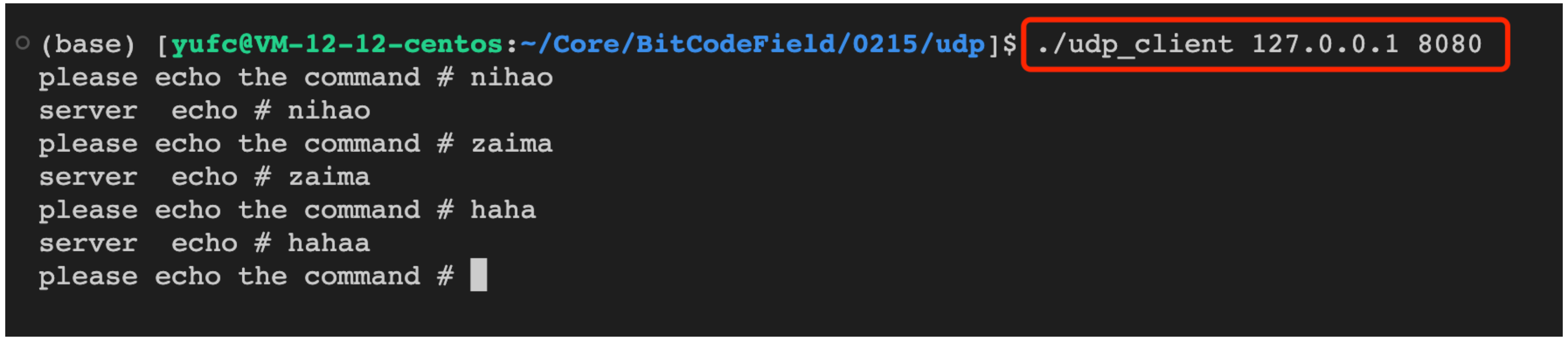
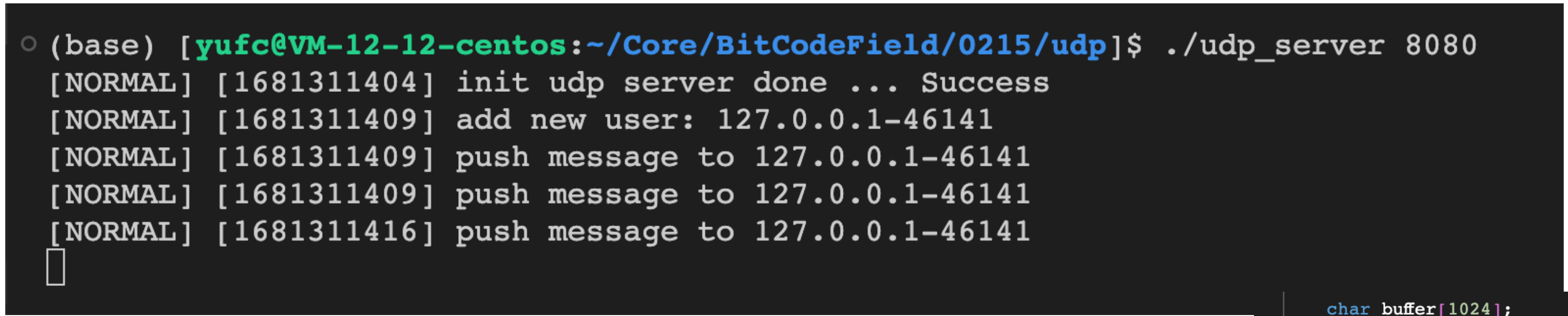
0216socket



```
(base) [yufc@VM-12-12-centos:~/Core/BitCodeField/0215/udp]$ ./udp_client 127.0.0.1 8080
please echo the command # nihao
server  echo # nihao
please echo the command # zaima
server  echo # zaima
please echo the command # haha
server  echo # hahaa
please echo the command # 
```

