

实验 1：利用 Socket，编写一个聊天程序

姓名：赵元鸣

学号：2211757

班级：0850

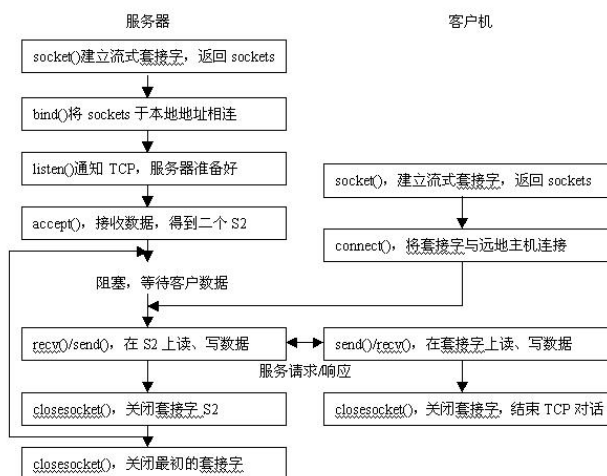
一. 实验要求

实验要求：（1）给出你聊天协议的完整说明。（2）利用 C 或 C++ 语言，使用基本的 Socket 函数完成程序。不允许使用 CSocket 等封装后的类编写程序。（3）使用流式 Socket 完成程序。（4）程序应有基本的对话界面，但可以不是图形界面。程序应有正常的退出方式。（5）完成的程序应能支持多人聊天，支持英文和中文聊天。（6）编写的程序应该结构清晰，具有较好的可读性。（7）现场演示。（8）提交程序源码、可执行代码和实验报告。

二. 聊天协议

协议设计

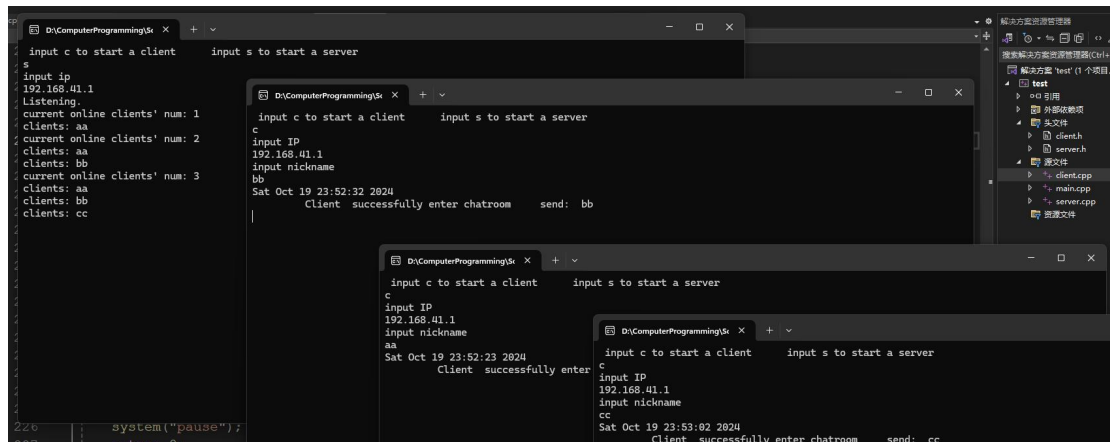
我用的是 tcp 协议，socket 为流式套接字。流式套接字（SOCK_STREAM）是一种网络编程接口，它提供了一种面向连接的、可靠的、无差错和无重复的数据传输服务。这种服务保证了数据按照发送的顺序被接收，使得数据传输具有高度的稳定性和正确性。通常用于那些对数据的顺序和完整性有严格要求的应用。通常由传输控制协议（TCP）来实现。TCP 协议通过建立连接、数据分包的编号和确认、以及重传机制等方式来确保数据的可靠传输。尽管这种服务提供了高度的可靠性，但它也可能导致较高的网络资源占用率。其工作流程图如下：



