计算机网络课程设计

学生学院 计算机学院

专业班级 19 级软件工程（1）班

学　　号 3119005028

学生姓名 魏耀辉

指导教师 姜文超

2021 年 06 月 25 日

计算机网络课程设计任务书

设计题目	**编程实现 FTP 服务器 ★★★**
已知技术参数和设计要求	设计要求： 1.客户端通过 Windows 的命令行访问 FTP 服务器。 2.FTP 服务器可以并发地服务多个客户。 3.至少实现对 FTP 命令 user、pass、dir、get 的支持。即用户注册、显示服务器端的文件列表、下载文件等。 4.FTP 服务器必须对出现的问题或错误做出响应。
设计内容与步骤	1.参考相关的 RFC，熟悉 FTP 规范； 2.学习多线程机制； 3.FTP 服务器结构设计； 4.FTP 服务器程序设计； 5.FTP 服务器程序调试； 6.课程设计任务书。
设计工作计划与进度安排	1.Socket 程序设计 4 小时 2.程序调试调试方法 4 小时 3.FTP 规范 4 小时 4.FTP 服务器结构设计 4 小时 5.FTP 服务器程序设计与调试 14 小时 6.课程设计报告 5 小时

参考资料

1、RFC 0959 -File Transfer Protocol.

2、Computer Networking－ A Top Down Approach Featuring the Internet

目录

一、 设计目的.....4
二、 设计内容.....4
（一）要求完成的任务：.....4
三、 设计步骤.....4
（一） 需求分析.....4
（二） 概要分析.....5
1. 设计思想.....5
2. 抽象数据类型定义.....5
3. 主程序流程.....5
4. 工程结构.....6
（三） 详细设计.....7
1. 实现概要设计的功能.....7
（四） 调试分析.....8
1. 调试中遇到的问题.....8
2. 问题的解决方式.....8
（五） 系统测试.....9
（六） 使用说明.....11
四、 经验与体会.....12
五、 重要数据结构及疑难说明.....12

一、设计目的

FTP(File Transfer Protocol)是TCP/IP网络上两台计算机传送文件的协议，FTP是在TCP/IP网络和INTERNET上最早使用的协议之一，它属于网络协议组的应用层。FTP客户机可以给服务器发出命令来下载文件，上载文件，创建或改变服务器上的目录，一般我们均是将我们电脑中的内容与服务器数据进行性传输。其实电脑与服务器是一样的，只是服务器上安装的是服务器系统，并且服务器稳定性与质量要求高些，因为服务器一般放在诸如电信等机房中，24小时都开机，这样我们才可以一直访问服务器中的相关信息。FTP服务器上传下载文件方便，容易实现远程共享文件，使得网络文件共享变得简单。

二、设计内容

（一）要求完成的任务：

- 参考相关的RFC，熟悉FTP规范；
- 学习多线程机制；
- FTP服务器结构设计；
- FTP服务器程序设计；
- FTP服务器程序调试；
- 课程设计任务书。

三、设计步骤

（一）需求分析

- 客户端通过Windows的命令行访问FTP服务器。
- FTP服务器可以并发地服务多个客户。
- 至少实现对FTP命令user、pass、dir、get的支持。即用户注册、显示服务器端的文件列表、下载文件等。
- FTP服务器必须对出现的问题或错误做出响应。

（二）概要分析

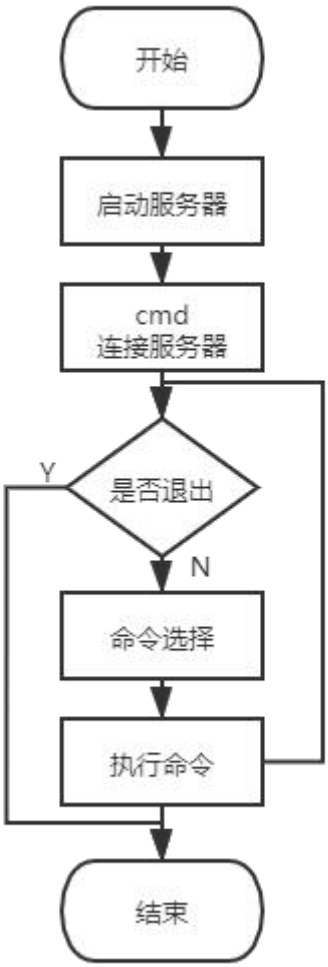
1. 设计思想

使用多线程来模拟多个用户进行对 FTP 服务器的操作，通过设计 FTP 服务器命令的接口来具体实现命令。通过用户类来管理提前存入的账号和密码，文件类进行文件大小的获取和文件的信息保存，User 类用于保存用户信息，Server 类和 ConnectionServer 类用于 FTP 服务器的搭建和监听。