当然，现在这个客户端是肯定有问题的

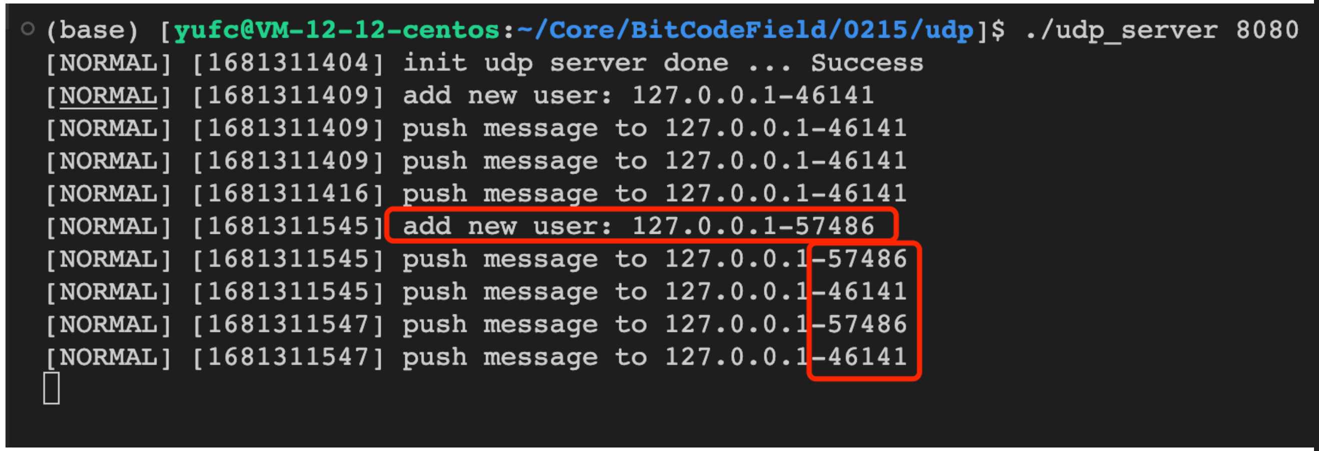
我们看客户端的代码就能看到如果客户端一直不输入



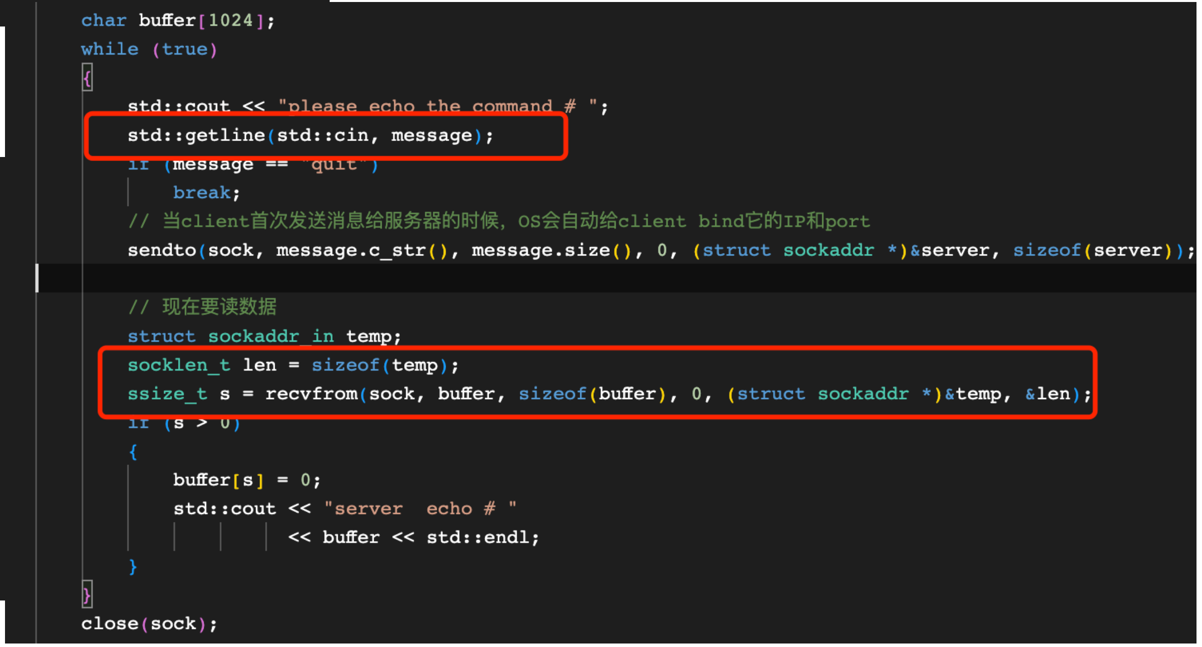
```
(base) [yufc@VM-12-12-centos:~/Core/BitCodeField/0215/udp]$ ./udp_server 8080
[NORMAL] [1681311404] init udp server done ... Success
[NORMAL] [1681311409] add new user: 127.0.0.1-46141
[NORMAL] [1681311409] push message to 127.0.0.1-46141
[NORMAL] [1681311409] push message to 127.0.0.1-46141
[NORMAL] [1681311416] push message to 127.0.0.1-46141
```

