

计算机网络第一次实验报告

- 姓名：刘沿辰
- 学号：2012543
- 年级：2020级

实验要求

1. 使用流式Socket，设计一个两人聊天协议，要求聊天信息带有时间标签。请完整地说明交互消息的类型、语法、语义、时序等具体的消息处理方式
2. 对聊天程序进行设计。给出模块划分说明、模块的功能和模块的流程图
3. 在Windows系统下，利用C/C++对设计的程序进行实现。程序界面可以采用命令行方式，但需要给出使用方法。编写程序时，只能使用基本的Socket函数，不允许使用对socket封装后的类或架构
4. 对实现的程序进行测试
5. 撰写实验报告，并将实验报告和源码提交至<http://cc-backend.nankai.edu.cn/>

协议设计

本程序是一个基于C++实现的网络聊天小程序，由于只支持两人聊天，所以协议的设计比较简单，设计如下：

MAGIC	DATA&TIME	INFORMATION
幻数，用于校验 消息类型	时间戳，存储了年 月日时分秒信息	发送的信息本身

前八位是MAGIC幻数，用于校验消息以及确认消息类别。目前，我的设定中幻数 "abcdefgh" 代表这条消息是一个带有时间戳的文字信息，并可以按照上图的方式解码。（由于目前该聊天软件仅支持文字信息，所以幻数只有这一个，作用也仅仅是增加软件扩展性，以及校验信息有没有传输错误）

之后是一个存储了年月日等信息的时间戳，长度为19位，实现代码如下：

```
to_string(tm_t->tm_year + 1900) + '.' +  
to_string(tm_t->tm_mon + 1) + '.' +  
to_string(tm_t->tm_mday) + "-" +  
to_string(tm_t->tm_hour) + ':' +  
to_string(tm_t->tm_min) + ':' +  
to_string(tm_t->tm_sec)
```

