

# 编程实验1：聊天程序的设计和实现

---

## 协议设计

本次实验采用TCP服务实现设计。

### 服务端

服务器实例化一个WSA，用来存储被WSAStartup函数调用后返回的Windows Sockets数据。

建立一个socket结构体，并绑定到一个特定的传输层服务。指定地址类型为AF\_INET（即IPv4），服务类型为SOCK\_STREAM（即TCP连接，提供序列化的、可靠的、双向连接的字节流，且支持带外数据传输），协议为0（系统自动选择）。

利用bind函数将IP地址和端口号和socket绑定到一起。

使用listen函数使socket进入监听状态，指定请求队列的最大程度为5。

初始化一个sockaddr\_in结构体处理网络通信地址，等待连接。

连接成功后开始实现通讯，最后要退出的时候关闭socket关闭WSA服务。

### 客户端

客户端实例化一个WSA，用来存储被WSAStartup函数调用后返回的Windows Sockets数据。

建立一个socket结构体，利用sockaddr\_in结构体处理网络通信地址，将套接字（即IP地址和端口号）绑定到创建的socket上。

使用connect函数远程连接指定网络地址，连接成功后实现通讯。

最后要退出的时候关闭socket关闭WSA服务。

## 程序设计

引入头文件winsock2.h，使用库函数，数据结构及一些定义。

引入头文件time.h，使用一些库函数获取当前时间。

由于版本问题，一些函数等在高版本的vs中不能使用，所以使用宏定义#define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS和#define \_CRT\_SECURE\_NO\_WARNINGS 1，解决报错问题。

链接Ws2\_32.lib库，显示加载 ws2\_32.dll。

### 服务端

首先实例化一个wsadata使用服务。

建立一个socket绑定到一个特定传输层服务。

使用bind函数将本地地址端口绑定到上面建立的socket。

使用listen函数开始监听。

使用accept函数接受一个特定socket请求等待队列中的连接请求。

连接成功后利用while循环实现轮流交谈。由于send函数和recv函数都是阻塞函数，所以循环内部先调用recv函数接收消息，再调用send函数发送消息，客户端与之相反。

其中使用字符串数组利用recv函数接受对方发来的消息，将接收到的内容输出到命令行窗口并输出接受时间，并且利用send函数将输入到字符串数组的内容发送出去，将发送时间显示到命令行窗口，且发送内容包括这个发送时间。

每一轮循环都判断发送内容和接受内容，只要由一方输入quit，双方程序跳出循环，结束服务，停止运行。

## 客户端

首先实例化一个wsadata使用服务。

建立一个socket绑定到一个特定传输层服务。

使用connect函数向一个特定socket发出连接请求。

连接成功后利用while循环实现轮流交谈。由于send函数和recv函数都是阻塞函数，所以循环内部先调用send函数发送消息，再调用recv函数接收消息，服务端与之相反。

连接成功后利用while循环实现轮流交谈，其中利用send函数将输入到字符串数组的内容发送出去，将发送时间显示到命令行窗口，且发送内容包括这个发送时间，使用字符串数组利用recv函数接受对方发来的消息，将接收到的内容输出到命令行窗口并输出接受时间。

每一轮循环都判断发送内容和接受内容，只要由一方输入quit，双方程序跳出循环，结束服务，停止运行。

## 程序实现

### 服务端

```
1  #define _WINSOCK_DEPRECATED_NO_WARNINGS
2  #define _CRT_SECURE_NO_WARNINGS 1
3  #include<iostream>
4  #include <winsock2.h>
5  #include<time.h>
6  #pragma comment(lib, "ws2_32.lib")
7  using namespace std;
8
9  int main()
10 {
11     //第一步：初始化
12     WSADATA wsaData; //实例化wsaData
13     WORD wVersionRequested;
14     wVersionRequested = MAKEWORD(2, 2); //指定版本
15     wVersionRequested=MAKEWORD(a,b)
16     WSStartup(wVersionRequested, &wsaData);
17
18     //第二步：建立一个socket，并绑定到一个特定的传输层服务
19     SOCKET sockSrv = socket(AF_INET, SOCK_STREAM, 0);
20     if (!sockSrv) cout << "建立socket失败" << endl;
21     else cout << "建立socket成功" << endl;
22
23     //第三步：bind，将一个本地地址绑定到指定的Socket
24     SOCKADDR_IN addrSrv;
25     addrSrv.sin_family = AF_INET;
26     addrSrv.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");
27     addrSrv.sin_port = htons(1234);
28     int cRes = bind(sockSrv, (SOCKADDR*)&addrSrv, sizeof(SOCKADDR));
29     if (SOCKET_ERROR == cRes) cout << "绑定失败" << endl;
30     else cout << "绑定成功" << endl;
```

```

30
31 //第四步: listen, 使socket进入监听状态, 监听远程连接是否到来
32 if (listen(sockSrv, 5))cout << "监听失败" << endl;
33 else cout << "监听成功" << endl;
34
35 //初始化一个客户端, 等待客户端的连接
36 SOCKADDR_IN addrClient;
37 int len = sizeof(sockaddr_in);
38 SOCKET sockConn = accept(sockSrv, (SOCKADDR*)&addrClient, &len);
39 if (!sockConn)cout << "连接失败" << endl;
40 else cout << "连接成功" << endl;
41
42 //成功后
43 cout << "1号" << endl;
44 cout << "-----" << endl;
45 while (1)
46 {
47     //接收, 给recvBuf加一个收到时间
48     char recvBuf[1024] = {};
49     recv(sockConn, recvBuf, 1024, 0);
50     cout << "2号: " << recvBuf;
51     time_t now_time = time(NULL);
52     tm* t_tm = localtime(&now_time);
53     cout << "                自己收到时间: " << asctime(t_tm);
54
55     //判断是否对方要退出
56     if (strlen(recvBuf) == 0)
57     {
58         cout << "要退出了" << endl;
59         break;
60     }
61
62     cout << "-----" << endl;
63
64     //发送
65     char sendBuf[1024] = {};
66     cout << "1号: ";
67     cin.getline(sendBuf, 1024);
68
69     //判断自己是否要退出
70     if (!strcmp("quit", sendBuf))
71     {
72         cout << "要退出了" << endl;
73         break;
74     }
75
76     //给sendBuf加一个发送时间
77     time_t now_time1 = time(NULL);
78     tm* t_tm1 = localtime(&now_time1);
79     cout << "                自己发送时间: " << asctime(t_tm1);
80     strcat(sendBuf, "                对方发送时间: ");
81     strcat(sendBuf, asctime(t_tm1));
82     send(sockConn, sendBuf, strlen(sendBuf), 0);
83
84     cout << "-----" << endl;
85 }
86 closesocket(sockConn);
87 closesocket(sockSrv);