我的程序流程为：

(1) 服务器建立：通过 ip 和 port 指定，建立一个服务器，然后监听

(2) 客户端建立：客户端需要输入服务器的 ip 地址，然后规定自己相关信息比如 port 与昵称



```
input c to start a client      input s to start a server
s
input ip
192.168.41.1
listening.
current online clients' num: 1
clients: aa
current online clients' num: 2
clients: aa
clients: bb
current online clients' num: 3
clients: aa
clients: bb
clients: cc

input c to start a client      input s to start a server
c
input IP
192.168.41.1
input nickname
bb
Sat Oct 19 23:52:32 2024
Client successfully enter chatroom send: bb

input c to start a client      input s to start a server
c
input IP
192.168.41.1
input nickname
aa
Sat Oct 19 23:52:23 2024
Client successfully enter

input c to start a client      input s to start a server
c
input IP
192.168.41.1
input nickname
cc
Sat Oct 19 23:53:02 2024
Client successfully enter chatroom send: cc
```

(3) 消息发送：分为公聊和私聊，都是客户端将信息通过 socket 传递给服务器然后由服务器分发。

信息开头为已存在昵称，则判定为私聊；开头为“world”，则判定为公聊

(4) 人员查询：服务器会实时输出人员变动，客户端可也通过相关指令得知当前人员

(5) 退出：客户端输入 exit 后会自动结束程序，其他客户端和服务器会输出退出信息。

在 tcp 协议的利用上，

服务器和客户端都需要使用 `WSAStartup()` 函数来初始化。服务器端首先创建套接字 `socket()`，并使用 `bind()` 将其绑定到 IP 地址和端口，之后进入循环监听状态。服务器使用 `listen()` 将套接字设置为监听状态，并将等待队列的最大长度设置为 5；接着，使用 `accept()` 函数接受新的连接请求，并用一个套接字存放新创建的套接字。

客户端同样需要创建套接字并绑定服务器信息，然后使用 `connect()` 连接远程服务器。客户端使用 `send()` 函数发送数据，并判断是否是退出信息。

服务器端和客户端都使用 `CreateThread()` 创建一个专门用来接收消息的线程，并使用 `send()` 和 `recv()` 函数进行消息的发送和接收。在通话过程中，一旦检测到退出指令，则使用 `CloseHandle()` 关闭线程，使用 `closesocket()` 关闭套接字，最终使用 `WSACleanup()` 释放 Socket DLL 资源。

三. 程序设计

主要实现了 `server.cpp` 与 `client.cpp` 两个文件，通过 `main.cpp` 统一运行。介绍主要代码：

1. Server

实现了如下函数

```
void setSIP(char* p);
```

```

bool isName(char* a);
bool cmpStr(char* a, char* b);
bool wantCList(char* a);
bool ClientExit(char* a);
bool isWorldChat(char* a);
DWORD WINAPI recMsg(LPVOID arg);
int serverStart();

```

以及存储 client 发送信息的结构体。

```

struct CInfo {
    int num;
    char name[10];
    bool online = false;
    int len;
};

```

RecMsg 函数处理 socket 信息（代码过长，这里讲述流程）：

- 接收消息：使用 `recv()` 函数从客户端接收数据，如果接收失败，则输出错误信息。
- 处理退出信息：如果接收到的消息以 'e' 开头，表示客户端已退出：解析退出信息，提取客户端的名称。更新在线客户端计数，输出退出提示。向其他在线客户端发送退出通知，并关闭该客户端的套接字。
- 处理获取在线客户端列表的请求：如果消息请求获取在线用户列表（通过 `wantCList()` 判断），构建一个包含所有在线用户的消息，并发送给请求的客户端。
- 处理聊天消息：提取时间戳和消息内容，解析发送者的名称。如果消息是全局聊天（通过 `isWorldChat()` 判断），则将消息广播给所有在线客户端。如果消息是针对特定用户，查找该用户并将消息发送给其套接字。如果目标用户不存在，通知发送者该用户不存在。
- 消息格式化：在发送和接收的消息中，使用字符数组和特定的格式（如长度前缀）组织数据，确保服务器能够正确解析。