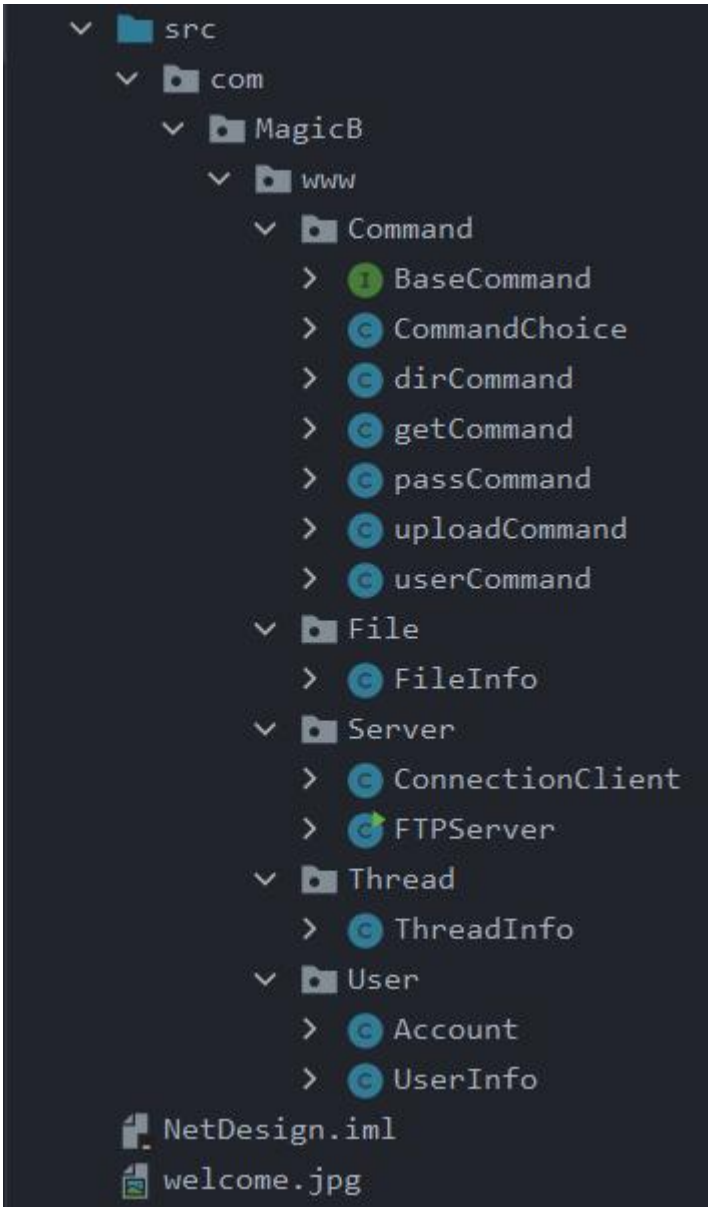
2. 抽象数据类型定义

本次课程设计中未涉及到抽象数据类型的定义

3. 主程序流程



4. 工程结构


src
com
MagicB
www
Command
BaseCommand
CommandChoice
dirCommand
getCommand
passCommand
uploadCommand
userCommand
File
FileInfo
Server
ConnectionClient
FTPServer
Thread
ThreadInfo
User
Account
UserInfo
NetDesign.iml
welcome.jpg

（三）详细设计

1. 实现概要设计的功能

```java
package com.MagicB.www.Command;

import com.MagicB.www.User.UserInfo;

import java.io.BufferedWriter;
//基础指令接口
public interface BaseCommand {
    public void Command(String data, BufferedWriter writer, UserInfo userInfo);
}
```

