FTP 实验文档

实验环境

- OS：Ubuntu 18.04
- Python版本：Python 3.6
- Python第三方库：PyQt5

程序结构

Server

- global.h

  global.h头文件中主要定义了全局变量，包括用户结构体数组等

- utils.h/c

  utils文件中为程序的主要响应指令的逻辑处理模块。包括解析指令、将指令分配给相应的处理函数、处理函数给出返回响应等。

- network.h/c、

  network中主要定义了和网络连接相关的函数，包括接受、断开连接、完成数据连接的传输等。

- main.c

  main中定义的为程序的主循环，通过select函数非阻塞地检查每一个连接是否有新的可读数据，有则读取该数据，同时若程序发出可写的信号，也响应相应的写函数

Client

- ui.py

  client方面使用PyQt5完成，所以在ui.py中定义的是PyQt相关的UI组件以及UI的更新、输入获取等，并发出信号交由view处理逻辑以及指令发出

- view.py

  view中定义的为处理网络请求及返回的类，接受ui中发出的信号，通过PyQt特有的信号、槽机制实现响应信号并向server发送相关指令并接受返回，或者读取、写入数据。

FTP 指令及项目实现

- Server

  在本次实验中，实现的FTP指令主要包括USER、PASS、QUIT、SYST、TYPE、MKD、CWD、PWD、RMD、RNFR、RNTO、PASV、PORT、REST、RETR、STOR、LIST。Server端遵守FTP协议标准对客户端的输入进行处理、返回应答。为了解决server端的阻塞问题，程序采用了select方法对每一个与客户端之间的指令连接进行循环监控，若select发现某一个指令连接产生了可读数据，就意味着该客户端发送来了新的指令，接下来程序交由processMsg函数进行处理，函数首先对指令内容进行解析，若是合法指令，则将指令及指令内容分配至相应的处理函数。在处理函数内，程序根据指令内容产生返回值，并通过指令连接返回至客户端。同时对

于数据连接，则通过相应的用户结构体中的各种变量判断是否为数据传输状态以及当前的传输进度等，通过这些值在主循环中调用相应的数据传输函数，完成数据传输。

- Client

  Client端使用PyQt的信号槽机制，通过在响应程序中的按钮点击等，发出相应的指令信号，从而进入槽函数中，在槽函数得到指令内容后，将指令包装好并发送给Server并接受返回，根据返回值的类型，程序给用户不同的提示信息，或者开放、结束数据连接等。

项目难点及解决方案

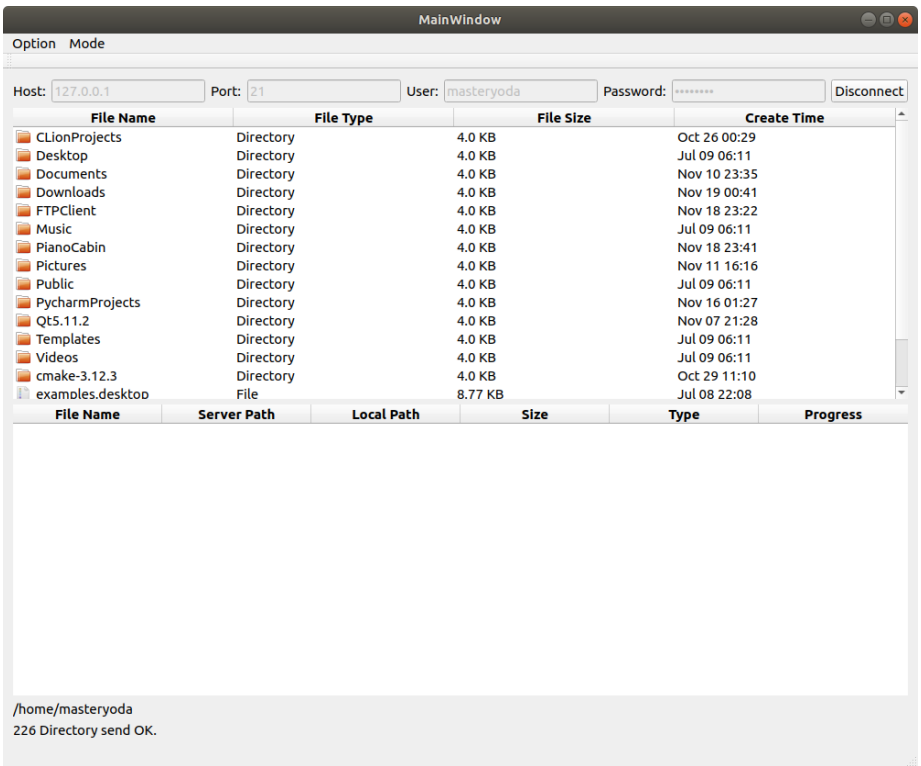
- 多用户连接以及非阻塞响应：

  Server端解决多用户接入以及文件传输的非阻塞方法是采用select函数监听各连接端口的可读数据，并通过一个用户结构体数组管理每一个用户的不同数据信息，包括其指令连接、数据连接的套接字等。在接收到指令时，首先获取相应的套接字的用户index，并将该index信息传入处理函数中，从而实现对不同用户连接的非阻塞处理。对于文件传输，则在每次主循环中传输定长的数据，通过不断的循环完成数据的传输，这期间同样能够完成对其他用户的指令的响应，保证了非阻塞性。

- 断点续传功能：

  断点续传功能通过FTP现有的REST指令实现，通过REST发送来的已传输数据大小，将文件指针调整至相应的位置，紧接着再通过RETR或者STOR指令继续进行相应的数据传输。

Client使用



- 首先使用Client一定要先进行连接，填入IP、端口、用户名及密码信息后，点击connect按钮即可完成连接
- 连接完成后，可以看到主程序分为两大主要窗口，上半部分显示当前目录的文件信息，在其中可以进行目录跳转、上传下载、新建、修改、剪切、删除目录等操作。其中目录跳转通过双击任意文件夹即可实现进入文件夹，通过点击子文件夹的".."目录即可回到父目录，而其他操作均通过在相应文件夹或文件上右键完成。下半部分窗口则是文件传输列表，显示目前未完成的文件传输内容。当有文件传输发生时，上半部分会被锁

定，不允许用户再进行相关的操作。用户通过右键传输项可以实现对传输项的控制，包括暂停、继续以及取消。

- PORT和PASV模式的切换放置于菜单栏的Option菜单中，点击PORT或PASV可以选择相应的模式。当选择PORT模式后，程序的HOST和PORT输入框会允许输入，用户完成输入后点击PORT按钮即可完成PORT连接，点击PASV则可再次切换回PASV模式。