**武汉大学计算机学院**

**本科生实验报告**

**计算机网络应用设计**

专 业 名 称 ：计算机科学与技术

课 程 名 称 ：计算机网络课程设计

团 队 名 称 ：计网课程设计第二小组

指 导 教 师 一：周浩

团 队 成 员 一：

团 队 成 员 二：

二○二一年七月

**郑 重 声 明**

本团队呈交的实验报告，是在指导老师的指导下，独立进行实验工作所取得的成果，所有数据、图片资料真实可靠。尽我所知，除文中已经注明引用的内容外，本实验报告不包含他人享有著作权的内容。对本实验报告做出贡献的其他个人和集体，均已在文中以明确的方式标明。本实验报告的知识产权归属于培养单位。

团队成员签名：

日期： 2021.7.19

摘 要

计算机网络应用设计实验的实验目的是使学生熟悉网络规划与设计的基本知识和方法、掌握网络系统软件与应用软件开发的方法，能将所学的操作系统、数据库、软件工程、计算机网络等方面的知识集成到一起，规划、安装、调试实际网络系统、开发实际软件系统。。

实验设计主要遵循面向系统的软件开发和面向网络应用的软件开发。

实验内容主要包括：

1. 学习使用FTP协议下载文件的流程
2. 学习相关类库的使用
3. 设计及实现FTP客户端
4. 设计及实现FTP服务器

实验结论为学习了FTP协议的相关内容，对其传输文件的方式有了更深刻的理解，完成了FTP客户端的设计同时还书写了FTP的服务器学习了socket编程，客户端实现了上传、下载和断点续传等功能，服务器可以对客户端的请求进行响应，可以创建删除用户，更改下载目录等。

**关键词：**FTP；客户端；服务器

目录

[1 实验目的和意义 5](#_Toc78712708)

[1.1 实验目的 5](#_Toc78712709)

[1.1.1 目的一：面向系统的软件开发 5](#_Toc78712710)

[1.1.2 目的二：面向网络应用的软件开发 5](#_Toc78712711)

[2.实验意义 5](#_Toc78712712)

[2 实验设计 6](#_Toc78712713)

[2.1 概述 6](#_Toc78712714)

[2.2 实验原理 6](#_Toc78712715)

[2.2.1 使用FTP协议下载文件的流程 6](#_Toc78712716)

[2.2.2 相关类库说明 8](#_Toc78712717)

[2.3 实验方案 10](#_Toc78712718)

[2.3.1 FTP客户端设计 10](#_Toc78712719)

[2.3.2 FTP服务器端设计 18](#_Toc78712720)

[结论 28](#_Toc78712721)

[1. 程序主要界面及结果 28](#_Toc78712722)

[2. 程序源程序 34](#_Toc78712723)

[客户端代码： 34](#_Toc78712724)

[服务器端代码： 49](#_Toc78712725)

[参考文献 71](#_Toc78712726)

# 1 实验目的和意义

## 1.1 实验目的

本实验是使学生熟悉网络规划与设计的基本知识和方法、掌握网络系统软件与应用软件开发的方法，能将所学的操作系统、数据库、软件工程、计算机网络等方面的知识集成到一起，规划、安装、调试实际网络系统、开发实际软件系统。

### 1.1.1 目的一：面向系统的软件开发

本实验是使学生掌握网络系统软件的开发方法、开发平台的使用、与实际数据库的集成方法。用JAVA/ VC++/C＃完成FTP客户端、SMTP客户端、POP3客户端三个系统程序。

### 1.1.2 目的二：面向网络应用的软件开发

利用软件工程的方法，设计一个小规模的应用系统，与具体数据库连接起来，从用户界面设计、数据库设计到处理流程设计，最终完成系统的编程。本实验将用网上书店作为例子，要求实现基于Web的远程功能（Web页面）和后端管理功能。

## 2.实验意义

该实验是理论知识和动手能力的综合体现。通过本实验，掌握网络系统软件、网络应用软件的开发方法、开发平台的使用、与实际数据库的集成方法。

# 2 实验设计

## 2.1 概述

FTP协议（文件传输协议）是一种网络数据传输协议，用于文件传输，可以将文件从一台计算机发送到另一台计算机。传输的文件格式多种多样，内容包罗万象，如电子报表、声音、程序和文档文件等。

早期认为的FTP是一个ARPA计算机网络上主机间文件传输的用户级协议。他的主要功能是方便主机间的文件传输，并且允许在其他主机上进行方便的存储和文件处理。

而根据现在的FTP的定义，FTP的主要功能为：

* 促进文件的共享，包括计算机程序和数据。
* 支持间接地使用远程计算机。
* 不会因为各类主机文件存储系统的差异而受影响。
* 可靠并且有效的传输数据。

FTP可以被用户在终端使用，通常是给程序使用的。FTP中主要采用了TCP协议和Telnet协议。

## 2.2 实验原理

### 2.2.1 使用FTP协议下载文件的流程

以一个典型的FTP会话模型为例，如图2-1所示。在本地主机前的用户，希望把文件传送到一台远程主机上或者从这台远程主机上获取下载一些文件，用户需要做的是提供一个登录名和一个登录密码来进行访问。身份认证信息确认后，他就可以在本地文件系统和远程文件系统之间传送文件。如图2-1所示，用户通过一个FTP用户代理与FTP服务器交换信息。用户必须首先提供远程主机的主机名，这样才能让本地主机中的FTP客户端进程建立一个与远程主机中的FTP服务器端进程之间的连接通信。紧接着用户提供用户名和密码，这些信息将被确认为FTP参数经由TCP连接传送到服务器上。认证信息得到确认后，该用户就有权在本地文件系统和远程文件系统之间复制文件。

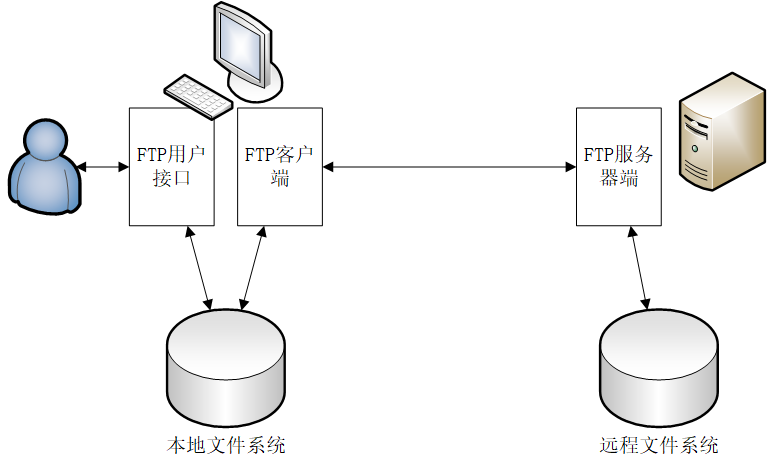


图2-1 典型FTP会话模型

一个完整的FTP文件传输需要建立两种类型的连接，一种为文件传输下命令，称为控制连接;另一种实现真正的文件传输，称为数据连接。这是与HTTP文件传送协议之间重要的差别，虽然它们有大多数共同特征，如都运行在TCP上。

1.控制连接:

当FTP客户端提出请求，需要与远程FTP服务器端实现.上传和下载文件时，它首先向服务器端的TCP端口号21发起一个建立连接的请求, FTP服务器接受来自客户端的请求。这个完成连接的建立过程就称为FTP控制连接。

2.数据连接

FTP控制连接建立之后，就开始传输文件，传输文件的连接称为FTP数据连接。FTP数据连接也就是FTP传输数据的所有过程，主要分为主动传输模式和被动传输模式。主动传输模式下，FTP服务器使用TCP端口号20与客户端的暂时端口进行连接，并传输数据，客户端只是处于接收数据状态。被动传输模式下，FTP服务器打开一个暂态端口等待客户端对其进行连接，并传输数据，服务器端不参与数据的主动传输，只是被动接收。图2-2形象地说明了FTP的控制连接和数据连接。

