**实验3-1 基于UDP服务设计可靠传输协议并编程实现**

1813069 郭怡霏

1. 实验目的：

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：

1. 建立连接；
2. 差错检测；
3. 确认重传。

流量控制采用停等机制，完成给定txt格式和jpg格式测试文件的单向传输（与TCP功能类似，在UDP服务上实现）。

1. 实验原理：

UDP 是User Datagram Protocol的简称， 中文名是用户数据报协议，是OSI（Open System Interconnection，开放式系统互联） 参考模型中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，IETF RFC 768是UDP的正式规范。UDP在IP报文的协议号是17。

UDP协议与TCP协议一样用于处理数据包，在OSI模型中，两者都位于传输层，处于IP协议的上一层。UDP有不提供数据包分组、组装和不能对数据包进行排序的缺点，也就是说，当报文发送之后，是无法得知其是否安全完整到达的。UDP用来支持那些需要在计算机之间传输数据的网络应用。包括网络视频会议系统在内的众多的客户/服务器模式的网络应用都需要使用UDP协议。UDP协议从问世至今已经被使用了很多年，虽然其最初的光彩已经被一些类似协议所掩盖，但即使在今天UDP仍然不失为一项非常实用和可行的网络传输层协议。

许多应用只支持UDP，如：多媒体数据流，不产生任何额外的数据，即使知道有破坏的包也不进行重发。当强调传输性能而不是传输的完整性时，如：音频和多媒体应用，UDP是最好的选择。在数据传输时间很短，以至于此前的连接过程成为整个流量主体的情况下，UDP也是一个好的选择。

总而言之，UDP的特点是发送快速但不一定传输完整，且不支持差错检测、确认重传等功能，因此将这些功能定义为上层的协议，通过携带校验和、不断地单向传输不同的数据包来实现。

1. 实验重点：
2. 建立连接；
3. 差错检测与损坏包重传；
4. 应用层的自定义协议。
5. 实验环境：
6. 操作系统：Oracle VM VirtualBox 6.1.14 r140239 (Qt 5.6.2)

Ubuntu 18.04.5 LTS (64 bit)

1. IDE：Visual Studio Code 1.51.1
2. 使用语言：C++
3. 使用库：stdio.h string.h stdlib.h unistd.h sys/types.h sys/socket.h sys/time.h netinet/in.h time.h pthread.h arpa/inet.h iostream fstream cstdlib ctime
4. 实验步骤：

自定义协议：

1. 建立连接：client获取到服务器的地址之后向server发送想要的文件名；server端创建UDP套接口、创建数据报socket、绑定套接口之后开始监听，如接收到client的文件名请求后即生成新数据包并发送；
2. 传输数据类型：完整的数据包是一个名为Pack的类，包括包头PackInfo和正式报文buf两部分，包头中包括数据包序列号id、报文长度buf\_size和校验和checksum，本次作业实现server向client传输文件，server发送的是完整数据包，而client回发的ack只有包头，通过改变其中的id来告知所需数据包；
3. 读写方式：因为需要传输txt文本文档和jpg图片，因此一律使用二进制读写形式；
4. 文件结束：server设置为每次只能发送一个文件，因此如果创建新包时，使用fread读取数据得到返回值为负，则说明文件已经读完，则设置最后一个数据包的id为-1，发送给客户端之后不再接收ack，直接关闭文件并退出；
5. 差错检测：server在发送数据包之前通过计算来确定校验和checksum，计算公式为head.checksum = ~(SERVER\_PORT & SERVER\_IP\_1 & SERVER\_IP\_2 & SERVER\_IP\_3 & SERVER\_IP\_4 & head.id & head.buf\_size)（将server的端口号、IPv4的四位、数据包id、报文长度这七个数按位与之后取反）；client不设置缓存，收到数据包之后检查计算得到的checksum是否正确，如计算结果不合则直接丢弃包，并返回该包的包头作为ack；
6. 确认重传：server端设置一个名为data的包来传输，使用停等机制，故收到的ack编号只会与将要发送的id相等或小1。设置一个while(1)的循环，每一次循环中先检查对方所需的数据包是否为将要发送的那一个（即有没有丢包错包情况）：如是，则将继续读取文件中长度不超过BUFF\_LEN（设为1024）的数据作为新包报文，设置好包头之后发送给client；否则，此时data的数据没有刷新，还是上一个包的数据，可更新校验和后重新发送（因为此时可能是错包的情况，checksum计算有误，因此重新计算）；
7. 结束：client收到id为-1的数据包表明文件传输结束，不再接收并退出。

