



**课 程 设 计**

课程名称\_\_\_\_\_**计算机网络** \_\_\_\_

题目名称\_ **编程实现FTP服务器(java)**

学生学院\_\_\_\_\_**计算机学院**\_\_\_\_\_\_\_\_\_

专业班级\_\_\_**17级计算机科学与技术(5班)\_**

学 号 **3117004622**

学生姓名\_\_\_\_\_\_\_\_\_**刘友滨**\_\_\_\_\_\_\_\_\_\_

指导教师\_\_\_\_\_\_ 何翠红\_\_\_\_\_\_\_\_\_\_\_

2019 年 5 月 20 日

目录

[一. 设计思想说明 3](#_Toc32091)

[1.1设计环境 3](#_Toc16357)

[1.2设计思想 3](#_Toc31944)

[二 系统结构 4](#_Toc19853)

[2.1功能模块图 4](#_Toc17954)

[2.2 系统流程图 5](#_Toc22567)

[三 关键程序说明 6](#_Toc14717)

[3.1 Main方法的以及所在类的说明 6](#_Toc28470)

[3.2 用户处理类(ClientDeal)说明 7](#_Toc8460)

[3.3 Global类说明 10](#_Toc12792)

[四 实现界面和功能以及屏幕截图 12](#_Toc28061)

[4.1实现的功能 12](#_Toc2655)

[4.2 登陆界面截图 13](#_Toc13363)

[五 调试过程中遇到的问题和解决的方法 14](#_Toc10459)

[六 测试与分析 14](#_Toc23851)

[6.1 测试登陆命令(USER \PASS) 14](#_Toc3920)

[6.2 测试文件夹的创建与删除（MKDIR \EMDIR） 15](#_Toc14800)

[6.3 测试查询/切换服务器当前工作目录命令（PWD\ CD） 16](#_Toc11508)

[6.4 测试查询文件以及文件夹的命令(DIR \LS) 17](#_Toc6988)

[6.5 测试删除服务器文件命令(DELETE) 18](#_Toc23017)

[6.6 测试文件上传命令(PUT) 19](#_Toc7762)

[6.7 测试文件下载命令(GET) 20](#_Toc14538)

[6.8 测试断开连接命令(CLOSE) 21](#_Toc14587)

[七 用户使用说明 21](#_Toc30089)

[7.1登陆说明 21](#_Toc17581)

[7.2 使用说明 21](#_Toc21579)

[八 总结 23](#_Toc19454)

[九 附录 24](#_Toc22574)

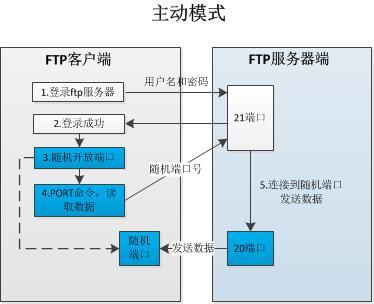
## 设计思想说明

### 1.1设计环境

本次课程设计是采用java语言实现FTP服务器，服务器的设计参考FTP的标准规范，开发环境是基于jdk8，开发软件采用eclipse。 客户端是基于windows命令行。

### 1.2设计思想

FTP服务器的有两种模式，一种模式是Standard（主动模式）、一种模式是Passive（被动模式）。本次课设仅实现主动模式，下面介绍主动模式的原理。



1.2.1主动模式的原理

Ftp服务器开启**21端口**监听客户端的连接请求，客户端发起请求与服务器的21端口建立tcp连接（注意客户端的连接端口号必须大于1024）。此时建立起来的连接是命令连接，该连接用于客户端给发送ftp命令以及接收来自客户端的响应信息，这个连接会一直维持客户端与服务器的通信。当客户端需要与服务器之间传输数据时，客户端会随机开一个端口号并向服务器发送PORT命令（包含客户端的ip地址与端口号），服务器接收到命令后会用**20端口**与客户端的指定端口号建立连接，此连接用于传输数据，数据传输完毕后此连接就会关闭。

