实验 3-1 基于UDP设计可靠传输协议并实现

一、协议设计

1. 规定客户端和服务器端的报文格式

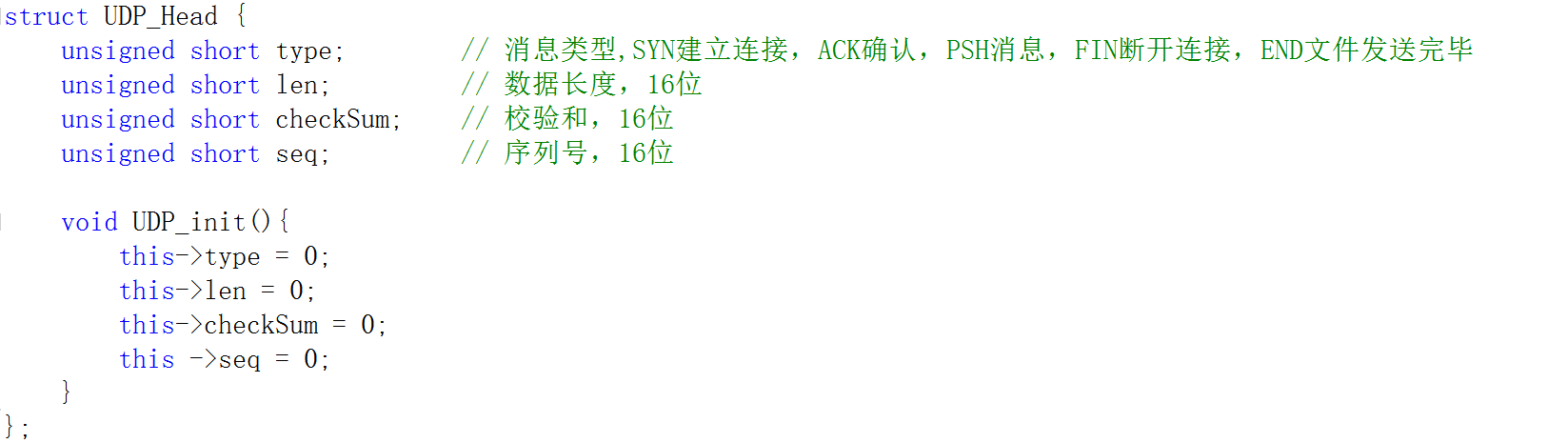
服务器和客户端为了便于交流，我们首先规定传输消息的报文格式。报文主要包括消息头和消息主体两部分。最终，发送消息缓冲区sendBuf和接收消息缓冲区recvBuf，它们的类型都为char 类型的字符数组。

1. UDP传输消息的消息头：设置消息头为一个struct结构体，每个成员变量都是16位。 type表示消息类型，用define宏定义了SYN建立连接，ACK确认，PSH消息，FIN断开连接，END文件发送完毕，SYN\_ACK表示服务器发来的连接确认，FIN\_ACK表示服务器发来的断开连接确认，END\_ACK表示服务器发来的文件传输完毕的确认。

len表示消息主体的字节数量，我们规定传输消息主体最大字节数为10240，也就是10KB。

chechSum是校验和，发送时先计算一遍校验和并取反，并存入消息头的这个成员变量中，接收方收到后，再计算一遍校验和，取反后如果为全0，则说明消息无误。之后在程序设计中详细介绍校验和函数。

