

**计算机网络及应用**

**编程实践**

**基于socket接口实现ftp协议**

|  |  |
| --- | --- |
| 成员： | 陈小灵 2000022715 |
|  | 陈彦冰 2000022772 |

目录

[**1 题目介绍** 3](#_Toc75604363)

[**2 人员分工** 3](#_Toc75604364)

[**3 设计** 3](#_Toc75604365)

[**3.1 应用系统基本功能与附加功能** 3](#_Toc75604366)

[**3.2 应用协议描述** 4](#_Toc75604367)

[**3.3 应用协议实现设计** 6](#_Toc75604368)

[**3.4 客户端C设计** 7](#_Toc75604369)

[**3.5 服务器S设计** 8](#_Toc75604370)

[**4 实现** 9](#_Toc75604371)

[**4.1 应用协议实现** 9](#_Toc75604372)

[**4.2 客户端C实现** 10](#_Toc75604373)

[**4.3 服务器S实现** 13](#_Toc75604374)

[**5 测试** 14](#_Toc75604375)

[**5.1 测试环境** 14](#_Toc75604376)

[**5.2 测试计划** 15](#_Toc75604377)

[**5.3 测试用例列表** 15](#_Toc75604378)

[**6 附录** 32](#_Toc75604379)

[**6.1 参考说明** 32](#_Toc75604380)

[**6.2 问题说明** 32](#_Toc75604381)

**1 题目介绍**

本次编程实践选取的题目为基于socket接口实现ftp协议，该题目具体要求包括：1）实现上传文件功能；2）实现下载文件功能；3）实现删除文件功能；4）实现查询服务器当前目录下文件信息功能；

**2 人员分工**

|  |  |  |
| --- | --- | --- |
|  | 陈小灵 | 陈彦冰 |
| 设计部分 |  | 应用协议设计，服务器端设计，客户端设计 |
| 实现部分 |  | 应用协议实现，服务器端代码实现，客户端代码实现 |
| 测试部分 | 测试计划和测试执行 |  |
| 课堂展示及文档汇总 | 测试部分的文档撰写，制作ppt | 设计和实现部分的文档撰写 |
| 分工百分比 | 30% | 70% |

**3 设计**

**3.1 应用系统基本功能与附加功能**

根据题目要求，应用系统包括的基本功能有：

1）上传文件功能（对应命令STOR）；2）下载文件功能（对应命令RETR）；3）删除文件功能（对应命令DELE）；4）查询服务器当前目录下文件信息功能（对应命令LIST）；

此外我们预计附加功能包括：

登录（默认用户使用其他功能前需要先进行登录，用户名和密码提交分别对应命令USER和PASS）；主被动模式选择（主动模式对应命令MYPORT，被动模式对应命令PASV）；从FTP服务器上退出登录，断开控制连接（对应命令QUIT）；显示当前服务器工作路径（对应命令PWD）；更改服务器工作路径（对应命令CWD）；客户端交互界面。