还有必要的判断函数 `wantCList(char* a)`：用于检查接收到的消息是否是请求在线客户端列表的指令。开头，返回 `true` 表示是请求，`false` 表示不是。`isWorldChat(char* a)` 用于判断消息是否是全局聊天消息。

`ClientExit(char* a)`：检查接收到的消息是否表示某个客户端的退出指令。

```

bool ClientExit(char* a) {
    int l1 = int(a[25]);
    int l2 = int(a[25 + l1 + 1]);
    int len = 0;

```

```

while (a[len] != '\0')
    len++;
    if ((len == (25 + 1 + 11 + 1 + 12 + 4)) && a[25 + 1 + 12 + 1 + 2] == 'e' && a[25
+ 1 + 11 + 1 + 2 + 1] == 'x' && a[25 + 1 + 11 + 1 + 2 + 2] == 'i' && a[25 + 1 + 11
+ 1 + 2 + 3] == 't')
        return true;
    else
        return false;
}

```

2. Client

Client 实现了如下函数：

```

void PrintClients(char* c_list);
char* setIP();
int charLen(char* a);
bool isExitting(char* a);
bool OtherExitting(char* a);
int clientStart();
DWORD WINAPI RecMsg(LPVOID);

```

RecMsg 函数如下：

```

DWORD WINAPI RecMsg(LPVOID) {
    while (1) {
        ::memset(c_rev_buffer, 0, c_buffer_size);
        if (recv(local_sock, c_rev_buffer, c_buffer_size, 0) < 0) {
            cout << "step 1 fail. Errcode: " << SOCKET_ERROR << ".\n";
            system("pause");
        }
        else if (c_rev_buffer[0] == 's')
            PrintClients(c_rev_buffer);
        else if (OtherExitting(c_rev_buffer)) {
            int len = int(c_rev_buffer[1]);
            char* exittingOne = new char[len + 1];
            for (int i = 0; i < len; i++)
                exittingOne[i] = c_rev_buffer[i + 2];
            exittingOne[len] = '\0';
            cout << "Client " << exittingOne << " exit from chat.\n";
        }

        else {
            char t_str[26];
            int idx = 0;
            for (; idx < 25; idx++)

```

```

        t_str[idx] = c_rev_buffer[idx];
        t_str[idx] = '\0';
        int len = int(c_rev_buffer[idx]);
        idx += 1;
        char* name = new char[len + 1];
        for (; idx < 26 + len; idx++)
            name[idx - 26] = c_rev_buffer[idx];
        name[idx - 26] = '\0';
        while (c_rev_buffer[idx] == '\0')
            idx++;
        cout << t_str << "\t Client " << name << "\t send:\t";
        while (c_rev_buffer[idx] != '\0') {
            cout << c_rev_buffer[idx];
            idx++;
        }
        cout << endl;
    }
}
return 0;
}

```

Client 调用函数中对 server 信息的判断流程如下:

```

if (connect(local_sock, (SOCKADDR*)&rem_adr, sizeof(rem_adr)) != SOCKET_ERROR)
{
    send(local_sock, sendName, int(usr_name[0] + 2), 0);
    HANDLE rcv_thread = CreateThread(NULL, NULL, RecMsg, NULL, 0, NULL);
    while (1) {
        memset(c_input_buffer, 0, c_buffer_size);
        char input[c_buffer_size];
        cin.getline(input, c_buffer_size);

        if (!strcmp(input, "getc_list\0")) {
            send(local_sock, input, 11, 0);
        }
        else {
            if (isExiting(input)) {
                char tmp[c_buffer_size];
                memset(tmp, 0, c_buffer_size);
                tmp[0] = 'e';
                strcat(tmp, usr_name);
                strcat(tmp, "exit\0");
                send(local_sock, tmp, sizeof(tmp), 0);
                cout << "Exit done.\n";
                closesocket(local_sock);
            }
        }
    }
}