seq表示发送端的序列号，在接收端也要进行校验。



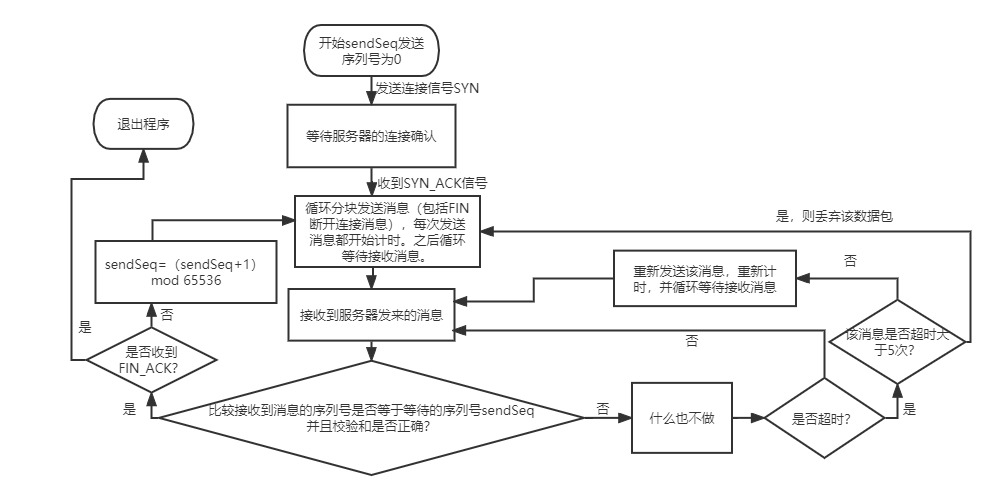
1. UDP传输消息的消息主体：文件传输的消息为char类型，我们规定如果消息字节数超过10240，那么分段发送，最大发送10240字节数，没有超过则发送现有字节数。
2. 客户端（发送端）和服务器（接收端）

使用了课程中讲授的rdt3.0协议进行设计，设计了建立连接、发送消息时的序列号检验、校验和检验、以及丢包时的超时重传的功能。

（1）建立连接：客户端（发送端）先发送SYN信号的消息给服务器（接收端），服务器再发送SYN\_ACK信号给客户端，如果两边都接收到消息并且序列号和校验和都正确，则连接成功。

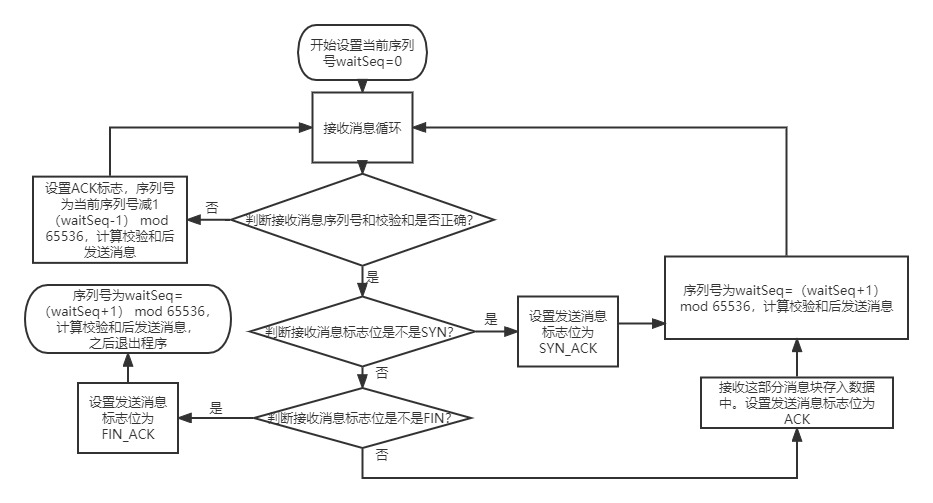
（2）客户端发送消息：一个大文件分成多个块循环发送消息，每次发送完消息后开始计时，之后进入等待接收端确认消息的循环：如果接收到确认消息，并且序列号和校验和都正确，则判断接收到消息的类型type，如果type是ACK类型，那么说明消息发送接收成功，序列号+1，跳出接收消息循环，发送下一个消息块。如果序列号或校验和有错误，什么都不做，等待超时重传。如果超时，重新传该消息，并重新计时，进入等待接收端确认消息的循环，超时重传如果超过5次，那么丢弃该消息包，传输下一个消息包。