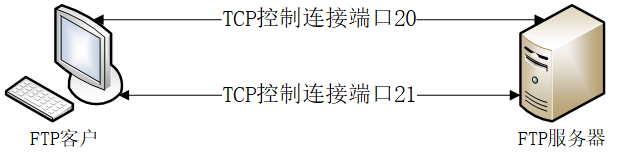


图2-1 典型FTP会话模型

### 2.2.2 相关类库说明

1. FtpWebRequest

FtpWebRequest类用于实现文件传输协议(FTP)客户端的操作,包括文件的删除、上传、下裁等。

表2-1 FtpWebRequest 类的重要方法

|  |  |
| --- | --- |
| 方法 | 说明 |
| Abort | 如果正在进行文件传输，用此方法来终止传输。如果没有进行任何操作，此方法不产生任何作用 |
| BeginGetResponse | 开始以异步方式向FTP服务器发送请求并从FTP服务器接收响应，建立控制连接 |
| Create | 初始化新的WebRequest对象 |
| EndGetResponse | 结束由BeginGetResponse 启动的挂起的异步操作 |
| Equals | 确定两个实例是否相等 |
| ToString | 返回表示当前对象的String |
| GetResponse | 返回FTP服务器响应 |

为了实现FTP功能的一般过程，先通过Create方法创建一个FtpWebRequest对象，指向FTP服务器的路径。

在创建好新的FtpWebRequest对象后，要设置FTP的执行方法，还有该类对象的属性设置，FtpWebRequest 的属性可以设置所开发的应用程序与相关FTP服务器之间的配置信息。

表2-2 FtpWebRequest 类的重要属性

|  |  |
| --- | --- |
| 名称 | 说明 |
| Credentials | 获取用于FTP服务器的通信凭证，指定登录FTP服务器的用户名和密码 |
| KeepAlive | 指定控制连接是应该关闭还是在请求完成之后关闭，默认为true |
| UseBinary | 指定文件的传输类型 |
| UsePassive | 指定数据传输的使用模式是主动模式还是被动模式 |
| ContentLength | 使FTP服务器将会提前预知文件的大小 |
| Method | 获取或设置要发送到FTP服务器的命令 |

2. FtpWebResponse

FtpWebResponse类用于封装文件传输协议(FTP)服务器对请求的响应，相当于FTP服务器端的操作实例。表2-3给出了FtpWebResponse 类的方法。

表2-3 FtpWebResponse 类的重要方法

|  |  |
| --- | --- |
| 名称 | 说明 |
| Close | 当在应用程序中启用网络跟踪功能后，此成员将输出跟踪信息 |
| Eguals | 确定两个对象实例是否相等 |
| GetResponseStream | 检索包含从FTP服务器上发送的响应数据的流 |
| ToString | 返回表示当前对象的String |

要获取FtpWebResponse 类的对象的实例，通过调用GetResponse 方法获取。GetResponse方法为WebResponse 的方法，当运用时，返回的对象必须强制转换成FtpWebResponsc。当应用程序不再需要FtpWebResponse对象时，调用Close 方法以释放FtpWebResponse所持有的资源。

表7-4 FtpWebResponse 类的属性

|  |  |
| --- | --- |
| 属性 | 说明 |
| BannerMessage | 获取在登录前建立连接时FTP服务器发送的消息 |
| ContentLength | 获取从FTP服务器上接收的数据的长度 |
| ContentType | 获取或设置接收的数据的内容类型 |
| ExitMessage | 获取FTP会话结束时服务器发送的消息 |
| IsFromCache | 获取一个布尔值，该值指示此响应是否为从缓存中获取的 |
| IsMutuallyAuthenticated | 获取一个布尔值，该值指示是否发生了相互身份验证 |
| LastModified | 获取FTP服务器上的文件的上次修改日期和时间 |
| ResponseUri | 获取对请求发送响应的URI |
| StatusCode | 获取从FTP服务器上发送的最新状态代码 |
| StatusDescription | 获取描述从FTP服务器发送的状态代码的文本 |
| WelcomeMessage | 获取身份验证完成时FTP服务器发送的消息 |

3. WebRequestMethods.Ftp

WebRequestMethods.Ftp类是.NET 2.0新增的类，它用于表示可与FTP请求一起使用的FTP协议方法的类型，其方法都是一些可重载的一般方法, 如equal()、GetHashCode()、ToString()等。对于该类，主要了解它的字段，它的每一个字段都代表了每一种具体的协议方法。如表2-5所示为WebRequestMethods.Ftp类的字段。

表2-5 WebRequestMethods.Ftp 类的字段

|  |  |
| --- | --- |
| 属性 | 说明 |
| AppendFile | 表示要用于将文件追加到FTP服务器上的现有文件的FTP APPE协议方法 |
| DeleteFile | 表示要用于删除FTP服务器上的文件的FTP DELE协议方法 |
| DownloadFile | 表示要用于从FTP服务器下载文件的FTP RETR协议方法 |
| GetFileSize | 表示要用于检索FTP服务器上的文件大小的FTP SIZE协议方法 |
| ListDirectory | 表示获取FTP服务器上的文件的简短列表的FTPNLIST协议方法 |
| ListDirectoryDetails | 表示获取FTP服务器上的文件的详细列表的FTP LIST协议方法 |
| MakeDirectory | 表示在FTP服务器上创建目录的FTP MKD协议方法 |
| PrintWorkingDircctory | 表示打印当前工作目录的名称的FTP PWD协议方法 |
| RemoveDirectory | 表示移除目录的FTP RMD协议方法 |
| Rename | 表示重命名目录的FTP RENAME协议方法 |
| UploadFile | 表示将文件上传到FIP服务器的FTP STOR协议方法 |
| UploadFileWithUniqueName | 表示将具有唯一名称的文件上传到FTP服务器的FTP STOU协议方法 |

## 2.3 实验方案

### 2.3.1 FTP客户端设计

在客户端的设计中，将采用图形化界面的方式来与服务器端进行交互，用户可以通过设计良好的界面，屏蔽掉FTP协议底层的繁复内容，简单操作即可达到目的。

#### 2.3.1.1需求分析

要设计一个友好的FTP客户端需要完成以下功能：

* 用户可以配置服务器的目标地址，指定FTP服务器，也可以使用默认地址。
* 用户可以使用预先分配的账号密码进行登录，获取操作文件的权限。
* 用户可以上传文件。
* 用户可以下载文件。
* 用户可以删除文件。
* 用户可以重命名文件。
* 用户可以查看客户端的所有文件。
* 用户可以查看客户端文件的详细信息。
* 用户在下载文件和上传文件时可以使用断点续传。

#### 2.3.1.2界面设计

我们为FTP客户端设计了两个活动的winform界面。

**1. Login.cs：**

手机屏幕的截图

描述已自动生成手机屏幕的截图

描述已自动生成

此Login界面的特点有：

1.界面清新，操作友好，逻辑清晰，用户容易上手。

2.使用了winform的无边框模式，既省去了原生界面繁重的边框，又可以让不同系统的用户3.使用完全相同的界面效果。

4.蓝-白配色简单优美，各式按钮的hover, click事件对应样式的改变。

5.所有的文本框均使用了none border, 但又在其下增加了一条高度为1的纯色label，在用户焦点锁定某一特定文本框时，label也会改变为蓝色，显示该文本框获得焦点，模拟了在C#中没有实现的仿Qt的lineEdit组件。

6.由于在登录时要进行一次FTP连接的尝试，当用户无法连接至服务器时，程序会经由异常处理模块显示出错误信息，而不会导致程序崩溃，大大加强了程序的健壮性。提示信息如下：