对应这些功能需求，用例图如图3.1所示。

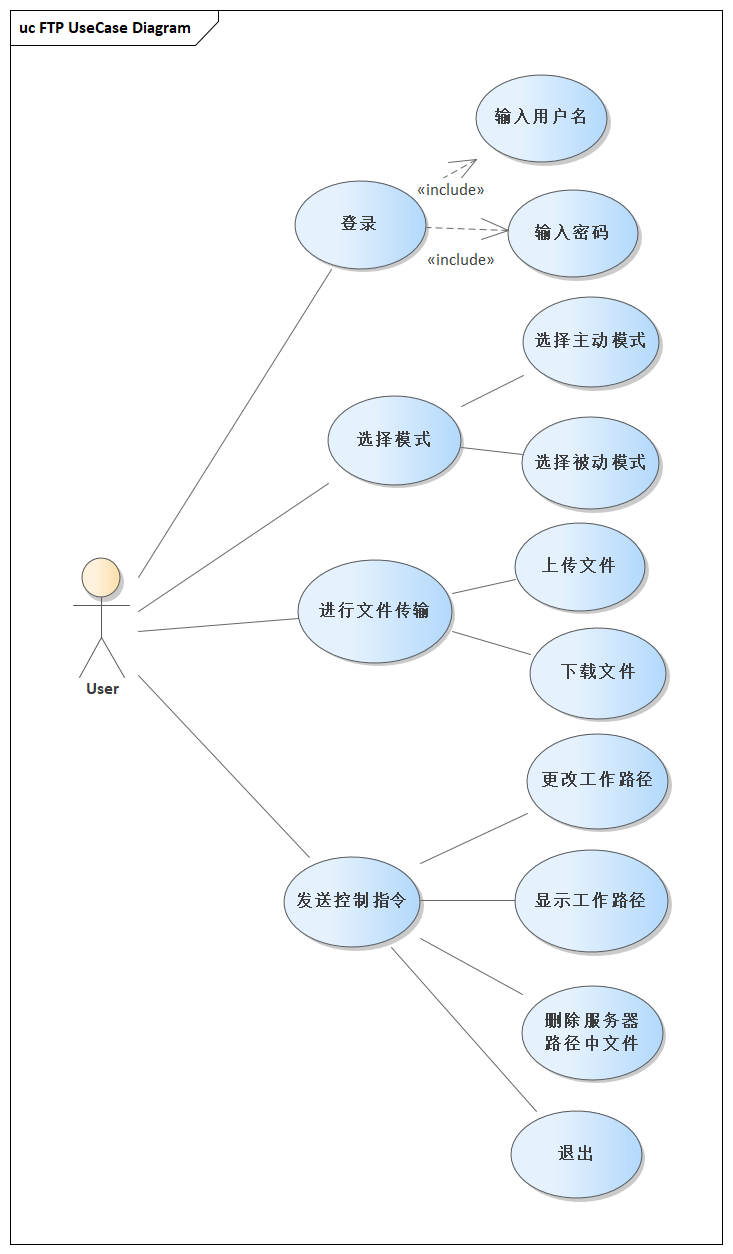


图3.1 FTP协议实现用例图

**3.2 应用协议描述**

FTP（File Transfer Protocol，文件传输协议）是一种client/server模式的应用层协议，作为网络共享文件的传输协议，其屏蔽了各计算机系统的细节，因而适合于在异构网络中任意计算机之间传送文件。

3.2.1 传输方式

FTP传输文件的方式包括ASCII和二进制两种。

ASCII传输方式假定用户正在拷贝的文件只包含简单ASCII码文本，如果在远程机器上运行的不是UNIX，当文件传输时ftp通常会自动地调整文件的内容以便于把文件解释成另一台计算机存储文本文件的格式。

二进制传输方式中保存文件的位序，以便原始和拷贝的是逐位一一对应的。二进制传输方式可以传送非文本的文件。

3.2.2 工作原理

FTP使用客户服务器方式。一个FTP服务器进程可同时为多个客户进程提供服务。FTP的服务器进程由两大部分组成：一个主进程，负责接受新的请求；若干个从属进程，负责处理单个请求。服务器主进程默认打开熟知端口（端口号为21），使客户进程能连接上，而从属进程用于处理客户进程发来的请求。

FTP协议是基于TCP传输层协议的，它在两台通信的主机之间使用了两条并行的 TCP 连接，一条是数据连接，用于数据传送；另一条是控制连接，用于传送控制信息。客户端有三个构件：用户接口、用户控制进程（User PI，用户协议解释器）和用户数据传送进程（User DTP，用户数据传输进程）。除主进程外，服务器有两个构件：服务器控制进程（Server PI，用户协议解释器）和服务器数据传送进程（Server DTP，用户数据传输进程）。在整个交互的 FTP 会话中，控制连接始终是处于连接状态的，数据连接则在每一次文件传送时先打开后关闭。

3.2.3 工作模式

FTP支持的模式包括主动模式和被动模式。在主动模式中，由客户端发送PORT命令，服务器根据PORT命令指定的客户端地址和端口号发起数据连接。在被动模式中，由客户端发送PASV命令，服务器返回监听的地址和端口号，由客户端发起数据连接。

工作流程主要包括两种情况。第一种情况如图3.2所示，在开始阶段，标准 FTP 命令由用户 PI 产生并通过控制连接传送到服务器进程，服务器 PI 向用户 PI 返回标准应答。FTP 命令指定数据连接参数和文件系统操作。用户DTP 在特定数据端口侦听，服务器开始数据连接并以指定的参数开始数据传输。

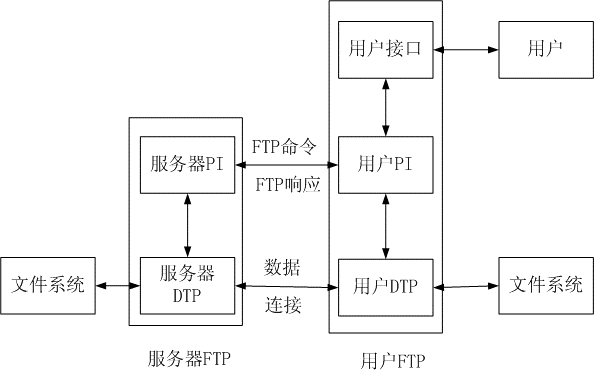


图3.2 FTP工作示意图情况一

另一种情况如图3.3所示，用户可以控制在两个远程主机之间传送文件。用户在两台主机间建立控制连接，然后规划数据连接。用这种方式，控制信息由用户PI 获得，但是数据在服务器 DTP 之间传送。

图3.3 FTP工作示意图情况二

**3.3 应用协议实现设计**

本次对FTP协议的实现在原协议的基础上展开，并在细节上进行一些修改，具体协议流程的示意图如图3.4所示。

原协议中，默认情况下FTP协议使用TCP端口中的 20和21这两个端口，其中20用于传输数据，21用于传输控制信息；如果采用主动模式，那么数据传输端口就是20，且由服务器发起数据连接，如果采用被动模式，则具体最终使用哪个端口要服务器端和客户端协商决定，且由客户端发起数据连接。

本实现中，保留21端口用于传输控制信息，而在传输数据信息时，无论是主动模式还是被动模式，服务器端都改为采用随机高位端口号与客户端连接，并且要求客户端在传输文件前（上传或下载）必须先进行主动或被动的模式选择。这一修改使原有主动模式也采用随机端口传输数据，不容易被监听。