我们看到发送端流程图如图所示：



（3）服务器接收消息：接收消息首先检验序列号是不是和等待的序列号相等，并且检验校验和是否为0。如果序列号或者校验和有误，那么发送ACK消息，并且消息的序列号为等待的序列号-1(mod 65536)。如果没有错误，进入类型判断。如果收到SYN，说明要建立连接，回SYN\_ACK标志消息，如果收到FIN，说明要断开连接，回FIN\_ACK标志消息，并退出程序，如果都不是以上两个，则说明接收到文件信息，将消息主体部分存入要写入文件的数据中，并且发送ACK标志消息表示确认收到消息。

接收端流程图如下：

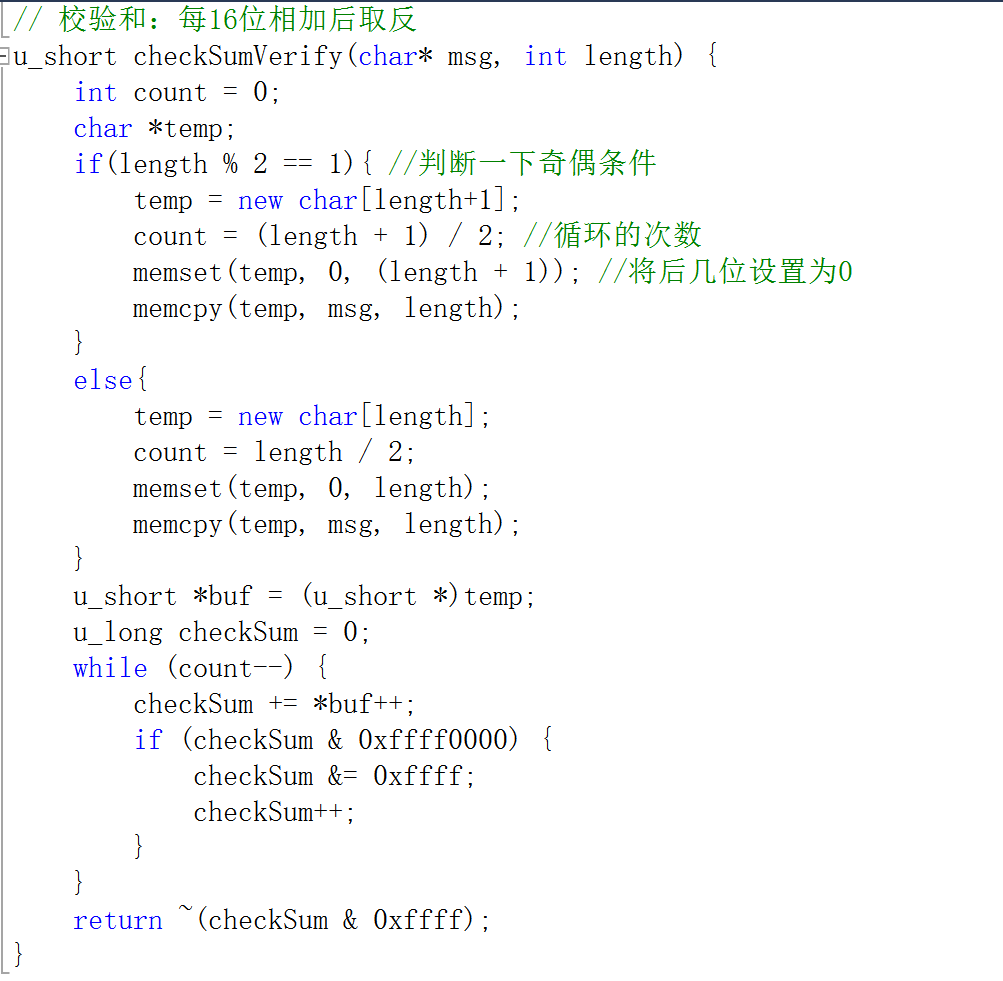


（4）断开连接：断开连接只需要两次挥手过程。客户端先向服务端发送FIN标志的消息，服务端收到后，发送FIN\_ACK标志的消息，则断开连接。