手机屏幕截图

描述已自动生成

以下列出主要的活动控件及作用：

|  |  |  |
| --- | --- | --- |
| 控件ID | 控件类名 | 描述 |
| IPBox | TextBox | 用户配置FTP服务器的IP地址，默认127.0.0.1 |
| IDBox | TextBox | 用户的ID，非必选项 |
| passwordBox | TextBox | 用户的密码，非必选项 |
| loginBtn | Button | 登录按钮，点击后向服务器发送一次LIST请求，成功后才可以进入下一界面 |
| exitBtn | Button | 退出按钮 |
| backBtn | Button | 返回按钮，可以返回到无认证模式 |
| useVerifyLabel | Button | 类似于Button，点击后切换到需认证模式 |

**2. MainWindow.cs：**

**社交网络的手机截图

描述已自动生成**

主要控件如下：

|  |  |  |
| --- | --- | --- |
| 控件ID | 控件类名 | 描述 |
| fileListBox | ListBox | 用于显示服务器上的所有文件 |
| uploadBtn | Button | 上传按钮 |
| getFileBtn | Button | 获取服务器上所有文件名的按钮 |
| renameBtn | Button | 重命名按钮 |
| downloadBtn | Button | 下载按钮 |
| deleteBtn | Button | 删除按钮 |
| detailBtn | Button | 查看文件细节按钮 |
| exitBtn | Button | 退出按钮 |
| backBtn | Button | 返回按钮，可以返回登录界面 |
| detailLabel | Label | 显示文件细节的标签，但不由detailBtn触发修改 |
| infoLabel | Label | 显示登录用户信息的标签 |
| filenameBox | TextBox | 用户输入重命名后的文件名 |

1.此界面沿用了整体设计时的蓝白配色，两个界面整体性很强，用户不会有割裂感。

手机屏幕的截图

描述已自动生成社交网络的手机截图

描述已自动生成

2.对ListBox的measure\_Item与Draw\_Item事件进行了重写，可以手动设置每一项显示的字体、前景色、背景色、大小, 当某一项被选中时还会显示高亮与字体加粗，用户使用时便可一目了然，具体效果如下：

手机屏幕截图

描述已自动生成 手机屏幕截图

描述已自动生成

3.沿用了Login界面的许多空间，蓝底白字的Button，hover与click事件样式改变。

4.对所有可能的服务器无响应情况进行了异常处理，程序不会因为服务器端的问题导致崩溃：

社交网站的手机截图

描述已自动生成

5.用户如果要删除或重命名文件时，会弹出一个确认对话框，防止用户的误操作，极大增强了界面的友好性与安全性。

社交网站的手机截图

描述已自动生成

6.用户在上传或者下载文件时可以使用断点续传功能。

社交网络的手机截图

描述已自动生成

#### 2.3.2.3业务逻辑

引入命名空间的语句：

using System;

using System.IO;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Text;

using System.Windows.Forms;

using System.Net;

实现业务逻辑的函数：

（1）实现文件上传的函数

首先从参数中获得需上传的文件名，然后用包含文件路径的URI和文件名来实例化FtpWebRequest 对象，在获得reqFTP对象后可以对它的一系列属性进行修改以满足文件传输的需要。在做完了准备工作后，把要上传的文件写到一个文件流中以让服务器端可以进行接收。在完成将文件内容从文件流写入上传流后，关闭文件流和FtpWebRequest流结束操作。

（2）实现删除功能的函数

首先实例化一个FtpWebRequest的对象,接着把该对象的Method属性设置成DeleteFile。然后,在reqFTP对象上调用GetResponseStream方法获得应答流。最后，把sr对象上的文件流从头到尾读到result上。

（3）获取文件详细信息列表的函数

在获得ftp 对象后，对它的Method属性设置成了ListDirectoryDetails,就可以列出该文件的详细信息了。

（4）实现下载功能的函数

首先在获得了FtpWebRequest对象以后，通过设置Method属性为DownloadFile，可以进行下载的动作。接着，获取了一个应答流以处理服务器上的文件。通过获得服务器端的文件长度，可以根据指定的尺寸(这里是2KB)单位进行文件下载，其实也就是把服务器端的文件写入到一个文件流上。最后，只需关闭文件流就可以完成下载的动作。

（5）获取文件长度的函数

首先根据输入的文件名指定具体的文件，在获得reqFTP对象之后，将该对象的Method 属性设置成GetFileSize，以便进行获取文件长度的动作。最后，通过得到文件的应答流来获取服务器端文件的大小。

（6）实现重命名功能的函数

首先得到一个FtpWebRequest 对象，并将它的Method属性设置成Rename，然后通过传入的原文件名和修改后的文件名进行修改的动作。

（7）实现新建目录功能的函数

首先根据uri 参数创建FtpWebRequest对象，并将其Method属性设置成MakeDirectory便可以进行创建目录的操作。

（8）实现断点续传功能的函数

断点续传功能是当用户下载大文件时，由于主观或外部原因中途停止下载，已经下载的部分将作为缓存内容存放在系统中，当用户再次下载时，可以从断点开始继续下载。

后期为了实现断点续传功能，我设计了一个类DownloadClass，含有一个方法download\_method,并含有多个成员变量，以此来启动多线程，用户便可以在下载中暂停下载。为了记录断点信息，创建了一个downloadState.dat文件，以以下格式存放下载状态：

filename\tfilepath

分别记录已缓存的文件的名称与缓存地址，当用户要启动下载时，就会读取这个文件内容，如果文件中存在该filename的记录，则启动断点续传模式，读取filepath的大小，设置FTP请求的ContentOffset字段，即发送REST指令，进行断点续传，当文件成功下载时，再将filename从.dat文件中删除即可，我将这些功能整合到一个DownloadHelper类中，含有以下三个静态方法：

checkDownloadData, 返回一个long类型变量，表示ContentOffset的值

checkDownlaodState, 检查该filename是否存在于断点续传文件中

editDownloadState, 将该filename从断点续传文件中删除。

然后再将ftp请求的method设置为DownloadFile，这样就会同时发送RETR和REST请求，开始断点续传了。

### 2.3.2 FTP服务器端设计

在对服务器的设计中，将采用控制台命令的方式来处理来自客户端的命令，使用这种方式可以更清晰地看出服务器是如何与客户端进行交互的。

#### 2.3.2.1需求分析

对于服务器而言，服务器端设计需求如下：

* 可以新增用户，并设置本地的主目录供客户端操作。
* 可以删除用户。
* 可以罗列出用户列表，显示合法用户及其主目录所在。
* 可以退出服务器控制台。
* 可以显示帮助文件，查看命令的说明。

以上这些功能是通过使用命令的方式在控制台上实现的，但是这只是服务器端针对用户管理的一个线程。对于服务器而言，最重要的任务莫过于能够提供客户端进行上传、下载等一系列功能，而我们是在另一个线程中提供对客户机的FTP命令进行解析并做出响应的。针对客户端的命令，实现的后台文件管理的线程的需求如下：

* 可以接受客户机连接服务器
* 可以接受用户进行远程登陆（针对合法的用户）
* 可以提供用户对主目录上的文件进行下载
* 可以允许用户将本地的文件上传至服务器上
* 可以允许用户查看主目录上的信息
* 可以允许用户在服务器上创建目录，更改目录和删除目录
* 可以获取文件长度

#### 2.3.2.2 FTP命令字和响应码

**FTP命令字**

1.连接操作命令

实验使用到的连接操作命令包括USER、PASS。

（1）USER命令

用于发送登录FTP服务器的用户名称，命令格式: USER username。

参数username是在FTP服务器数据库上的账号，登录FTP服务器前先要建立一个账号连接。在客户端连接到FTP服务器的端口(端口号21),并接收到一个由代码220开头的行后，表示与服务器连接建立成功，即可以向服务器发送USER与PASS命令以登录FTP服务器。

