计算机网络实验报告3-1

姓名：高祎珂　　学号：2011743

实验要求

利用**数据报套接字**在用户空间实现面向连接的可靠数据传输，功能包括：**建立连接、差错检测、确认重传。**流量控制采用**停等机制，完成给定测试文件的传输。**

功能设计

1、数据报套接字UDP:

UDP是User Datagram Protocol的简称,中文名是用户数据报协议,是OSI参考模型中的传输层协议,它是一种无连接的传输层协议,提供面向事务的简单不可靠信息传送服务.

2、建立连接:

根据TCP的三次握手和四次挥手原则，进行发送端与接收端的连接和断连。

3、差错检测

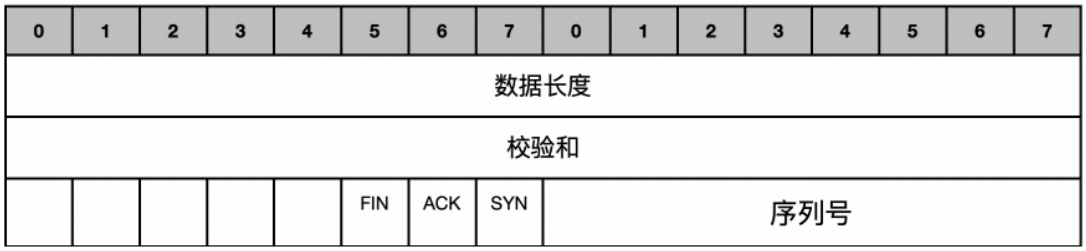
计算校验和，实现可靠数据传输。本次实验设计数据的UCP报文头是分开传输的，在握手，挥手期间，只传UDP报文头，在数据传输过程中，要加入数据段计算校验和。

4、确认重传

使用rdt3.0实现，设计一个定时器，根据序列号确认，以及超时重传。超时重传指的是在发送数据报文段后开始计时，到等待确认应答到来的那个时间间隔。如果超过这个时间间隔，仍未收到确认应答，发送端将进行数据重传。