定义了基础命令接口，方便了后续命令的实现。

```java
package com.MagicB.www.Command;
//命令选择类
public class CommandChoice {
    public static BaseCommand commandChoice(String choice){
        BaseCommand command=null;
        switch (choice){
            case "user":{
                command=new userCommand();
                break;
            }
            case "pass":{
                command=new passCommand();
                break;
            }
            case "dir":{
                command=new dirCommand();
                break;
            }
            case "get":{
                command=new getCommand();
                break;
            }
            case "upload":{
                command=new uploadCommand();
                break;
            }
        }
        return command;
    }
}
```

通过命令选择类，来 new 对应命令的对象，只要将对应的命令接口完备就可以直接使用该命令。

（四）调试分析

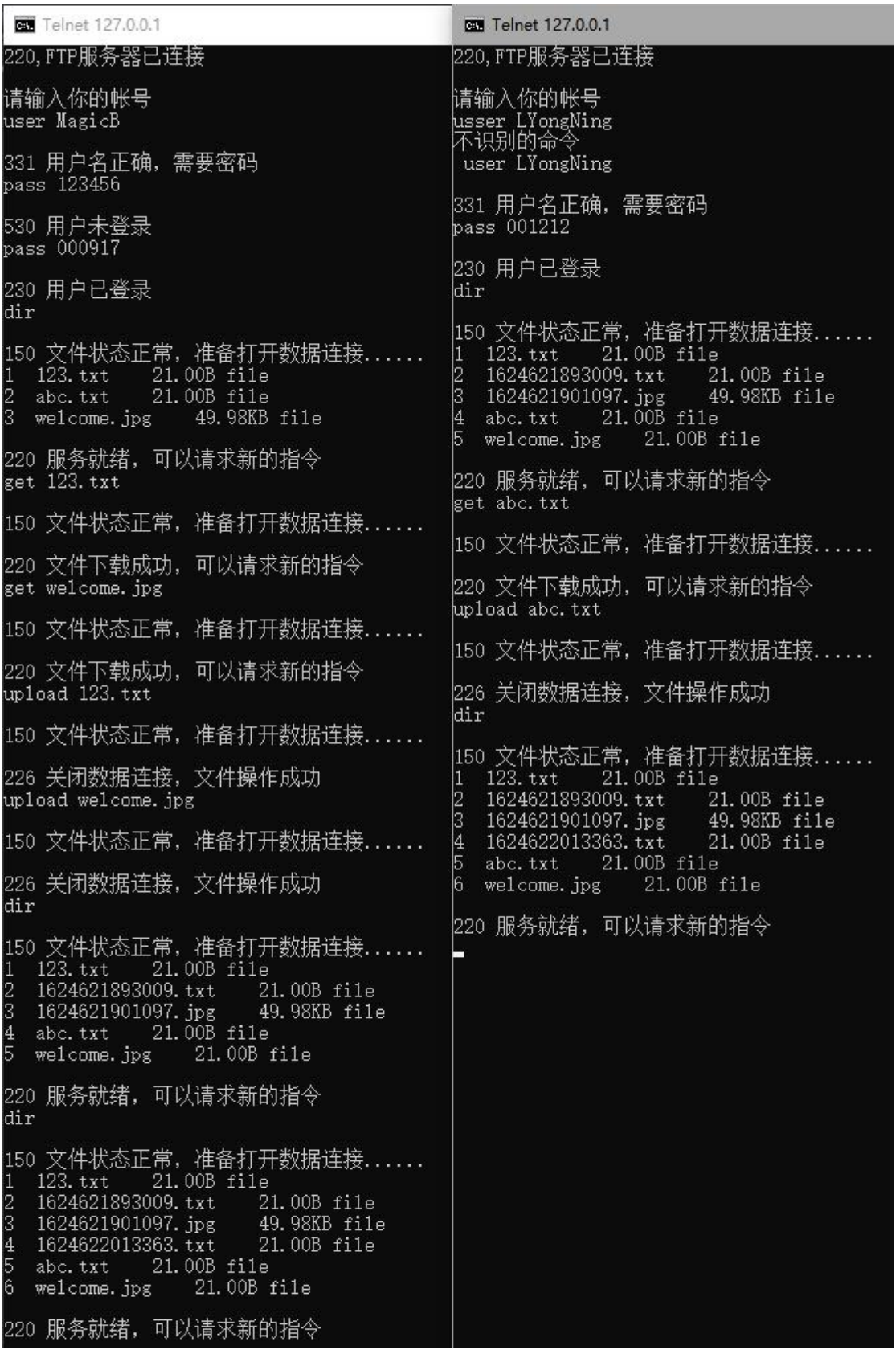
1. 调试中遇到的问题

①一开始打算用比较熟悉的 C 语言进行编写，但后来发现 C 语言针对多线程的部分比较难以实现，并不能达到一个很好的效果。②在使用 Java 后对多线程的使用也不太熟悉，在线程创建的时候不能很好的使用 Java 自带的方法。③基础命令类的编写过于繁琐，传入的参数不相同

2. 问题的解决方式

①转用对多线程编程更为友好的 Java 语言，只需要编写相应的类来实现线程的 Runnable 接口就可以实现了②上网查阅了线程 Thread 类中的使用方法，采用线程池进行管理线程③编写了基础命令的接口，只需要实现接口就可以方便的调用不同的命令

（五）系统测试

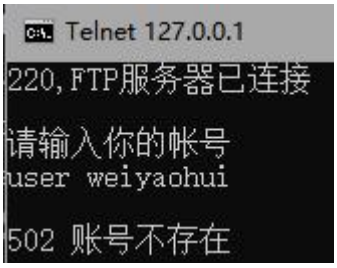

Telnet 127.0.0.1
220,FTP服务器已连接
请输入你的帐号
user MagicB
331 用户名正确，需要密码
pass 123456
530 用户未登录
pass 000917
230 用户已登录
dir
150 文件状态正常，准备打开数据连接.....