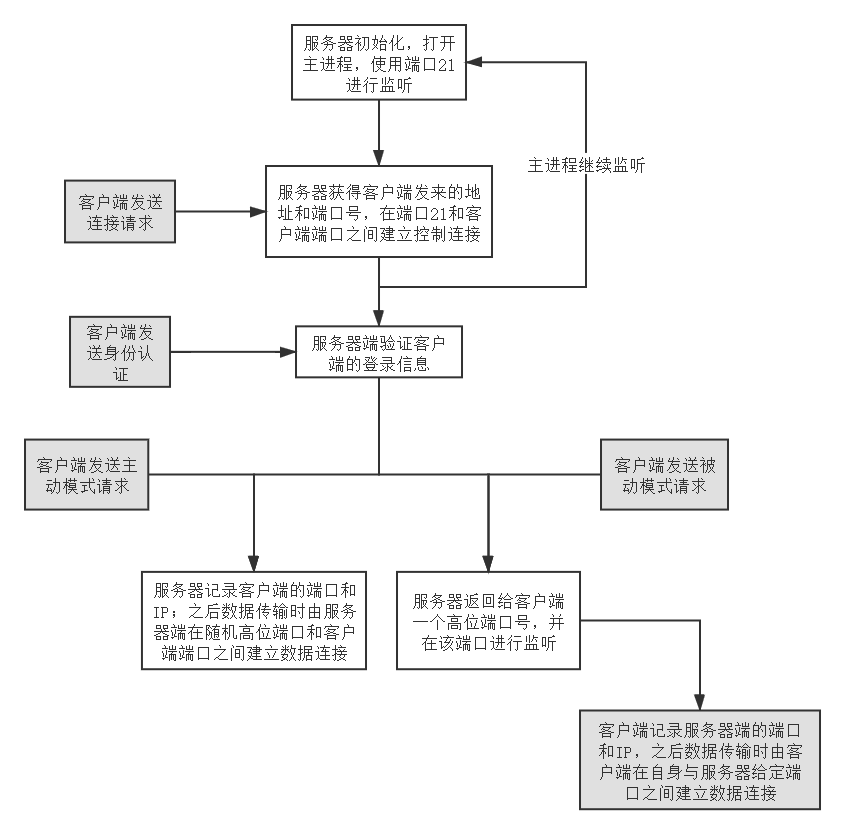


图3.4 应用协议流程示意图

**3.4 客户端C设计**

客户端需要向服务器的21端口发送连接请求，具有连接到服务器的功能；客户端向服务器端发送指令报文，在用户处可采用交互界面的方式进行，用户通过设定好的按钮进行操作以限制部分请求的顺序；客户端需要分别支持主动模式与被动模式下的运行，在两种模式下都需要完成各指令的发送和响应接收，在被动模式情况下发起数据连接；使用二进制流进行文件传输，且每次仅限于单个文件传输，不支持直接传输文件夹。具体的功能模块结构图如图3.5所示。

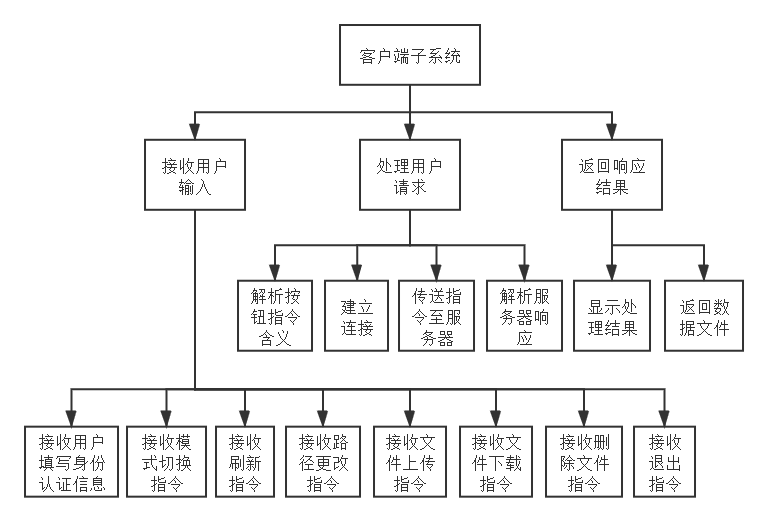


图3.5客户端功能模块结构图

**3.5 服务器S设计**

服务器端需要开启主进程以监听来自客户端的请求，具有建立控制连接的功能；对应功能需求，服务器端需要处理相应的指令，根据处理结果返回相应的响应代码和参数；在主动模式情况下发起数据连接；使用二进制流进行文件传输，且每次仅限于单个文件传输，不支持直接传输文件夹。具体的功能模块结构图如图3.6所示。