二、程序功能实现

在本小节主要介绍程序中主要实现的功能。

1. 校验和函数



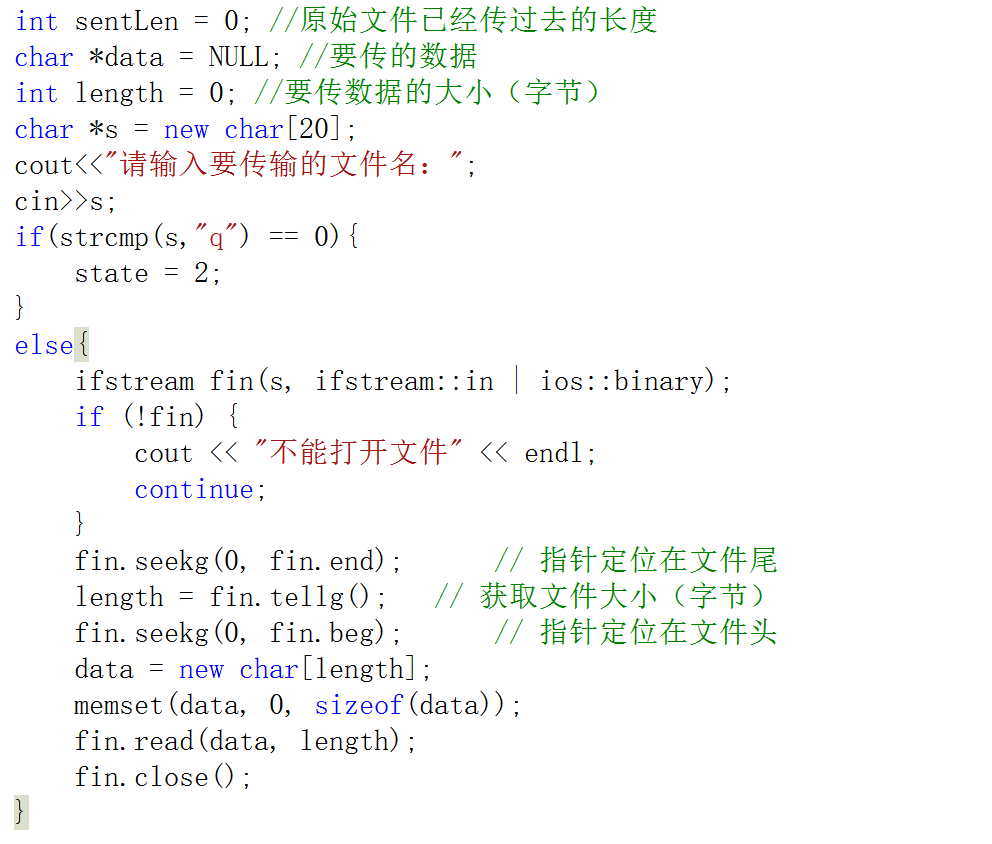
传入的参数是char 类型的消息，length是消息的字节长度。由于校验时我们是消息的每个16位数据进位相加后取反，char 类型是1个字节8位，我们要将消息分成16位的段。因此，首先要判断一下length的奇偶性，如果是奇数需要给结尾补0来保证16为格式对齐，需要先扩展一个length+1的空间，count是要循环的次数，count = (length+1)/2；如果是偶数则count直接除以2即可。用usigned long类型checkSum承接每个16位进位相加后的和，在循环中，每加16位的数据，判断一下是否超出16位，如果是则取后16位并末位+1。最后返回总和checkSum后16位并取反。