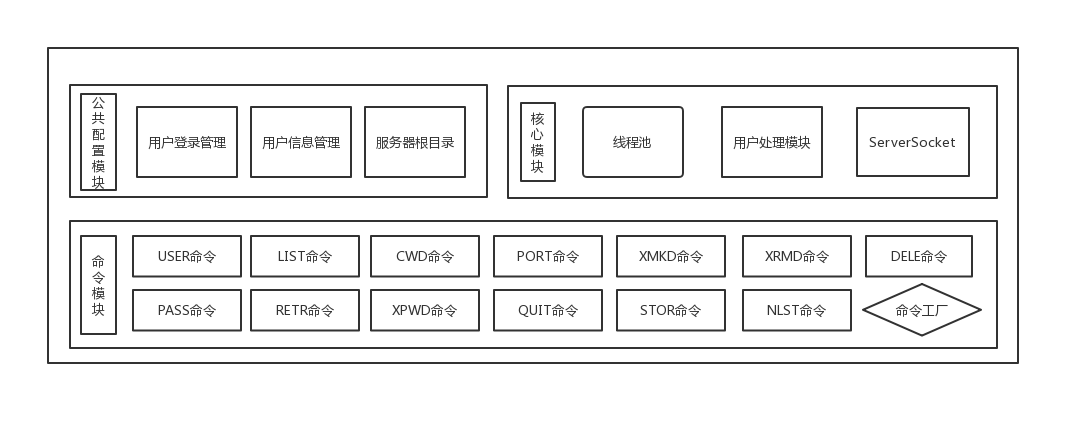
#### 1.2.2 Java层面的设计思想

基于ftp服务的主动模式原理以及规范，遵循面向对象设计原则以及软件开发低耦合原则，本次开发采用了如下的设计。

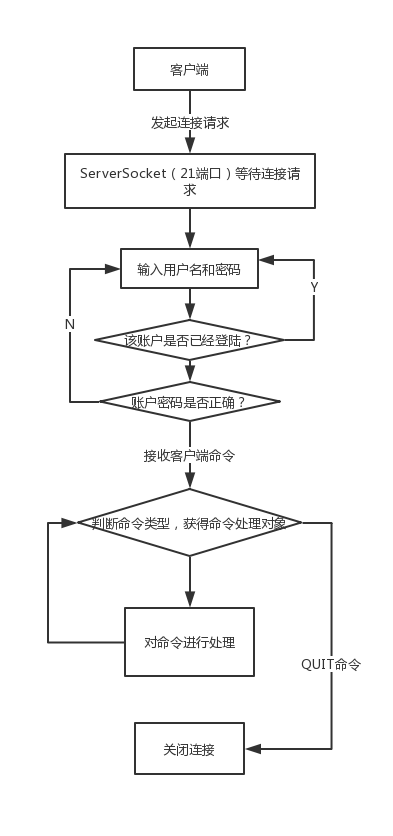
1. 采用java的Socket Api实现网络的连接通信。
2. 采用多线程机制（线程池）实现并发地服务多个用户。
3. 采用简单工厂的设计模式，将代码的耦合度降低，便于修改复用。
4. 服务器维护了一个ServerSocket用于接收客户端的连接请求。接收到请求后将获得的socket连接交给用户处理类（ClientDeal）的一个对象并将此对象交给线程池进行管理。
5. 用户处理类的对象主要负责维护客户连接的一些信息，以及接收客户端的命令，并通过命令工厂获得对象的命令处理对象进行处理。
6. 同时设计了一个Global类用于管理公共信息，如：客户的账号信息。

## 二 系统结构

### 2.1功能模块图



### 2.2 系统流程图



## 三 关键程序说明

### 3.1 Main方法的以及所在类的说明

|  |
| --- |
| public class FtpServer {    //开启一个线程池,用于并发的处理用户连接请求  private static ExecutorService ThreadPool = Executors.newFixedThreadPool(Constant.THREADPOOLSIZE);  //声明一个服务器套接字，用于接收响应客户端的请求连接  public static ServerSocket server = null;    public static void main(String[] args) throws IOException {  //开启ServerSocket服务. 监听客户端的请求  server = new ServerSocket(Constant.LISTENPORT);  //初始化ftp服务全局参数  Global.initGlobalMessgae();    //循环不断监听连接请求的到来  while(true) {  //阻塞地获取连接  Socket socket = server.accept();  //将客户端连接交给线程池管理  ThreadPool.execute(new ClientDeal(socket));  }      }  } |