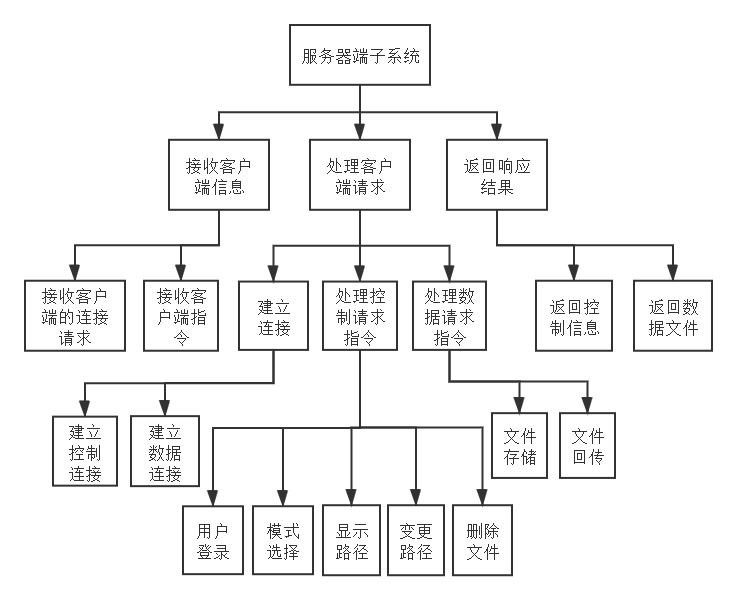


图3.6服务器端功能模块结构图

**4 实现**

本项目采用JAVA语言进行代码实现。由于实现时大体结构已在前文“应用协议实现设计”中解释，且实现时指令和方法对应，每个指令有不同的实现流程，各指令实现均没有复杂的逻辑，此处不再补充算法流程图。

**4.1 应用协议实现**

本实现中支持的控制指令和对应参数如下表4.1所示。控制指令和参数之间要求使用空格进行分隔，且在同一行进行输入。

表4.1 应用协议指令列表

|  |  |  |  |
| --- | --- | --- | --- |
| 指令 | 参数 | 响应代码及含义 | 含义 |
| USER | 具体用户名 | 331 请输入密码  504 用户名不存在 | 输入用户名，要求客户端在任何指令之前必须首先完成 |
| PASS | 用户名对应的密码 | 230 登录成功  504 登录失败 | 输入密码，在USER之后紧接着使用 |
| MYPORT | IP:PORT | 225 打开数据连接  425 无法打开数据连接 | 设置主动模式，要求登录后传输文件前设置模式（MYPORT或PASV），向服务器发送IP及端口 |
| PASV | 无 | 2277 进入被动模式返回服务器IP和端口  225 打开数据连接  425无法打开数据连接 | 设置被动模式，要求登录后传输文件前设置模式（MYPORT或PASV） |
| QUIT | 无 | 221 退出网络 | 断开控制连接和数据连接 |
| STOR | 文件名 | 125 开始传输文件  250 文件上传完成 | 上传文件到服务器（若有同名文件则添加后缀“(1)”） |
| RETR | 文件名 | 125 开始传输文件  250 文件回传完成  450 该文件不存在 | 从服务器下载文件 |
| DELE | 文件名 | 250 文件删除完成  350 文件删除出错  450 该文件不存在 | 从服务器删除文件（或文件夹） |
| LIST | 无 | 213 当前目录下文件列表显示完成 | 查询服务器当前目录下文件信息 |
| PWD | 无 | 212 显示当前路径 | 显示当前服务器工作路径 |
| CWD | 当前路径的子路径（相对路径）或“..” | 212 路径更改完成  550 路径不存在  504 无效命令参数 | 更改服务器工作路径（仅限已有文件夹路径） |