在发送端，计算校验和后存入消息头，并进行传输。在接收端，计算校验和，如果消息传输正确，那么校验和16位应该全是0。

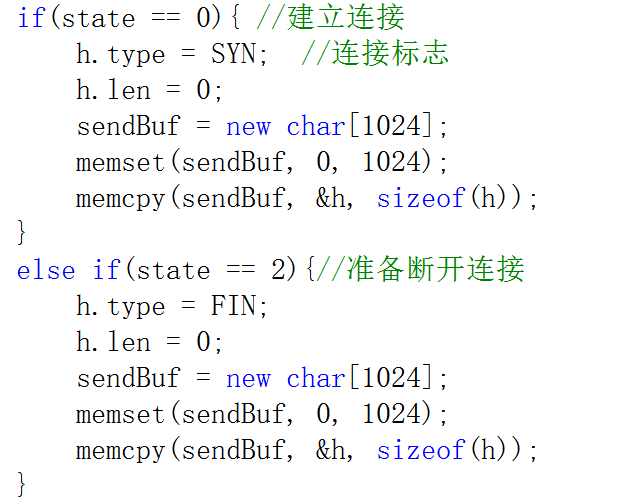
1. 客户端

主要设置三层while循环来进行：第一层while循环用来输入文件名，并且每一次循环来传输一个文件；第二层循环用来发送消息，由于一个文件可能比较大，我们又设置每次最大传输消息主体的字节数不超过10KB，因此一个文件需要分多个消息进行传输，因此设置了这个循环；第三层循环用来接收消息，当发送了消息之后，我们就进入接收消息循环，经过序列号检验和校验和检验后：如果收到服务器发来的ACK消息正确，则序列号+1，并跳出这层循环继续发送消息；如果收到的ACK消息有误，什么也不做等待超时；如果检测到超时，那么重新发送该文件并进行重新计时，依然进行着第三层循环来等待接收消息，值得注意的是，如果重传超过5次，就选择丢弃该数据包，跳出循环发送下一个消息。

在第一层循环中，主要是以二进制的形式读取文件，输入文件名之后用ifstream流fin打开并读取文件，将文件的所有数据存入data中，文件长度存入length中（单位：字节）。如果输入的文件名是’q’，表示断开连接，之后发送消息时客户端就会发送FIN标志位的消息给服务端。代码展示如下：



在第二层循环中，根据状态值判断，设置state为0时表示还没有进行连接，state为1时表示已经连接，可以发送文件消息，state为2时表示断开连接。



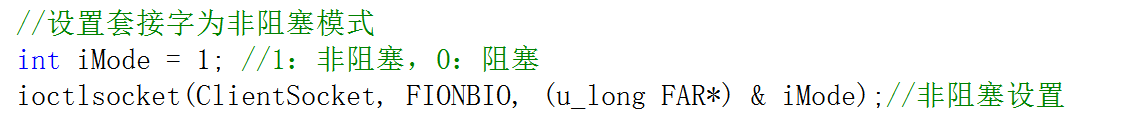
state为1时传输消息，就像前面说的，需要分块发送，计算校验和之后发送。

在第三层循环中，用来接收服务器收到的消息。同步判断一下序列号和校验和，如果正确则跳出循环，继续发送消息。如果有误，则什么都不做，等待超时重传。如果超时，则重新发送数据，并进行重新计时。





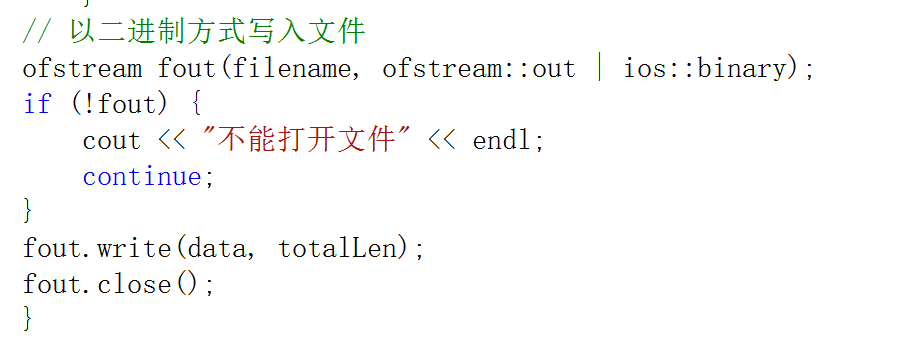
这里我们注意到，要设置socket为非阻塞模式，以保证在第三层循环接收消息的时候不会被recvfrom函数阻塞。



