1.ServerºÝ¡G

#ifndef UNICODE

#define UNICODE

#endif

#define WIN32\_LEAN\_AND\_MEAN

#include <winsock2.h>

#include <Ws2tcpip.h>

#include <stdio.h>

// Link with ws2\_32.lib

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

int main()

{

int iResult = 0;

WSADATA wsaData;

SOCKET RecvSocket;

sockaddr\_in RecvAddr;

unsigned short Port = 27015;

char RecvBuf[1024];

int BufLen = 1024;

sockaddr\_in SenderAddr;

int SenderAddrSize = sizeof(SenderAddr);

//-----------------------------------------------

// Initialize Winsock

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

if (iResult != NO\_ERROR) {

wprintf(L"WSAStartup failed with error %d\n", iResult);

return 1;

}

//-----------------------------------------------

// Create a receiver socket to receive datagrams

RecvSocket = socket(AF\_INET, SOCK\_DGRAM, IPPROTO\_UDP);

if (RecvSocket == INVALID\_SOCKET) {

wprintf(L"socket failed with error %d\n", WSAGetLastError());

return 1;

}

//-----------------------------------------------

// Bind the socket to any address and the specified port.

RecvAddr.sin\_family = AF\_INET;

RecvAddr.sin\_port = htons(Port);

RecvAddr.sin\_addr.s\_addr = htonl(INADDR\_ANY);

iResult = bind(RecvSocket, (SOCKADDR\*)&RecvAddr, sizeof(RecvAddr));

if (iResult != 0) {

wprintf(L"bind failed with error %d\n", WSAGetLastError());

return 1;

}

//-----------------------------------------------

// Call the recvfrom function to receive datagrams

// on the bound socket.

wprintf(L"Receiving datagrams...\n");

iResult = recvfrom(RecvSocket,

RecvBuf, BufLen, 0, (SOCKADDR\*)&SenderAddr, &SenderAddrSize);

if (iResult == SOCKET\_ERROR) {

wprintf(L"recvfrom failed with error %d\n", WSAGetLastError());

}

//-----------------------------------------------

// Close the socket when finished receiving datagrams

wprintf(L"Finished receiving. Closing socket.\n");

iResult = closesocket(RecvSocket);

if (iResult == SOCKET\_ERROR) {

wprintf(L"closesocket failed with error %d\n", WSAGetLastError());

return 1;

}

//-----------------------------------------------

// Clean up and exit.

wprintf(L"Exiting.\n");

WSACleanup();

return 0;

}

2.ClientºÝ¡G

#ifndef UNICODE

#define UNICODE

#endif

#define WIN32\_LEAN\_AND\_MEAN

#include <winsock2.h>

#include <Ws2tcpip.h>

#include <stdio.h>

#include <stdlib.h>

// Link with ws2\_32.lib

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

int main()

{

int iResult;

WSADATA wsaData;

SOCKET SendSocket = INVALID\_SOCKET;

sockaddr\_in RecvAddr;

unsigned short Port = 27015;

char SendBuf[1024];

int BufLen = 1024;

//----------------------

// Initialize Winsock

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

if (iResult != NO\_ERROR) {

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

return 1;

}

//---------------------------------------------

// Create a socket for sending data

SendSocket = socket(AF\_INET, SOCK\_DGRAM, IPPROTO\_UDP);

if (SendSocket == INVALID\_SOCKET) {

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

WSACleanup();

return 1;

}

//---------------------------------------------

// Set up the RecvAddr structure with the IP address of

// the receiver (in this example case "192.168.1.1")

// and the specified port number.

RecvAddr.sin\_family = AF\_INET;

RecvAddr.sin\_port = htons(Port);

inet\_pton(AF\_INET, "192.168.42.124", (void\*)&RecvAddr.sin\_addr.S\_un.S\_addr);

//---------------------------------------------

// Send a datagram to the receiver

wprintf(L"Sending a datagram to the receiver...\n");

iResult = sendto(SendSocket,

SendBuf, BufLen, 0, (SOCKADDR\*)&RecvAddr, sizeof(RecvAddr));

if (iResult == SOCKET\_ERROR) {

wprintf(L"sendto failed with error: %d\n", WSAGetLastError());

closesocket(SendSocket);

WSACleanup();

return 1;

}

//---------------------------------------------

// When the application is finished sending, close the socket.

wprintf(L"Finished sending. Closing socket.\n");

iResult = closesocket(SendSocket);

if (iResult == SOCKET\_ERROR) {

wprintf(L"closesocket failed with error: %d\n", WSAGetLastError());

WSACleanup();

return 1;

}

//---------------------------------------------

// Clean up and quit.

wprintf(L"Exiting.\n");

WSACleanup();

system("pause");

return 0;

}