```

```
88 WSACleanup();
89 }
```

## 客户端

```
1  #define _WINSOCK_DEPRECATED_NO_WARNINGS
2  #define _CRT_SECURE_NO_WARNINGS 1
3  #include<iostream>
4  #include <winsock2.h>
5  #include<time.h>
6  #pragma comment(lib, "ws2_32.lib")
7  using namespace std;
8
9  int main()
10 {
11     //第一步：初始化
12     WSADATA wsaData;
13     WORD wVersionRequested;
14     wVersionRequested = MAKEWORD(2, 2);
15     WSStartup(wVersionRequested, &wsaData);
16
17     //第二步：创建一个socket
18     SOCKET sockClient = {};
19     sockClient = socket(PF_INET, SOCK_STREAM, 0);
20     if (sockClient == SOCKET_ERROR)cout << "socket创建失败" << endl;
21     else cout << "socket创建成功" << endl;
22
23     //第三步：向一个特定的socket发出建连请求
24     sockaddr_in addrSrv;
25     addrSrv.sin_family = PF_INET;
26     addrSrv.sin_addr.S_un.S_addr = inet_addr("127.0.0.1");
27     addrSrv.sin_port = htons(1234);
28     int cRes = connect(sockClient, (SOCKADDR*)&addrSrv, sizeof(SOCKADDR));
29     if (SOCKET_ERROR == cRes)cout << "连接失败" << endl;
30     else cout << "连接成功" << endl;
31
32     cout << "2号" << endl;
33     cout << "-----" << endl;
34
35     while (1)
36     {
37         //发送
38         char sendBuf[1024] = {};
39         cout << "2号: ";
40         cin.getline(sendBuf, 1024);
41
42         //判断自己是否要退出
43         if (!strcmp("quit", sendBuf))
44         {
45             cout << "要退出了" << endl;
46             break;
47         }
48
49         //给sendBuf加一个发送时间
50         time_t now_time1 = time(NULL);
51         tm* t_tm1 = localtime(&now_time1);
52         cout << "                自己发送时间: " << asctime(t_tm1);
```

```

53     strcat(sendBuf, "                对方发送时间: ");
54     strcat(sendBuf, asctime(t_tm1));
55     send(sockClient, sendBuf, strlen(sendBuf), 0);
56
57     cout << "-----" << endl;
58
59     //接收, 给recvBuf加一个收到时间
60     char recvBuf[1024] = {};
61     recv(sockClient, recvBuf, 1024, 0);
62     cout << "1号: " << recvBuf;
63     time_t now_time = time(NULL);
64     tm* t_tm = localtime(&now_time);
65     cout << "                自己收到时间: " << asctime(t_tm);
66
67     //判断是否对方要退出
68     if (strlen(recvBuf) == 0)
69     {
70         cout << "要退出了" << endl;
71         break;
72     }
73
74     cout << "-----" << endl;
75 }
76
77 //关闭socket
78 closesocket(sockClient);
79 //结束使用socket, 释放socket dll资源
80 WSACleanup();
81 }

```

## 程序运行说明

### 运行说明

- 1号为服务端, 2号为客户端
- 一人一句, 客户端先说
- 每个人说完后都会将发送时间一起发送给对方, 并将发送时间输出到命令行
- 对方收到后, 会接收到发送时间并输出, 并将他的接收时间输出到命令行
- 最多能输入977个char型字符 (因为指定字符串长度为1024, 其中中文字符每个占两个char, 将时间和输入内容一起发送, 所以时间部分占了46)
- 任意一方输入quit, 两个都退出