协议设计

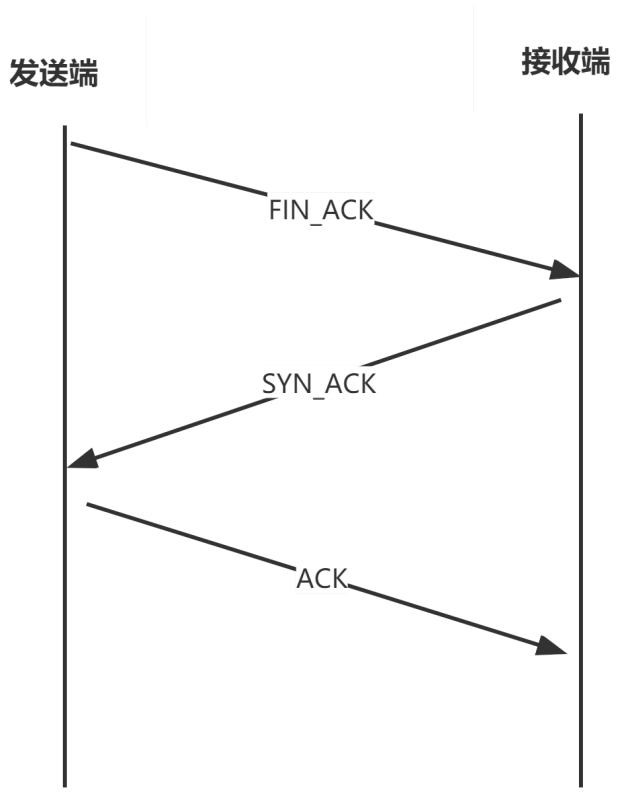
- **报文格式**



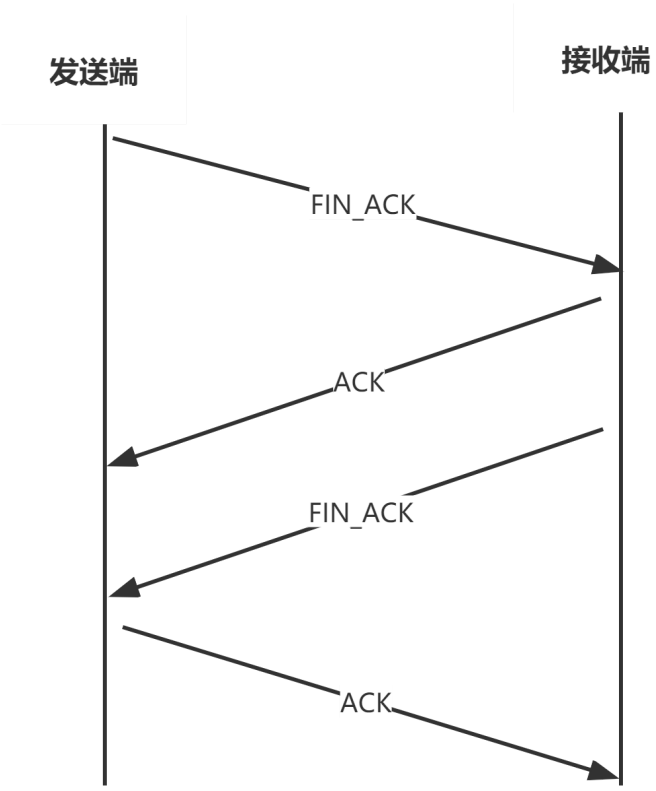
前16位为数据长度，用于记录数据区大小，17-32位位校验和，用于检验传输的正确性，33-40为标志位，40-48位为传输的数据包的序列号。

- **连接与断开**

- **三次握手**



- **四次挥手**



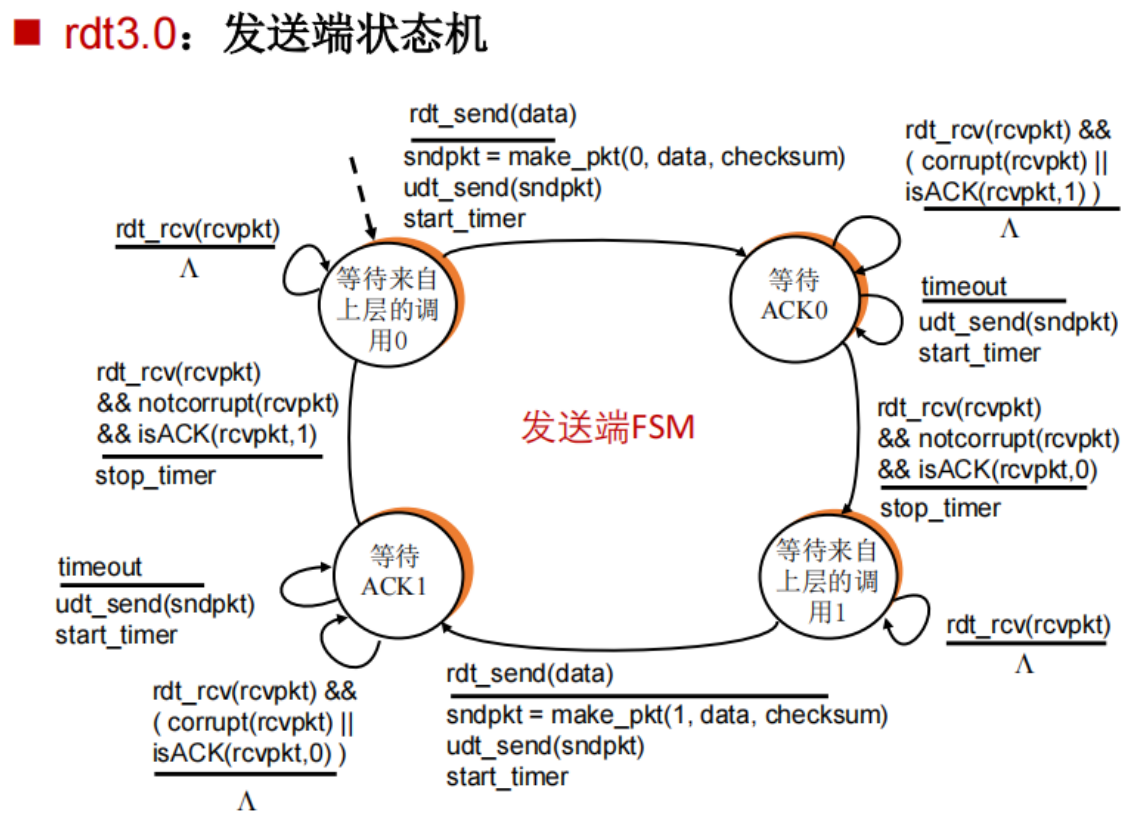
- **数据传输**

  数据传输发怂端和接收端军参照rdt3.0进行实现。数据在传输时，将一个文件分为数个包进行分段传输，每个包内容为数据头+数据，且考虑到文件内容较大，每次只读取文件的一部分到缓存进行传输。