（2）PASS命令

用于向远程服务器发送用户的密码，该命令在USER 之后使用，命令格式: PASS password。

参数password是登录用户的账户密码，当用户名和密码同时输入正确时，FTP 服务器登录成功。在发送完USER命令并接收到正确的响应后，需要立即执行PASS命令发送账户密码。

2.设置模式命令

（1）TYPE命令

用于确定数据的传输方式，-般使用TYPE命令在ASCII和二进制模式之间切换。二进制模式类型在发送和接收任何类型的文件时都可以使用，ASCII类型只在传送文本时使用。

（2） PORT命令

用于为数据连接指定一个IP地址和本地端口,用户和服务器都有默认的数据端口，在正常情况下无须用到这个命令，但在执行一些命令时，如RETR,需要额外建立一个数据通道传送数据。命令格式: PORT hl,h2,h3,h4,pl,p2。

3.目录操作命令

（1）CWD命令

用于改变远程文件系统的指定路径。命令格式: CWD folder。

参数folder是新的目录名字。

（2）LIST命令

用于列出指定目录中的子目录和文件信息。如果没有指定目录的名字，系统默认列出当前目录下的所有子目录和文件信息并返回客户端。

（3）NLST命令

用于让服务器给客户发送一份目录列表，返回内容只包括文件名称。

（4）CUDP命令

用于把当前目录改成远程文件系统的根目录，此命令无需改变登录、登录信息或传输参数。

（5）PWD命令

用于在应答中返回当前工作的目录名称，该命令没有参数。

（6）DELE命令

用于删除目录或者文件。DELE后面要跟删除的文件或者目录名参数。

（7）RNFR命令

对旧路径重命名

（8）RNTO命令

对新路径重命名

4.发送操作命令

STOR命令用于向服务器发送一个文件。

5.获取操作命令

（1）RETR命令

用于从服务器接收一个文件。

（2）REST命令

用于标识出文件内的数据点，将从这个点开始继续传送文件。REST不会引起文件传送，而只是跳到文件中指定的数据检验点上。

**FTP响应码**

|  |  |
| --- | --- |
| 响应码 | 解释说明 |
| 110 | 新文件指示器上的重启标记 |
| 120 | 服务器准备就绪的时间（分钟数） |
| 125 | 打开数据连接，开始传输 |
| 150 | 打开连接 |
| 200 | 成功 |
| 202 | 命令没有执行 |
| 211 | 系统状态回复 |
| 212 | 目录状态回复 |
| 213 | 文件状态回复 |
| 214 | 帮助信息回复 |
| 215 | 系统类型回复 |
| 220 | 服务就绪 |
| 221 | 退出网络 |
| 225 | 打开数据连接 |
| 226 | 结束数据连接 |
| 227 | 进入被动模式（IP 地址、ID 端口） |
| 230 | 登录因特网 |
| 250 | 文件行为完成 |
| 257 | 路径名建立 |
| 331 | 要求密码 |
| 332 | 要求帐号 |
| 350 | 文件行为暂停 |
| 421 | 服务关闭 |
| 425 | 无法打开数据连接 |
| 426 | 结束连接 |
| 450 | 文件不可用 |
| 451 | 遇到本地错误 |
| 452 | 磁盘空间不足 |
| 500 | 无效命令 |
| 501 | 错误参数 |
| 502 | 命令没有执行 |
| 503 | 错误指令序列 |
| 504 | 无效命令参数 |
| 530 | 未登录网络 |
| 532 | 存储文件需要帐号 |
| 550 | 文件不可用 |
| 551 | 不知道的页类型 |
| 552 | 超过存储分配 |
| 553 | 文件名不允许 |

#### 2.3.2.3 业务逻辑设计

引入命名空间的语句：

using System;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Net;

using System.Net.Sockets;

using System.IO;

using System.Threading;

using System.Threading.Tasks;

UserInfo类：

这个类封装了用户名、密码和主目录。凡是涉及这三者的操作，都可以转换为对其成员的动作。

FtpState类：

这个类负责描述FTP各个状态标志，如登陆状态、等待密码状态等。这些标志对程序的执行流程都有非常关键的作用。

FtpServer类：

（1）FtpServer的构造函数：

在FtpServer类中，首先定义了两个动态数组，在客户端接入连接后，users数组保存后台用来处理ftp命令线程的句柄。而userinfo则保存表示每个接入的客户端信息，每个数组元素就是一个Userinfo类的成员。而initDir是指初始化的主目录路径，并在Main函数里设置默认值为c盘根目录。

在FtpServer的构造函数中，首先构造一个FtpConsole对象，它的基本作用是在控制台上显示命令和密码。在启动了FtpConsole类的CobsoleThread线程之后，将可以看到控制台可以处理关于用户管理的命令。接着，执行loaduserinfo方法，它的作用将在后面描述。然后通过tcplistener来监听是否有请求接入的客户端，并把接入后的tcplistener对象传给incoming，在准备好读写流后，将进行通信的incoming对象和接入序号传递给FtpHandler类的构造函数，启动了该类的FtpHandler线程后，将使程序具有可以在后台处理ftp命令的功能，并将该线程句柄加到users数组中。

（2）加载用户信息的函数

创建一个服务线程并启动该线程。把该线程加到users中。

Loaduserinfo方法即加载用户信息方法，会获取一个user.cfg文件，该文件保存用户名以及密码等信息。在得到该文件句柄后，对他进行解析，把用户名、密码等字段做逐一的拆分并传递给相应的变量。最后，把userinfo对象加到动态数组中，也就是说，没调用一次该方法就在用户信息这个数组中增加一个Userinfo对象。

FtpHandler类：

这个类负责处理用户在客户端请求的命令，然后通过服务器处理返回操作的响应码。

（1）初始化参数

进行诸如套接字、工作目录、主机IP地址的初始化等工作，为解析输入作准备。

（2）解析输入命令字符串函数

将输入的字符串以某种方式进行解析，得到输入的命令名，然后进行匹配，根据不同的命令名返回不同的响应码。

（3）验证路径的函数

首先根据输入的信息判断是否存在有该路径，如果不存在即可报错。

在验证的过程中先获取路径,然后通过得到文件名与之组合，得到该文件的绝对路径。

（4）校验密码的函数

根据输入的密码判断正确与否，正确返回true，错误返回false。

（5）USER命令的业务逻辑函数

当用户输入USER 时，显示331 User name okay, need password,并等待密码验证。

（6）PASS命令的业务逻辑函数

当用户输入PASS命令后，判别密码是否合法并执行相应的登录动作。

（7）错误函数

未识别当前命令时，显示命令错误

（8）CDUP命令的业务逻辑函数即寻找当前目录的业务逻辑函数

找到ftp的工作目录，如果该目录没有上级目录则会在控制台上打印错误语句。

（9）CWD命令的业务逻辑函数即判别目录的函数

根据输入的命令判断是否存在相应的工作目录，如果存在将标志置为真，否则置为假，并在控制台上打印相关信息。

（10）QUIT命令的业务逻辑函数即退出命令的函数

（11）PORT命令的业务逻辑函数即判别格式的函数

判断检验输入的参数是否有格式上的问题，如果有，则在控制台上打印相关信息。

（12）LIST命令的业务逻辑函数即命令列表的函数

获取指定目录的文件详细信息并开始传输。若成功则提示226 Transfer complete!,否则显示451 Requested action aborted: local error in processing。

（13）NLIS命令的业务逻辑函数

获取指定目录的所有文件名并开始传输。若成功则提示226 Transfer complete!,否则显示451 Requested action aborted: local error in processing。