1. 服务端

服务端和客户端的接收消息这部分代码相似，当接收到消息时，判断一下序列号和校验和，如果都正确，则发送对应的ACK标志和当前序列号，否则发送ACK标志和前一个序列号，表示让客户端进行重传。

最终当收到END标志位的消息时，表示一个文件传输结束，发送END\_ACK标志位的消息给客户端后，跳出循环。随着END标志位消息传来的还有文件名，接收到文件名之后，用ofstream打开并写入文件。注意是以二进制形式写入文件，也和之前的读入文件类似：



三、程序测试

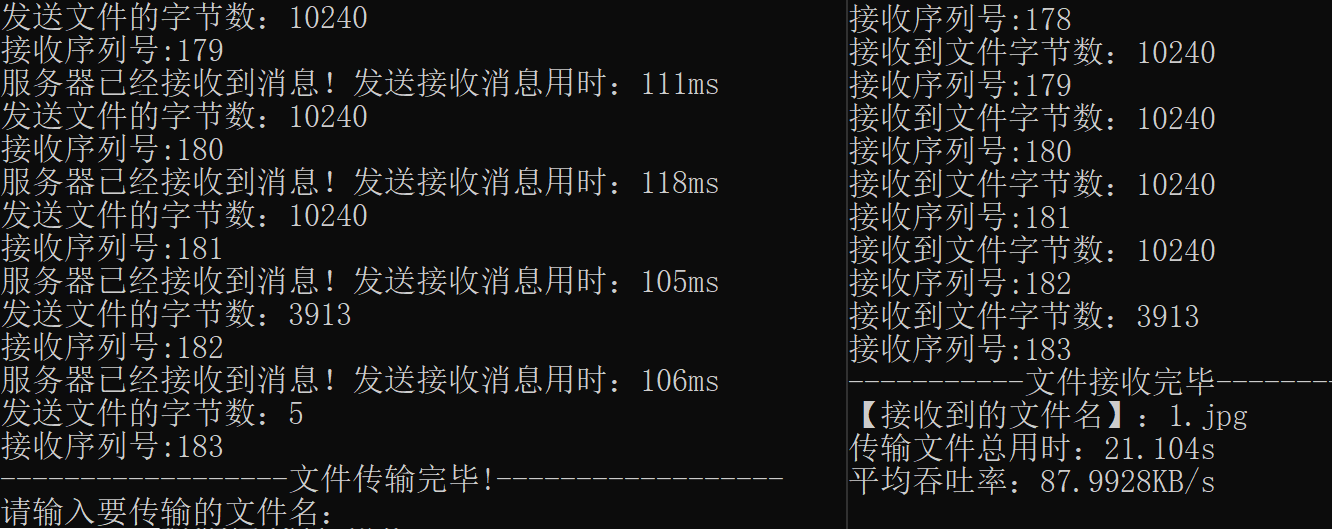
使用方法：

首先将要测试的文件放在源码同层路径下。先执行服务器端，再打开客户端。在客户端有提示，输入测试的文件名，回车后就进行了传输，之后在两边都可以看到传输时的消息日志记录。如果输入文件名是”q”，表示客户端发送断开连接消息。