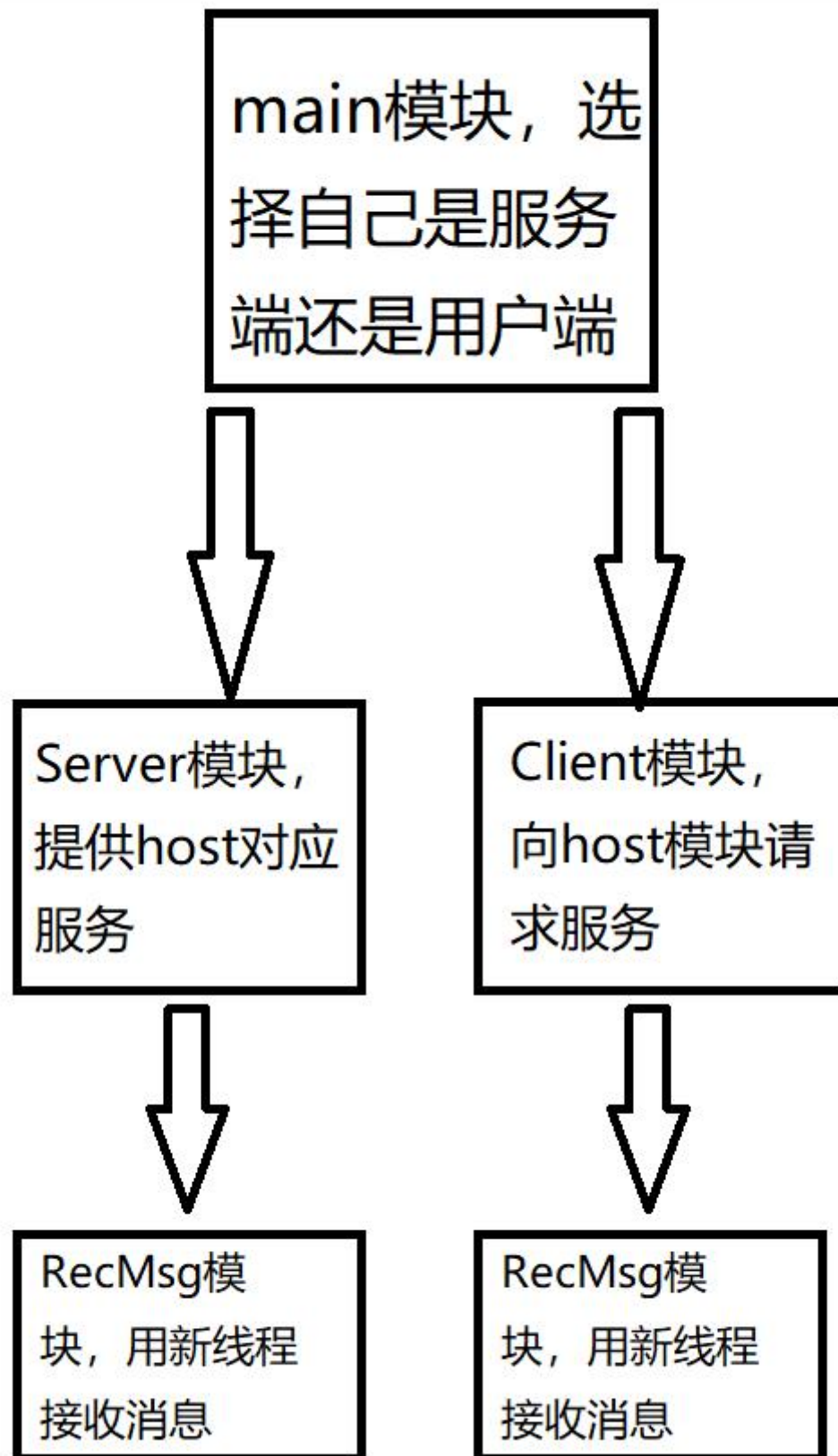
由此可以得到一个由这些信息拼成的字符串

最后一部分以 , Msg: 开头, 后面跟用户输入的字符串

在收到消息时, 程序会先用前八位校验消息类型, 然后读取时间戳, 最后以 , Msg: 来分割出用户消息, 完成消息的解码。

各模块功能

程序总共分为main, Server, Client和RecMsg四个模块, 调用关系如下:



1. main模块仅用于分支选择，用户仅需使用同样的exe，即可选择主机或客机来使用聊天服务。主要分支代码如下：

```
bool is_host = false, is_client = false;  
string idt = "";
```

```

while (!is_host && !is_client) {
    cin >> idt;
    if (idt == "host") {
        cout << "You are the host!" << endl;
        is_host = true;
    }
    else if (idt == "client") {
        cout << "You are the client!" << endl;
        is_client = true;
    }
    else {
        cout << "Invalid input, please try again ( host / client)" <<
endl;
    }
}

```

2. Server提供host相关服务，即绑定端口ID，接收并处理客户端请求等。
前期的处理流程主要是

创建socket->绑定ID和端口->接收用户端请求->建立连接->开始聊天

聊天时主要工作的代码如下：

```

// Receive message
thread in_out[1];
in_out[0] = thread(RecMsg, clientSocket);

// Send message
while (true) {
    string data;
    getline(cin, data);
    time_t now = time(NULL);
    tm* tm_t = localtime(&now);
    data = "abcdefgh" +
        to_string(tm_t->tm_year + 1900) + '.' +
        to_string(tm_t->tm_mon + 1) + '.' +
        to_string(tm_t->tm_mday) + "-" +
        to_string(tm_t->tm_hour) + ':' +
        to_string(tm_t->tm_min) + ':' +
        to_string(tm_t->tm_sec) +

```

```

        ", Msg: " + data;
    const char* sendData;
    sendData = data.c_str();
    send(clientSocket, sendData, strlen(sendData), 0);
}

