report（stage2）

实验内容

进一步优化 FTP 服务器和客户端：

- 服务器支持多客户端连接；
- 服务器支持非阻塞传输大文件；
- 服务器支持断点续传；
- 客户端有用户友好的 GUI。

运行环境

1. 服务器：
   - 操作系统：ubuntu16.04
   - 内存：1GB
   - 编译器：gcc 5.4.0

2. 客户端：
   - 操作系统：Windows 10 64 位

实现原理

1. 服务器非阻塞传输大文件：

   将需要用到的文件与套接字描述符加入读写状态判断集合中，在循环中使用 select()函数判断每次循环时文件与套接字的读写状态，并作出相应处理，从而让不同文件与套接字的处理事件并行执行。

2. 服务器支持多客户端连接：

   让服务器的监听套接字在主进程中循环监听，若接收到客户端的连接请求，则使用 fork()函数创建子进程，在子进程中处理该客户端的各种事件。

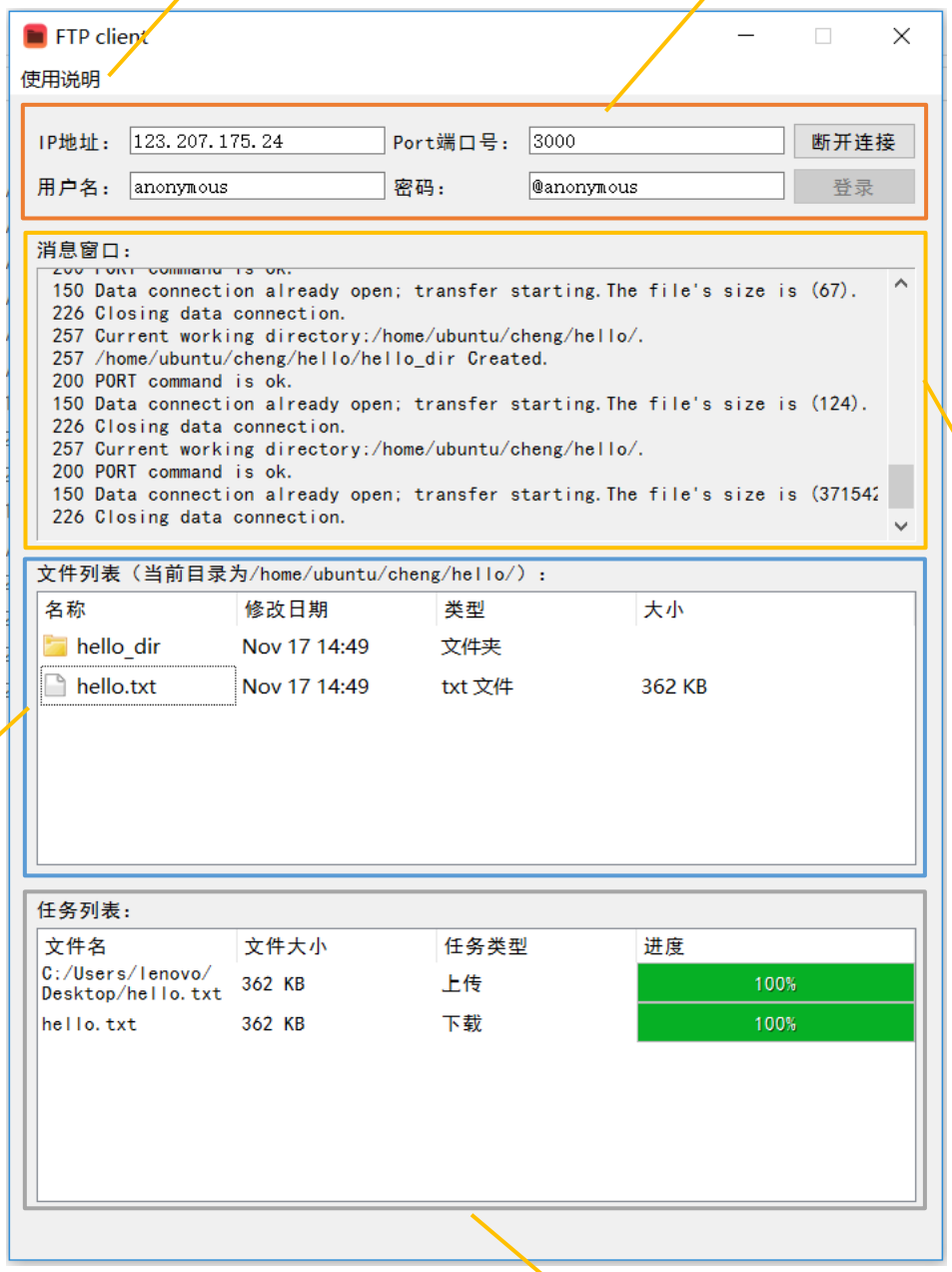
3. 服务器实现断点续传：

   实现“REST”指令，在数据传输前告知服务器已有文件的大小，从而设置文件的读写指针；之后再继续进行之前的数据传输即可。

客户端应用程序界面说明

点击查看使用说明。

输入 ip 地址和端口号进行连接；输入用户名和密码登录。



查看服务器返回的消息提示。

显示当前目录下的文件列表，具体交互说明见 README 文档。

显示任务列表，具体交互说明见 README 文档。

实验的难点与反思

1. **客户端数据接收问题。**客户端的实现中我使用了 QtcpSocket 类，在下载文件时，无论是采用信号槽机制还是多线程机制，都会出现数据接收不完全的情况。这里我的解决办法是在服务器传输数据时注明包的大小，客户端接收完成后向服务器发送完成指令，服务器再传下一个包。但是这就导致这个服务器应用程序不具有普适性。
2. **界面交互友好性与稳定性的考虑。**因为界面交互需要考虑用户的各方面需求，还要防止用户的某些操作使程序出现奇怪的 bug，以及还要考虑不同网络状态下程序的运行情况，所以在实现过程中要考虑很多问题并做大量测试。