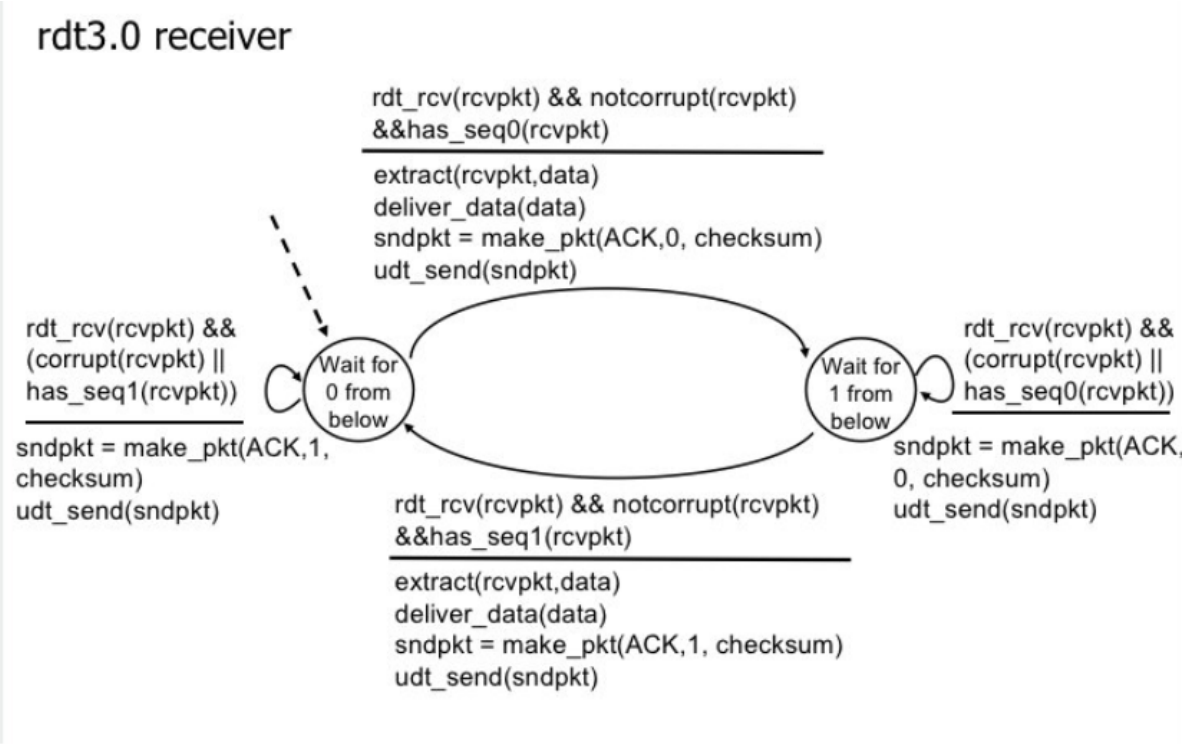
  发送端在传输时，需要接受到上一个发送包序号的ACK=1且确认无误后才能发送下一个数据包，若超时或者校验和不对则进行重传，如果未收到想要的数据包则等待；接收端接收到了一个数据包，先要进行校验，如果检查无误，则向发送放返回该序列号的ACK=1。

发送端状态机

使用rdt3.0实现可靠数据传输



接收端状态机



代码实现

计算校验和

```
u_short cksum(u_short* UDP_header, int size)
{
    // size以字节计数，但是这里要用16byte    u_short
    // 这里使用的函数都是以字节计数的
    unsigned short* a = (unsigned short*)UDP_header;
    unsigned int sum = 0;
    for (int i = 0; i < size / 2; i++) {
        sum += (unsigned int)a[i];
        if (sum & (0xFFFF0000)) {
            sum &= 0xFFFF;
            sum++;
        }
    }
    return ~(((unsigned short)sum) & 0XFFFF);
}
```