**说明：**

1. 声明创建了一个ServerSocket套接字，用于监听客户端的连接。这个套接字所

用的的端口为默认的命令端口（20),也就是这个套接字所获得的连接是命令连接。

1. ServerSocket每accept一个Socket。程序就会创建一个用户处理类的对象处理这个Socket。
2. Main方法还通过维持了一个线程池实现并发处理用户请求。

### 3.2 用户处理类(ClientDeal)说明

|  |
| --- |
| public class ClientDeal implements Runnable{    @SuppressWarnings("unused")  //客户端与服务器端的连接  private Socket socket = null;  private InputStream ips = null;  private OutputStream ous = null;  private BufferedReader reader = null;  private BufferedWriter writer = null;    //当前连接对应的用户名  private String name = null;    //判断是否已经登陆  private volatile boolean isLogin = false;    //当前服务器工作目录  private String currentPath = Global.rootPath;    //客户端的ip地址  private String ip = null;    //客户端数据传输端口  private int port;    //记录是否第一次访问  private boolean isFirstConn = true;    //省略 set get方法  public ClientDeal(Socket socket) {  try {  this.socket = socket;  //获取套接字的输入输出流  ips = socket.getInputStream();  ous = socket.getOutputStream();  } catch (IOException e) {  CloseUtil.close(ips);  CloseUtil.close(ous);  CloseUtil.close(socket);  }  }    /\*\*  \* 当用户调用quit命令是时调用此方法进行连接清理工作  \*/  public void clear() {  CloseUtil.close(reader);  CloseUtil.close(writer);  CloseUtil.close(ips);  CloseUtil.close(ous);  CloseUtil.close(socket);  }      @Override  public void run() {  try {  reader = new BufferedReader(new InputStreamReader(ips,"GB2312"));  writer = new BufferedWriter(new OutputStreamWriter(ous,"GB2312"));  } catch (UnsupportedEncodingException e1) {  e1.printStackTrace();  }    while(!socket.isClosed()) {  //第一次访问(连接成功) 提示用户  if(isFirstConn) {  try {  writer.write("220 连接ftp服务器成功 .---------------\r\n");  writer.flush();  } catch (IOException e) {  e.printStackTrace();  }  //标志为已经访问过  isFirstConn = false;  }else {    if(!socket.isClosed()) {  //接收客户端的命令  String command = null;  try {  //阻塞地等待客户端命令  command = reader.readLine();  } catch (IOException e) {  clear();  e.printStackTrace();  }  System.out.println(command);    if(command!=null) {//获取到命令command  //分割命令  String[] datas = command.split(" ");  //通过命令工厂获得响应的命令处理类  Command commandDeal = CommandFactory.getCommand(datas[0]);    //一开始在没有登陆的情况下只能使用 user pass 命令  if(isLogin||commandDeal instanceof UserCommand||commandDeal instanceof PassCommand) {  //判断命令是否存在  If(commandDeal==null) {  try {  writer.write("503 命令不存在，请重新输入 .---------------\r\n");  writer.flush();  } catch (IOException e) {  e.printStackTrace();  }    }else {  //命令存在则 获取命令的数据信息  String data = "";  if(datas.length>=2) {  data = datas[1];  }  //调用命令处理类的处理方法  try {  commandDeal.deal(writer, data, this);  } catch (IOException e) {  e.printStackTrace();  }  }    }else {//未登陆状态  try {  writer.write("530 执行该命令需要先登陆 .---------------\r\n");  writer.flush();  } catch (IOException e) {  e.printStackTrace();  }  }  }    }else {  //连接断开则退出循环  break;    }  }  }  }  } |

**说明：**

1. 用户处理类存放了一个用户连接的信息，比如用户ip地址、数据传输端口号、用户名、 登 陆标志、Socket对象......
2. 在run方法中实现的功能是接收客户端命令，通过命令工厂获得对应的命令处理对象，处理客户请求

### 3.3 Global类说明

|  |
| --- |
| public class Global {    //记录存在的用户 用户名---密码  public static Map<String,String> users = new ConcurrentHashMap<>();    //记录已经登陆的用户  public static Set<String> loginUsers = new HashSet<>();    //服务器根目录  public static String rootPath = new File("").getAbsolutePath();    /\*\*\*  \* 初始化全局参数  \*/  public static void initGlobalMessgae() {  DocumentBuilderFactory df = DocumentBuilderFactory.newInstance();  DocumentBuilder db;  Document doc = null;  try {  //读取配置文件  db = df.newDocumentBuilder();  doc = db.parse("src/config.xml");    } catch (ParserConfigurationException e) {  e.printStackTrace();  } catch (SAXException e) {  e.printStackTrace();  } catch (IOException e) {  e.printStackTrace();  }  //获取根节点  Element root = doc.getDocumentElement();    /\*\*\*  \* 获取用户信息  \*/  NodeList list = root.getElementsByTagName("user");  for(int i=0;i<list.getLength();i++) {  Node n = list.item(i);  //判断是否为真实节点  if(n.getNodeType()==Node.ELEMENT\_NODE) {  String name = null;  String pwd = null;  //获取用户具体信息  NodeList ulist = n.getChildNodes();  for(int j=0;j<ulist.getLength();j++) {  Node ns = ulist.item(j);  //判断是否为真实节点  if(ns.getNodeType()==Node.ELEMENT\_NODE) {  if(ns.getNodeName().equals("name")) {  name = ns.getTextContent();  }else if(ns.getNodeName().equals("pwd")) {  pwd = ns.getTextContent();  }  }    }  //将已配置的用户信息保存到map容器中  users.put(name, pwd);  }  }    /\*\*  \* 获取服务器目录  \*/  NodeList dirList = root.getElementsByTagName("root");  Node dir = dirList.item(0);  rootPath = dir.getTextContent();    }    } |

