

## 2-2 加餐 - 验证 UDP - windows 作为 client 访问 Linux

完整的测试代码链接(包括服务器端): <https://gitee.com/whb-helloworld/linux-plus-meal/tree/master/windows-udp-tcp>

### UDP client 样例代码

```
C++
#include <iostream>
#include <cstdio>
#include <thread>
#include <string>
#include <cstdlib>
#include <WinSock2.h>
#include <Windows.h>

#pragma warning(disable : 4996)

#pragma comment(lib, "ws2_32.lib")

std::string serverip = ""; // 填写你的云服务器 ip
uint16_t serverport = 8888; // 填写你的云服务开放的端口号

int main()
{
    WSADATA wsd;
    WSStartup(MAKEWORD(2, 2), &wsd);

    struct sockaddr_in server;
    memset(&server, 0, sizeof(server));
    server.sin_family = AF_INET;
    server.sin_port = htons(serverport); //?
    server.sin_addr.s_addr = inet_addr(serverip.c_str());

    SOCKET sockfd = socket(AF_INET, SOCK_DGRAM, 0);
    if (sockfd == SOCKET_ERROR)
```

```
{
    std::cout << "socket error" << std::endl;
    return 1;
}
std::string message;
char buffer[1024];
while (true)
{
    std::cout << "Please Enter@ ";
    std::getline(std::cin, message);
    if(message.empty()) continue;
    sendto(sockfd, message.c_str(), (int)message.size(), 0,
(struct sockaddr *)&server, sizeof(server));
    struct sockaddr_in temp;
    int len = sizeof(temp);
    int s = recvfrom(sockfd, buffer, 1023, 0, (struct sockaddr
*)&temp, &len);
    if (s > 0)
    {
        buffer[s] = 0;
        std::cout << buffer << std::endl;
    }
}

closesocket(sockfd);
WSACleanup();
return 0;
}
```

```
whb@bite01:~/linux-plus-meal/windows-udp-tcp$ ./udp_server 8888
113.132.212.215:38071# hello
113.132.212.215:38071# i am udp client
113.132.212.215:38071# goo
113.132.212.215:38071# good
113.132.212.215:38071# good man
```

服务器端

```
test.cpp
7  #include <Windows.h>
8
9  #pragma warning(disable : 4996)
10
11 #pragma comment(lib, "ws2_32.lib")
12
13 uint16_t serverport = 8888;
```

D:\test\test\_code\Vs4\Debug

```
Please Enter@ hello
hello[udp server echo message]
Please Enter@ i am udp client
i am udp client[udp server echo message]
Please Enter@ goo
goo[udp server echo message]
Please Enter@ good
good[udp server echo message]
Please Enter@ good man
good man[udp server echo message]
Please Enter@
```

vs客户端  
windows上  
启动编译运行

- 注意：一定要开放云服务器对应的端口号，在你的阿里云或者腾讯云或者华为云的网站后台中开放。
- 我们发现可以 udp tcpclient(Windows)和 tcpserver(Linux)可以通信

**C++**

**WinSock2.h** 是 **Windows Sockets API**（应用程序接口）的头文件，用于在 **Windows** 平台上进行网络编程。它包含了 **Windows Sockets 2**（**Winsock2**）所需的数据类型、函数声明和结构定义，使得开发者能够创建和使用套接字（**sockets**）进行网络通信。

在编写使用 **Winsock2** 的程序时，需要在源文件中包含 **WinSock2.h** 头文件。这样，编译器就能够识别并理解 **Winsock2** 中定义的数据类型和函数，从而能够正确地编译和链接网络相关的代码。

此外，与 **WinSock2.h** 头文件相对应的是 **ws2\_32.lib** 库文件。在链接阶段，需要将这个库文件链接到程序中，以确保运行时能够找到并调用 **Winsock2 API** 中实现的函数。

在 **WinSock2.h** 中定义了一些重要的数据类型和函数，如：

**WSADATA**: 保存初始化 **Winsock** 库时返回的信息。

**SOCKET**: 表示一个套接字描述符，用于在网络中唯一标识一个套接字。

**sockaddr\_in**: **IPv4** 地址结构体，用于存储 **IP** 地址和端口号等信息。

**socket()**: 创建一个新的套接字。

**bind()**: 将套接字与本地地址绑定。

**listen()**: 将套接字设置为监听模式，等待客户端的连接请求。

**accept()**: 接受客户端的连接请求，并返回一个新的套接字描述符，用于与客户端进行通信。

**C++**

**WSAStartup** 函数是 **Windows Sockets API** 的初始化函数，它用于初始化 **Winsock** 库。该函数在应用程序或 **DLL** 调用任何 **Windows** 套接字函数之前必须首先执行，它扮演着初始化的角色。

以下是 **WSAStartup** 函数的一些关键点：

它接受两个参数：**wVersionRequested** 和 **lpWSAData**。**wVersionRequested** 用于指定所请求的 **Winsock** 版本，通常使用 **MAKEWORD(major, minor)** 宏，其中 **major** 和 **minor** 分别表示请求的主版本号 and 次版本号。**lpWSAData** 是一个指向 **WSADATA** 结构的指针，用于接收初始化信息。

如果函数调用成功，它会返回 **0**；否则，返回错误代码。

**WSAStartup** 函数的主要作用是向操作系统说明我们将使用哪个版本的 **Winsock** 库，从而使得该库文件能与当前的操作系统协同工作。成功调用该函数后，**Winsock** 库的状态会被初始化，应用程序就可以使用 **Winsock** 提供的一系列套接字服务，如地址家族识别、地址转换、名字查询和连接控制等。这些服务使得应用程序可以与底层的网络协议栈进行交互，实现网络通信。

在调用 **WSAStartup** 函数后，如果应用程序完成了对请求的 **Socket** 库的使用，应调用 **WSACleanup** 函数来解除与 **Socket** 库的绑定并释放所占用的系统资源。