其中，主动模式的指令与被动模式的响应代码格式较为特殊，会在附录中另行解释。

**4.2 客户端C实现**

客户端端实现的类图如下图4.1所示。

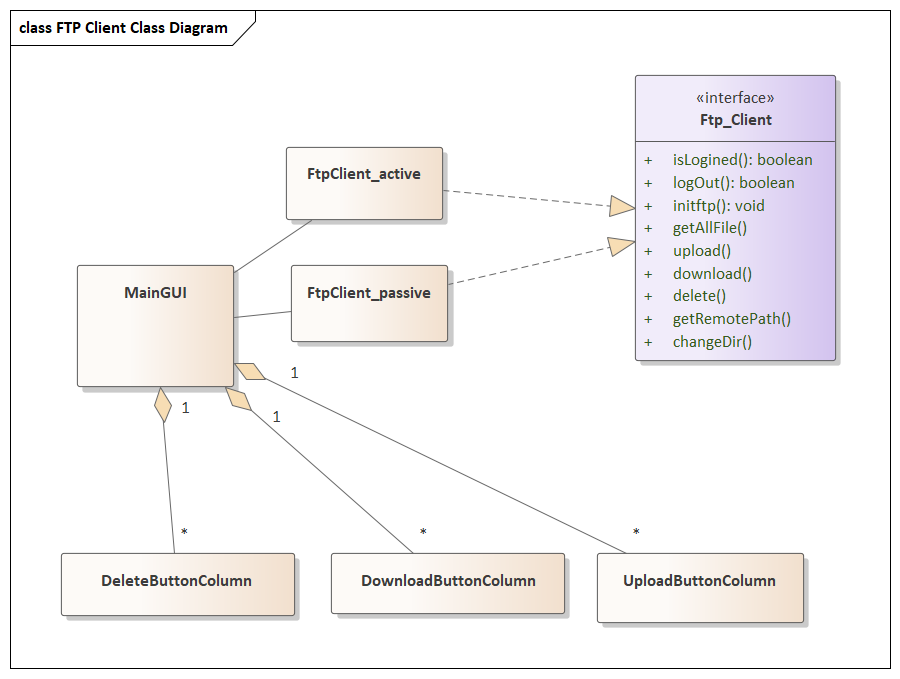


图4.1 客户端类图

其中，MainGUI类基于JAVA的Swing工具包实现了客户端的交互界面，DeleteButtonColumn类、DownloadButtonColumn类、UploadButtonColumn类对应删除、下载、上传按钮，由于交互界面中这几个按钮需要出现多次，因此单列为类。Ftp\_Client接口规定了客户端不同模式均需要实现的方法，isLogined方法用于获取是否登录，initftp方法用于执行USER和PASS的登录指令，logOut方法用于执行QUIT退出指令，getAllFile方法用于执行LIST指令获取和刷新文件列表，upload方法用于STOR指令上传本地文件，download方法用于RETR指令下载文件到本地，delete方法用于DELE指令删除文件，getRemotePath用于PWD指令显示服务器的远程工作目录，changeFir用于CWD指令更改服务器的远程工作目录。FtpClient\_active类和FtpClient\_passive类实现了Ftp\_Client接口，分别对应主动模式和被动模式下客户端的实现，其构造函数完成了客户端与服务器端的初始控制连接。

客户端工作时的交互界面示意图如图4.2和4.3。

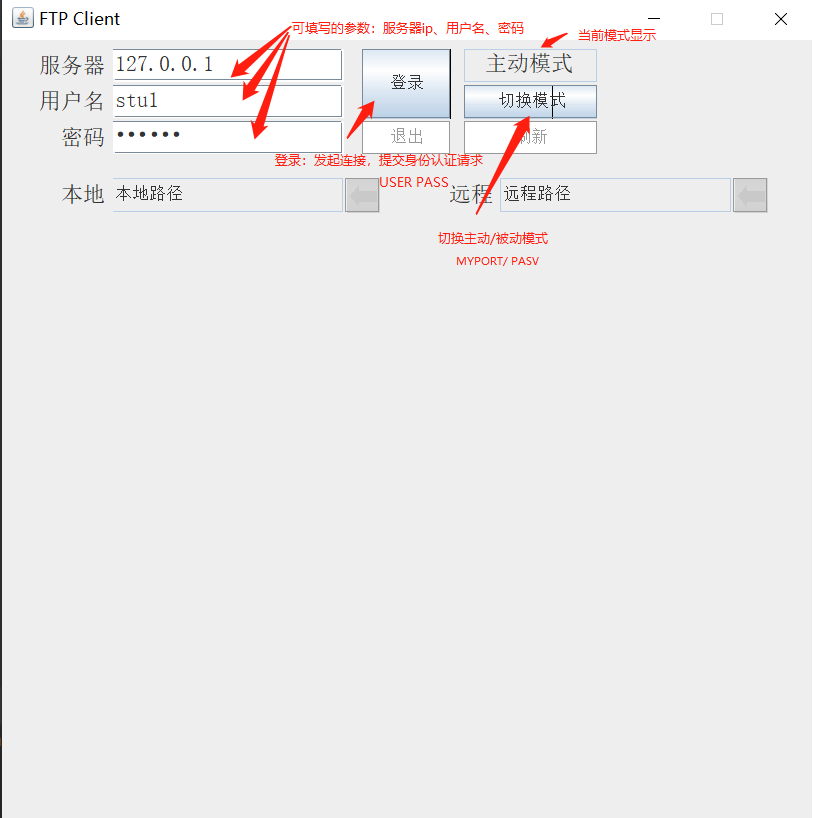


图4.2 客户端交互界面登录前情况

登录前，客户端交互界面上可使用的部分包括：填写服务器IP、用户名、密码；切换模式（主/被动）；登录。限制只能在登录前切换模式，并通过登录按钮一并确认发送。其中包括的指令有USER、PASS、MYPORT、PASV。点击登录同时完成客户端与服务端控制连接的建立，客户端路径的显示与服务器路径的显示（PWD）、客户端路径下文件列表的显示和服务器端路径下文件列表的显示（LIST）。