**说明：**

1. Globla主要是存放程序运行的公共信息 比如 合法的用户信息、已登录的用户、服务器根目录
2. 合法用户的信息 以及 服务器根目录都是存放在src目录下的config.xml配置文件中，通过DocumentBuilder 获取得到信息

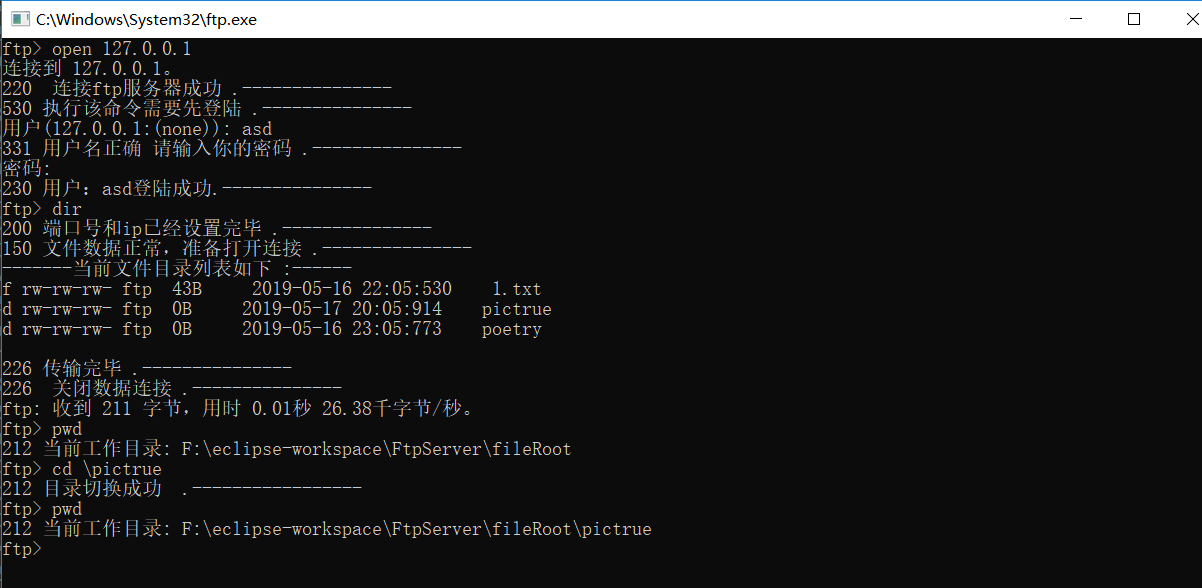
## 四 实现界面和功能以及屏幕截图

### 4.1实现的功能

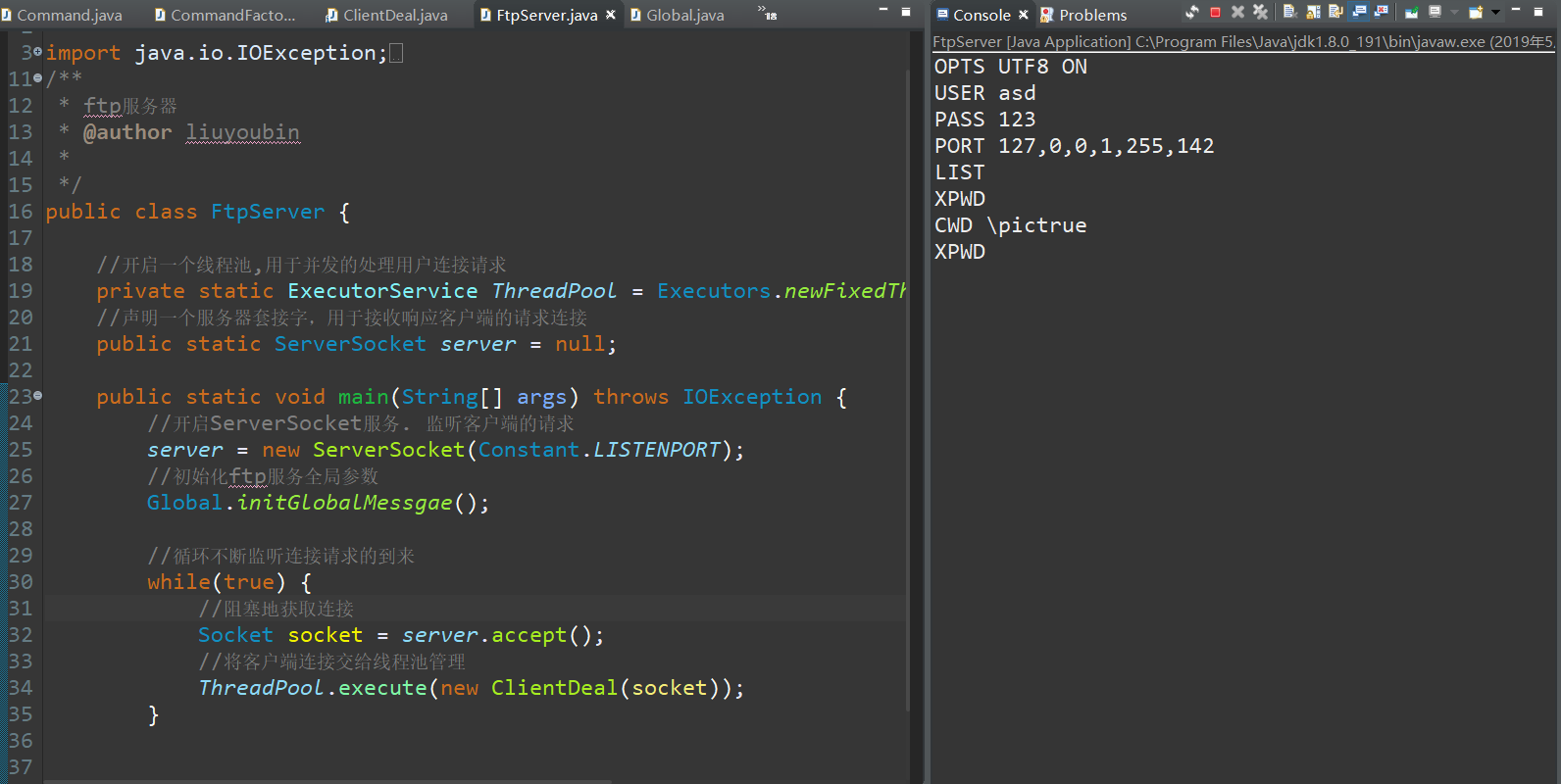
1. 用户可以通过windows命令行登陆 使用**user** 和 **pass** 命令输入正确的账号密码就可以登陆
2. 用户可以使用**pwd** 命令确定当前所在的服务器工作目录 ，使用**cd**命令切换服务器工作目录
3. 用户可以使用**midir**在服务器当前目录下建立目录，使用**rmdir**删除指定的空目录
4. 用户可以使用**get**命令从服务器上下载文件到指定的本地目录下，可以使用**put**命令上传本地文件到服务器指定的目录下，可以使用**delete**命令删除服务器上的指定文件
5. 用户可以使用**dir** 查看指定目录下的文件夹以及文件，使用**ls**命令查看指定目录下的所有文件

### 4.2 登陆界面截图

**客户端通过windows命令行登陆界面**



**服务器运行界面**



## 五 调试过程中遇到的问题和解决的方法

**问题一：**在接收到客户端请求后，客户端一直处于接收信息的状态无法发送命令，而服务器则一直阻塞

**原因：**通过调试发现服务器是阻塞在通过socket获得的输入流之中，而客户端则是一直处于接收信息的状态。在查看ftp规范后才明白原来客户端是通过状态码来确定一次通信是否完成，只有接收到状态码才能进行下一条命令的发送否则就会一直处于信息接收状态。