数据头

```
struct HEADER
{
    u_short sum = 0;//校验和 16位
    u_short datasize = 0;//所包含数据长度 16位
    unsigned char flag = 0;
    //八位，使用后三位，排列是FIN ACK SYN
    unsigned char SEQ = 0;
    //八位，传输的序列号，0~255，超过后mod
    HEADER() {
        sum = 0;//校验和 16位
        datasize = 0;//所包含数据长度 16位
```

```
        flag = 0;
        //八位，使用后三位，排列是FIN ACK SYN
        SEQ = 0;
    }
};
```

发送端发包

```
void send_package(SOCKET& socketClient, SOCKADDR_IN& servAddr, int& servAddrlen,
char* message, int len, int& order) {
    HEADER header;
    char* buffer = new char[MAXSIZE + sizeof(header)];
    for (int i = 0; i < MAXSIZE + sizeof(header); i++) {
        buffer[i] = 0;
    }
    header.datasize = len;
    header.SEQ = unsigned char(order);//序列号
    memcpy(buffer, &header, sizeof(header));
    memcpy(buffer + sizeof(header), message, sizeof(header) + len);
    u_short check = cksum((u_short*)buffer, sizeof(header) + len);//计算校验和
    header.sum = check;
    memcpy(buffer, &header, sizeof(header));
    sendto(socketClient, buffer, len + sizeof(header), 0, (sockaddr*)&servAddr,
servAddrlen);//发送
    cout << "Send message " << len << " bytes!" << " FLAG:" <<
state[int(header.flag)] << " SEQ:" << int(header.SEQ) << " CHECKSUM:" <<
int(header.sum) << endl;
    clock_t start = clock();//记录发送时间
    //接收ack等信息
    while (true) {
        u_long mode = 1;
        ioctlsocket(socketClient, FIONBIO, &mode);
        // 超时机制
        int times = 0;
        while (times < 5 && recvfrom(socketClient, buffer, MAXSIZE +
sizeof(header), 0, (sockaddr*)&servAddr, &servAddrlen) <= 0)
        {
            // 未收到消息，超时重传
            if (clock() - start > MAX_TIME)
            {
                sendto(socketClient, buffer, len + sizeof(header), 0,
(sockaddr*)&servAddr, servAddrlen);//发送
                cout << "TIME OUT! ReSend message " << len << " bytes! FLAG:" <<
state[int(header.flag)] << " SEQ:" << int(header.SEQ) << endl;
                times++;
                start = clock();//记录发送时间
            }
        }
        //cout << "收到了消息" << endl;
        if (times == 5) {
            cout << "发送出错" << endl;
            return;
        }
        // 得到了数据包
        memcpy(&header, buffer, sizeof(header));//缓冲区接收到信息，读取
```

```
        //检查是否是想要的ACK以及差错检验
        //cout << header.SEQ <<"      "<< header.flag << "      " <<
cksum((u_short*)buffer, sizeof(header) + len) << endl;
        if (header.SEQ == u_short(order) && header.flag == ACK)
        {
            //修改flag，检查校验和
            header.flag = INIT;
            memcpy(buffer, &header, sizeof(header));
            if (cksum((u_short*)buffer, sizeof(header) + len) == check)
            {
                header.flag = ACK;
                cout << "Send has been confirmed! FLAG:" <<
state[int(header.flag)] << " SEQ:" << int(header.SEQ) << endl;
                break;
            }

        }
        // 序列号不对，或者检验和不对，等待
        else {
            // 收到的消息不对，超时重传
            if (clock() - start > MAX_TIME)
            {
                header.datasize = len;
                header.SEQ = u_char(order);//序列号
                header.flag = u_char(0x0);
                memcpy(buffer, &header, sizeof(header));
                memcpy(buffer + sizeof(header), message, sizeof(header) + len);
                u_short check = cksum((u_short*)buffer, sizeof(header) + len);//
计算校验和
                header.sum = check;
                memcpy(buffer, &header, sizeof(header));
                sendto(socketClient, buffer, len + sizeof(header), 0,
(sockaddr*)&servAddr, servAddrlen);//发送
                cout << "Message Error! ReSend message " << len << " bytes!
FLAG:" << state[int(header.flag)] << " SEQ:" << int(header.SEQ) << endl;
                start = clock();//记录发送时间
            }
            continue;
        }
    }
    u_long mode = 0;
    ioctlsocket(socketClient, FIONBIO, &mode);//改回阻塞模式

}
```

