1.¦øªA¾¹ºÝ¡G

#undef UNICODE

#define WIN32\_LEAN\_AND\_MEAN

#include <windows.h>

#include <winsock2.h>

#include <ws2tcpip.h>

#include <stdlib.h>

#include <stdio.h>

// Need to link with Ws2\_32.lib

#pragma comment (lib, "Ws2\_32.lib")

// #pragma comment (lib, "Mswsock.lib")

#define DEFAULT\_BUFLEN 512

#define DEFAULT\_PORT "27015"

int \_\_cdecl main(void)

{

WSADATA wsaData;

int iResult;

SOCKET ListenSocket = INVALID\_SOCKET;

SOCKET ClientSocket = INVALID\_SOCKET;

struct addrinfo\* result = NULL;

struct addrinfo hints;

int iSendResult;

char recvbuf[DEFAULT\_BUFLEN];

int recvbuflen = DEFAULT\_BUFLEN;

// Initialize Winsock

iResult = WSAStartup(MAKEWORD(2, 2), &wsaData);

if (iResult != 0) {

printf("WSAStartup failed with error: %d\n", iResult);

return 1;

}

ZeroMemory(&hints, sizeof(hints));

hints.ai\_family = AF\_INET;

hints.ai\_socktype = SOCK\_STREAM;

hints.ai\_protocol = IPPROTO\_TCP;

hints.ai\_flags = AI\_PASSIVE;

// Resolve the server address and port

iResult = getaddrinfo(NULL, DEFAULT\_PORT, &hints, &result);

if (iResult != 0) {

printf("getaddrinfo failed with error: %d\n", iResult);

WSACleanup();

return 1;

}

// Create a SOCKET for connecting to server

ListenSocket = socket(result->ai\_family, result->ai\_socktype, result->ai\_protocol);

if (ListenSocket == INVALID\_SOCKET) {

printf("socket failed with error: %ld\n", WSAGetLastError());

freeaddrinfo(result);

WSACleanup();

return 1;

}

// Setup the TCP listening socket

iResult = bind(ListenSocket, result->ai\_addr, (int)result->ai\_addrlen);

if (iResult == SOCKET\_ERROR) {

printf("bind failed with error: %d\n", WSAGetLastError());

freeaddrinfo(result);

closesocket(ListenSocket);

WSACleanup();

return 1;

}

freeaddrinfo(result);

iResult = listen(ListenSocket, SOMAXCONN);

if (iResult == SOCKET\_ERROR) {

printf("listen failed with error: %d\n", WSAGetLastError());

closesocket(ListenSocket);

WSACleanup();

return 1;

}

// Accept a client socket

ClientSocket = accept(ListenSocket, NULL, NULL);

if (ClientSocket == INVALID\_SOCKET) {

printf("accept failed with error: %d\n", WSAGetLastError());

closesocket(ListenSocket);

WSACleanup();

return 1;

}

// No longer need server socket

closesocket(ListenSocket);

// Receive until the peer shuts down the connection

do {

iResult = recv(ClientSocket, recvbuf, recvbuflen, 0);

if (iResult > 0) {

printf("Bytes received: %d\n", iResult);

// Echo the buffer back to the sender

iSendResult = send(ClientSocket, recvbuf, iResult, 0);

if (iSendResult == SOCKET\_ERROR) {

printf("send failed with error: %d\n", WSAGetLastError());

closesocket(ClientSocket);

WSACleanup();

return 1;

}

printf("Bytes sent: %d\n", iSendResult);

}

else if (iResult == 0)

printf("Connection closing...\n");

else {

printf("recv failed with error: %d\n", WSAGetLastError());

closesocket(ClientSocket);

WSACleanup();

return 1;

}

} while (iResult > 0);

// shutdown the connection since we're done

iResult = shutdown(ClientSocket, SD\_SEND);

if (iResult == SOCKET\_ERROR) {

printf("shutdown failed with error: %d\n", WSAGetLastError());

closesocket(ClientSocket);