**解决办法：**在每一次命令交互之后向客户端发送条带状态码的信息

**问题二：**接收到客户端的port命令后进行设置，在之后的程序运行时会抛出端口号溢出的异常。

**原因：**在ftp的规范中中规定Port命令后六个数字用逗号隔开,其中前四个数字组成IP地址,第5个数字乘以256再加上第6个数字为实际的端口号。然而在程序中直接将后两个数字拼接成端口号

**解决办法：**修改获得端口号的代码逻辑

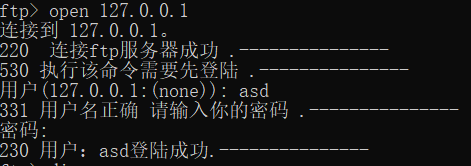
## 六 测试与分析

### 测试登陆命令(USER \PASS)

**测试一：**

输入：正确的账号密码 asd/123

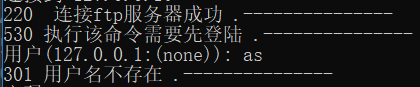
结果：登陆成功

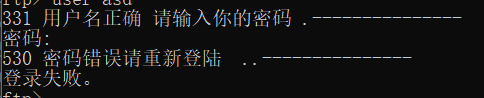


**测试二：**

输入：错误的账号密码 as/12

结果：登陆失败





**测试三：**

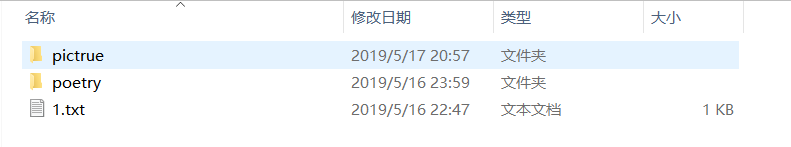
输入：已登录的账号 asd

结果：登陆失败



### 测试文件夹的创建与删除（MKDIR \EMDIR）

**创建文件夹前文件目录情况：**



**使用midir命令创建文件夹aaa:**



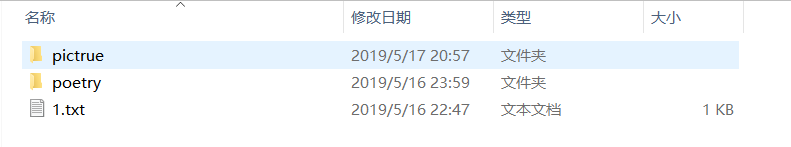
**创建文件夹后文件目录情况：**



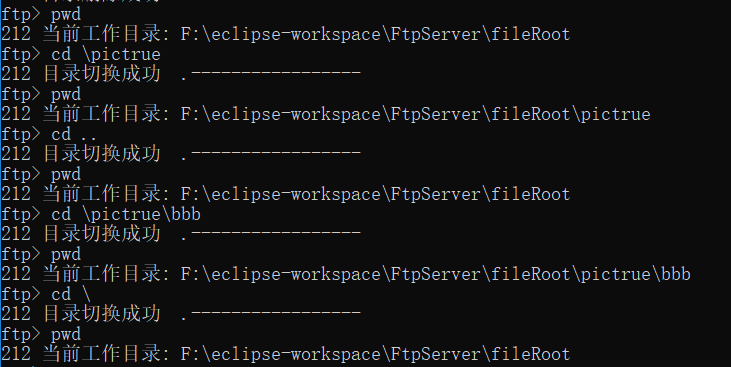
**使用rmdir删除文件夹aaa：**



**删除文件夹后文件目录情况：**



### 测试查询/切换服务器当前工作目录命令（PWD\ CD）



**分析：**

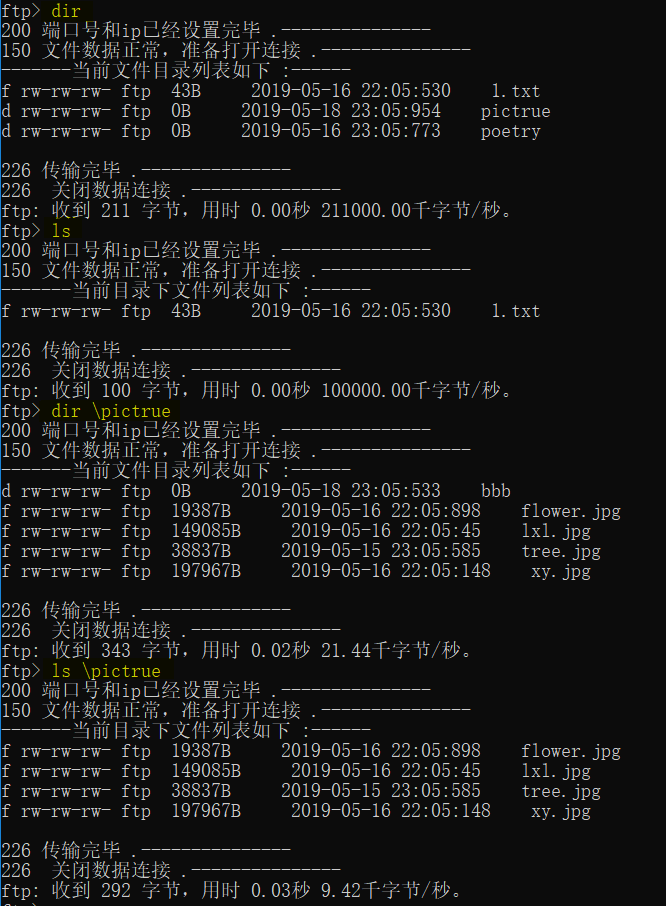
1. **通过上面的测试可以知道pwd命令可以用来查询服务器当前工作目录**

**2. cd 命令加上“\文件夹名称” 可以切换到指定子目录**

**3. cd命令加上“..” 可以切换到上一级目录**

**4. cd命令加上 “\” 可以切换到服务器根目录**

### 测试查询文件以及文件夹的命令(DIR \LS)

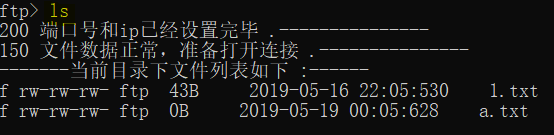


**分析：**

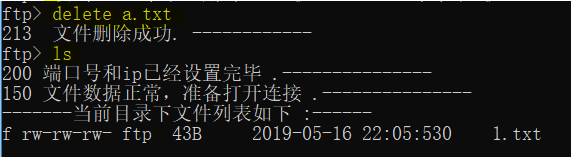
1. **dir命令可以查询指定目录下的所有文件夹以及文件，当不带参数时查询当前工作目录**
2. **Ls命令可以查询指定目录下的所有文件，当不带参数时查询当前工作目录**