接收端收包

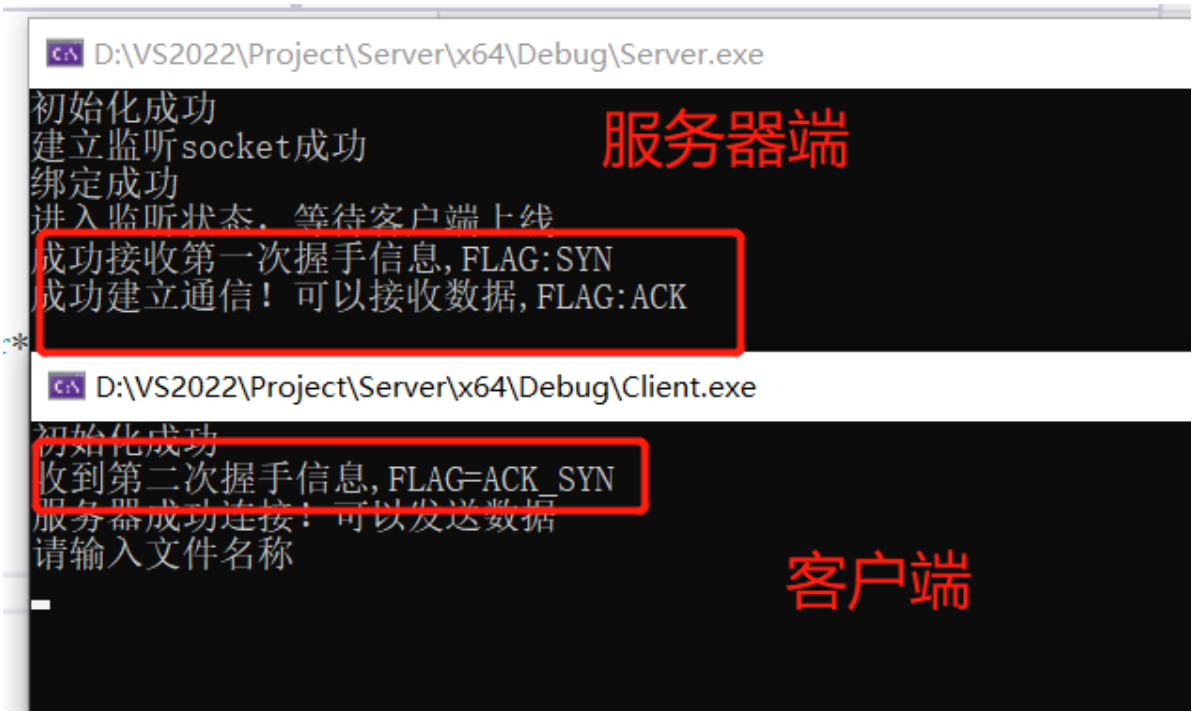
```
int RecvMessage(SOCKET& sockServ, SOCKADDR_IN& ClientAddr, int& ClientAddrLen,
char* message) {
    long int all = 0;//文件长度
    HEADER header;
    char* buffer = new char[MAXSIZE + sizeof(header)];
    int seq = 0;
    int index = 0;
    while (true) {
```

```
        for (int i = 0; i < MAXSIZE + sizeof(header); i++) {
            buffer[i] = 0;
        }
        int length = recvfrom(sockServ, buffer, sizeof(header) + MAXSIZE, 0,
(sockaddr*)&ClientAddr, &ClientAddrLen);//接收报文长度
        //cout << "收到了文件" << length;
        memcpy(&header, buffer, sizeof(header));

        //判断是否是结束
        if (header.flag == OVER && cksum((u_short*)&header, sizeof(header)) ==
0)
        {
            cout << "文件接收完毕" << endl;
            break;
        }
        // 数据包flag是初始化状态，为0    确认校验和
        if (header.flag == unsigned char(0) && cksum((u_short*)buffer,
length)==0&& seq == int(header.SEQ)) {
            cout << "Rece message " << length - sizeof(header) << " bytes!Flag:"
<< flagmap[int(header.flag)] << " SEQ : " << int(header.SEQ) << " CHECKSUM:" <<
int(header.sum) << endl;
            // 把数据写入message
            char* temp = new char[length - sizeof(header)];
            memcpy(temp, buffer + sizeof(header), length - sizeof(header));
            memcpy(message + all, temp, length - sizeof(header));
            all = all + int(header.datasize);


            // 校验和重新置0，返回ACK
            header.flag = ACK;
            header.sum = 0;
            memcpy(buffer, &header, sizeof(header));
            // 发送ACK
            sendto(sockServ, buffer, sizeof(header)+ MAXSIZE, 0,
(sockaddr*)&ClientAddr, ClientAddrLen);
            cout << "Send to Clinet ACK:" << (int)header.SEQ << " SEQ:" <<
(int)header.SEQ << " CHECKSUM:" << int(header.sum)<< endl;
            seq++;
            if (seq > 255)
            {
                seq = seq - 256;
            }
        }
        //校验和不对，或者SEQ不对
        else {
            //说明出了问题，返回ACK
            header.flag = ACK;
            if (cksum((u_short*)buffer, length) != 0)
                header.flag = INIT;
            memcpy(buffer, &header, sizeof(header));
            //重发该包的ACK
            sendto(sockServ, buffer, sizeof(header)+MAXSIZE, 0,
(sockaddr*)&ClientAddr, ClientAddrLen);
            cout << "Some ERROR:" << (int)header.SEQ << " SEQ:" <<
(int)header.SEQ << endl;
```