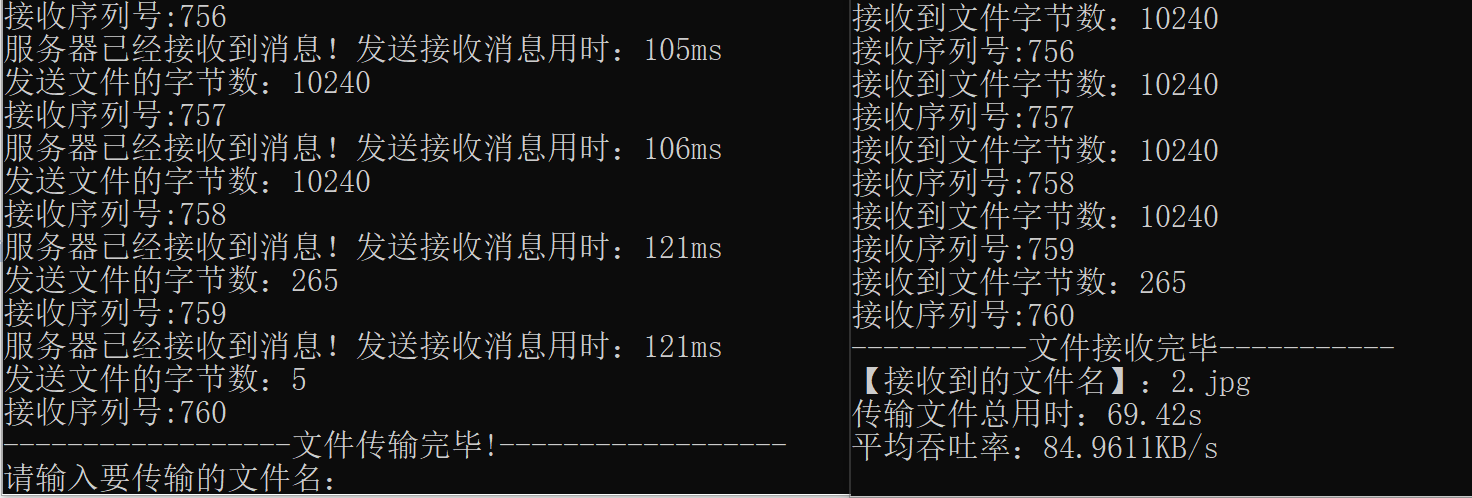
最终结果展示：

给定四个测试文件进行传输（左边客户端（发送端），右边服务器（接收端））

（1）第一个文件传输完毕时截图：此时用了183个序列号，除去连接用掉1个序列号，一共传输了182次将文件传输完毕。传输文件总用时：21.104s，平均吞吐率：87.9928KB/s



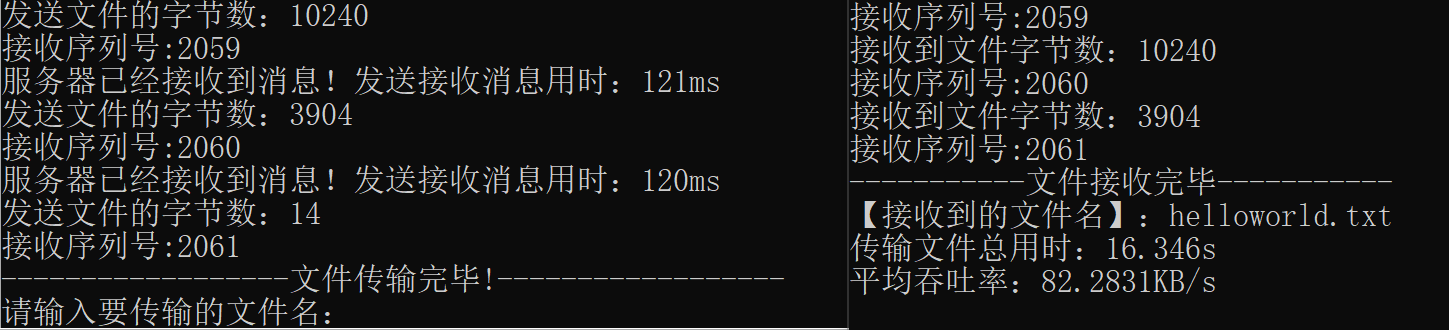
1. 第二个文件传输完毕时截图：此时用到第760个序列号，一共传输了577次将文件传输完毕。传输文件总用时：69.42s，平均吞吐率：84.9611KB/s