```

可以看到代码新开了一个线程调用RecMsg模块接收消息，而用主线程来发送消息

3. Client模块即为用户模块，可以向Server发出服务申请并获取聊天服务。整体流程和Server模块非常相近，不过前期处理流程变为：

创建socket->绑定ID和端口->向host发送服务请求->建立连接->开始聊天

后文的发送消息和接收消息部分和Server完全一致

4. 最后是一个RecMsg模块，这个模块被Server和Client都调用了。由于接收消息和发送消息都是在死循环中完成，并且等待消息时都是卡住进程，所以这部分的主要功能就是打开一个新线程，然后用这个线程内的循环来接收消息，接收消息部分如下：

```

char msg[200]; // save message
while (true) {
    int num = recv(soc, msg, 200, 0);
    // 判断消息是否为空
    msg[num] = '\0';
    // 检查幻数
    string magic = "abcdefgh";
    string t_magic = "";
    for (int i = 0; i < 8; i++) {
        t_magic += msg[i];
    }
    if (magic == t_magic) {
        int pos = 0;
        for (int i = 8; i < num; i++) {
            if (msg[i] == 'M') {
                if (msg[i + 1] == 's') {
                    pos = i + 5;
                    break;
                }
            }
        }
        if (pos < num) {

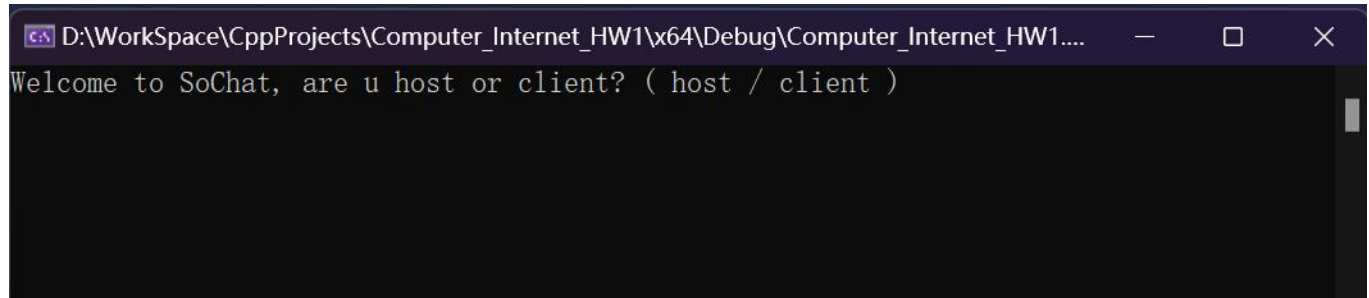
```

```
        cout << msg << endl;  
    }  
}  
}
```

程序最开始会检查幻数是否正确，然后过滤时间戳检查消息，最后输出非空消息。

程序界面展示及运行说明

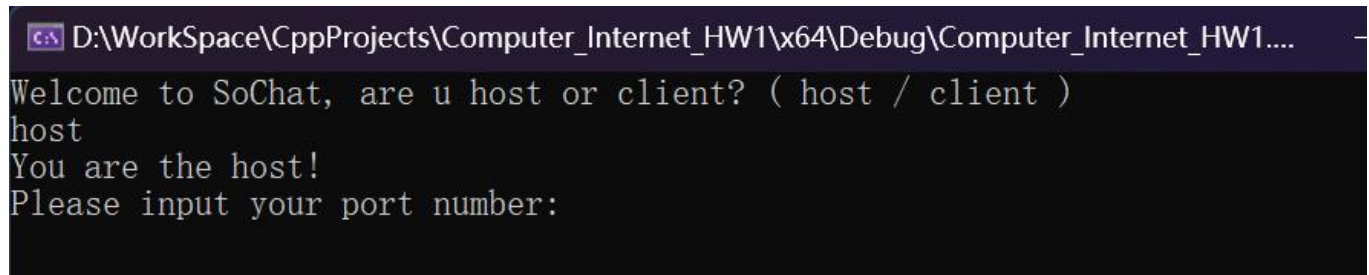
程序开始界面，选择自己是主机还是客户机（输入host或client）



```
D:\Workspace\CppProjects\Computer_Internet_HW1\x64\Debug\Computer_Internet_HW1....  
Welcome to SoChat, are u host or client? ( host / client )
```