代码就会阻塞在getline那里
但是我们期望的聊天软件不是这样的，我们就算不输入，不发，也弄收消息，发的同时也能收消息

如果现在我们再开一个服务端



```
(base) [yufc@VM-12-12-centos:~/Core/BitCodeField/0215/udp]$ ./udp_server 8080
[NORMAL] [1681311404] init udp server done ... Success
[NORMAL] [1681311409] add new user: 127.0.0.1-46141
[NORMAL] [1681311409] push message to 127.0.0.1-46141
[NORMAL] [1681311409] push message to 127.0.0.1-46141
[NORMAL] [1681311416] push message to 127.0.0.1-46141
[NORMAL] [1681311545] add new user: 127.0.0.1-57486
[NORMAL] [1681311545] push message to 127.0.0.1-57486
[NORMAL] [1681311545] push message to 127.0.0.1-46141
[NORMAL] [1681311547] push message to 127.0.0.1-57486
[NORMAL] [1681311547] push message to 127.0.0.1-46141
```



```cpp
char buffer[1024];
while (true)
{
    std::cout << "please echo the command # ";
    std::getline(std::cin, message);
    if (message == "quit")
        break;
    // 当client首次发送消息给服务器的时候，OS会自动给client bind它的IP和port
    sendto(sock, message.c_str(), message.size(), 0, (struct sockaddr *)&server, sizeof(server));

    // 现在要读数据
    struct sockaddr_in temp;
    socklen_t len = sizeof(temp);
    ssize_t s = recvfrom(sock, buffer, sizeof(buffer), 0, (struct sockaddr *)&temp, &len);
    if (s > 0)
    {
        buffer[s] = 0;
        std::cout << "server  echo # "
                  << buffer << std::endl;
    }
}
close(sock);
```

所以，我们应该引入多线程

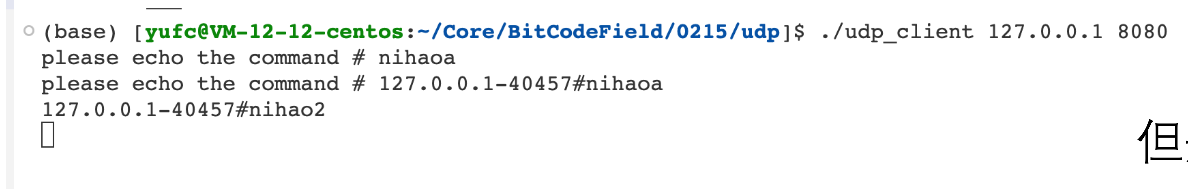
收信息用一个线程，发信息用一个线程
这样就没问题了

```cpp
static void *udpSend(void *args)
{
    int sock = *(int*)((ThreadData*)args)->__args;
    std::string name = ((ThreadData*)args)->__name;

    std::string message;
    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());

    // 收信息+发送
    while (true)
    {
        std::cout << "please echo the command # ";
        std::getline(std::cin, message);
        if (message == "quit")
            break;
        // 当client首次发送消息给服务器的时候，OS会自动给client bind它的IP和port
        sendto(sock, message.c_str(), message.size(), 0, (struct sockaddr *)&server, sizeof(server));
    }
    return nullptr;
}
```

```cpp
static void *udpRecv(void *args)
{
    int sock = *(int*)((ThreadData*)args)->__args;
    std::string name = ((ThreadData*)args)->__name;

    char buffer[READ_SEND_MAX_SIZE];
    while(true)
    {
        // 现在要读数据
        struct sockaddr_in temp;
        socklen_t len = sizeof(temp);
        ssize_t s = recvfrom(sock, buffer, sizeof(buffer), 0, (struct sockaddr *)&temp, &len);
        if (s > 0)
        {
            buffer[s] = 0;
            std::cout << buffer << std::endl;
        }
    }
}
```