（14）TYPE命令的业务逻辑函数即得到命令类型的函数

判断命令的类型采用的是二进制模式还是图像模式等，并打印相关信息。

（15）RETR命令的业务逻辑函数即初始化工作环境的函数

主要作用是初始化工作环境，判断以二进制模式还是图像模式来执行。并使用网络流将输入的命令写到控制台上。如果程序成功，则会显示“226 Transfer complete !”的字样。否则，出现的语句应该是451 Requested action aborted: local error in processing。

（16）STOR命令的业务逻辑函数

根据模式存储发送到服务器的文件。

（17）PWD命令的业务逻辑函数

获取当前目录。

（18）NOOP命令的业务逻辑函数

返回200 OK表示命令确认。

（19）ABOR命令的业务逻辑函数

关闭套接字，当出现错误时，会在控制台上打印451 Requestedaction aborted: local error in processing语句。

（20）DELE命令的业务逻辑函数即删除文件函数

先判断用户所指向的文件路径是否合法，如果不合法，则不允许删除，并提示出错。如果存在该文件，则将其删除。

（21）增加尾缀的函数

给字符串加尾缀，以/为结束符

（22）构造函数

（23）线程函数

先在构造函数中得到传入的Tcpclient对象以进行针对指定客户端的数据传输，在HandleThread线程中，负责将客户端的命令传送到服务器上并进行命令的翻译，即将命令和参数分离。

（24）REST命令的业务逻辑函数

后期实验中我们添加了断点续传功能，进行断点续传时，会发送一条REST指令并带有下载的偏移量，在读到REST指令后，直接将该long变量存放在offset成员变量中，在下一步进行RETR指令时，即可读到offset非零，在传送时就可以从offset开始了。注意的是，这样每次传送结束都要将offset重新置零，因为不是所有的FTP请求都带有REST指令。

FtpConsole类：

该类承担在控制台上的所有业务逻辑。

（1）退出函数

退出控制台采用return方法返回。

（2）列出用户列表的函数

（3）验证是否是合法用户的函数

根据传入的用户名称验证是否与文件中的名称匹配。

（4）增加用户的函数

根据用户的输入信息保存其资料，并在userinfo对象中保存信息，接着打印用户添加成功的语句。然后，设置密码，若密码为空，则会报错。以上工作完成后，为用户设置操作的主目录。以后该用户的文件传输都只能在工作目录下进行。如果目录不存在，则会显示出错信息，最后保存该用户信息。

（5）保存用户信息函数

将用户输入的信息写道一个叫user.cfg的文件上，保存的时候按照规定的数据结构进行，以便于以后提取用户信息时采用。

（6）删除用户的函数

首先输入要删除的用户名的名称，如果输入的名称不存在则提示出错。

（7）显示帮助的函数

用户输入help命令即可弹出相应的信息，并可以进一步通过出入具体命令查询该命令的用法。

（8）出错函数

（9）构造函数

（10）控制台线程函数

首先读入命令，打印一个->间隔符号，然后解析输入的命令，返回响应码，每个响应码针对一个不同的操作。

（11）解析输入的函数

在这个FtpConsole类中，实现了很多命令，如adduser，listuser命令。它首先打开一个ConsoleThread线程，这个线程负责接收控制台的命令并调用其中的parseInput进行解析，然后根据返回值判断应该调用何种命令的函数。

# 结论

## 1. 程序主要界面及结果

启动server：

手机屏幕的截图

描述已自动生成

添加一个监听用户，并设置用户名与密码，设置目录地址：

电脑截图

描述已自动生成

启动client,输入IP地址（用环回地址测试）及刚才设置的用户名密码，点击登录：

手机屏幕的截图

描述已自动生成

登录成功后，右上角会提示当前用户的欢迎语。当前目录下还没有文件，此时点击上传文件按钮，选择需要上传的文件：

社交网站的手机截图

描述已自动生成

确认后，显示上传成功，并且已经显示出来：

社交网络的手机截图

描述已自动生成社交网络的手机截图

描述已自动生成

点击文件后，右侧显示出文件名与文件的详细信息，也可以通过点击细节按钮，获取TCP的原始数据。

社交网站的手机截图

描述已自动生成社交网站的手机截图

描述已自动生成

再上传多个文件后，选中文件后点击下载按钮，选择保存路径与文件名，下载成功后，可以在对应目录下找到下载的文件。

社交网站的手机截图

描述已自动生成手机屏幕截图

描述已自动生成

选择需要重命名的文件，填写重命名后的新文件名，点击确认：

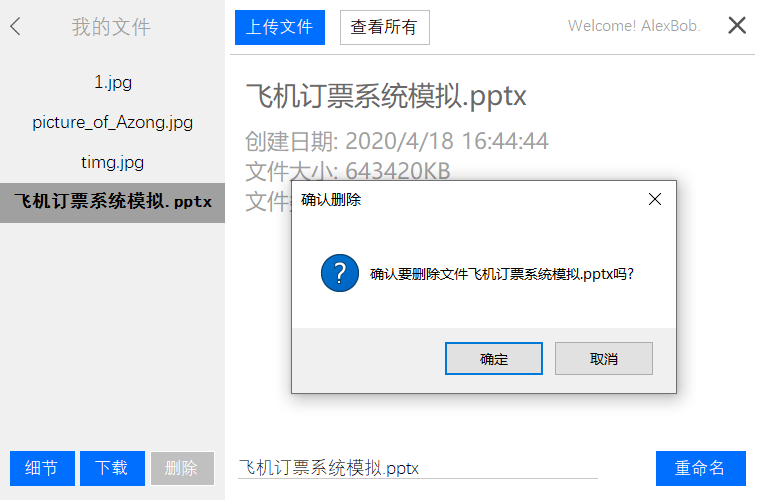
社交网络的手机截图

描述已自动生成社交网站的手机截图

描述已自动生成社交网络的手机截图

描述已自动生成

选择需要删除的文件，点击删除按钮：



另外，后期我们也增加实现了断点续传的功能，让程序拥有更加强大的能力。

手机屏幕截图

描述已自动生成

社交网络的手机截图

描述已自动生成

截图里有图片

描述已自动生成

手机屏幕截图

描述已自动生成截图里有图片

描述已自动生成

## 2. 程序源程序

客户端:

[MainWindow.cs](#mw)(710 lines), [Form1.cs](#f1)(120 lines)

界面设计:

服务器端:

[Program.cs](#server)(1450 lines)

### 客户端代码：

ftpclient/MainWindow.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.IO;

using System.Net;

namespace FTPApplication\_WHU

{

public partial class MainWindow : Form

{

private Form1 parent;

private Thread downloadThread;

Point mouseOff;//用于获取鼠标位置

bool leftFlag;//移动标识

public MainWindow(Form1 parent)

{

this.parent = parent;

this.parent.Hide();

this.StartPosition = FormStartPosition.CenterScreen;

InitializeComponent();

if (this.parent.ftpUserID == "test")

{

this.fileListBox.Items.Add("测试用例1");

this.fileListBox.Items.Add("测试用例2");

this.fileListBox.Items.Add("测试用例3");

}

else

{

this.fileListBox.Items.Clear();

}

}

private void Upload(string filename)//上传方法

{

FileInfo fileInf = new FileInfo(filename);

string uri = "ftp://" + this.parent.ftpServerIP + "/" + fileInf.Name;

FtpWebRequest reqFTP;

reqFTP = (FtpWebRequest)FtpWebRequest.Create(new Uri("ftp://" + this.parent.ftpServerIP + "/" + fileInf.Name));

reqFTP.Credentials = new NetworkCredential(this.parent.ftpUserID, this.parent.ftpPassword);

reqFTP.KeepAlive = false;

reqFTP.UsePassive = false;

reqFTP.Method = WebRequestMethods.Ftp.UploadFile;

reqFTP.UseBinary = true;

