详细设计说明书

1 引言

1.1 编写目的

根据《需求规格说明书》、《概要设计说明书》，在仔细考虑讨论之后，我们对《基于JAVA的FTP项目开发》的功能划分、数据结构、项目总体结构的实现有了进一步的想法。我们将这些想法记录下来，作为详细设计说明书，为进一步设计项目、编写代码打下基础。

在前一阶段《概要设计说明书》中，已解决了实现该系统需求的程序模块设计问题。包括如何把该系统划分成若干个模块、决定各个模块之间的接口、模块之间传递的信息，以及数据结构、模块结构的设计、系统数据结构设计、系统出错处理设计等。在以下的详细设计报告中将对在本阶段中对系统所做的所有详细设计进行说明。

在本阶段中，确定应该如何具体地实现所要求的项目，从而在编码阶段可以把这个描述直接翻译成用具体的程序语言书写的程序。主要的工作有：根据在《需求分析说明书》中所描述的数据、功能、运行、性能需求，并依照《概要设计说明书》所确定的处理流程、总体结构和模块外部设计，设计软件系统的结构设计、逐个模块的程序描述（包括各模块的功能、性能、输入、输出、算法、程序逻辑、接口等等）。

1.2 背景

本项目的名称：基于JAVA的FTP项目开发

本项目的提出者：综合实训指导老师——黄钰

本项目的开发者：朱传波、占鹏、李澎、田尧

本项目的使用者：教师和学生

1.3 定义

J2SE ： Java2 Standard Edition

FTP（File Transfer Protocol）：文件传输协议

Netbeans：一种集成开发环境，包含绘图工具

1.4 参考资料

（1） 程良伦主编，《网络工程概论》，机械工业出版社，2007

（2） （美）埃史尔 陈昊鹏，《Thinking in java》中文版，机械工业出版社，2007

（3） 飞思科技，《Java TCP/IP 应用开发详解》，清华大学出版社，2001

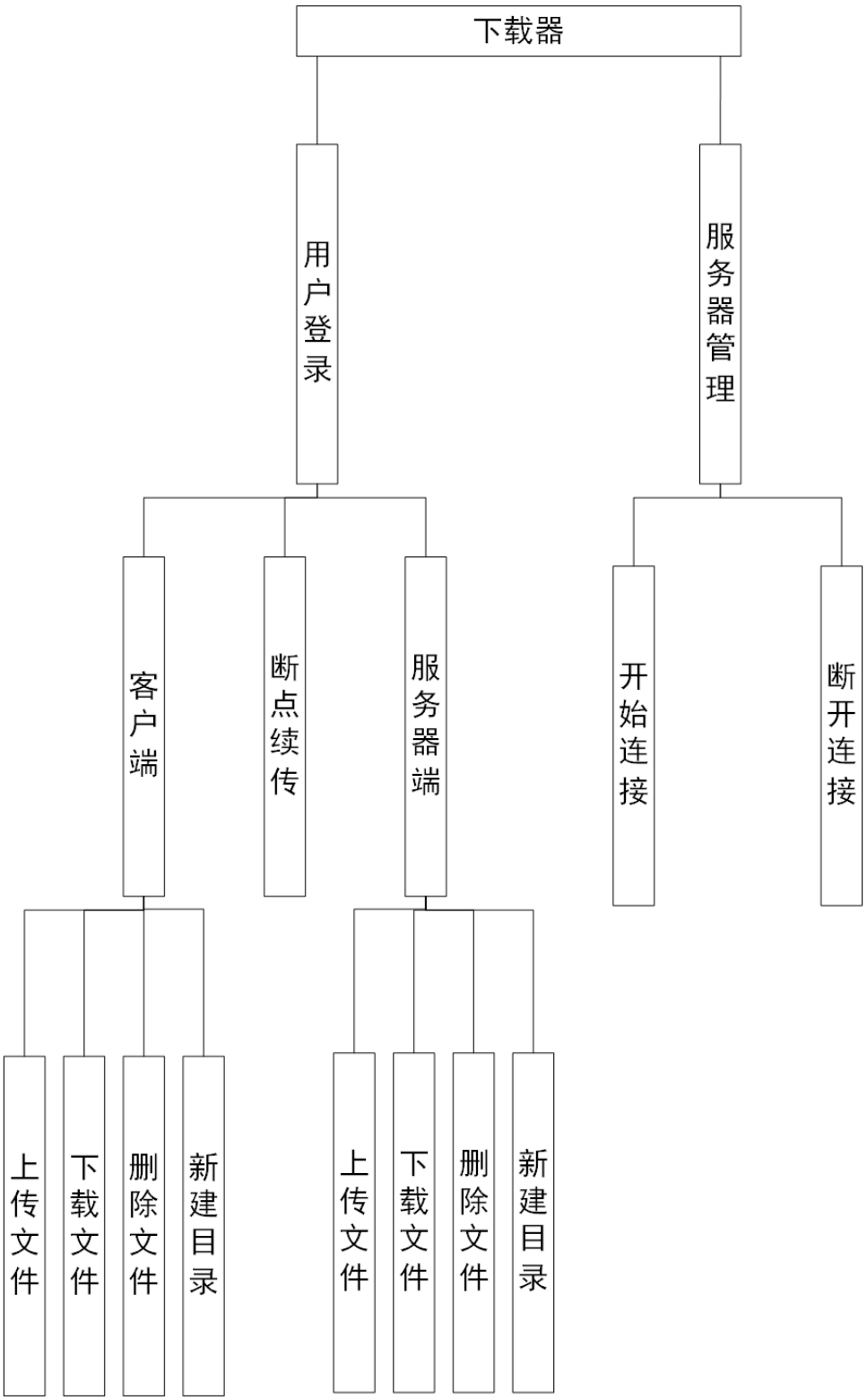