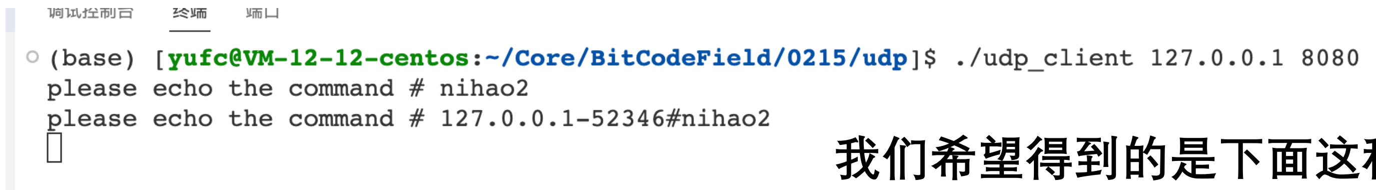
Udp是全双工的 -> 可以同时进行收发而不受干扰

最后，结果和我们期望的一样



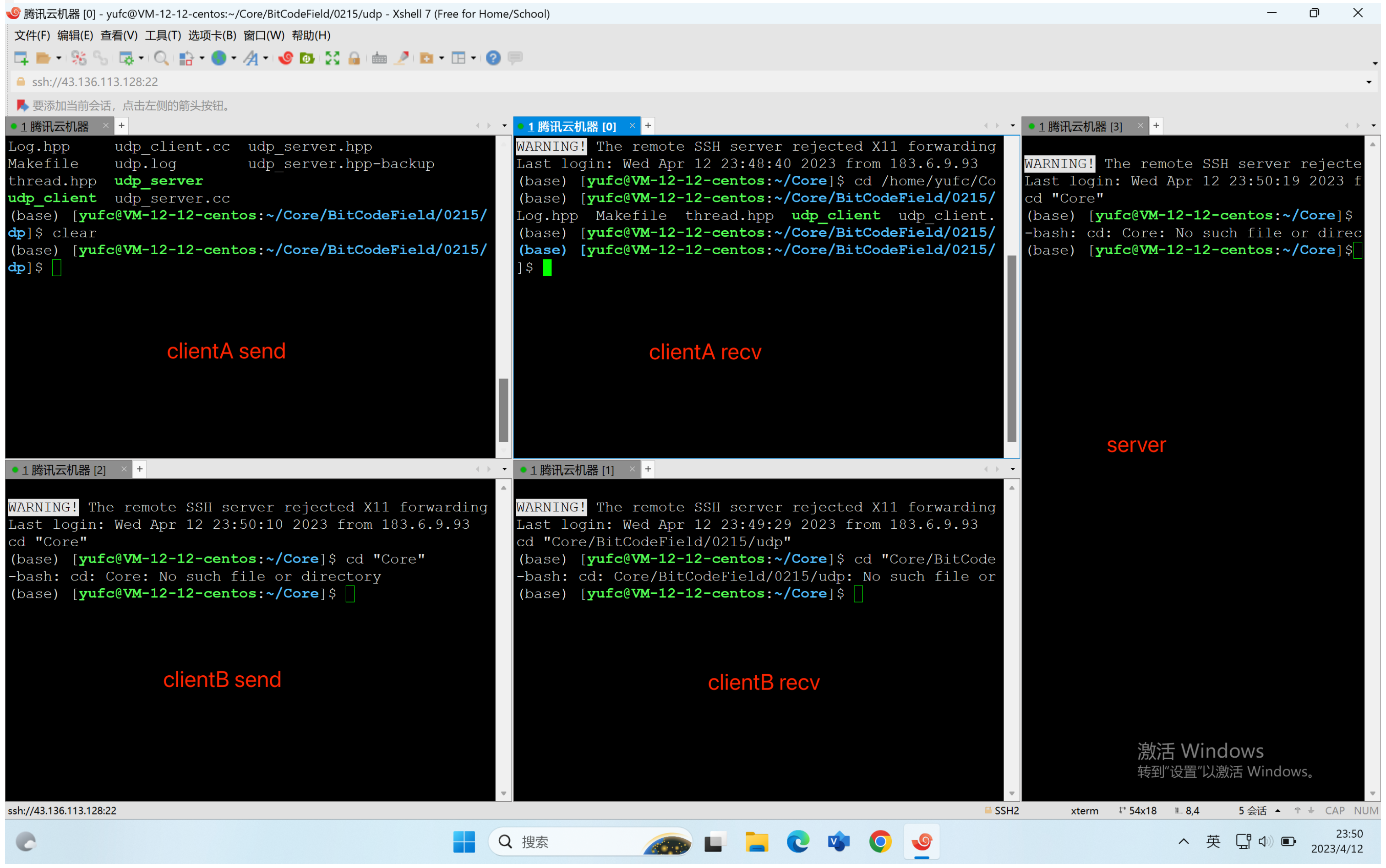
```
(base) [yufc@VM-12-12-centos:~/Core/BitCodeField/0215/udp]$ ./udp_client 127.0.0.1 8080
please echo the command # nihaoa
please echo the command # 127.0.0.1-40457#nihaoa
127.0.0.1-40457#nihao2
```

但是这样消息和发的地方都混在一起了， 怎么办



```
(base) [yufc@VM-12-12-centos:~/Core/BitCodeField/0215/udp]$ ./udp_client 127.0.0.1 8080
please echo the command # nihao2
please echo the command # 127.0.0.1-52346#nihao2
```

我们希望得到的是下面这种效果！



```
Log.hpp      udp_client.cc   udp_server.hpp
Makefile     udp.log         udp_server.hpp-backup
thread.hpp   udp_server
udp_client   udp_server.cc
(base) [yufc@VM-12-12-centos:~/Core/BitCodeField/0215/
dp]$ clear
(base) [yufc@VM-12-12-centos:~/Core/BitCodeField/0215/
dp]$
```