//指定主动方式

reqFTP.UsePassive = false;

reqFTP.ContentLength = fileInf.Length;

int buffLength = 2048;

byte[] buff = new byte[buffLength];

int contentLen;

//打开一个文件流来读入上传的文件

FileStream fs = fileInf.OpenRead();

try

{

Stream strm = reqFTP.GetRequestStream();

contentLen = fs.Read(buff, 0, buffLength);

while (contentLen != 0)

{

strm.Write(buff, 0, contentLen);

contentLen = fs.Read(buff, 0, buffLength);

}

strm.Close();

fs.Close();

MessageBox.Show("上传成功！");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "上传出错");

}

}

public void DeleteFTP(string fileName)//删除功能

{

try

{

string uri = "ftp://" + this.parent.ftpServerIP + "/" + fileName;

FtpWebRequest reqFTP;

reqFTP = (FtpWebRequest)FtpWebRequest.Create(new Uri("ftp://" + this.parent.ftpServerIP + "/" + fileName));

reqFTP.Credentials = new NetworkCredential(this.parent.ftpUserID, this.parent.ftpPassword);

reqFTP.KeepAlive = false;

reqFTP.Method = WebRequestMethods.Ftp.DeleteFile;

reqFTP.UsePassive = false;

string result = String.Empty;

FtpWebResponse response = (FtpWebResponse)reqFTP.GetResponse();

long size = response.ContentLength;

Stream datastream = response.GetResponseStream();

StreamReader sr = new StreamReader(datastream);

result = sr.ReadToEnd();

sr.Close();

datastream.Close();

response.Close();

MessageBox.Show("删除成功!");

}

catch

{

MessageBox.Show("删除失败，刷新或稍后再试!");

}

}

public string[] GetFileList()//获取文件列表方法

{

string[] downloadFiles;

StringBuilder result = new StringBuilder();

FtpWebRequest reqFTP;

try

{

reqFTP = (FtpWebRequest)FtpWebRequest.Create(new Uri("ftp://" + this.parent.ftpServerIP + "/"));

reqFTP.UseBinary = true;

reqFTP.Credentials = new NetworkCredential(this.parent.ftpUserID, this.parent.ftpPassword);

reqFTP.Method = WebRequestMethods.Ftp.ListDirectory;

reqFTP.UsePassive = false;

WebResponse response = reqFTP.GetResponse();

StreamReader reader = new StreamReader(response.GetResponseStream());

string line = reader.ReadLine();

while (line != null)

{

result.Append(line);

result.Append("\n");

line = reader.ReadLine();

}

result.Remove(result.ToString().LastIndexOf('\n'), 1);

reader.Close();

response.Close();

return result.ToString().Split('\n');

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

downloadFiles = null;

return downloadFiles;

}

}

private string[] GetFilesDatailList(string filename)//获取文件详细信息列表

{

// if failed ,return null info.

string[] ret = null;

try

{

StringBuilder result = new StringBuilder();

FtpWebRequest ftp;

ftp = (FtpWebRequest)FtpWebRequest.Create(new Uri("ftp://" + this.parent.ftpServerIP + "/"));

ftp.Credentials = new NetworkCredential(this.parent.ftpUserID, this.parent.ftpPassword);

ftp.Method = WebRequestMethods.Ftp.ListDirectoryDetails;

ftp.UsePassive = false;

WebResponse response = ftp.GetResponse();

StreamReader reader = new StreamReader(response.GetResponseStream());

string line = reader.ReadLine();

while (line != null)

{

string[] info = line.Split('\t');

if (info[0] == filename)

{

ret = info;

break;

}

else

{

line = reader.ReadLine();

}

}

return ret;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return ret;

}

}

private void Download(string filePath, string fileName, long offset = 0)//下载方法

{

FtpWebRequest reqFTP;

Uri u = new Uri("ftp://" + parent.ftpServerIP + "/" + fileName);

try

{

FileStream outputStream;

if (File.Exists(filePath))

{

outputStream = new FileStream(filePath, FileMode.Append);

}

else

{

outputStream = new FileStream(filePath + "\\" + fileName, FileMode.Create);

}

reqFTP = (FtpWebRequest)FtpWebRequest.Create(u);

reqFTP.Method = WebRequestMethods.Ftp.DownloadFile;

reqFTP.UseBinary = true;

reqFTP.UsePassive = false;

if (offset != 0)

{

reqFTP.ContentOffset = offset;

}

reqFTP.Credentials = new NetworkCredential(this.parent.ftpUserID, this.parent.ftpPassword);

FtpWebResponse response = (FtpWebResponse)reqFTP.GetResponse();

Stream ftpStream = response.GetResponseStream();

long cl = response.ContentLength;

download\_class dc = new download\_class();

dc.filePath = filePath;

dc.fileName = fileName;

dc.ftpStream = ftpStream;

dc.outputStream = outputStream;

dc.response = response;

dc.total = cl;

dc.offset = offset;

downloadThread = new Thread(dc.download\_method);

downloadThread.Start();

}

catch(Exception e)

{

}

}

private string GetFileSize(string filename)//获取文件长度方法

{

string fileSize;

FtpWebRequest reqFTP;

try

{

reqFTP = (FtpWebRequest)FtpWebRequest.Create(new Uri("ftp://" + this.parent.ftpServerIP + "/" + filename));

reqFTP.UseBinary = true;

reqFTP.Credentials = new NetworkCredential(this.parent.ftpUserID, this.parent.ftpPassword);

reqFTP.Method = WebRequestMethods.Ftp.GetFileSize;

reqFTP.UsePassive = false;

WebResponse response = reqFTP.GetResponse();

StreamReader reader = new StreamReader(response.GetResponseStream());

fileSize = reader.ReadLine();

reader.Close();

response.Close();

return fileSize;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

return null;

}

}

private void Rename(string currentFilename, string newFilename)//重命名方法

{

FtpWebRequest reqFTP;

try

{

reqFTP = (FtpWebRequest)FtpWebRequest.Create(new Uri("ftp://" + this.parent.ftpServerIP + "/" + currentFilename));

reqFTP.Method = WebRequestMethods.Ftp.Rename;

reqFTP.RenameTo = newFilename;

reqFTP.UseBinary = true;

reqFTP.UsePassive = false;

reqFTP.Credentials = new NetworkCredential(this.parent.ftpUserID, this.parent.ftpPassword);

FtpWebResponse response = (FtpWebResponse)reqFTP.GetResponse();

Stream ftpStream = response.GetResponseStream();

ftpStream.Close();

response.Close();

MessageBox.Show("重命名成功!");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/\*\*

\* 注意，以下函数与功能实现无关

\*

\*/

// 用于拖动窗口

private void Form1\_MouseDown(object sender, MouseEventArgs e)

{

mouseOff = new Point(e.X, e.Y);//获取当前鼠标位置

leftFlag = true;//用于标记窗体是否能移动(此时鼠标按下如果说用户拖动鼠标则窗体移动)

}

private void Form1\_MouseMove(object sender, MouseEventArgs e)

{

if (leftFlag)

{

Location = new Point(Control.MousePosition.X - mouseOff.X, Control.MousePosition.Y - mouseOff.Y);

}

}

private void Form1\_MouseUp(object sender, MouseEventArgs e)

{

if (leftFlag)

{

leftFlag = false; //释放鼠标标识为false 表示窗体不可移动

}

}

// 当此界面关闭时，同时关闭隐藏着的父窗口

private void MainForm\_FormClosed(object sender, FormClosedEventArgs e)

{

this.parent.Close();

}

// listbox的表现效果，也许Qt和WPF可以写的很简单

private void fileListBox\_DrawItem(object sender, DrawItemEventArgs e)

{

Color foreColor;