```

```

        closesocket(rem_sock);
        WSACleanup();
        system("pause");
        return 0;
    }
    else {
        int i = 0;
        while (input[i] != ':' && input[i] != '\0')
            i++;
        char* objC = new char[i + 2];
        objC[0] = char(i);
        for (int j = 1; j <= i; j++)
            objC[j] = input[j - 1];
        objC[i + 1] = '\0';
        i += 1;
        int j = i;
        while (input[i] != '\0') {
            c_input_buffer[i - j] = input[i];
            i += 1;
        }
        c_input_buffer[i - j] = '\0';
    }

```

```

        time_t now = time(0);
        char* timer = ctime(&now);
        memset(c_send_buffer, 0, c_buffer_size);
        strcat(c_send_buffer, timer);
        strcat(c_send_buffer, objC);
        strcat(c_send_buffer, usr_name);
        strcat(c_send_buffer, c_input_buffer);
        send(local_sock, c_send_buffer, c_buffer_size, 0);
        memset(c_send_buffer, '\0', c_buffer_size);
        memset(c_input_buffer, '\0', c_buffer_size);
    }
}
}
else {
    cout << "step 5 fail: " << SOCKET_ERROR << ".\n";
    return 0;
}

```

在这个客户端主函数的判断流程中，主要对输入的信息进行处理和分类，具体流程如下：

1. 连接到服务器：使用 `connect()` 函数尝试与远程地址建立连接。如果连接成功，程序继续执行，否则输出错误信息并返回。

2. 发送用户名：连接成功后，使用 `send()` 函数将用户名发送到服务器。
3. 创建接收线程：调用 `CreateThread()` 创建一个线程，用于接收来自服务器的消息，这样主线程可以专注于处理用户输入。
4. 进入主输入循环：进入一个无限循环，等待用户输入消息。
5. 处理用户输入：
获取客户端列表：如果用户输入 "getc_list"，调用 `send()` 发送该请求给服务器，等待服务器返回在线客户端列表。

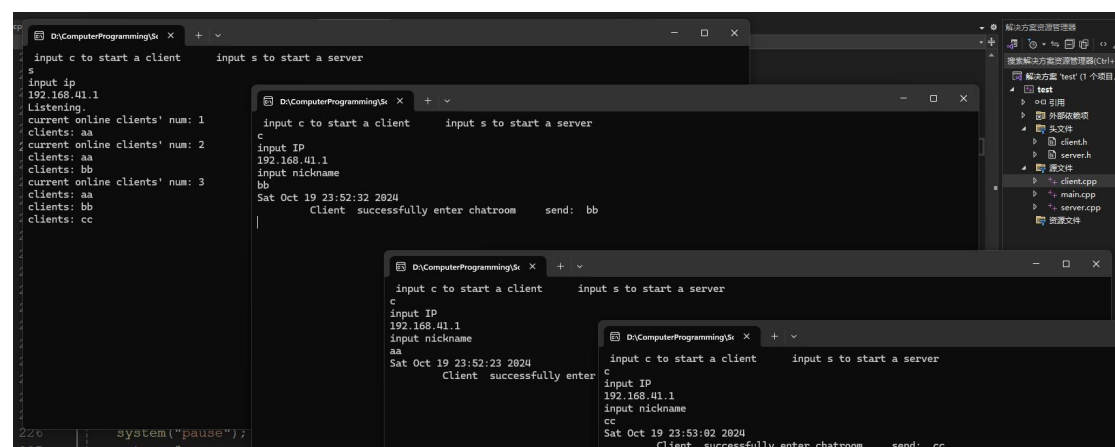
处理退出指令：如果检测到用户输入的内容表示退出（通过 `isExiting()` 函数判断），则构造一个包含退出信息的信息，发送给服务器，并关闭套接字和清理资源。程序输出退出信息并结束。

处理聊天消息：如果输入既不是获取客户端列表请求，也不是退出指令，程序会解析输入内容：

- 查找输入中的冒号：位置，用于分割接收对象和消息内容。
- 构建 `objC` 字符串，表示目标客户端的名称，并将其长度存储在第一个字符中。
- 提取消息内容到 `c_input_buffer`。
- 获取当前时间，格式化为字符串 `timer`。
- 组合时间戳、目标客户端名称、用户名和消息内容到 `c_send_buffer` 中。
- 发送组合后的消息到服务器。