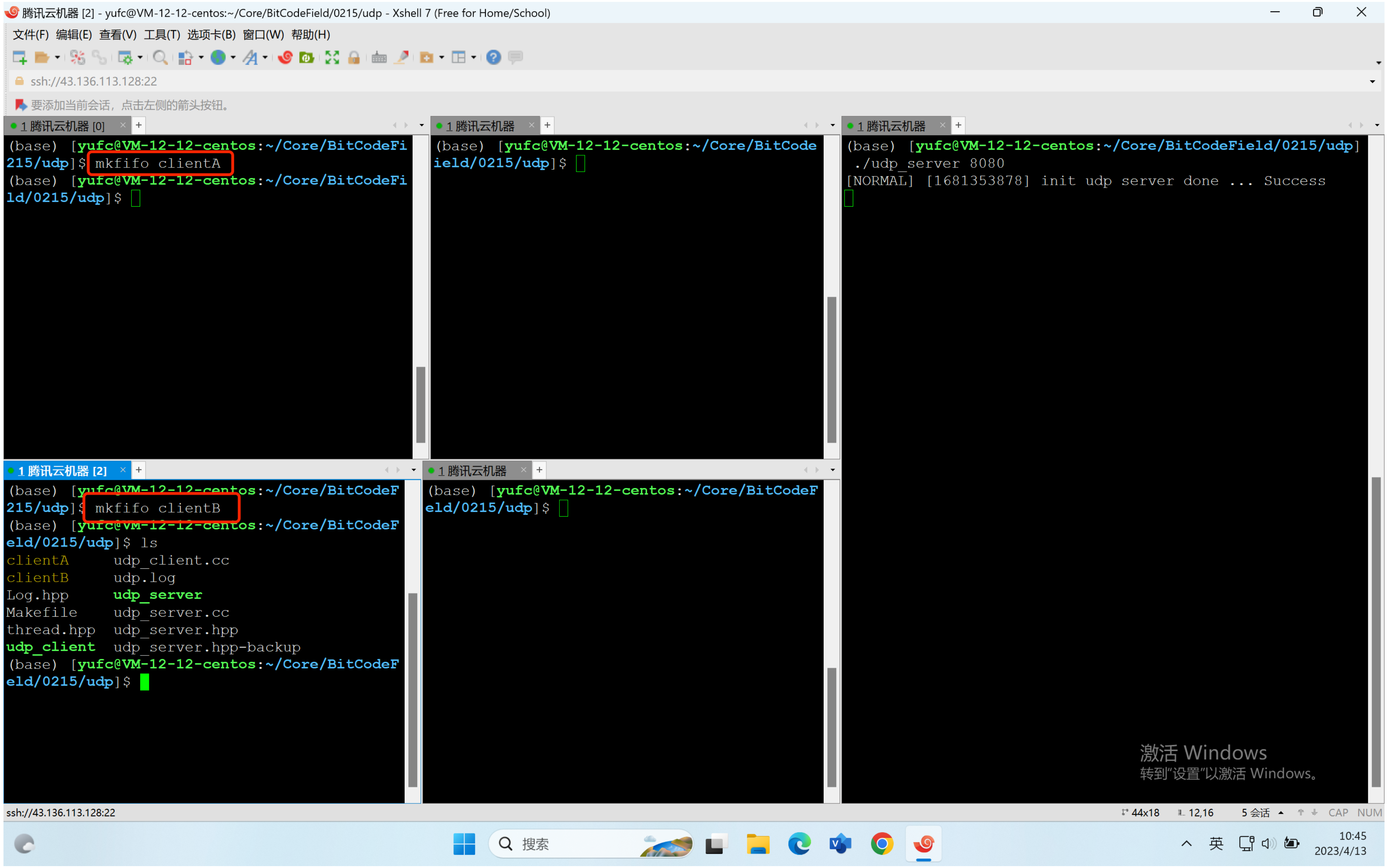
```
WARNING! The remote SSH server rejected X11 forwarding
Last login: Wed Apr 12 23:48:40 2023 from 183.6.9.93
(base) [yufc@VM-12-12-centos:~/Core]$ cd /home/yufc/Co
(base) [yufc@VM-12-12-centos:~/Core/BitCodeField/0215/
Log.hpp  Makefile  thread.hpp  udp_client  udp_client.
(base) [yufc@VM-12-12-centos:~/Core/BitCodeField/0215/
(base) [yufc@VM-12-12-centos:~/Core/BitCodeField/0215/
]$
```

```
WARNING! The remote SSH server rejecte
Last login: Wed Apr 12 23:50:19 2023 f
cd "Core"
(base) [yufc@VM-12-12-centos:~/Core]$
-bash: cd: Core: No such file or direc
(base) [yufc@VM-12-12-centos:~/Core]$
```

```
WARNING! The remote SSH server rejected X11 forwarding
Last login: Wed Apr 12 23:50:10 2023 from 183.6.9.93
cd "Core"
(base) [yufc@VM-12-12-centos:~/Core]$ cd "Core"
-bash: cd: Core: No such file or directory
(base) [yufc@VM-12-12-centos:~/Core]$
```

```
WARNING! The remote SSH server rejected X11 forwarding
Last login: Wed Apr 12 23:49:29 2023 from 183.6.9.93
cd "Core/BitCodeField/0215/udp"
(base) [yufc@VM-12-12-centos:~/Core]$ cd "Core/BitCode
-bash: cd: Core/BitCodeField/0215/udp: No such file or
(base) [yufc@VM-12-12-centos:~/Core]$
```