### 测试删除服务器文件命令(DELETE)

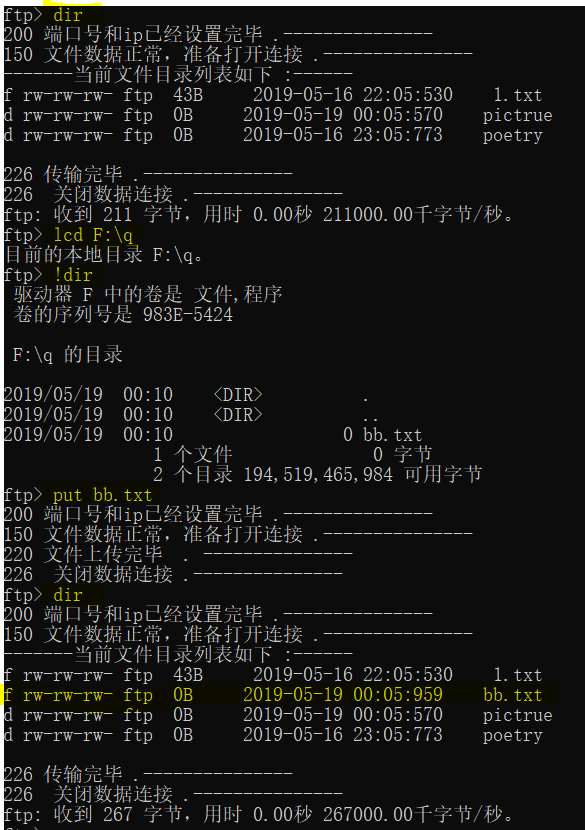
**文件删除前：**



**删除文件a.txt:**



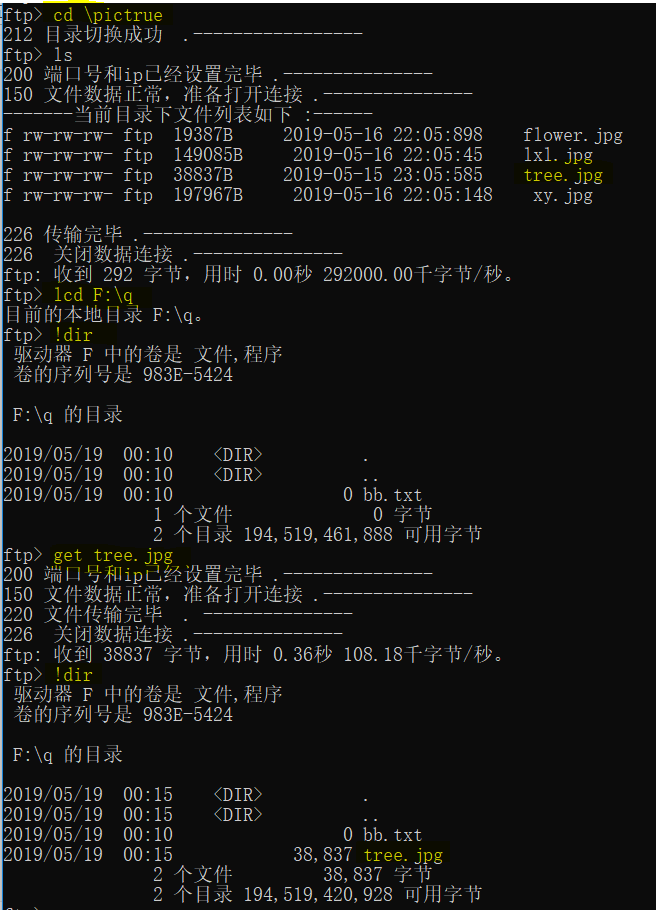
### 测试文件上传命令(PUT)



**分析：**

**在上传文件请需要先用lcd命令定位到客户端的目录下，然后使用put命令+指定文件名称，将文件上传到服务器工作目录下。可以使用！dir命令查看客户端当前目录下的文件情况**

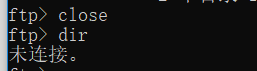
### 测试文件下载命令(GET)