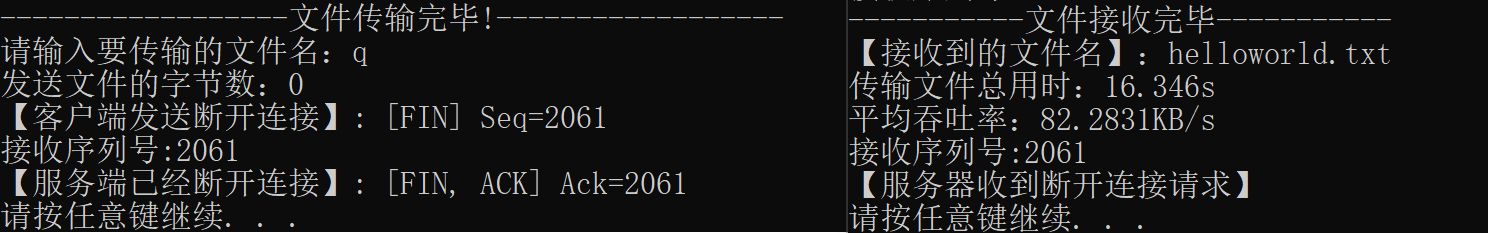
1. 第三个文件传输完毕时截图：此时用到第1929个序列号，一共传输了1169次将文件传输完毕。传输文件总用时：144.579s，平均吞吐率：82.7783KB/s



1. 第四个文件传输完毕时截图：此时用到第2061个序列号，一共传输了132次将文件传输完毕。传输文件总用时：16.346s，平均吞吐率：82.2831KB/s



1. 最后，再客户端（发送端）输入q表示断开连接。

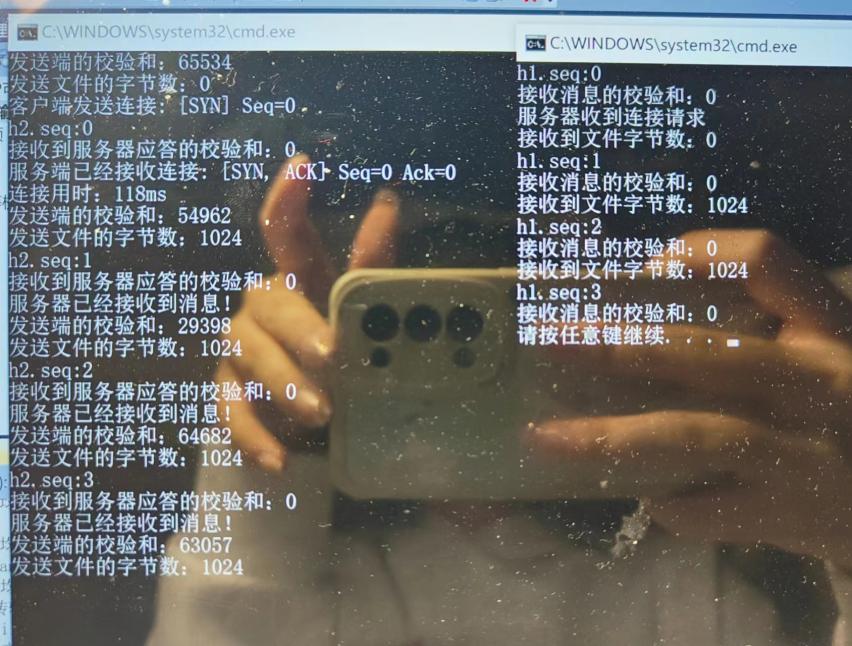


我们打开服务器端的文件夹，发现可以看到我们传输过来的四个文件。



1. 遇到的问题及解决办法

遇到的问题，由于之前校验和函数编写不正确，导致，服务端有直接退出程序的现象：



发现问题出在没有末位补零，可能导致在读取内存过程中出现错误，程序退出。之后，在校验和函数中先判断了消息长度的奇偶性，并在奇数末位补零，这样在char类型向unsigned short类型转换过程中就不会出现错误，最终解决了问题。