聊天的时候发送和接收分开！



腾讯云机器 [2] - yufc@VM-12-12-centos:~/Core/BitCodeField/0215/udp - Xshell 7 (Free for Home/School)

文件(F) 编辑(E) 查看(V) 工具(T) 选项卡(B) 窗口(W) 帮助(H)
ssh://43.136.113.128:22
要添加当前会话，点击左侧的箭头按钮。
1 腾讯云机器 [0]
1 腾讯云机器
1 腾讯云机器
1 腾讯云机器 [2]
1 腾讯云机器

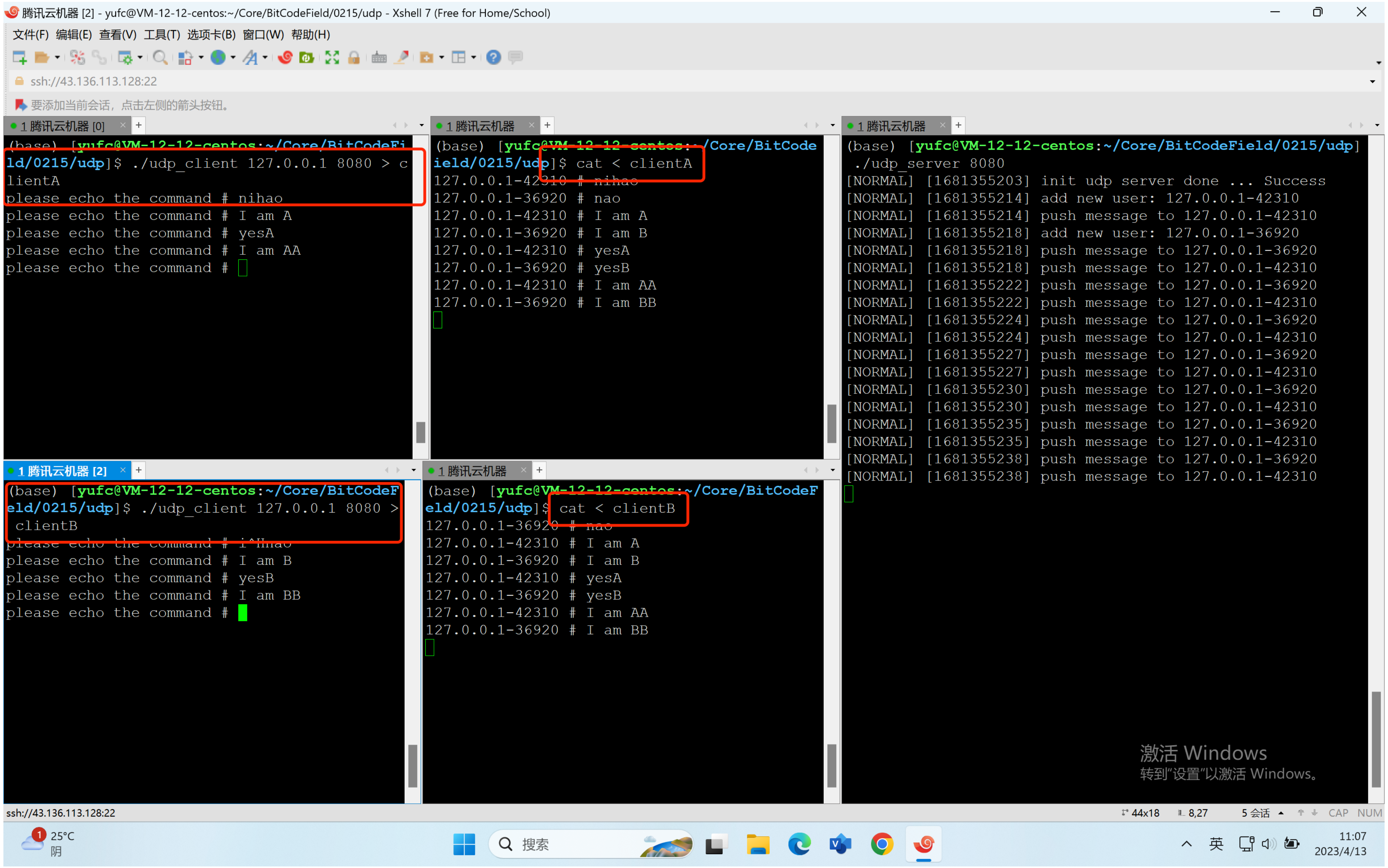
```
(base) [yufc@VM-12-12-centos:~/Core/BitCodeFi
215/udp]$ mkfifo clientA
(base) [yufc@VM-12-12-centos:~/Core/BitCodeFi
ld/0215/udp]$ 
```

```
(base) [yufc@VM-12-12-centos:~/Core/BitCode
ield/0215/udp]$ 
```

```
(base) [yufc@VM-12-12-centos:~/Core/BitCodeField/0215/udp]
 ./udp_server 8080
[NORMAL] [1681353878] init udp server done ... Success
```

```
(base) [yufc@VM-12-12-centos:~/Core/BitCodeF
215/udp]$ mkfifo clientB
(base) [yufc@VM-12-12-centos:~/Core/BitCodeF
eld/0215/udp]$ ls
clientA      udp_client.cc
clientB      udp.log
Log.hpp      udp_server
Makefile     udp_server.cc
thread.hpp   udp_server.hpp
udp_client   udp_server.hpp-backup
(base) [yufc@VM-12-12-centos:~/Core/BitCodeF
eld/0215/udp]$ 
```

```
(base) [yufc@VM-12-12-centos:~/Core/BitCodeF
eld/0215/udp]$ 
```