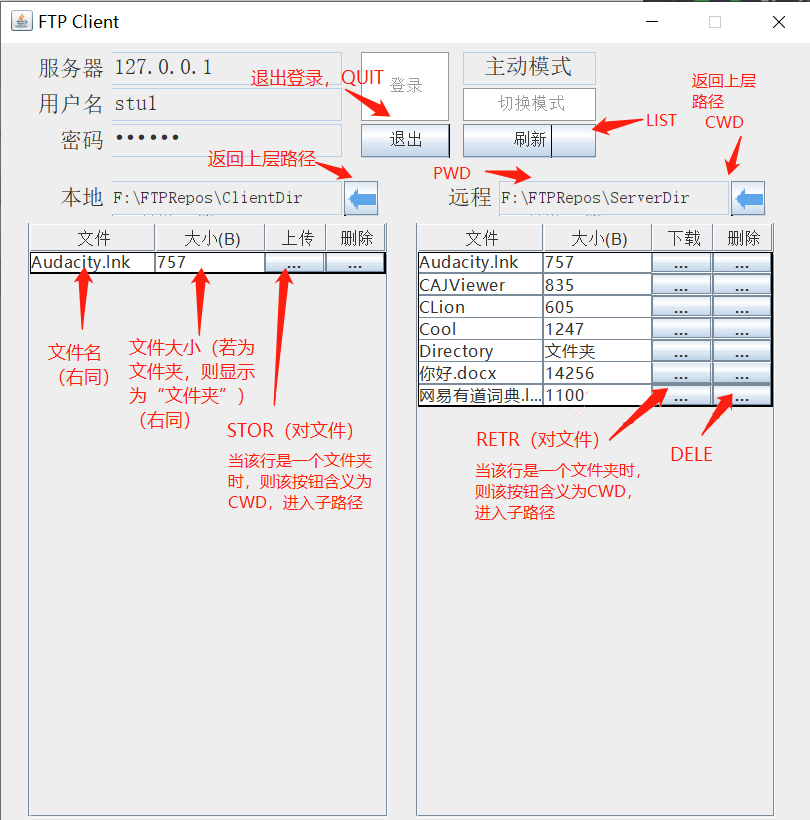


图4.3 客户端交互界面登录后情况

登录后，客户端交互界面上可使用的部分包括：退出、刷新、本地和远程路径显示、本地和远程的返回上层路径、上传本地文件、删除本地文件、下载远程文件、删除远程文件、更改本地路径至子文件夹、更改远程路径至子文件夹。限制文件的上传、下载操作不包括文件夹。其中包括的指令有QUIT、LIST、PWD、CWD、STOR、DELE、RETR。

**4.3 服务器S实现**

服务器端实现的类图如图4.4所示，图中标记出了Server端所用的类及主要的方法和变量。

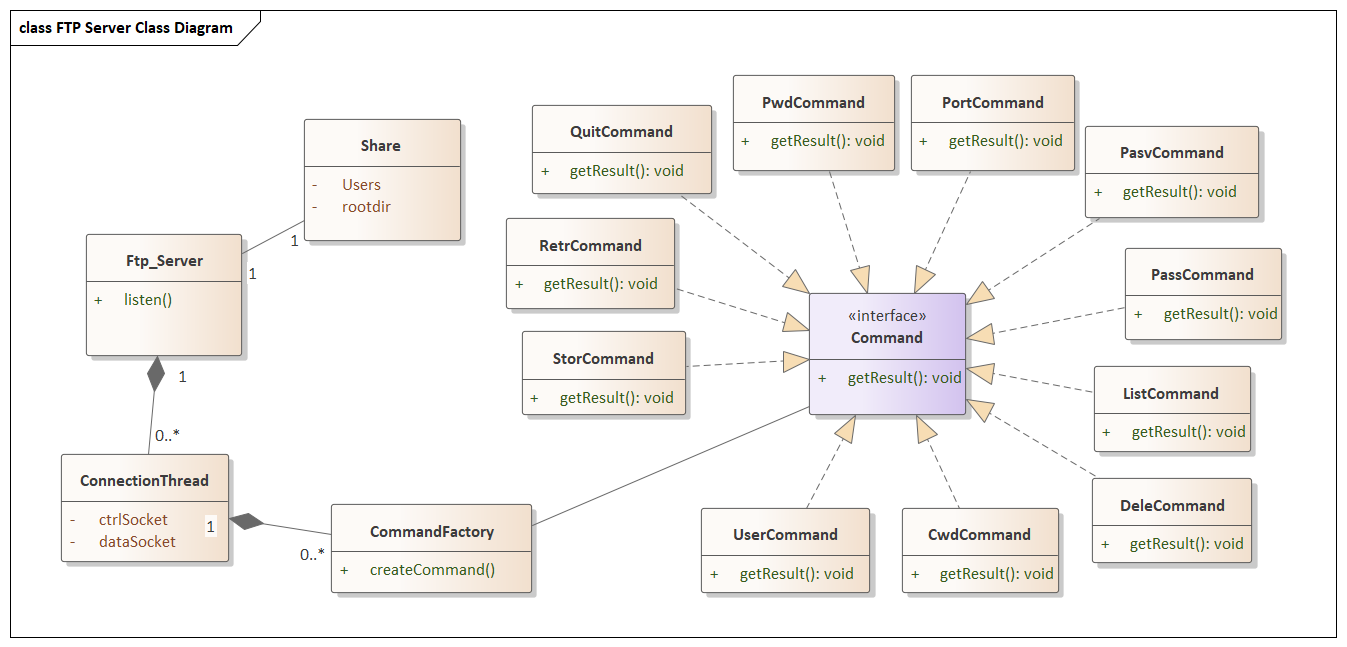


图4.4 服务器端类图

其中，Ftp\_Server类实现了服务器主进程，用于接收客户端的连接请求。Share类存放了服务器端中的用户信息和工作路径作为共享变量。ConnectionThread类负责管理和建立控制连接和数据连接，接收连接上传输的指令等信息，对连接上传输的指令进行中继。CommandFactory类实现了不同指令的分流。Command接口规定了收到来自客户端指令时固定的处理操作需要的函数和相应参数。UserCommand类、PassCommand类、PortCommand类、PasvCommand类、QuitCommand类、StorCommand类、RetrCommand类、DeleCommand类、ListCommand类、PwdCommand类、CwdCommand类分别实现了对USER、PASS、MYPORT、PASV、QUIT、STOR、RETR、DELE、LIST、PWD、CWD各具体指令的处理和响应。使用时，客户端通过Socket连接向服务器端发送请求，初始应与21端口建立连接。

