creation failed. Error Code: %d\n", WSAGetLastError());
        return 1;
    }

    // Configure server address
    server_addr.sin_family = AF_INET;
    server_addr.sin_addr.s_addr = INADDR_ANY;
    server_addr.sin_port = htons(PORT);

    // Bind socket
    if (bind(server_fd, (struct sockaddr *)&server_addr, sizeof(server_addr)) == SOCKET_ERROR) {
        printf("Bind failed. Error Code: %d\n", WSAGetLastError());
        closesocket(server_fd);
        WSACleanup();
        return 1;
    }

    // Listen for incoming connections
    if (listen(server_fd, 5) == SOCKET_ERROR) {
        printf("Listen failed. Error Code: %d\n", WSAGetLastError());
        closesocket(server_fd);
        WSACleanup();
        return 1;
    }

    printf("Server listening on port %d...\n", PORT);

    // Accept connection
    if ((client_fd = accept(server_fd, (struct sockaddr *)&client_addr, &addr_len)) == INVALID_SOCKET)
{
        printf("Accept failed. Error Code: %d\n", WSAGetLastError());
        closesocket(server_fd);
        WSACleanup();
        return 1;
    }

    printf("Client connected!\n");

    // Echo loop
    while (1) {
        int bytes_received = recv(client_fd, buffer, BUFFER_SIZE, 0);
        if (bytes_received <= 0) {
            printf("Connection closed or error occurred.\n");
            break;
        }
        buffer[bytes_received] = '\0';
        printf("Received: %s\n", buffer);

        // Send back the same message
        send(client_fd, buffer, bytes_received, 0);
    }

    // Cleanup
    closesocket(client_fd);
    closesocket(server_fd);
    WSACleanup();

    return 0;
}
```


C:\Users\pusal\OneDrive\Doc
Default: C:\Users\pusal\OneDrive\Documents\29c.exe
ctrl+alt+1
C:\Users\pusal\OneDrive\Doc

```
Server listening on port 8080...
Client connected!
Received: hii
Received: how are u
```

```
Connected to server!
Enter message: hii
Echoed back: hii
Enter message: how are u
Echoed back: how are u
Enter message:
```