WSACleanup();

return 1;

}

// cleanup

closesocket(ClientSocket);

WSACleanup();

return 0;

}

2.«È¤áºÝ¡G

#define WIN32\_LEAN\_AND\_MEAN

#include <windows.h>

#include <winsock2.h>

#include <ws2tcpip.h>

#include <stdlib.h>

#include <stdio.h>

// Need to link with Ws2\_32.lib, Mswsock.lib, and Advapi32.lib

#pragma comment (lib, "Ws2\_32.lib")

#pragma comment (lib, "Mswsock.lib")

#pragma comment (lib, "AdvApi32.lib")

#define DEFAULT\_BUFLEN 512

#define DEFAULT\_PORT "27015"

int \_\_cdecl main(int argc, char\*\* argv)

{

WSADATA wsaData;

SOCKET ConnectSocket = INVALID\_SOCKET;

struct addrinfo\* result = NULL,

\* ptr = NULL,

hints;

const char\* sendbuf = "this is a test";

char recvbuf[DEFAULT\_BUFLEN];

int iResult;

int recvbuflen = DEFAULT\_BUFLEN;

// Validate the parameters

if (argc != 2) {

printf("usage: %s server-name\n", argv[0]);

return 1;

}

// Initialize Winsock

iResult = WSAStartup(MAKEWORD(2, 2), &wsaData);

if (iResult != 0) {

printf("WSAStartup failed with error: %d\n", iResult);

return 1;

}

ZeroMemory(&hints, sizeof(hints));

hints.ai\_family = AF\_UNSPEC;

hints.ai\_socktype = SOCK\_STREAM;

hints.ai\_protocol = IPPROTO\_TCP;

// Resolve the server address and port

iResult = getaddrinfo(argv[1], DEFAULT\_PORT, &hints, &result);

if (iResult != 0) {

printf("getaddrinfo failed with error: %d\n", iResult);

WSACleanup();

return 1;

}

// Attempt to connect to an address until one succeeds

for (ptr = result; ptr != NULL; ptr = ptr->ai\_next) {

// Create a SOCKET for connecting to server

ConnectSocket = socket(ptr->ai\_family, ptr->ai\_socktype,

ptr->ai\_protocol);

if (ConnectSocket == INVALID\_SOCKET) {

printf("socket failed with error: %ld\n", WSAGetLastError());

WSACleanup();

return 1;

}

// Connect to server.

iResult = connect(ConnectSocket, ptr->ai\_addr, (int)ptr->ai\_addrlen);

if (iResult == SOCKET\_ERROR) {

closesocket(ConnectSocket);

ConnectSocket = INVALID\_SOCKET;

continue;

}

break;

}

freeaddrinfo(result);

if (ConnectSocket == INVALID\_SOCKET) {

printf("Unable to connect to server!\n");

WSACleanup();

return 1;

}

// Send an initial buffer

iResult = send(ConnectSocket, sendbuf, (int)strlen(sendbuf), 0);

if (iResult == SOCKET\_ERROR) {

printf("send failed with error: %d\n", WSAGetLastError());

closesocket(ConnectSocket);

WSACleanup();

return 1;

}

printf("Bytes Sent: %ld\n", iResult);

// shutdown the connection since no more data will be sent

iResult = shutdown(ConnectSocket, SD\_SEND);

if (iResult == SOCKET\_ERROR) {

printf("shutdown failed with error: %d\n", WSAGetLastError());

closesocket(ConnectSocket);

WSACleanup();

return 1;

}

// Receive until the peer closes the connection

do {

iResult = recv(ConnectSocket, recvbuf, recvbuflen, 0);

if (iResult > 0)

printf("Bytes received: %d\n", iResult);

else if (iResult == 0)

printf("Connection closed\n");

else

printf("recv failed with error: %d\n", WSAGetLastError());

} while (iResult > 0);

// cleanup

closesocket(ConnectSocket);

WSACleanup();

return 0;

}