腾讯云机器 [2] - yufc@VM-12-12-centos:~/Core/BitCodeField/0215/udp - Xshell 7 (Free for Home/School)

ssh://43.136.113.128:22

```
(base) [yufc@VM-12-12-centos:~/Core/BitCodeFi
ld/0215/udp]$ ./udp_client 127.0.0.1 8080 > c
lientA
please echo the command # nihao
please echo the command # I am A
please echo the command # yesA
please echo the command # I am AA
please echo the command #
```

```
(base) [yufc@VM-12-12-centos:~/Core/BitCode
ield/0215/udp]$ cat < clientA
127.0.0.1-42310 # nihao
127.0.0.1-36920 # nao
127.0.0.1-42310 # I am A
127.0.0.1-36920 # I am B
127.0.0.1-42310 # yesA
127.0.0.1-36920 # yesB
127.0.0.1-42310 # I am AA
127.0.0.1-36920 # I am BB
```

```
(base) [yufc@VM-12-12-centos:~/Core/BitCodeField/0215/udp]
 ./udp_server 8080
[NORMAL] [1681355203] init udp server done ... Success
[NORMAL] [1681355214] add new user: 127.0.0.1-42310
[NORMAL] [1681355214] push message to 127.0.0.1-42310
[NORMAL] [1681355218] add new user: 127.0.0.1-36920
[NORMAL] [1681355218] push message to 127.0.0.1-36920
[NORMAL] [1681355218] push message to 127.0.0.1-42310
[NORMAL] [1681355222] push message to 127.0.0.1-36920
[NORMAL] [1681355222] push message to 127.0.0.1-42310
[NORMAL] [1681355224] push message to 127.0.0.1-36920
[NORMAL] [1681355224] push message to 127.0.0.1-42310
[NORMAL] [1681355227] push message to 127.0.0.1-36920
[NORMAL] [1681355227] push message to 127.0.0.1-42310
[NORMAL] [1681355230] push message to 127.0.0.1-36920
[NORMAL] [1681355230] push message to 127.0.0.1-42310
[NORMAL] [1681355235] push message to 127.0.0.1-36920
[NORMAL] [1681355235] push message to 127.0.0.1-42310
[NORMAL] [1681355238] push message to 127.0.0.1-36920
[NORMAL] [1681355238] push message to 127.0.0.1-42310
```

```
(base) [yufc@VM-12-12-centos:~/Core/BitCodeF
eld/0215/udp]$ ./udp_client 127.0.0.1 8080 >
 clientB
please echo the command # i^Hnao
please echo the command # I am B
please echo the command # yesB
please echo the command # I am BB
please echo the command #
```

```
(base) [yufc@VM-12-12-centos:~/Core/BitCodeF
eld/0215/udp]$ cat < clientB
127.0.0.1-36920 # nao
127.0.0.1-42310 # I am A
127.0.0.1-36920 # I am B
127.0.0.1-42310 # yesA
127.0.0.1-36920 # yesB
127.0.0.1-42310 # I am AA
127.0.0.1-36920 # I am BB
```

