FTP实验报告

1.FTP命令实现

在server上实现了所有题目要求的FTP命令：

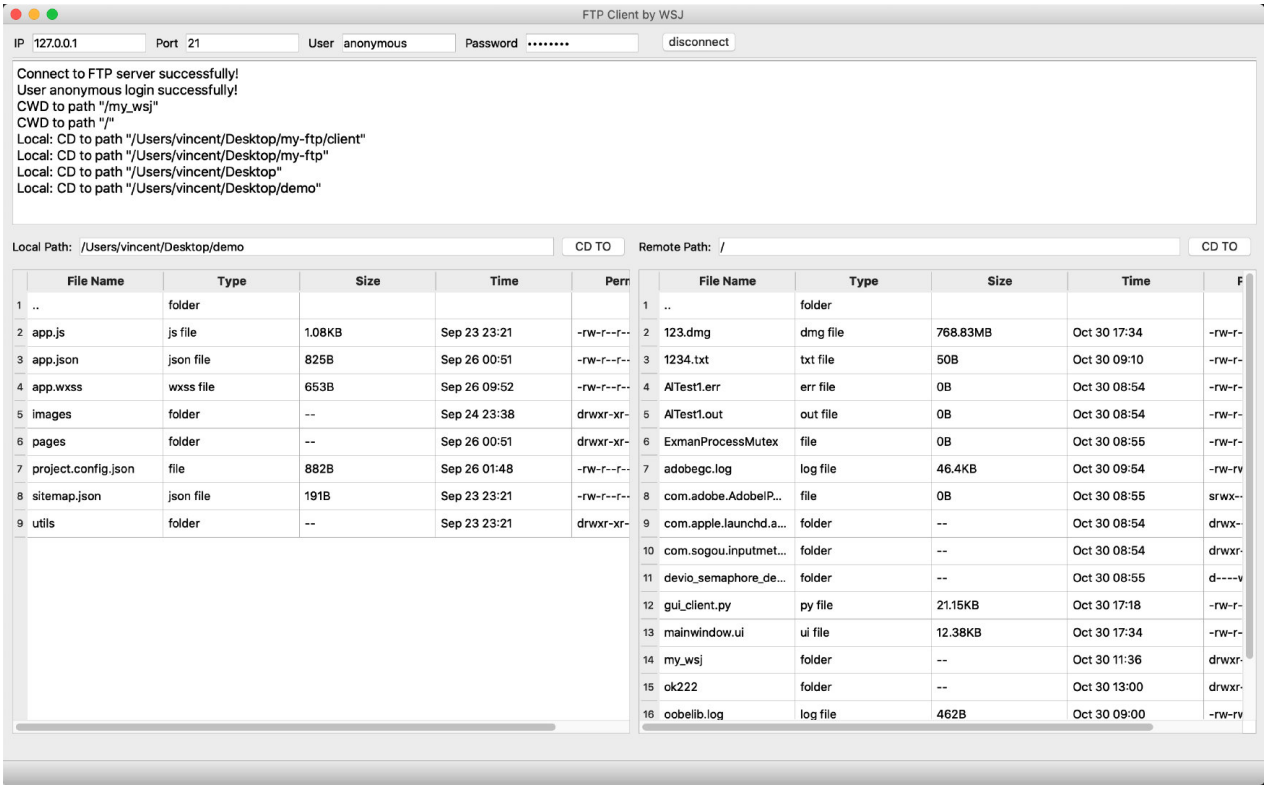
```
["USER","PASS","STOR","RETR","QUIT","SYST","TYPE","PORT","PASV",
"LIST","RNFR","RNTO","MKD","CWD","PWD","RMD",]
```

此外还实现了`DELE`和`REST`,`CDUP`等命令

在client上实现了以下命令：

```
["USER","PASS","STOR","RETR","QUIT","SYST","TYPE","PORT",
        "PASV","LIST","REST","RNFR","RNTO","MKD","CWD","PWD","RMD",
        "login","ls","upload","download","mv","pwd"]
```