Font font;

e.DrawBackground();

if ((e.State & DrawItemState.Selected) == DrawItemState.Selected)

{//如果当前行为选中行。

//绘制选中时要显示的蓝色边框。

Color c = SystemColors.ControlDark;

foreColor = Color.Black; //Color.FromArgb(6, 82, 121);

font = new Font("黑体",11, FontStyle.Bold);

e.Graphics.FillRectangle(new SolidBrush(c), e.Bounds);//绘制背景

}

else

{

font = e.Font;

foreColor = e.ForeColor;

}

// e.DrawFocusRectangle();

StringFormat strFmt = new System.Drawing.StringFormat();

strFmt.Alignment = StringAlignment.Center; //文本垂直居中

strFmt.LineAlignment = StringAlignment.Center; //文本水平居中

e.Graphics.DrawString(fileListBox.Items[e.Index].ToString(), font, new SolidBrush(foreColor), e.Bounds, strFmt);

}

private void fileListBox\_MeasureItem(object sender, MeasureItemEventArgs e)

{

e.ItemHeight = 40;

}

// 点击关闭

private void closeBtn\_Click(object sender, EventArgs e)

{

this.Close();

}

// 删除

private void deleteBtn\_Click(object sender, EventArgs e)

{

if (fileListBox.SelectedItem == null)

{

MessageBox.Show("请选择你需要删除的文件!");

}

else

{

string fileName = fileListBox.SelectedItem.ToString();

DialogResult dr = MessageBox.Show($"确认要删除文件{fileName}吗?", "确认删除",

MessageBoxButtons.OKCancel, MessageBoxIcon.Question);

if (dr == DialogResult.OK)

{

DeleteFTP(fileName);

getFileBtn\_Click(null, null);

}

}

}

// 获取文件列表

private void getFileBtn\_Click(object sender, EventArgs e)

{

string[] filenames = this.GetFileList();

fileListBox.Items.Clear();

try

{

foreach (string filename in filenames)

{

fileListBox.Items.Add(filename);

}

}

catch

{

}

}

// 上传文件

private void uploadBtn\_Click(object sender, EventArgs e)

{

OpenFileDialog opFilDIg = new OpenFileDialog();

if (opFilDIg.ShowDialog() == DialogResult.OK)

{

Upload(opFilDIg.FileName);

getFileBtn\_Click(null, null);

}

}

// 下载文件

private void downloadBtn\_Click(object sender, EventArgs e)

{

if (fileListBox.SelectedItem == null)

{

MessageBox.Show("请选择需要下载的文件!");

return;

}

string fileName = fileListBox.SelectedItem.ToString();

if (fileName != string.Empty)

{

}

else

{

MessageBox.Show("请选择下载的文件！");

}

long offset;

if ((offset= DownloadHelper.checkDownloadState(fileName)) != -1)

{

string path = DownloadHelper.checkDownloadFile(fileName);

Download(path,fileName, offset);

}

else

{

FolderBrowserDialog fldDlg = new FolderBrowserDialog();

fldDlg.ShowDialog();

Download(fldDlg.SelectedPath, fileName);

}

}

private void renameBtn\_Click(object sender, EventArgs e)

{

if (fileListBox.SelectedItem == null)

{

MessageBox.Show("必须选中你想要重命名的文件！");

}

else

{

string currentFileName = fileListBox.SelectedItem.ToString();

string newFileName = filenameBox.Text.ToString();

DialogResult dr = MessageBox.Show($"确认要重命名文件{currentFileName}->{newFileName}吗?", "确认重命名",

MessageBoxButtons.OKCancel, MessageBoxIcon.Question);

if (dr == DialogResult.OK)

{

if (newFileName.Trim() != "")

{

Rename(currentFileName, newFileName);

getFileBtn\_Click(null, null);

}

else

{

MessageBox.Show("重命名不能为空!");

}

}

}

}

// 返回到父界面（可以修改IP和用户信息), 同时隐藏该界面

private void backBtn\_Click(object sender, EventArgs e)

{

this.parent.Show();

this.Hide();

}

// 如果此界面可见，则父界面不可见，同时在这里设置欢迎语

private void MainWindow\_VisibleChanged(object sender, EventArgs e)

{

if (this.Visible)

{

this.parent.Hide();

}

if (this.parent.ftpUserID == "test")

{

this.InfoLabel.Text = "Test Mode.";

}

else

{

this.InfoLabel.Text = $"Welcome! {this.parent.ftpUserID}. ";

}

}

//closeBtn的hover特效

private void closeBtn\_MouseEnter(object sender, EventArgs e)

{

this.closeBtn.BackgroundImage = (System.Drawing.Image)Properties.Resources.ResourceManager.GetObject("close\_clicked");

}

private void closeBtn\_MouseLeave(object sender, EventArgs e)

{

this.closeBtn.BackgroundImage = (System.Drawing.Image)Properties.Resources.ResourceManager.GetObject("close");

}

// 当重新选中文件时，把文件名填入filenamebox内（默认不重命名)

private void fileListBox\_SelectedIndexChanged(object sender, EventArgs e)

{

if (fileListBox.SelectedItem != null)

{

string filename = fileListBox.SelectedItem.ToString();

filenameBox.Text = filename;

string[] details = GetFilesDatailList(filename);

titleLabel.Text = filename;

restartLabel\_set(filename);

if (details == null)

{

DetailLabel.Text = "无法获取文件的详细信息！请重试";

}

else

{

DetailLabel.Text = $"创建日期: {details[1]}\n文件大小: {details[2]}KB\n文件类型: {details[3]}";

}

}

}

private void DetailLabel\_Click(object sender, EventArgs e)

{

}

private void MainWindow\_Load(object sender, EventArgs e)

{

}

private void detail\_Click(object sender, EventArgs e)

{

if (fileListBox.SelectedItem == null)

{

MessageBox.Show("请选择文件!");

}

else

{

string filename = fileListBox.SelectedItem.ToString();

string[] details = GetFilesDatailList(filename);

if (details == null)

{

MessageBox.Show("无法获取文件的详细信息！请重试");

}

else

{

MessageBox.Show(

$"Raw response fropm server to {parent.ftpUserID}:\n{details[0]}\n{details[1]}\n{details[2]}\n{details[3]}");

}

}

}

private void pauseBtn\_Click(object sender, EventArgs e)

{

if (downloadThread.IsAlive)

{

downloadThread.Abort();

MessageBox.Show($"已停止下载!");

if (fileListBox.SelectedItem != null)

{

string fn = fileListBox.SelectedItem.ToString();

restartLabel\_set(fn);

}

}

}

private void restartLabel\_set(string filename)

{

restarLabel.Text = "";

long offset;

if ((offset = DownloadHelper.checkDownloadState(filename)) != -1)

{

restarLabel.Text = $"可以启用断点续传,已下载{offset / 1024.0}KB";

downloadBtn.Text = "续传";

}

else

{

downloadBtn.Text = "下载";

}

}

private void clearCacheBtn\_Click(object sender, EventArgs e)

{

if (fileListBox.SelectedItem != null)

{

string fn = fileListBox.SelectedItem.ToString();

DownloadHelper.editDownloadData(fn);

restartLabel\_set(fn);

}

}

}

public class DownloadHelper

{

public static void editDownloadData(string filename)

{

FileStream fs = new FileStream("downloadState.dat", FileMode.OpenOrCreate);

string all = "";

StreamReader sr = new StreamReader(fs);

string line = null;

while ((line = sr.ReadLine()) != null)

{

string[] data = line.Split('\t');

if (data[0] != filename)

{

all += line;

all += "\n";

}

}

sr.Close();

fs.Close();

FileStream fs\_2 = new FileStream("downloadState.dat", FileMode.Create);

StreamWriter sw = new StreamWriter(fs\_2);

sw.Write(all);

sw.Close();

fs\_2.Close();