三. 演示

服务端与客户端创建：



```
input c to start a client      input s to start a server
s
input IP
192.168.41.1
Listening.
current online clients' num: 1
clients: aa
current online clients' num: 2
clients: bb
current online clients' num: 3
clients: aa
clients: bb
clients: cc

c
input IP
192.168.41.1
input nickname
bb
Sat Oct 19 23:52:32 2024
Client successfully enter chatroom  send: bb

input c to start a client      input s to start a server
c
input IP
192.168.41.1
input nickname
cc
Sat Oct 19 23:52:23 2024
Client successfully enter

input c to start a client      input s to start a server
c
input IP
192.168.41.1
input nickname
cc
Sat Oct 19 23:53:02 2024
Client successfully enter chatroom  send: cc
```

私聊： client bb 指定 cc 为对象，然后将发送时间，发件人等信息呈现给 cc

```
D:\ComputerProgramming\Sc x + v
input c to start a client      input s to start a server
c
input IP
192.168.41.1
input nickname
aa
Sat Oct 19 23:52:23 2024
Client successfully enter chatroom    send: aa
bb:你好

D:\ComputerProgramming\Sc x + v
input c to start a client      input s to start a server
c
input IP
192.168.41.1
input nickname
bb
Sat Oct 19 23:52:32 2024
Client successfully enter chatroom    send: bb
Sun Oct 20 00:24:45 2024
Client aa        send: 你好
```

公屏聊天: cc 发送

```
D:\ComputerProgramming\Sc x + v
input c to start a client      input s to start a server
c
input IP
192.168.41.1
input nickname
cc
Sat Oct 19 23:53:02 2024
Client successfully enter chatroom    send: cc
world:hello guys.
Sun Oct 20 00:25:24 2024
Client cc        send: hello guys.
```

aa 与 bb 以及客户端输出公屏聊天

```
input c to start a client      input s to start a server
s
input ip
192.168.41.1
listening.
current online clients' num: 1
clients: aa
current online clients' num: 2
clients: aa
clients: bb
current online clients' num: 3
clients: aa
clients: bb
clients: cc
Client cc        send world msg At: Sun Oct 20 00:25:24 2024
hello guys.
```



```
D:\ComputerProgramming\Sc x + v
input c to start a client      input s to start a server
c
input IP
192.168.41.1
input nickname
bb
Sat Oct 19 23:52:32 2024
Client successfully enter chatroom      send: bb
Sun Oct 20 00:24:45 2024
Client aa      send: 你好
Sun Oct 20 00:25:24 2024
Client cc      send: hello guys.
```

```
D:\ComputerProgramming\Sc x + v
input c to start a client      input s to start a server
c
input IP
192.168.41.1
input nickname
aa
Sat Oct 19 23:52:23 2024
Client successfully enter chatroom      send: aa
bb:你好
Sun Oct 20 00:25:24 2024
Client cc      send: hello guys.
```

aa 退出: aa 输入 exit 退出

```
input c to start a client      input s to start a server
c
input IP
192.168.41.1
input nickname
aa
Sat Oct 19 23:52:23 2024
Client successfully enter chatroom      send: aa
bb:你好
Sun Oct 20 00:25:24 2024
Client cc      send: hello guys.
exit
Exit done.
step 1 fail. Errcode: -1.
请按任意键继续. . . 请按任意键继续. . .
```

其他客户端与 server 反馈:

Server:

```
Client aa      left
there are 2      client(s) left
```

Client:

```
input c to start a client    input s to start a server
c
input IP
192.168.41.1
input nickname
bb
Sat Oct 19 23:52:32 2024
    Client successfully enter chatroom    send:  bb
Sun Oct 20 00:24:45 2024
    Client aa        send:  你好
Sun Oct 20 00:25:24 2024
    Client cc        send:  hello guys.
Client aa exit from chat.
```