**分析：**

**在从服务器下载文件之前，需要先用lcd命令指定接收文件的目录，然后使用get+指定文件名称 将文件下载到客户端指定目录下**

### 测试断开连接命令(CLOSE)



## 七 用户使用说明

### 7.1登陆说明

**本次实现的ftp服务器是基于widows命令行的，用户需要使用windows命令行窗口进行登陆使用**

**登陆步骤：**

1. **打开windows自带的ftp命令行**

**2. 输入 open [服务器ip地址] 21 连接服务器**

**Ps: 端口号如果不填则默认 21**

**3.按照提示输入合法的用户名和密码即可登陆，用户名和密码在config.xml文件中配置**

**Ps:如果第一次登陆失败，之后则需要手动使用USER 和 PASS命令登陆**

### 7.2 使用说明

**本次ftp服务器仅实现了主动模式的部分指令，指令的使用方式如下：**

|  |  |  |
| --- | --- | --- |
| **命令** | **参数说明** | **作用** |
| **USER 参数** | **用户名** | **登陆服务器时使用，用于向服务器传输用户名** |
| **PASS 参数** | **密码** | **登陆服务器时使用，用于向服务器传输密码** |
| **MKDIR 参数** | **文件夹名称** | **在当前服务器工作目录下创建指定的文件夹，文件夹名称不能与当前目录下的文件夹重名** |
| **RMDIR 参数** | **文件夹名称** | **删除当前服务器工作目录下的指定空文件夹，文件夹必须为空** |
| **PWD** | **-** | **查看服务器当前工作目录** |
| **CD 参数** | **\ : 切换到服务器根目录**  **.. : 切换到上一级目录，但不能跳出根目录**  **\文件夹名 :切换到当前目录的指定子目录下，可以指定多级目录** | **切换到指定的服务器工作目录** |
| **DIR [参数]** | **指定的子目录，必须以 “ \文件夹名\文件夹名” 的形式**  **Ps :当参数为空时，指定的目录为服务器当前工作目录** | **查询指定目录下的所有文件和文件夹的详细信息** |
| **LS [参数]** | **指定的子目录，必须以 “ \文件夹名\文件夹名” 的形式**  **Ps :当参数为空时，指定的目录为服务器当前工作目录** | **查询指定目录下的所有文件的详细信息** |
| **DELETE [参数]** | **文件名称** | **删除服务器当前工作目录的指定文件** |
| **GET 参数** | **文件名称** | **下载服务器当前工作目录的指定文件到本地目录下，需要配合lcd命令使用** |
| **PUT 参数** | **文件名称** | **上传当前本地目录的指定文件到服务器的工作目录下，需要配合lcd命令使用** |
| **CLOSE** | **-** | **客户端与服务器断开连接** |
| **LCD 参数** | **本地文件目录** | **指定到本地的文件目录** |
| **！DIR** | **-** | **查看当前本地目录下的所有文件和文件夹是详细信息** |

## 八 总结

**特色与优点**

**本次课程设计采用了java语言设计FTP服务器，在开发过程中注重面向对象的开发思想，在代码的设计上可以体现面向对象的思想。例如：在工具类的开发上体现了封装思想，在命令的设计上体现了多态。在开发中还合理地利用jdk的API，例如：使用线程池管理线程。**

**同时还采用了一定的设计模式来进行代码优化。比如在ftp命令处理类对象的获取上，采用了简单工厂模式来进行获得。通过简单工厂模式将获得的逻辑判断与业务分开，提高了代码的可读性与复用性，降低了代码耦合度。**

**另一方面，本次课程设计还将用户的信息与代码分开，同过配置文件进行管理。**

**问题与不足**

**本次的课程设计仅仅只是实现了FTP服务器的主动模式中的一部分命令，还有很大的扩展空间。**

**同时在代码的设计上还不够完善，例如：在命令的处理设计上只能接受一参数的指令，不够灵活。**

**进一步的展望与设想**

**在以后可以接着实现更多的指定，实现被动传输模式。同时也可以对并发进行优化，让服务器能够有更优的性能。**

## 九 附录

**com.lyb.command:**

**Command.java**

**CommandFactory.java**

**CwdCommand.java**

**DeleCommand.java**

**DirCommand.java**

**GetCommand.java**

**LsCommand.java**

**MkdirCommand.java**

**PassCommand.java**

**PortCommand.java**

**PwdCommand.java**

**QuitCommand.java**

**RmdirCommand.java**

**StorCommand.java**

**UserCommand.java**

**com.lyb.constant:**

**Constant.java**

**com.lyb.global:**

**Global.java**

**com.lyb.server:**

**FtpServer.java**

**ClientDeal.java**

**com.lyb.util:**

**CloseUtil.java**

**StringUtil.java**

**TimeUtil.java**