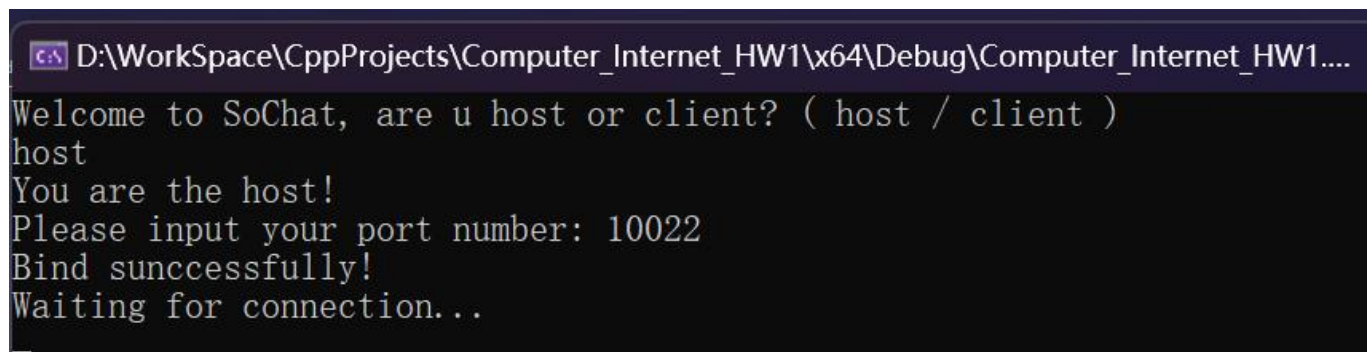
若输入错误，系统会让用户重新输入。

此处以主机为例演示，于是输入host



```
D:\Workspace\CppProjects\Computer_Internet_HW1\x64\Debug\Computer_Internet_HW1....  
Welcome to SoChat, are u host or client? ( host / client )  
host  
You are the host!  
Please input your port number:
```

接下来输入想要绑定的接口号进行bind



```
D:\Workspace\CppProjects\Computer_Internet_HW1\x64\Debug\Computer_Internet_HW1....  
Welcome to SoChat, are u host or client? ( host / client )  
host  
You are the host!  
Please input your port number: 10022  
Bind sunccessfully!  
Waiting for connection...
```

可以看到绑定成功后会提示等待用户机

打开另一个窗口，选择client，同样输入端口号

```
D:\Workspace\CppProjects\Computer_Internet_HW1\x64\Debug\Computer_Internet_HW...  
Welcome to SoChat, are u host or client? ( host / client )  
client  
You are the client!  
Please input your port number: 10022
```

然后连接成功建立，显示Accept

```
D:\Workspace\CppProjects\Computer_Internet_HW1\x64\Debug\Computer_Internet_HW1.exe  
Welcome to SoChat, are u host or client? ( host / client )  
host  
You are the host!  
Please input your port number: 10022  
Bind sunccessfully!  
Waiting for connection...  
Accept: 127.0.0.1  
_
```

接下来就可以互相发消息了

```
D:\Workspace\CppProjects\Computer_Internet_HW1\x64\Debug\Computer_Internet_HW1....  
Welcome to SoChat, are u host or client? ( host / client )  
client  
You are the client!  
Please input your port number: 10022  
Connected!  
abcdefgh2022.10.22-22:45:57, Msg: 你好!  
很高兴认识你!  
我是xxx  
来自xx省xx市  
abcdefgh2022.10.22-22:46:39, Msg: 我是yyy  
abcdefgh2022.10.22-22:46:45, Msg: 来自yy省yy市
```

主机端也能收到消息


```
C:\> D:\WorkSpace\CppProjects\Computer_Internet_HW1\x64\Debug\Computer_Internet_HW1.exe
Welcome to SoChat, are u host or client? ( host / client )
host
You are the host!
Please input your port number: 10022
Bind sunccessfully!
Waiting for connection...
Accept: 127.0.0.1
你好!
abcdefgh2022.10.22-22:46:4, Msg: 很高兴认识你!
abcdefgh2022.10.22-22:46:24, Msg: 我是xxx
abcdefgh2022.10.22-22:46:35, Msg: 来自xx省xx市
我是yyy
来自yy省yy市
```

实验过程中遇到的问题及分析

1. 最开始我编写程序的设想是分开写一个Server.exe和一个Client.exe，但是后来我意识到这样做一方面不方便链接调试，并且如果有人来用这个聊天软件，一方需要下载Server而另一方需要下载Client，这是一件很不合理的事情，于是我将这些内容内置，用户使用相同的.exe，然后在使用时区分Server和Client即可。
2. 最开始实现的程序版本不支持多线程，于是问题凸显：发消息就不能收消息，收消息就不能发消息。这种结构最好的结果就是收一句发一句，每方每次只能发一条消息。这依然是很不符合逻辑的设计，于是我将发消息和收消息抽象为两个模块，并使用多线程来并行执行两个模块，对发送消息和收消息去耦合化。
3. 在我编写程序时，最开始随便写的端口很容易被占用，后来发现了三件事：

- (1) 10000以上的端口很少被占用，随便挑基本都可以
 - (2) 1024以下的端口就不要用了.....已经被操作系统占有了
 - (3) 端口下标最大为65535，但是超过也没关系，系统会自动对输入的数据取余，也就是说若访问65540号端口，实际上访问的是 $65540 \% 65535 = 5$ 号端口。
4. 网络编程中很多工作是不确定的，并且是需要并行的。我们在设计网络应用的时候，不同模块之间的耦合程度以及相互访问的关系必须着重考量，否则将难以设计出好用的网络程序。