（4） 陈刚，《Eclipse 从入门到精通》，清华大学出版社，2005

（5） James Rumbaugh Ivar Jacobson Grady Booch，《UML 参考手册》，机械工业出版社，2001

（6） 需求分析说明书

（7） 概要设计说明书

2 程序系统的结构



图：下载器功能的结构图

3 客户端设计说明

3.1 模块描述

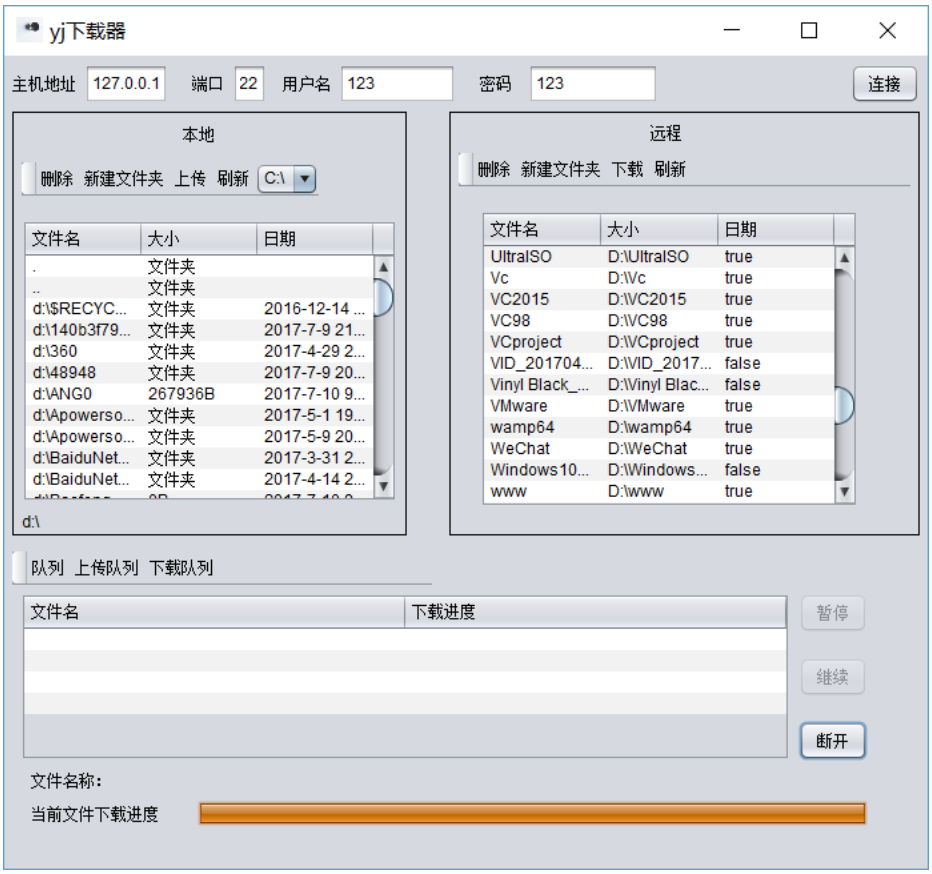
实现客户端读取必要对象的设置，以及参数的选取

3.2 功能

读取本地服务器中的文件，生成文件列表目录，对本地文件目录列表进行操作，实现与服务器端的连接，对远程服务器端数据以及文件进行上传下载操作。

3.3 交互的模块

客户端与服务器端通过界面进行交互，设计的界面如下：



图：用户界面设计

3.4 模块设计

主要的类的对象：

a. 定义 connectSocket（连接堆栈），控制连接，用于传送和响应命令
b. 定义 dataSocket（数据堆栈），数据连接，用于数据传输
c. 定义 BufferedReader inData,控制连接中用于读取返回信息的数据流
d. 定义 BufferedWriter outData,控制连接中用于传送用户命令的数据流
e. Response 字符串，封装客户端与服务器端间的返回信息
f. remoteHost 字符串，远程主机名（IP 地址）
g. remotePort, 通信端口号，默认为 21
h. remotePath 远程路径
i. user 用户名
j. passWord 用户密码
k. rootPath 根路径
l. currentPath 当前路径
m. Logined 判断是否登录服务器的标志

关于客户端的用户登录：

1、对象的获取与传递

a. 远程主机设置服务器域名
b. 客户端处获得服务器域名
c. 远程主机设置端口号
d. 客户端处获得端口号
e. 设置远程目录路径
f. 客户端获取当前的远程目录路径