编译：

在 vscode 命令行打开当前文件夹，输入如下代码：



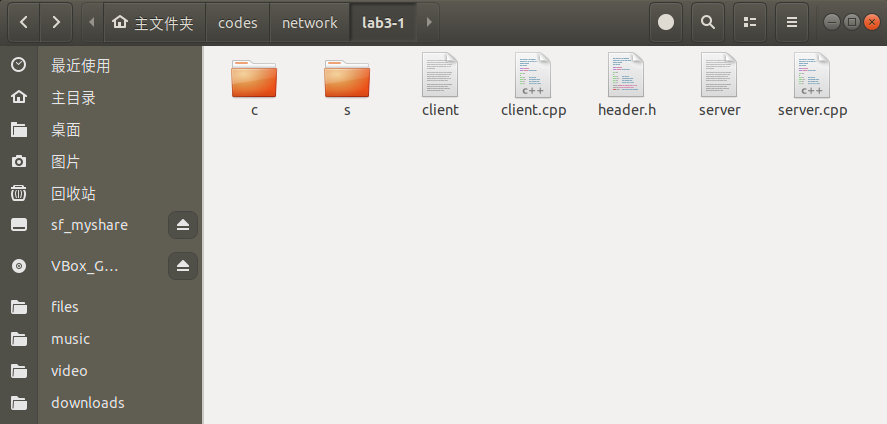
运行：

在命令行打开当前文件夹，输入如下代码，即可运行：