### 服务端截图 (客户端主动退出)

```
Microsoft Visual Studio 调试控制台
建立socket成功
绑定成功
监听成功
连接成功
1号
-----
2号: hello, 服务端          对方发送时间: Sat Oct 23 19:46:59 2021
                        自己收到时间: Sat Oct 23 19:46:59 2021
-----
1号: hello, 客户端          自己发送时间: Sat Oct 23 19:47:14 2021
-----
2号: 我要退出了哦          对方发送时间: Sat Oct 23 19:47:30 2021
                        自己收到时间: Sat Oct 23 19:47:30 2021
-----
1号: 好的                    自己发送时间: Sat Oct 23 19:47:33 2021
-----
2号:                        自己收到时间: Sat Oct 23 19:47:37 2021
要退出了
C:\Users\86158\source\repos\socket1\Debug\socket1.exe (进程 16724) 已退出, 代码为 0。
要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口...
```

客户端截图 (客户端主动退出)

```
Microsoft Visual Studio 调试控制台
socket创建成功
连接成功
2号
-----
2号: hello, 服务端          自己发送时间: Sat Oct 23 19:46:59 2021
-----
1号: hello, 客户端          对方发送时间: Sat Oct 23 19:47:14 2021
                        自己收到时间: Sat Oct 23 19:47:14 2021
-----
2号: 我要退出了哦          自己发送时间: Sat Oct 23 19:47:30 2021
-----
1号: 好的                    对方发送时间: Sat Oct 23 19:47:33 2021
                        自己收到时间: Sat Oct 23 19:47:33 2021
-----
2号: quit                  自己发送时间: Sat Oct 23 19:47:33 2021
要退出了
C:\Users\86158\source\repos\socket2\Debug\socket2.exe (进程 10276) 已退出, 代码为 0。
要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口...
```

服务端截图 (服务端主动退出)

```
Microsoft Visual Studio 调试控制台
建立socket成功
绑定成功
监听成功
连接成功
1号
-----
2号: 你好, 服务端          对方发送时间: Sat Oct 23 19:50:24 2021
                        自己收到时间: Sat Oct 23 19:50:24 2021
-----
1号: 你好, 客户端          自己发送时间: Sat Oct 23 19:50:30 2021
-----
2号: 这次你来说退出吧      对方发送时间: Sat Oct 23 19:50:41 2021
                        自己收到时间: Sat Oct 23 19:50:41 2021
-----
1号: 好的                  自己发送时间: Sat Oct 23 19:50:50 2021
-----
2号: 说出来                对方发送时间: Sat Oct 23 19:50:54 2021
                        自己收到时间: Sat Oct 23 19:50:54 2021
-----
1号: quit
要退出了

C:\Users\86158\source\repos\socket1\Debug\socket1.exe (进程 19444) 已退出, 代码为 0。
要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口. . .
```

客户端截图（服务端主动退出）

```
Microsoft Visual Studio 调试控制台
socket创建成功
连接成功
2号
-----
2号: 你好, 服务端          自己发送时间: Sat Oct 23 19:50:24 2021
-----
1号: 你好, 客户端          对方发送时间: Sat Oct 23 19:50:30 2021
                        自己收到时间: Sat Oct 23 19:50:30 2021
-----
2号: 这次你来说退出吧      自己发送时间: Sat Oct 23 19:50:41 2021
-----
1号: 好的                  对方发送时间: Sat Oct 23 19:50:50 2021
                        自己收到时间: Sat Oct 23 19:50:50 2021
-----
2号: 说出来                自己发送时间: Sat Oct 23 19:50:54 2021
-----
1号:                        自己收到时间: Sat Oct 23 19:50:57 2021
要退出了

C:\Users\86158\source\repos\socket2\Debug\socket2.exe (进程 20060) 已退出, 代码为 0。
要在调试停止时自动关闭控制台, 请启用“工具”->“选项”->“调试”->“调试停止时自动关闭控制台”。
按任意键关闭此窗口. . .
```

## 总结

实验过程中由于错将流式类型SOCK\_STREAM写成数据报式类型SOCK\_DGRAM, 而且一直没看出来导致浪费很长时间, 所以还是要细心。

使用函数localtime和asctime报错, 由于目前版本不支持这样的函数, 所以要在前面加一个宏定义#define \_CRT\_SECURE\_NO\_WARNINGS 1。

使用函数inet\_addr报错, 也是由于目前版本不支持这样的函数, 所以要在前面加一个宏定义#define \_WINSOCK\_DEPRECATED\_NO\_WARNINGS。

助教在检查作业时问：“为什么你的服务端先说？”，我一开始没明白为什么这么问，后来助教问“那服务端和客户端区别是什么”，我当时没反应过来。回来之后理清思路想了一下，服务端为客户端服务，向客户端提供资源，所以我猜想助教的意思是应该让客户端先请求先说话（因为本次实验要实现聊天，所以也没想那么多）。最后我将服务端和客户端发送顺序调换了一下。