g. 获取设定的用户名和用户密码

2、服务器的连接

需要建立控制连接，若没有控制连接，则生成获得新的控制连接，包括字符输入流以及字符输出流（即客户端处需要输入远程主机的 IP 以及端口号，然后服务器端在判断客户端输入的远程主机 IP 和端口号是否正确后，若正确则返回服务器成功连接的信息，若错误则返回连接失败的信息）

3、用户的登录

首先需要有控制连接即服务器连接成功的建立。若服务器没有连接则输入登录信息需要返回“服务器尚未连接，请先连接！”的信息。

若服务器已成功连接，则可以输入登录信息，由客户端处输入登录信息，然后由response 封装用户名以及用户密码然后经通道传送到服务器端，服务器端接收验证登录信息请求后，验证传送过来的用户名以及密码与所对应的密码是否一致，若一致则logined 的值变化为 true,返回“登录成功”；若不一致，则 logined 的值不变，返回“用户名或密码错误”；且需抓取登录失败的异常，若出现登录异常，则返回“登录失败！”的信息。

4、获取远程服务器的目录列表

如果服务器未连接或没有登录则无法获取远程服务器的目录列表，若已连接且登陆成功则在客户端可以通过点击显示远程文件列表的请求，而若服务端接受了这个请求则可以在客户端显示远程主机的文件列表。在实现了获取远程服务器的目录列表后，可以删除已有的目录、可以创建新的未重名的目录。而在远程服务器的目录处新建目录后，将讯号回送到客户端，可以在客户端通过刷新重新接收到新的目录列表。

上传下载文件

上传文件

由客户端向远程服务器的目录上传文件。在控制连接连接成功且正确获取服务器 IP 地址以及端口号、用户名与密码验证成功后，客户端才有向远程服务器上传文件的前提，在客户端处选择指定文件点击了上传的按钮后，上传请求经控制连接传送远程服务器端，远程服务器端再发送类似同意请求的信号经控制连接返回到客户端，文件得以上传，上传完成后在屏幕中输入“文件上传成功”字样。
上传完成后，关闭传送通道。