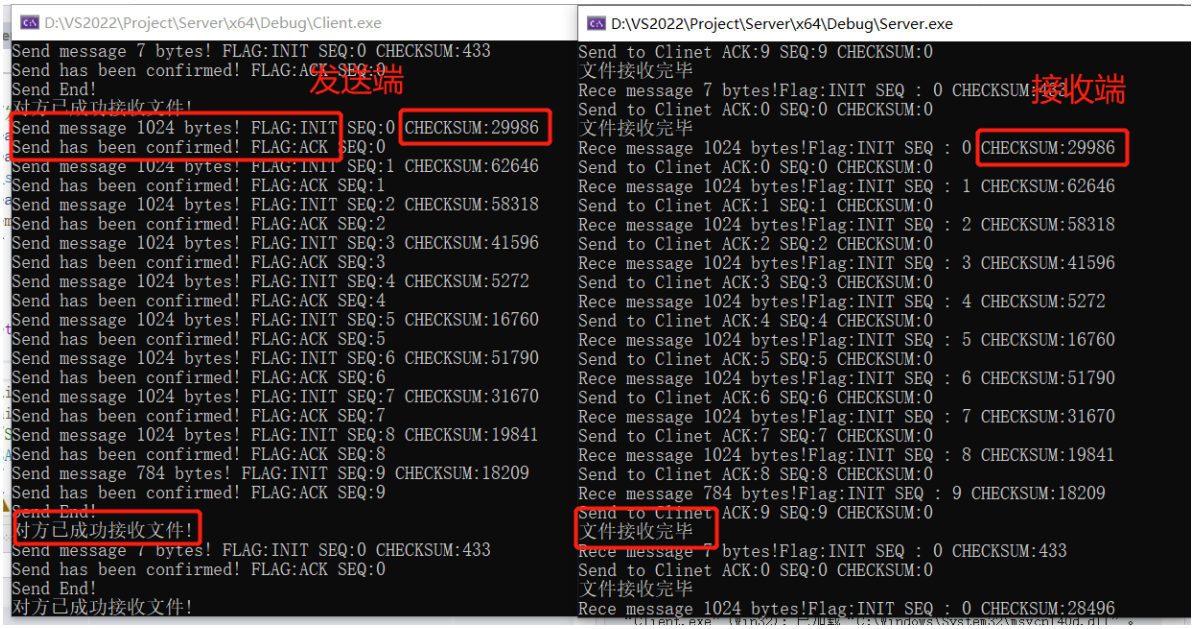
```
            continue;//丢弃该数据包

        }


    }
    //发送OVER信息
    header.flag = OVER;
    header.sum = 0;
    u_short temp = cksum((u_short*)&header, sizeof(header));
    header.sum = temp;
    memcpy(buffer, &header, sizeof(header));
    if (sendto(sockServ, buffer, sizeof(header), 0, (sockaddr*)&ClientAddr,
ClientAddrLen) == -1)
    {
        return -1;
    }
    return all;
}
```