**5 测试**

**5.1 测试环境**

Idea

JDK15

OS: Windows 10

CPU: AMD Ryzen 7 4800U with Radeon Graphics 1.80 GHz

RAM: 16.0 GB

SSD: 256G SSD固态硬盘

**5.2 测试计划**

将会测试以下功能：

（1）客户端交互界面

（2）登录功能—主动模式（默认用户使用其他功能前需要先进行登录，用户名和密码提交分别对应命令USER和PASS；主动模式对应命令MYPORT，被动模式对应命令PASV））

（3）登录功能—被动模式

（4）退出登录（对应命令QUIT）

（5）刷新功能： 查询当前目录下文件信息功能，更新客户端页面的文件列表（对应命令LIST）

（6）上传文件功能（对应命令STOR），测试多种不同类型的文件；

（7）下载文件功能（对应命令RETR）；

（8）删除文件功能（对应命令DELE）；

（9）显示服务器工作路径（对应命令PWD）

（10）切换服务器工作路径（对应命令CWD）

（11）其他

**5.3 测试用例列表**

|  |  |
| --- | --- |
| 测试用例编号 | 01 |
| 测试内容 | 客户端交互界面；显示服务器当前工作路径 |
| 预置条件 | 无 |
| 操作步骤 | 编译运行MainGUI.java |
| 预期结果 | 显示FTP客户端页面 |
| 实际结果 |  |
| 是否通过 | 是 |

|  |  |
| --- | --- |
| 测试用例编号 | 02 |
| 测试内容 | 登录—主动模式：当用户名和密码不匹配时 |
| 预置条件 | 运行FtpServer.java和MainGUI.java，弹出客户端窗口 |
| 操作步骤 | 填写以下信息：服务器：127.0.0.1；用户名：stu1；密码：000001 |
| 预期结果 | 客户端登录失败，弹出错误提示窗口 |
| 实际客户端结果 |  |
| 实际服务端结果 |  |
| 是否通过 | 是 |

|  |  |
| --- | --- |
| 测试用例编号 | 03 |
| 测试内容 | 登录—主动模式 |
| 预置条件 | 运行FtpServer.java和MainGUI.java，弹出客户端窗口 |
| 操作步骤 | 填写以下信息：服务器：127.0.0.1；用户名：stu1；密码：000000 |
| 预期结果 | 客户端登录成功；服务端打印登录信息和文件列表 |
| 实际客户端结果 |  |
| 实际服务端结果 |  |
| 是否通过 | 是 |

|  |  |
| --- | --- |
| 测试用例编号 | 04 |
| 测试内容 | 登录—被动模式 |
| 预置条件 | 运行FtpServer.java和MainGUI.java，弹出客户端窗口 |
| 操作步骤 | 点击“切换模式”，再点击登录 |
| 预期结果 | 登录成功 |
| 实际客户端结果 | 登录失败 |
| 实际服务端结果 |  |
| 是否通过 | 否 |

|  |  |
| --- | --- |
| 测试用例编号 | 05（回归测试） |
| 测试内容 | 登录—被动模式 |
| 代码修改 | 将"227 进入被动模式 127.0.0.1:" 改成"2277 进入被动模式 127.0.0.1:" |
| 预置条件 | 运行FtpServer.java和MainGUI.java，弹出客户端窗口 |
| 操作步骤 | 点击“切换模式”，再点击登录 |
| 预期结果 | 登录成功 |
| 实际客户端结果 |  |
| 实际服务端结果 |  |
| 是否通过 | 是 |

|  |  |
| --- | --- |
| 测试用例编号 | 06 |
| 测试内容 | 退出登录，断开控制连接 |
| 预置条件 | 运行FtpServer.java和MainGUI.java |
| 操作步骤 | 点击交互页面中的“退出”按钮 |
| 预期结果 | 服务端打印：QUIT 结束连接 |
| 实际结果 | 服务端打印：QUIT 结束连接 |
| 是否通过 | 是 |

|  |  |
| --- | --- |
| 测试用例编号 | 07 |
| 测试内容 | 刷新功能（查询本地路径和远程路径中的当前文件信息） |
| 预置条件 | 本地路径和远程路径中添加或删除新文件 |
| 操作步骤 | 点击“刷新”按钮 |
| 预期结果 | 客户端界面中显示更新后的本地文件列表：服务端打印当前文件列表 |
| 实际客户端结果 |  |
| 实际服务端结果 | 打印：LIST |
| 是否通过 | 是 |

|  |  |
| --- | --- |
| 测试用例编号 | 08 |
| 测试内容 | 上传文件功能--jpg文件 |
| 预置条件 | 运行FtpServer.java和MainGUI.java |
| 操作步骤 | 点击上传sea.jpg |
| 预期结果 | 远程文件路径下，文件列表中新增sea.jpg |
| 实际客户端结果 |  |
| 实际服务端结果 |  |
| 是否通过 | 是 |

|  |  |
| --- | --- |
| 测试用例编号 | 09 |
| 测试内容 | 上传文件功能--txt文件 |
| 预置条件 | 本地路径中有“测试.txt”，远程路径下无该文件 |
| 操作步骤 | 点击上传“测试.txt” |
| 预期结果 | 远程文件路径下，文件列表中新增“测试.txt” |
| 实际客户端结果 |  |
| 实际服务端结果 |  |
| 是否通过 | 是 |