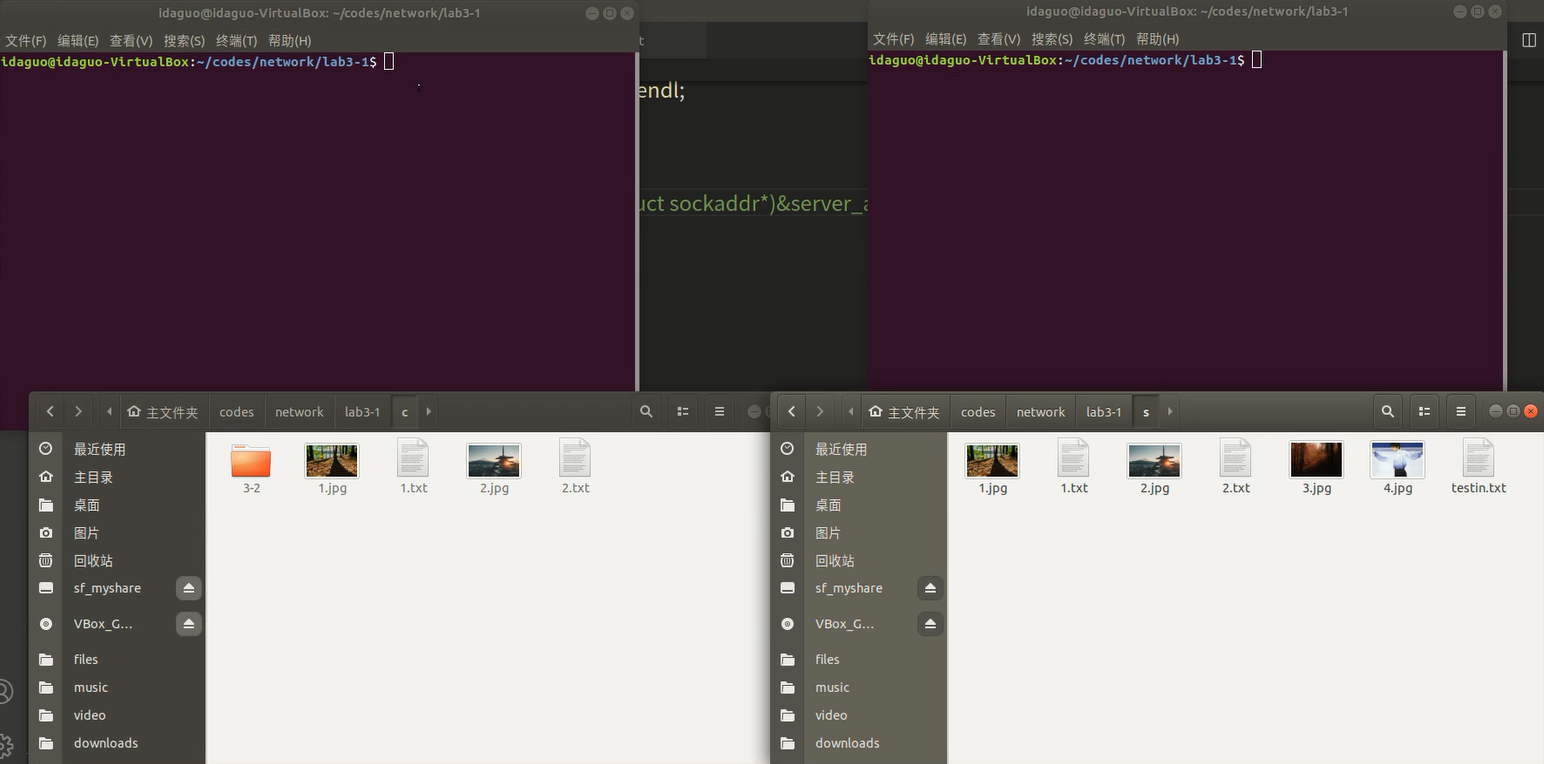




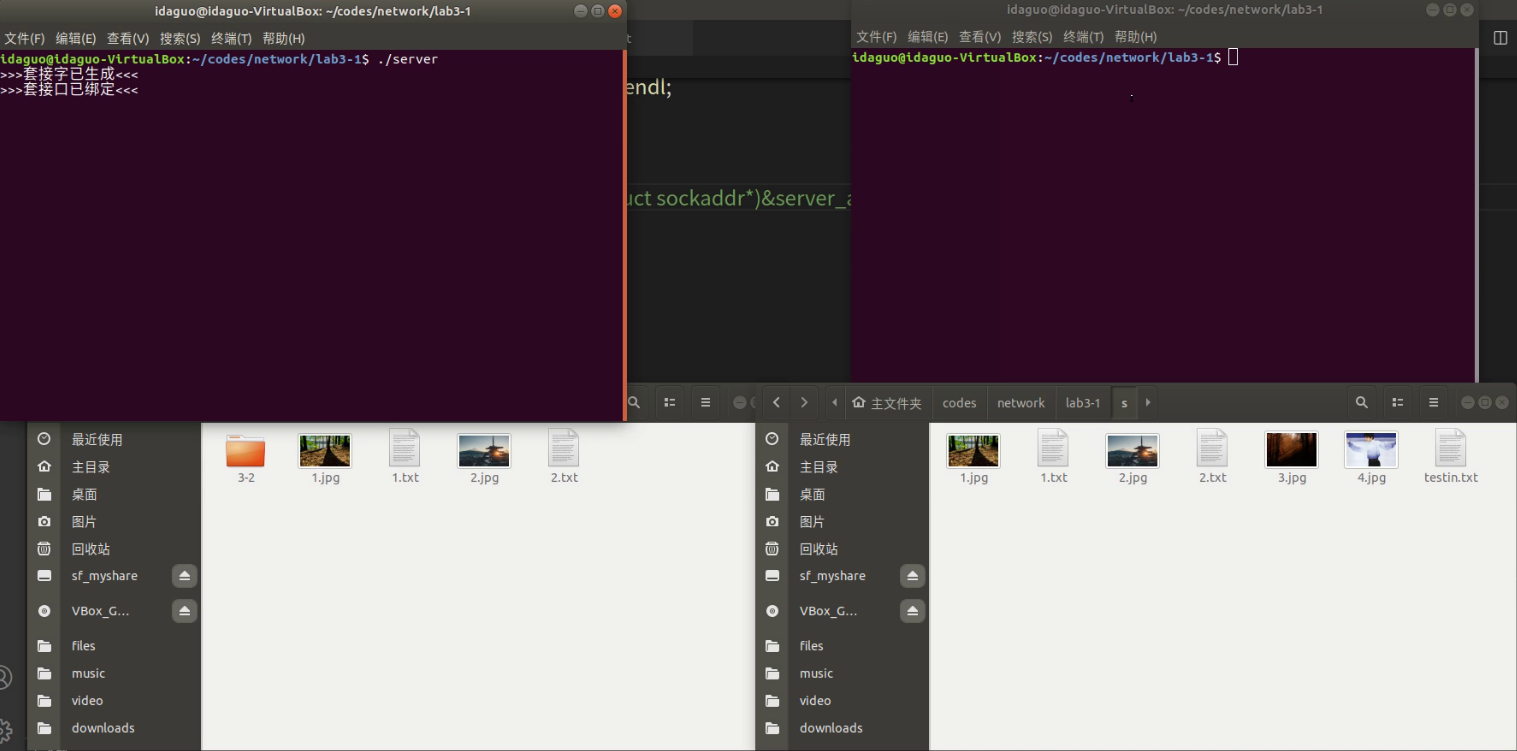
1. 实验结果：