GUI程序运行示意图如下，具体运行将在当面验收演示：



2.使用说明

见各个src目录下的README文件

3.文件结构介绍

client/src目录：

client.py: 定义了client类及其方法，也可以直接启动脚本

gui_client.py: GUI客户端脚本

mainwindow.ui: pyqt的UI文件，可以用qt designer打开编辑

Ui_mainwindow.py: UI文件自动生成的py代码

server/src目录：

server.c: 服务器的启动代码

handle.h: 定义了服务器对于不同命令的处理函数

util.h: 定义了一些工具函数

Makefile

4.主要思路与难点介绍：

server:

对于C语言实现的server，主要思路是根据协议，使用socket与客户端通信，接受客户端的命令并进行操作。

- 对于每个连接的客户端，我都维护了一个State的结构体，里面存储了诸如socekt,ip,连接状态以及目录等信息。由于server需要支持多个服务器同时连接，我才用了多进程的fork()方法：
  - fork()方法是linux系统中的一个方法，当调用该方法时，会将当前进程进行一份拷贝，此时生成了两个进程，并构造出一个进程链表，调用fork的进程为父进程，有一个指针(fpid)指向子进程。因此在每次accept到不同客户端的连接请求时，主进程都会fork一次，复制出一个子进程，使用子进程去处理新的客户端，然后杀死之前的父进程使其跳出阻塞状态。由此当有n个客户端同时连接server时，会有n个子进程处理对应的client，一个进程处于等待连接的阻塞状态。
- 与python的socket连接不上，查阅资料发现是由于端口的大小端表示不同，因此需要用htons函数进行转换
- 在收到命令后，通过handle函数解析并做出不同的操作与回复。一开始在程序结束后总要等一会释放端口才能被再次使用，后来查阅资料得知，可以使用以下语句使得端口释放后立即可以被使用

```
int reuse = 1;
setsockopt(serverSocket, SOL_SOCKET, SO_REUSEADDR, &reuse, sizeof reuse);
```

- 为了在ctr-c杀死进程后释放socket，我使用singal函数接受并处理SIGINT,SGTERM两个信号，这样退出程序后不会导致抵制占用。
- 目录权限的控制：为了安全起见，FTP根目录之外的目录不应当被访问到，这就需要对路径进行一些限制，我做了以下处理：
  - 对于..等相对路径，如果在根目录则不支持，会报错
  - 对于所有输入的服务器实际地址的绝对路径，都不支持，例如当根目录是"/tmp"时，其下有一个文件夹”123“，对于server来说,”/tmp/123“是不可接受的，"/123"和”123“是可以接受的
  - 判断一个目录的是否是可访问的：如果根目录的绝对路径是该目录的前缀串，则该目录可访问。

- 断点续传，通过REST命令设置断点，该断点是在state结构体中储存的，当REST后，再次执行RETR命令，文件起点指针会设置为断点处。
- 经过测试FileZilla客户端可以连接我的服务器并且执行相关操作，我很欣慰。

client:

- 客户端方面，为了逻辑更加清晰，我首先编写了一个纯命令行式客户端，在client.py中定义了Client类，通过命令行式交互确保client的功能正常。为了方便使用，我的客户端定义了一些额外命令：例如`ls`实际上是PORT或者PASV命令+LIST命令,`dowload`命令结合了PORT/PASV和RETR
- 为了提高程序的鲁棒性，我对于意外进行了处理，及时在客户端输入一些非法命令也能够正确捕捉并做出反应。
- 在GUI程序的编写上，我使用了pyqt5，其中遇到一个bug是文本框的刷新渲染有问题，在google之后我终于在qt的官方论坛的一个ISSUE的评论区角落找到了解决方法：使用pyqt 5.10版本，过高过低都会有问题=_=
- 在GUI界面上，我参考了著名开源软件FileZilla的设计，用户可以在客户端中访问本地和远程的目录，并且通过鼠标点击执行一些操作。
- GUI界面的一个主要问题是，在下载上传文件的时候，如果下载和主逻辑放在一个线程中会造成主界面卡死，因此需要对下载上传等命令单独开辟线程。由于FTP协议天然只能同时传输一个文件，因此在客户端上我设定了当有文件进行传输时，其余传输任务被拒绝。
- 很多操作对于文件与文件夹并不相同，因此我通过文件后缀和文件权限对其进行了区分

4.总结

本次试验任务量很大，其中还进行了延期，我对于socket和FTP的认知有了极大提高，收获也很多，尤其是发现自己的客户端和服务端可以和标准的server和client连接正常使用时，感到及其欣慰。