|  |  |
| --- | --- |
| 测试用例编号 | 10 |
| 测试内容 | 上传文件功能—MP3文件 |
| 预置条件 | 本地路径中有“薛之谦-绅士.mp3”，远程路径下无该文件 |
| 操作步骤 | 点击上传“薛之谦-绅士.mp3” |
| 预期结果 | 远程文件路径下，文件列表中新增“薛之谦-绅士.mp3” |
| 实际客户端结果 |  |
| 实际服务端结果 |  |
| 是否通过 | 是 |

|  |  |
| --- | --- |
| 测试用例编号 | 11 |
| 测试内容 | 上传文件功能—PDF文件 |
| 预置条件 | 本地路径中有“习题课.pdf”，远程路径下无该文件 |
| 操作步骤 | 点击上传“习题课.pdf” |
| 预期结果 | 远程文件路径下，文件列表中新增“习题课.pdf” |
| 实际客户端结果 |  |
| 实际服务端结果 |  |
| 是否通过 | 是 |

|  |  |
| --- | --- |
| 测试用例编号 | 12 |
| 测试内容 | 删除文件功能—本地路径 |
| 预置条件 | 本地路径下有很多文件 |
| 操作步骤 | 点击删除按钮 |
| 预期结果 | 本地路径下的文件全部删除 |
| 实际客户端结果 |  |
| 实际服务端结果 | 服务端打印：LIST |
| 是否通过 | 是 |

|  |  |
| --- | --- |
| 测试用例编号 | 13 |
| 测试内容 | 删除文件功能—远程路径 |
| 预置条件 | 远程路径下有很多文件 |
| 操作步骤 | 点击删除按钮，只留下sea.jpg和Audacity.lnk |
| 预期结果 | 远程路径下删除了很多文件，只剩下sea.jpg和Audacity.lnk |
| 实际客户端结果 |  |
| 实际服务端结果 |  |
| 是否通过 | 是 |

|  |  |
| --- | --- |
| 测试用例编号 | 14 |
| 测试内容 | 下载文件功能：把文件从远程路径下载到本地路径 |
| 预置条件 | 远程路径下有很多文件 |
| 操作步骤 | 点击下载“计网第三章.pdf”和“李佳莹 - 你笑起来真好看.mp3” |
| 预期结果 | 本地路径下新增“计网第三章.pdf”和“李佳莹 - 你笑起来真好看.mp3” |
| 实际客户端结果 |  |
| 实际服务端结果 |  |
| 是否通过 | 是 |

|  |  |
| --- | --- |
| 测试用例编号 | 15 |
| 测试内容 | 切换文件路径 |
| 预置条件 | 登录ftp |
| 操作步骤 | 点击返回上一级的箭头，返回文件上一层；点击文件夹对应的“…”，进入子文件夹 |
| 预期结果 | 返回上一层成功；进入子文件夹目录成功 |
| 实际客户端结果 |  |
| 实际服务端结果 |  |
| 是否通过 | 是 |

|  |  |
| --- | --- |
| 测试用例编号 | 16 |
| 测试内容 | 下载同名文件 |
| 预置条件 | 远程路径和本地路径下存在同名文件：“测试文件.docx”，但其内容和大小不同 |
| 操作步骤 | 点击下载“测试文件.docx” |
| 预期结果 | 本地路径下，原同名文件被覆盖 |
| 实际客户端结果 |  |
| 实际服务端结果 |  |
| 是否通过 | 是 |

|  |  |
| --- | --- |
| 测试用例编号 | 17 |
| 测试内容 | 上传同名文件 |
| 预置条件 | 远程路径和本地路径下存在同名文件：“开发文档.docx”，但其内容和大小不同 |
| 操作步骤 | 点击上传“开发文档.docx” |
| 预期结果 | 远程路径下，原同名文件被保留，新上传文件被自动命名为“开发文档.docx(1)” |
| 实际客户端结果 |  |
| 实际服务端结果 |  |
| 是否通过 | 是 |

|  |  |
| --- | --- |
| 测试用例编号 | 18 |
| 测试内容 | 删除服务器端不存在的文件（文件夹里已删除 文件列表未刷新） |
| 预置条件 | 远程路径下，“pic1.JPG”已被删除，但是客户端列表还未刷新 |
| 操作步骤 | 点击删除“pic1.JPG” |
| 预期结果 | 弹出窗口，告知文件不存在 |
| 实际客户端结果 |  |
| 实际服务端结果 |  |
| 是否通过 | 是 |

**6 附录**

**6.1 参考说明**

本实现参考了GitHub上https://github.com/muziyongshixin/FTP-client-and-server-use-JAVA 的代码，在原有基础上更改了一部分设计，修改并增加了相应功能和处理，整体代码书写架构可能有相似性。

**6.2 问题说明**

本实现中遇到了与参考代码相似的问题，即主动模式选择发送“PORT”时会自动断开控制连接，被动模式服务器返回响应代码“227”时也会使控制连接断开。推测是由于控制连接建立涉及到的服务器21端口作为原FTP协议使用的特殊端口，可能本身会对连接上传输的报文有一定的处理，尝试将控制连接更换为“5521”端口则不会出现这一情况。本实现最终保留了21端口的设计，但将主动模式指令更改为“MYPORT”，被动模式响应代码更改为“2277”则不再出错。