1 123.txt 21.00B file
2 abc.txt 21.00B file
3 welcome.jpg 49.98KB file
220 服务就绪，可以请求新的指令
get 123.txt
150 文件状态正常，准备打开数据连接.....
220 文件下载成功，可以请求新的指令
get welcome.jpg
150 文件状态正常，准备打开数据连接.....
220 文件下载成功，可以请求新的指令
upload 123.txt
150 文件状态正常，准备打开数据连接.....
226 关闭数据连接，文件操作成功
upload welcome.jpg
150 文件状态正常，准备打开数据连接.....
226 关闭数据连接，文件操作成功
dir
150 文件状态正常，准备打开数据连接.....
1 123.txt 21.00B file
2 1624621893009.txt 21.00B file
3 1624621901097.jpg 49.98KB file
4 abc.txt 21.00B file
5 welcome.jpg 21.00B file
220 服务就绪，可以请求新的指令
dir
150 文件状态正常，准备打开数据连接.....
1 123.txt 21.00B file
2 1624621893009.txt 21.00B file
3 1624621901097.jpg 49.98KB file
4 1624622013363.txt 21.00B file
5 abc.txt 21.00B file
6 welcome.jpg 21.00B file
220 服务就绪，可以请求新的指令
Telnet 127.0.0.1
220,FTP服务器已连接
请输入你的帐号
usser LYongNing
不识别的命令
user LYongNing
331 用户名正确，需要密码
pass 001212
230 用户已登录
dir
150 文件状态正常，准备打开数据连接.....
1 123.txt 21.00B file
2 1624621893009.txt 21.00B file
3 1624621901097.jpg 49.98KB file
4 abc.txt 21.00B file
5 welcome.jpg 21.00B file
220 服务就绪，可以请求新的指令
get abc.txt
150 文件状态正常，准备打开数据连接.....
220 文件下载成功，可以请求新的指令
upload abc.txt
150 文件状态正常，准备打开数据连接.....
226 关闭数据连接，文件操作成功
dir
150 文件状态正常，准备打开数据连接.....
1 123.txt 21.00B file
2 1624621893009.txt 21.00B file
3 1624621901097.jpg 49.98KB file
4 1624622013363.txt 21.00B file
5 abc.txt 21.00B file
6 welcome.jpg 21.00B file
220 服务就绪，可以请求新的指令