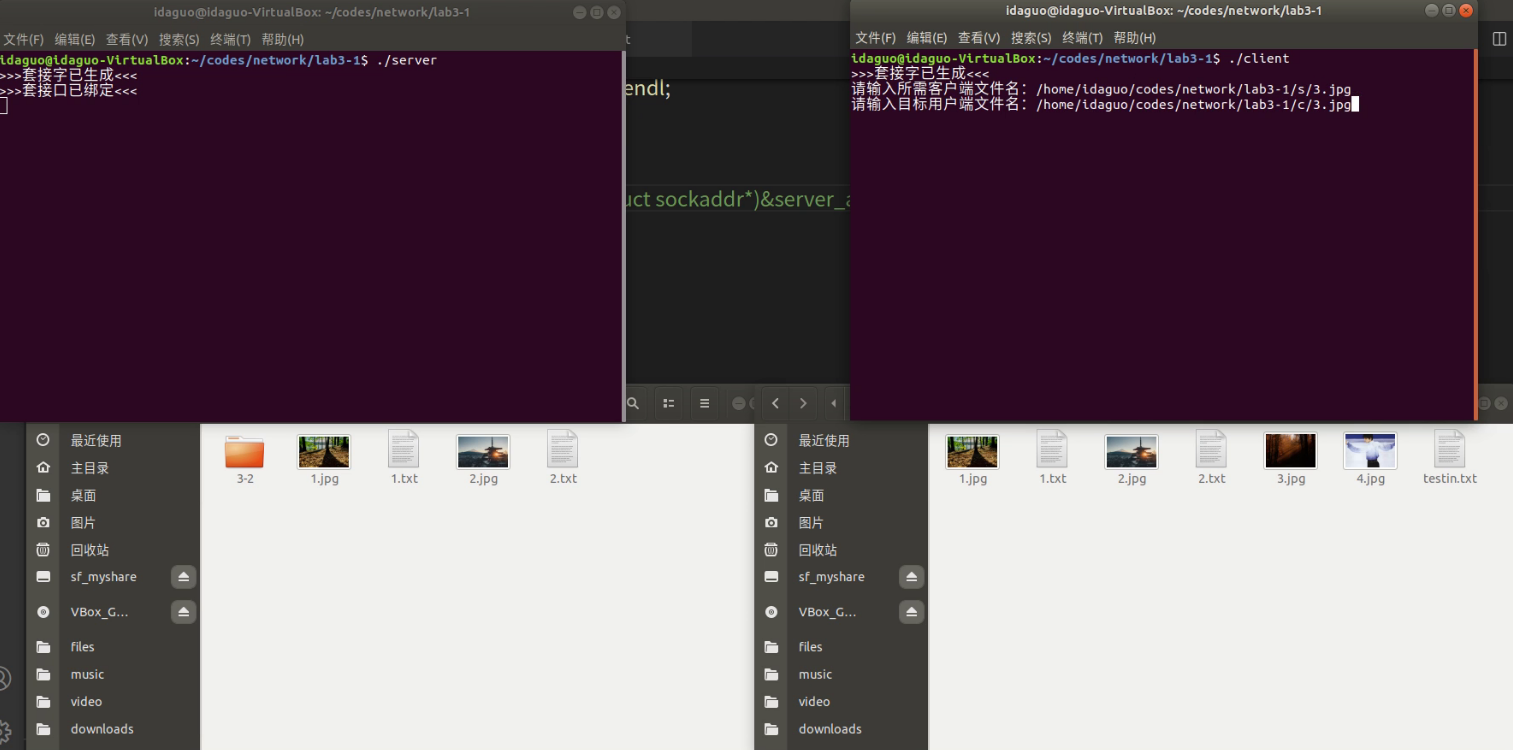
以上是文件列表，分别是客户端文件夹、服务器文件夹、客户端可执行文件、客户端源代码、服务器可执行文件、服务器源代码。通过在命令行输入./xxx（xxx 为可执行文件名）来运行。