下载文件

从客户端下载远程服务器目录上的文件并保存到客户端的目录中。在客户端处浏览远程服务器目录上的文件，从客户端处选取指定的文件并点击下载按钮，下载请求经控制连接传送到远程服务器端，远程服务器端再发送类似同意请求的信号经控制连接返回客户端后，下载功能正式成立。下载过程中可以在客户端面板中查看各个指定下载文件的下载进度，选择指定任务选择暂停按钮，则可以暂停该文件的下载，若下载完成，则会有“下载成功”这个信息显示在客户端屏幕上。
下载成功后，关闭传送通道。

4 服务器端设计说明

4.1 总思路

1.1 将主动模式和被动模式集成到服务器的不同方法中,即不同的方法使用不同的连接模式

1.2 通过将服务器的不同状态区分开来,每个状态只能执行一定的操作,即操作的类型由当前服务器的状态所限定.同时为每个操作设定不同的数字,使用数字来选择不同的操作

1.3 PORT 中文称为主动模式，工作的原理： FTP 客户端连接到 FTP 服务器的 21 端口，发送用户名和密码登录，登录成功后要 list 列表或者读取数据时，客户端随机开放一个端口（1024 以上），发送 PORT 命令到 FTP 服务器，告诉服务器客户端采用主动模式并开放端口；FTP 服务器收到 PORT 主动模式命令和端口号后，通过服务器的 20 端口和客户端开放的端口连接，发送数据.

1.4 PASV 是 Passive 的缩写，中文成为被动模式，工作原理：FTP 客户端连接到 FTP 服务器的 21 端口，发送用户名和密码登录，登录成功后要 list 列表或者读取数据时，发送 PASV 命令到 FTP 服务器， 服务器在本地随机开放一个端口（1024 以上），然后把开放的端口告诉客户端， 客户端再连接到服务器开放的端口进行数据传输.

1,5 客户端传来命令后,服务器根据其开头字母来分别进行不同的操作,同时将其参数保存,以便操作使用

4.2 FTP 服务器模块

2.1 服务器控制模块:主要用于对服务器所进行的操作的控制(如选择不同的操作),以及监听服务器的状态;

2.2 数据传输模块:主要包括 传输模式的设置,传输格式的设置,以及调用文件处理模块来实现文件的上传下载

2.3 命令信号传输模块:主要用于客户端的连接以及命令的传输和处理

2.4 文件处理模块:读取,处理文件.分别实现文件的上传下载.以及服务器端的文件的新建和删除

2.5 用户登录模块:主要实现用户的登录.

3.各模块详细功能

4.3 服务器控制模块

客户端向服务器端发送的命令,经过控制模块的处理,调用相应的方法.

当客户端发送出指令时,服务器将指令转化为指令号,调用相应的方法,在方法调用的同时,更新服务器的状态;

指令号	对应的方法	功能
-1	ErrCMD()	返回”语法错误”的信息
4	commandCDUP()	返回上一层的目录
6	commandCWD()	改变工作目录
7	commandQUIT()	退出程序
9	commandPORT()	主动模式
11	commandTYPE()	文件传输类型设置
14	commandRETR()	下载文件
15	commandSTOR()	上传文件
22	commandABOR()	关闭数据传输的 dataSocket
23	commandDELE()	删除服务器上的文件
25	commandMKD()	建立新的目录
26	commandLIST()	获取列表
32	commandNOOP()	返回”命令正确”信息
33	commandPWD()	显示当前的目录
100	commandPASV()	被动模式

表：指令方法功能表

4.2 数据传输模块

根据客户端传来的命令,判断传输格式类型,从而选择将传输格式设置为 ASCII 码或二进制形式

4.3 命令信号模块

服务器监听指定端口,等待客户端的连接.接收到客户端的请求后,创建线程,打开用于数据传输的端口.21 端口用于控制,20 端口用于和客户端传输数据

属性类型	属性名称	描述
Socket	ctrlSocket	用于控制的套接字
Socket	dataSocket	用于传输的套接字
ServerSocket	serverSocket	服务器端监听
String	cmd	存放指令
String	param	存放指令后的参数
String	reply	服务器的返回信息
String	requestfile	请求的文件
int	i	指令号

