FTP实验文档

2016013256 张佳麟

Server

- 实现指令

  除三个与数据传输相关的指令外，每个指令拥有一个对应的处理函数，在接收到客户端传来的请求时，利用`strtok`对原始字符串进行分割处理，通过指令调用不同的处理函数并将参数传入。

1. USER，PASS

   默认接受用户名为anonymous，密码为任意邮箱，利用正则表达式进行匹配。

2. QUIT

   断开客户端与服务器的连接。

3. TYPE，SYST

   SYST返回系统信息，TYPE I设置数据传输模式模式。

4. MKD，CWD，RMD，PWD

   进行目录的增删改查操作。

5. RNFR，RNTO

   通过两个连续指令来实现目标文件（夹）的路径以及名称更改。

6. PORT，PASV

   通过上述两个指令建立服务器与客户端之间的数据连接。

7. LIST

   列出服务器当前路径下的所有文件（夹）的相关信息。

8. RETR，STOR

   从服务器当前目录下载文件/上传本地文件到服务器当前目录。

9. REST

   在服务器处设置传输文件的偏移，必须与上述RETR，STOR联用。

- 重点难点

1. 多用户支持

   服务器中维护一个存有多个客户端信息的数组，每个客户端结构体中存有用于指令传输，监听，数据传输的描述符以及其它需要用到的信息，在服务器运行时，通过select对可读描述符进行选择，依次执行不同客户端发送的请求或数据。服务器运行时主循环伪代码如下：

```
while(1)
{
    select(max_fd+1, &fd_select, NULL, NULL, &timeout)
    是否有新客户端连入，有则建立连接;
    for(int i=0; i<MAX_CLIENT_NUM; ++i)
    {
        控制描述符i是否可读，可读则执行指令;
        监听描述符i是否可读，可读则建立数据连接并关闭监听;
        根据客户端结构体中的变量判断是否进行文件传输;
    }
}
```

服务器可最多支持`MAX_CLIENT_NUM`个客户端同时接入并及时响应请求。

2. 传输文件不阻塞服务器

   上述过程中，在传输文件时，并不会一次性传输完成整个文件而导致服务器被阻塞，而是在传输文件时一次读/写至多`bufsize`大小的数据并进行传输，完成一个`buf`的传输后即退出，直至下一次运行到对应client的数据传输处理函数再继续传输。当判断读/写已经完成后，会关闭数据传输并修改状态标志。

   注：根据ftp文档规定，当一个客户端与服务器之间正在传输文件时，服务器会忽略该客户端发送的其它请求直至数据传输完成。“不阻塞”在这个层面上的实现为服务器不会阻塞在单一文件的传输过程中，而不是在传输过程中继续响应同一客户端发出的其它指令。

3. 恢复传输

   服务器支持客户端用REST指令设置文件传输偏移量，并要求下一条指令一定为RETR或STOR，否则将偏移量恢复为0。通过REST和文件传输指令的配合使用来实现从指定位置继续文件的传输

Client

- 指令实现：与服务器实现指令相同
- 重点难点：

1. gui

   图形界面使用pyqt完成，且界面逻辑与ftp客户端逻辑基本分离，另外实现一个继承自`QObject`的类接受界面发送的信号（如按钮点击等）并根据不同的信号将一组指令发送至服务器，处理结果后再以信号的形式返回到界面。

2. 指令封装

   gui的客户端不要求用户直接输入具体指令，可点击的按钮，选项都封装了一组指令来完成对应的功能。封装情况如下：

   connect：初始化，包括USER，PASS，TYPE，PWD，LIST

   disconnect：QUIT

   系统信息：SYST

   剪切，粘贴/重命名：RNFR，RNTO

   删除：RMD

   创建文件夹：MKD

点击文件夹进入：CWD，LIST

上传/下载：STOR/RETR

继续传输：REST+STOR/RETR

3. 断点续传

   客户端维护一个数组，其中每一项都是一个字典，对应一个传输文件的相关信息，包括文件名，传输偏移，传输方向（上传/下载），传输状态（是否完成）等。用户在gui界面上可以选择当前尚未传输完成的任务继续进行，也可暂停当前正在传输的文件，开始其它任务，此类操作均会更新数组中的对应数据。