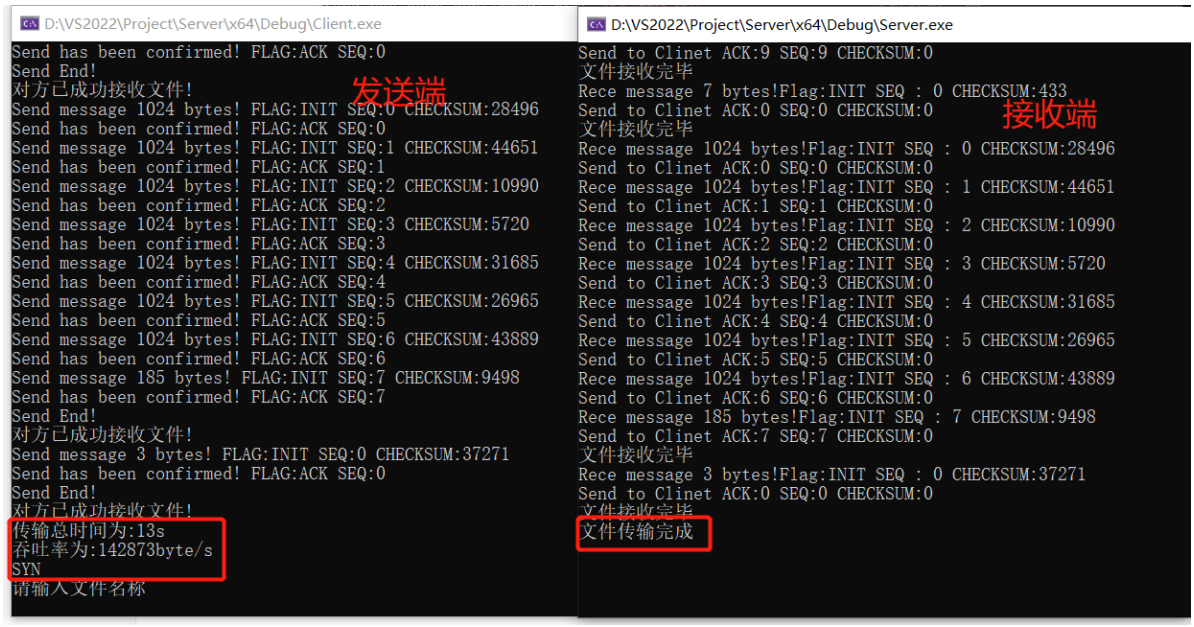
结果展示

握手展示



数据传输展示

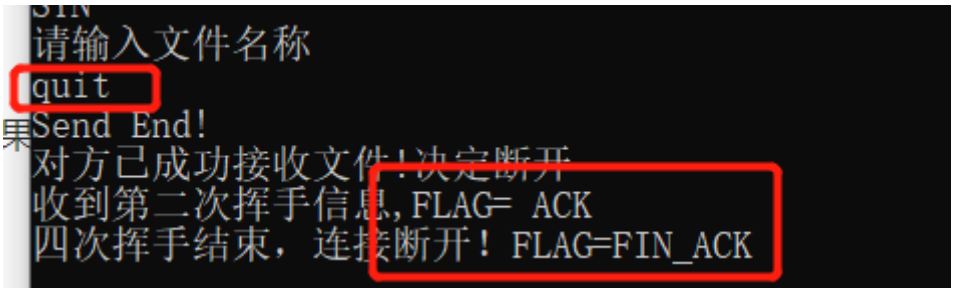




名称	修改日期	类型	大小
.vs	2022/11/16 21:44	文件夹	
x64	2022/11/18 0:20	文件夹	
1.jpg	2022/11/19 18:32	JPG 图片文件	1,814 KB
2.jpg	2022/11/18 18:32	JPG 图片文件	5,761 KB
3.jpg	2022/11/18 18:32	JPG 图片文件	11,689 KB
helloworld.txt	2022/11/19 14:01	文本文档	1,617 KB
server.cpp	2022/11/19 13:45	C++ source file	11 KB
Server.sln	2022/11/17 0:10	Microsoft Visual ...	3 KB
Server.vcxproj	2022/11/17 0:10	VC++ Project	7 KB
Server.vcxproj.filters	2022/11/17 0:10	VC++ Project Fil...	1 KB
Server.vcxproj.user	2022/11/16 21:44	USER 文件	1 KB

最终可以得到正常数据，文件大小和原来的都一样。

挥手展示



可以看到基本功能进行了实现。