这样，一个基于udp实现的，群聊模型就出来了

当然我们还可以思考
在服务端

如果我们维护一个message队列
然后把读和写也像客户端一样解耦

即，一个线程读message，放到队列中
然后另一个线程从队列中取message，广播给所有人

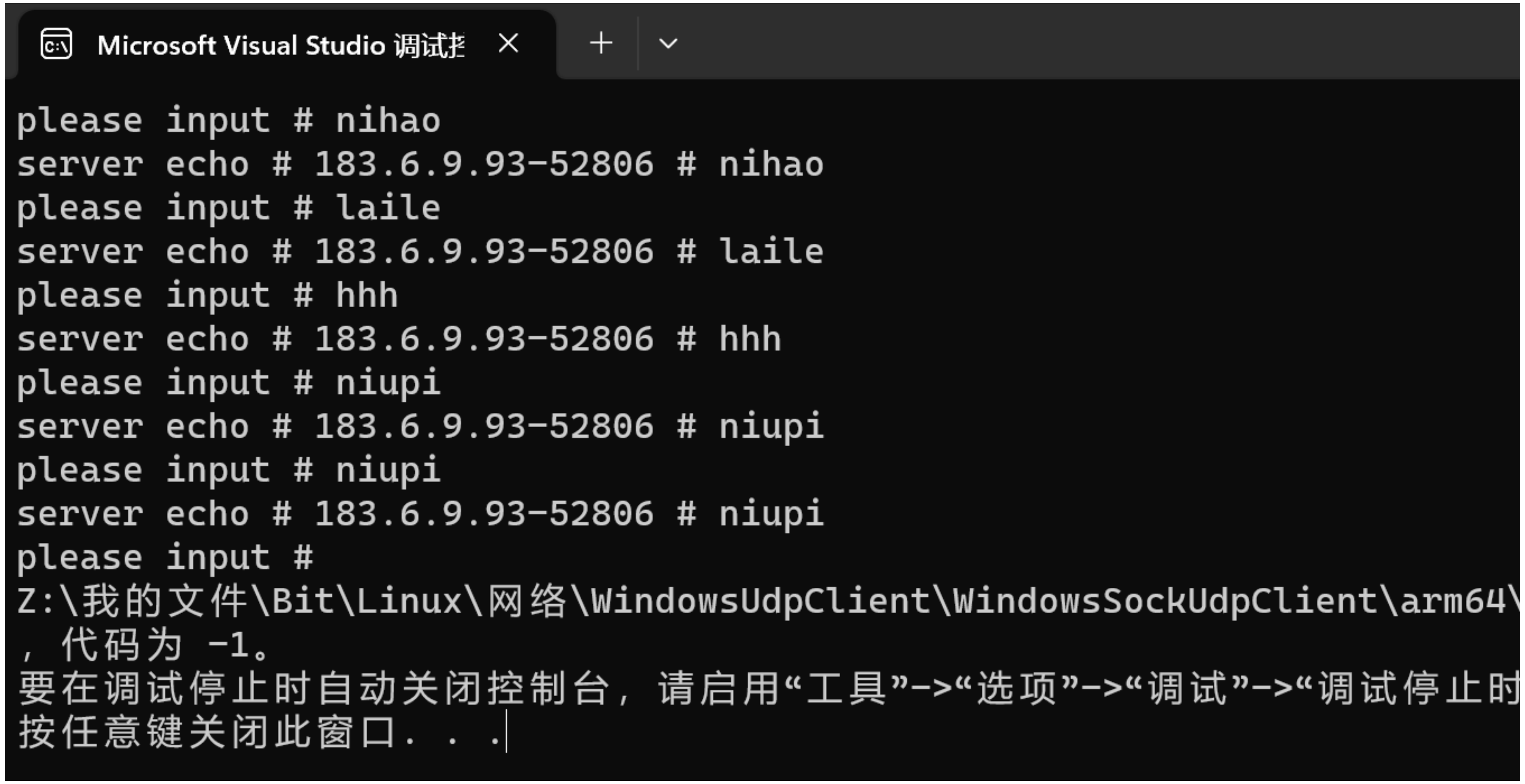
这个本质，就是，生产者消费者模型！

现在我想整一个，windows和linux下进行网络通信的udp代码

我们的服务端还是用linux的

我们只需要在win下弄个客户端就行了

详细内容请见vs2022上的代码



Microsoft Visual Studio 调试掐

```
please input # nihao
server echo # 183.6.9.93-52806 # nihao
please input # laile
server echo # 183.6.9.93-52806 # laile
please input # hhh
server echo # 183.6.9.93-52806 # hhh
please input # niupi
server echo # 183.6.9.93-52806 # niupi
please input # niupi
server echo # 183.6.9.93-52806 # niupi
please input #
Z:\我的文件\Bit\Linux\网络\WindowsUdpClient\WindowsSockUdpClient\arm64\
，代码为 -1。
要在调试停止时自动关闭控制台，请启用“工具”->“选项”->“调试”->“调试停止时
按任意键关闭此窗口. . .
```