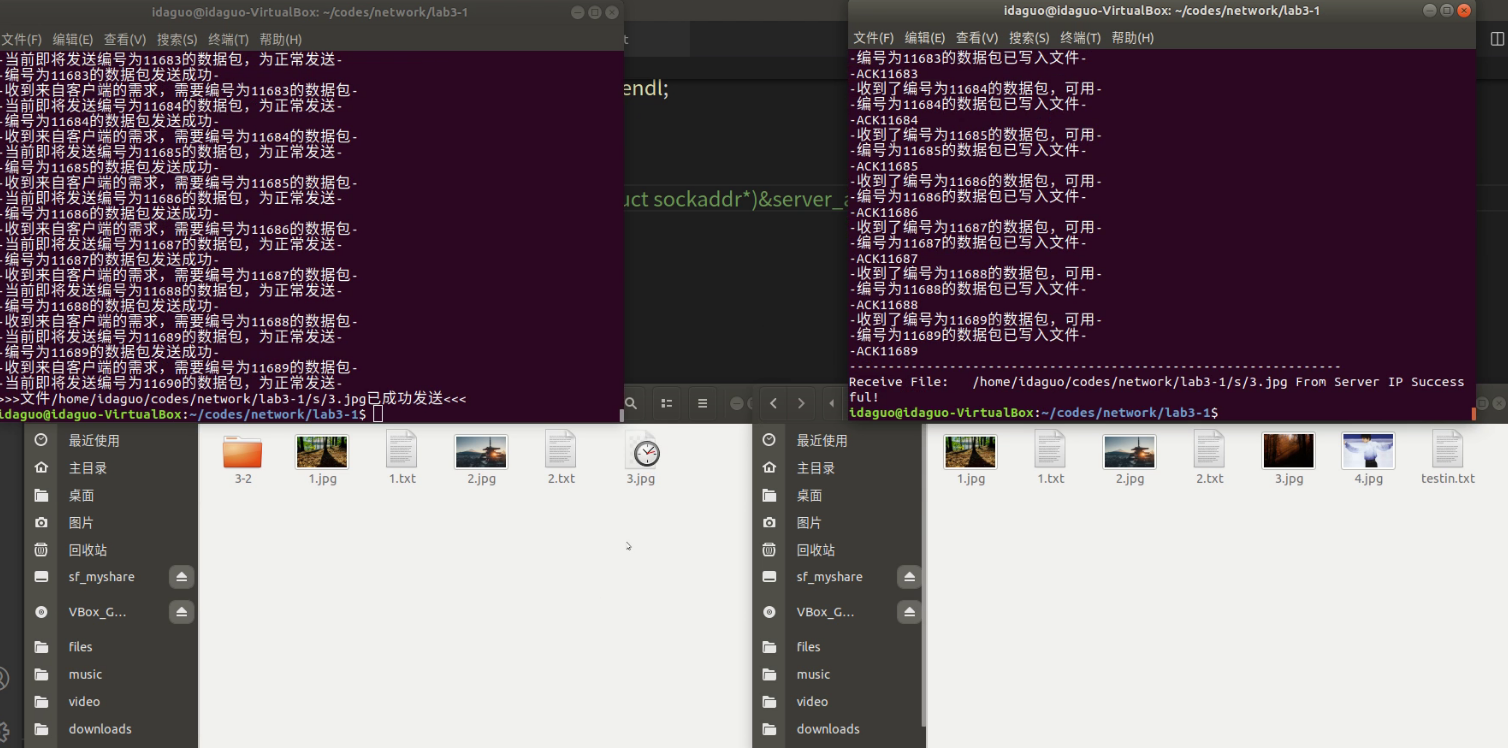
传输文件前的client文件夹和server文件夹



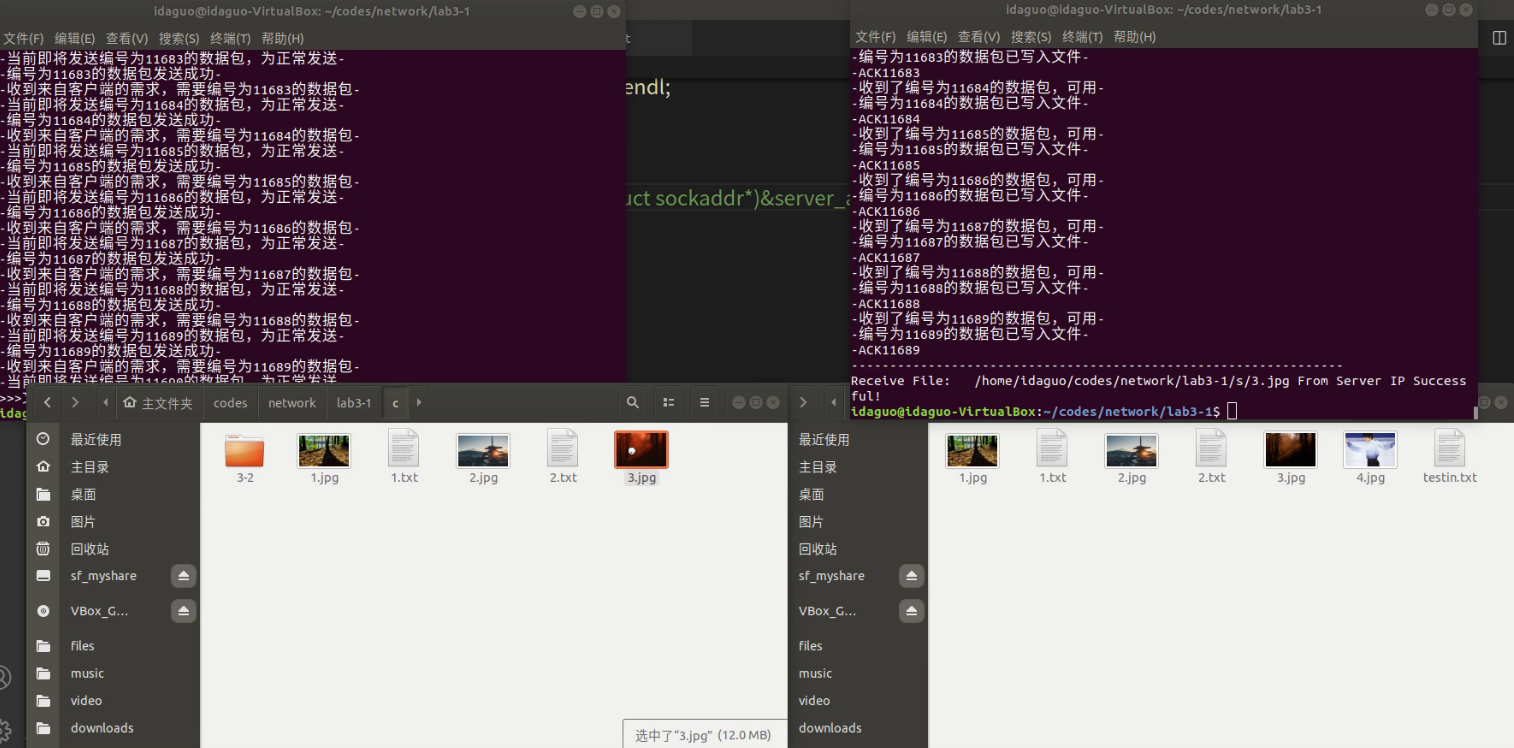
（1）服务器完成初始化设置开始监听



（2）客户端初始化，由用户输入希望得到的文件和要保存的位置



（3）名为3.jpg的文件传输结束，服务器和客户端都退出



（4）传输结束，client文件夹有了被传过来的3.jpg文件，可正常显示

1. 问题与不足：

起初设置client端只能接收一份文件，接收完自动退出，而server端不退出，但发现传输图片时会出现屏闪和Ubuntu的连续提示音，图片成功接收之后想再次运行client，发现什么也传不了，将server重启后可以传txt文件但不能传图片，必须将client可执行程序删除后重新编译才可以，因此设置server也是自动退出。