系统测试说明

1. 连接 FTP 服务器后显示 220，FTP 服务器已连接，表示本机已经连接到 FTP 服务器。

2. 同时使用两个 cmd 命令模拟多用户登录情况，如图所示可以同时登录。

3. 提示输入账号，若账号存在，返回 331，提示输入密码，若账号不存在，返回 502，账号不存在，重新输入账号



4. 输入账号对应的密码，若密码正确，返回 230，提示用户已登录，否则返回 530，用户未登录。

5. 使用 dir 命令可以查看 FTP 服务器目录下的文件，初始文件为 123.txt、abc.txt、welcome.jpg。

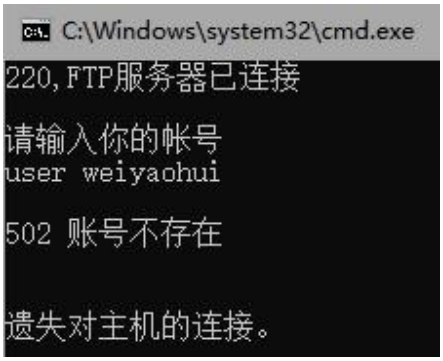
6. get 命令会返回 150.打开数据连接，下载完成后返回 220，可以请求新的指令。

7. upload 命令上传文件到 FTP 服务器，若文件名相同，则将上传的新文件名改为时间戳。

8. 无论哪一个用户上传了新的文件，另一个用户都能使用 dir 命令看到。

9. usser 不是命令，会提示不识别的命令。

10. 若断开 FTP 服务器，cmd 显示遗失对主机的连接



（六）使用说明

①点击启动 FTP 服务器按钮②cmd 中输入 telnet 127.0.0.1 25③输入对应需要的命令操作

四、经验与体会

本次的课程设计总体来说难度中上，在设计的前期由于第一次接触多线程编程而需要学习的成本导致前期的进度比较缓慢。回顾本次的课程设计，关键点在于将命令和账户文件等进行抽象处理，通过命令接口的定义让后续的命令类有了更好的实现。

本次课程设计加深了我对计算机网路中 TCP/IP 的思考与学习，对 Java 的网络 socket 编程加深了理解，并了解了 TCP/IP 协议在现实中的实现意义。另外，通过本次课程设计，学习到了 Java 的一些多线程的使用方法。

五、重要数据结构及疑难说明

user 命令类

```
public class userCommand implements BaseCommand{
    @Override
    public void Command(String username, BufferedWriter writer, UserInfo userInfo) {
        if(Account.hasUser(username)){//调用Account中的hasUser方法查找是否有该用户
            System.out.println(username+" 该用户存在");
            try{
                writer.write("\n331 用户名正确，需要密码\r\n");
                writer.flush();
                userInfo.setName(username);
            }
            catch (IOException e){
                e.printStackTrace();
            }
        }
        else{
            System.out.println("用户不存在");
```

```
            try{
                writer.write("\n502 账号不存在\r\n");
                writer.flush();
            }
            catch (IOException e){
                e.printStackTrace();
            }
        }
    }
}
```

pass 命令类

```
public class passCommand implements BaseCommand{
    @Override
    public void Command(String password, BufferedWriter writer, UserInfo
userInfo) {

if(password.equals(Account.getPassword(userInfo.getName()))){//判断密
码是否和该用户名存的密码一致
            System.out.println(password+"密码正确");
            try{
                writer.write("\n230 用户已登录\r\n");
                writer.flush();
                userInfo.setPassword(password);
            }
            catch (IOException e){
                e.printStackTrace();
            }
        }
```

```
        else{
            System.out.println(password+"密码不正确");
            try{
                writer.write("\n530 用户未登录\r\n");
                writer.flush();
            }
            catch (IOException e){
                e.printStackTrace();
            }
        }
    }
}
```

