**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;**

**}**

**A screenshot of a computer screen

AI-generated content may be incorrect.**