FTP实验报告

软件72 邵璟之 2016010853

1. 开发环境

- ubuntu:16.04
- python:3.5.2
- gcc:5.4.0

2. 运行指令

```
cd server
make
sudo ./server

cd client
python3 main.py
```

3. 功能实现

本实验分为服务器和客户端两部分，服务端使用C语言实现，客户端使用Python 3 解释器实现。 所有指令均遵循RFC959 标准，在测试时先通过与标准FTP 服务器 （pyftpdlib）和客户端（ftplib, The Python Standard Library）测试，保证协议的正确性。本实验中服务器通过多线程方式实现，使得可以同时处理接 受多个客户端的请求。

3.1 服务端

- 服务端实现要求的全部14个命令，并增加了REST指令：
    1. 当命令是USER 或者PASS 时，接受客户端传来的用户名和密码，但并不校验用户名和密码的正确性。
    2. 当命令是SYST 时，返回服务器使用的操作系统。
    3. 当命令是PWD，CWD 时，实现与工作目录相关的操作。前者是打印工作目录，后者是更改工作目录。
    4. 当命令是MKD，RMD 时，实现创建新文件夹，删除文件夹。
    5. 当命令是TYPE 时，设置数据传输格式。考虑到绝大多数情况都是使用二进制方式传输，这里只实现了二进制传输格式。
    6. 当命令是PORT 时，服务器进入主动传输模式。连接客户端指定的端口。
    7. 当命令是PASV 时，服务器进入被动传输模式。服务器监听一个端口， 并且把监听的端口返回告诉客户端，并且由客户端来连接。
    8. 当命令是RETR 或STOR 时，分别是从服务器获取一个文件或者存储一 个文件到服务器。获取文件的时候使用操作系统的接口，这样对于文件不存在的情况可以正常处理，不会导致程序崩溃。对于客户端命令以及数据通道的正确性也有相应的校验，确保程序不会崩溃。
    9. 当命令是LIST 时，返回工作目录的文件列表，也使用了操作系统的接口。
    10. 当命令是RNFR 或RNTO 时，用户可以对一个文件进行重命名。可以通过linux自带ftp连接进行测试。
    11. 当命令是QUIT 时，退出服务器，并将用户状态初始化。
    12. 当命令是REST 时，记录user当前已经传输了文件的大小，下次重新传输同一个文件时可以跳过这一部分，支持断点续传。
- 支持多用户同时登录
- 多线程上传文件
- 通过autograde.py评分脚本

3.2 客户端

- 客户端功能封装在FTPclient类中。实现了USER, PASS, RETR, STOR, QUIT, SYST, TYPE, PORT, PASV, MKD, CWD, PWD, LIST, RMD, RNFR, RNTO等命令，另外实现了DELE, HELP命令。通过封装好的TCP类来建立连接。
    1. login 方法用来登录服务器。判断用户名正确时再发送密码，完成登录，禁止用户名为空。客户端会记录是否已经登录，如果没有登录，将会提示先进行登录。
    2. quit 方法用来退出服务器，同时将状态初始化，防止退出后再调用其他操作导致错误。
    3. help 方法用来显示帮助信息。
    4. syst 方法向服务器发送SYST指令。
    5. pwd 和cwd 方法用来打印和更改工作目录。客户端通过给服务端发送指令获取当前工作目录，或更改工作目录。
    6. mkd 和 rmd 方法用来新建和删除路径。客户端通过给服务端发送MKD和RMD指令实现。
    7. pasv 方法通过给服务端发送PASV指令进入被动连接模式。pasv方法会拿到服务器提供的地址和端口，建立数据通道。
    8. 实现了port方法，客户端可以指定ip和port进行连接。
    9. list 方法用来打印工作目录的文件列表，按照RFC959 通过数据通道传输。
    10. retr 和stor 方法分别用来从服务器获取一个文件或者存储一个文件到服务器。
    11. rnfr 和rnto 方法允许用户对文件进行重命名。通过向服务端发送FNFR、RNTO指令进行调用。
- gui支持连接服务器，登录，显示当前文件夹目录，上传文件，下载文件，新建文件夹，删除文件夹

4. 文件结构

4.1 服务端

- server.h, server.c定义用户信息相关的内容，开启服务器相关的类和函数
- commands.h, commands.c定义要求实现的14个命令相关的函数，并实现14个函数，定义部分返回信息的内容
- api.h, api.c定义commands.c中需要用到的部分函数
- main.c定义根目录，开启服务器，处理传入的参数。

4.2 客户端

- tcp.py定义TCP类，用于建立TCP连接。
- client.py用于实现客户端与服务端对应的指令。
- MainWindow.py实现ui的基本布局。
- main.py实现ui中按钮对应的功能，以及最终界面的调用。

5. 其他

- 能同时接受来自多个主机的客户端连接，多用户能同时传输、读取数据
- 服务器能够处理多个连接，实现方式是使用pthreads，为每个用户开一个单独的线程
- 服务器通过REST函数配合user_info类中记录的skip_byte信息实现断点续传。