dir 命令类

```
//服务器目录下全部文件
public class dirCommand implements BaseCommand{
    @Override
    public void Command(String data, BufferedWriter writer, UserInfo
userInfo) {
        File file=new File(Account.getRootDir());
        //如果目录下没有文件
        if(!file.isDirectory()){
            try{
                writer.write("\n450 文件不存在\r\n");
                writer.flush();
            }
            catch (IOException e){
                e.printStackTrace();
            }
```

```
        }
        //如果文件存在，输出文件字符串
        else{
            StringBuffer dirList=new StringBuffer();
            int count=1;
            for(String item: file.list()){
                File itemFile=new File(file+File.separator+item);
                String fileSize= FileInfo.getFileSize(itemFile);//调用
文件信息类中的获取文件大小方法
                if(fileSize.equals("")||fileSize==null){
                    fileSize="dir";
                }
                else{
                    fileSize+=" file";
                }
                dirList.append(count+"  "+item+"    "+fileSize);
                dirList.append("\r\n");
                count++;
            }
            try{
                writer.write("\n150 文件状态正常，准备打开数据连
接......\r\n");
                writer.flush();
                writer.write(dirList+"\r\n");
                writer.write("220 服务就绪，可以请求新的指令\r\n");
                writer.flush();
                System.out.println(dirList.toString());
            }
            catch (IOException e){
```

```
                e.printStackTrace();
            }
        }
    }
}
```

get 命令类

```
public class getCommand implements BaseCommand{
    @Override
    public void Command(String datas, BufferedWriter writer, UserInfo
userInfo) {
        File file = new
File(Account.getRootDir()+File.separator+datas);
        File rootFile = new
File(System.getProperty("user.home")+File.separator);
        File lastFileName = new
File(System.getProperty("user.home")+File.separator+datas);
        String name = System.getProperty("user.home")+"\\"+datas;
        if (file.exists()) {
            //检测文件是否有同名的文件
            for (String item:rootFile.list()){
                item = System.getProperty("user.home")+"\\"+item;
                //获取文件名
                if (item.equals(name)){
                    //如果文件重名，使用时间的方法来重命名文件
                    File temp = new
File(System.getProperty("user.home")+"\\"+
Calendar.getInstance().getTimeInMillis()+"."+item.substring(item.last
IndexOf(".")+1));
```

```
                lastFileName.renameTo(temp);
            }
        }
        try {
            writer.write("\n150 文件状态正常，准备打开数据连接......\r\n");
            writer.flush();
            Socket socket = new Socket("127.0.0.1", 25);
            OutputStream outputStream = socket.getOutputStream();
            FileInputStream inputStream = new FileInputStream(file);
            int length;
            byte[] buff = new byte[1024];
            while((length = inputStream.read(buff))!=-1){
                outputStream.write(buff, 0, length);
            }
            //文件传输完成后，关闭输入输出流，关闭 socket 服务
            inputStream.close();
            outputStream.close();
            socket.close();
            writer.write("\n220 文件下载成功，可以请求新的指令\r\n");
            writer.flush();
        }
        catch (IOException e) {
            e.printStackTrace();
        }

    }
```

```
        else{
            try {
                writer.write("\n220 文件不存在，可以请求新的指令
\r\n");
                writer.flush();
            }
            catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

upload 命令类

```
public class uploadCommand implements BaseCommand{
    @Override
    public void Command(String data, BufferedWriter writer, UserInfo
userInfo) {
        File file=new File(Account.getRootDir());
        try{
            writer.write("\n150 文件状态正常，准备打开数据连
接......\r\n");
            writer.flush();
            File                                    lastFileName=new
File(Account.getRootDir()+"\\"+data);
            String name=Account.getRootDir()+data;
            for(String item: file.list()){
                item=Account.getRootDir()+item;
                if(item.equals(name)){
```

```
                    File    tempName=new    File(Account.getRootDir()+
Calendar.getInstance().getTimeInMillis()+"."+item.substring(item.last
IndexOf(".")+1));
                    lastFileName.renameTo(tempName);
                    System.out.println(lastFileName);
                }

            }
            RandomAccessFile                              uploadFile=new
RandomAccessFile(lastFileName,"rw");
            Socket tempSocket=new Socket("127.0.0.1",25);
            InputStream uploadSocket=tempSocket.getInputStream();
            byte []Buffer=new byte[1024];
            int length;
            if((length=uploadSocket.read(Buffer))!=-1){
                uploadFile.write(Buffer,0,length);
            }
            System.out.println("文件传输成功");
            uploadFile.close();
            uploadSocket.close();
            tempSocket.close();
            writer.write("\n226 关闭数据连接，文件操作成功\r\n");
            writer.flush();
        }
        catch (IOException e){
            e.printStackTrace();
        }
    }
}
```