表：命令信号模块属性列表

4.4 文件处理模块

函数名	描述
commandTYPE()	设置文件传输格式
commandRETR()	从服务器下载文件
commandSTOR()	上传文件到服务器
commandDELE()	删除服务器上的指定文件
commandMKD()	建立目录
commandLIST()	获取文件列表
commandPWD()	当前目录

表：文件处理方法表

4.5 用户登录模块

函数名	描叙
commandUSER()	输入用户名
commandPASS()	输入密码
isUserLogin()	判断登录信息是否正确

表：用户登录模块方法列表

属性类型	属性名	描述
String	user	用户名
String	password	密码
int	state	表示服务器当前状态

表：用户登录模块属性列表

5 断点续传功能设计说明

5.1 模块描述

实现文件传输过程中的文件突然出现中断，以及后续的上传的处理。

5.2 功能

文件传输过程中的出现中断，下次下载时可以直接在下载的断开点继续开始传输，不用从头开始

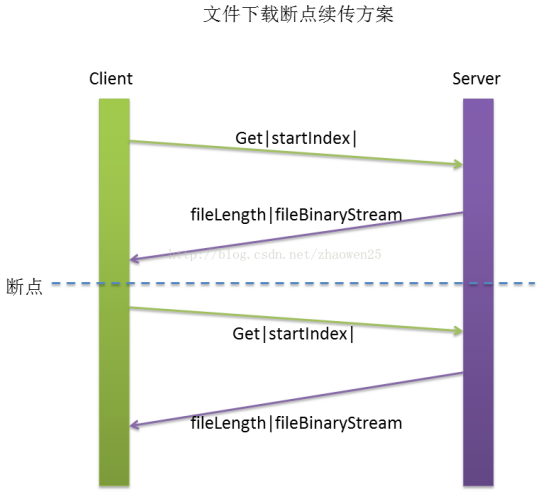
在文件下载传输过程中，将一个大的下载的文件分成多个部分，进行多个部分的同时下载（此处也可以采用多线程下载），当出现网络问题或者其他的问题导致不能下载即任务被暂停，此时记录下载的暂停位置即断点位置，在恢复下载即续传的时候，可以直接从记录的断点处下载或上传，这样处理，如果碰到网络故障，可以从已经上传或下载的部分开始继续上传下载以后未上传下载的部分，而没有必要重头开始上传下载。可以节省时间，提高速度。

5.3 交互的模块

客户端，服务器端

5.4 模块设计

断点续传简单示意图：



图：断点续传流程图

【分析】：客户端与服务器端进行套接字的连接，然后客户端发送请求下载的请求即图中流【Get|startindex|】中包括下载文件的具体信息以及此文件在本地的已有大小，客户端通过流【fileLength|fileBinaryStream】将文件具体大小以及保存在服务器端的断点位置发送给客户端，至此，数据传输前的准备工作已完成，接下来即是设置缓冲区长度进行文件的下载。

主要的类的对象：

File 创建的 f 对象，用于表示下载或上传的文件；

StringWriter 创建的 sw 对象，用于保存客户端传来的断点信息；

RandomAccessFile 创建的 accessFile 对象，用于创建文件输出流；

主要方法：

String 的 valueOf()方法，用于将文件的总长度转换为字符串形式；

String 的 getBytes()方法，用于将文件总长度的字符串形式转换成二进制形式；

OutputStream 的 write()方法,用于将表示文件总长度的直接写入流 downloadOut 中；

OutputStream 的 flush()方法，用于刷新 downloadOut 流并强制写出所有缓冲的输出字节;

OutputStream 的 close()方法，用于关闭 downloadOut 并释放与此流有关的所有系统资源;

RandomAccessFile 的 skipBytes()方法，用于跳过断点前字节不进行传输；的方法，用于；

RandomAccessFile 的 read()方法，用于将最多 len 个数据字节从下载文件读入 byte 数组并计入 realLength 中；

inputStream 的 close()方法，用于关闭 downloadIn 流并释放与该流关联的所有系统资源；

ServerSocket 的 close()方法，用于关闭此套接字 Socket;

File 的 length()方法，用于返回此抽象路径名表示的文件的长度；

总结：断点续传仅完成了一部分，因为没有数据库的建立，所以无法进行程序关闭后的断点续传，只是放在一个小的循环体中。