}

public static string checkDownloadFile(string filename)

{

FileStream fs = new FileStream("downloadState.dat", FileMode.OpenOrCreate);

StreamReader sr = new StreamReader(fs);

string line = null;

while ((line = sr.ReadLine()) != null)

{

string[] data = line.Split('\t');

if (data[0] == filename)

{

// 进行断点续传

FileInfo fi = new FileInfo(data[1]);

long ret = fi.Length;

sr.Close();

fs.Close();

return data[1];

}

}

sr.Close();

fs.Close();

return null;

}

public static long checkDownloadState(string filename)

{

// 检查文件的状态，决定是否进行断点续传

FileStream fs = new FileStream("downloadState.dat", FileMode.OpenOrCreate);

StreamReader sr = new StreamReader(fs);

string line = null;

while ((line = sr.ReadLine()) != null)

{

string[] data = line.Split('\t');

if (data[0] == filename)

{

// 进行断点续传

FileInfo fi = new FileInfo(data[1]);

long ret = fi.Length;

sr.Close();

fs.Close();

return ret;

}

}

sr.Close();

fs.Close();

return -1;

}

}

public class download\_class

{

// 多线程方法，用于创建多线程下载文件，使大文件的下载可以终止。

public string fileName { get; set; }

public string filePath { get; set; }

public long offset { get; set; }

public Stream ftpStream { get; set; }

public long total { get; set; }

public FileStream outputStream { get; set; }

public FtpWebResponse response { get; set; }

public void download\_method()

{

try

{

outputStream.Seek(0, SeekOrigin.End);

int bufferSize = 2048;

int readCount;

byte[] buffer = new byte[bufferSize];

readCount = ftpStream.Read(buffer, 0, bufferSize);

while (readCount > 0)

{

outputStream.Write(buffer, 0, readCount);

readCount = ftpStream.Read(buffer, 0, bufferSize);

}

DownloadHelper.editDownloadData(fileName);

ftpStream.Close();

}

catch (ThreadAbortException)

{

// 被Abort终止后，需要记入dat日志中，下次可以进行断点续传

FileStream fs = new FileStream("downloadState.dat", FileMode.OpenOrCreate);

StreamWriter sr = new StreamWriter(fs);

sr.WriteLine($"{fileName}\t{outputStream.Name}");

sr.Close();

fs.Close();

}

finally

{

outputStream.Close();

}

}

}

}

ftpclient/Form1.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Diagnostics;

using System.Resources;

using System.Net;

namespace FTPApplication\_WHU

{

public partial class Form1 : Form

{

/\*\*

\* This form includes user login set IP address of FTP server.

\* Once the control connection is done, it will turn to another form to do ftp operations

\*

\* What I'm thinking about is to create a .dat file to store the IDs and IPs user used to input, but anyway, it's optional.

\* Another suggestion is to initialize the ftpSeverIP for default thus we don't need to create .dat file.

\*

\*

\*

\*\*/

public string ftpServerIP;

public string ftpUserID;

public string ftpPassword;

private MainWindow child;

Point mouseOff;//用于获取鼠标位置

bool leftFlag;//移动标识

public Form1()

{

InitializeComponent();

this.IPBox.Text = "127.0.0.1";

// ResourceManager rm = new ResourceManager("My\_resources", typeof(Resources).Assembly);

this.hideUser(true);

this.StartPosition = FormStartPosition.CenterScreen;

}

private void Form1\_Load(object sender, EventArgs e)

{

}

private void button1\_Click(object sender, EventArgs e)

{

/\*\*

\* Try to connect to the server

\*\*/

// 我在想要不要这里就测试一下是否能连接服务器

ftpServerIP = IPBox.Text.Trim();

ftpUserID = IDBox.Text.Trim();

ftpPassword = passwordBox.Text;

bool success = false;

if (ftpUserID == "test")

{

// 进入测试模式，不需要输入用户名和密码就可以对界面进行调试

success = true;

}

else

{

// 必须要验证成功，才能够进入到主界面

FtpWebRequest reqFTP;

try

{

reqFTP = (FtpWebRequest)FtpWebRequest.Create(new Uri("ftp://" + ftpServerIP + "/"));

reqFTP.UseBinary = true;

reqFTP.Credentials = new NetworkCredential(ftpUserID, ftpPassword);

reqFTP.Method = WebRequestMethods.Ftp.ListDirectoryDetails;

reqFTP.UsePassive = false;

WebResponse response = reqFTP.GetResponse();

response.Close();

success = true;

}

catch

{

MessageBox.Show("无法连接Ftp服务器\n要仅测试界面，请在用户名输入test.");

success = false;

// 如果连接失败，是无法进入到主界面的

}

}

if (success)

{

if (child == null)

{

child = new MainWindow(this);

}

child.Show();

}

}

private void hideUser(bool isHide)

{

if (isHide)

{

this.label2.Hide();

this.label3.Hide();

this.line1.Hide();

this.line2.Hide();

this.IDBox.Hide();

this.IDBox.Clear();

this.passwordBox.Clear();

this.passwordBox.Hide();

this.backBtn.Hide();

this.button1.Location = new Point(42, 360);

this.label5.Show();

}

else

{

this.label2.Show();

this.label3.Show();

this.line1.Show();

this.line2.Show();

this.label5.Hide();

this.IDBox.Show();

this.label5.Hide();

this.backBtn.Show();

this.passwordBox.Show();

this.button1.Location = new Point(42, 541);

}

}

private void exitBtn\_Click(object sender, EventArgs e)

{

this.Close();

}

// hover effect (not as easy as QT)

private void exitBtn\_MouseEnter(object sender, EventArgs e)

{

this.exitBtn.BackgroundImage = (System.Drawing.Image)Properties.Resources.ResourceManager.GetObject("close\_clicked");

}

private void exitBtn\_MouseLeave(object sender, EventArgs e)

{

this.exitBtn.BackgroundImage = (System.Drawing.Image)Properties.Resources.ResourceManager.GetObject("close");

}

// make this form move. from csdn

private void Form1\_MouseDown(object sender, MouseEventArgs e)

{

if (e.Y <= 40)

{

mouseOff = new Point(e.X, e.Y);//获取当前鼠标位置

leftFlag = true;//用于标记窗体是否能移动(此时鼠标按下如果说用户拖动鼠标则窗体移动)

}

}

private void Form1\_MouseMove(object sender, MouseEventArgs e)

{

if (leftFlag)

{

Location = new Point(Control.MousePosition.X - mouseOff.X, Control.MousePosition.Y - mouseOff.Y);

}

}

private void Form1\_MouseUp(object sender, MouseEventArgs e)

{

if (leftFlag)

{

leftFlag = false; //释放鼠标标识为false 表示窗体不可移动

}

}

private void box\_GetFocus(object sender, EventArgs e)

{

//System.Windows.Forms.TextBox tb = (System.Windows.Forms.TextBox)sender;

if (sender as System.Windows.Forms.TextBox == this.IDBox)

{

this.line1.BackColor = Color.FromArgb(0, 111, 255);

}

else if(sender as System.Windows.Forms.TextBox == this.passwordBox)

{

this.line2.BackColor = Color.FromArgb(0, 111, 255);

}

else if (sender as System.Windows.Forms.TextBox == this.IPBox)

{

this.line3.BackColor = Color.FromArgb(0, 111, 255);

}

}

private void box\_LostFocus(object sender, EventArgs e)

{

if (sender as System.Windows.Forms.TextBox == this.IDBox)

{

this.line1.BackColor = System.Drawing.SystemColors.AppWorkspace;

}

else if (sender as System.Windows.Forms.TextBox == this.passwordBox)

{

this.line2.BackColor = System.Drawing.SystemColors.AppWorkspace;

}

else if (sender as System.Windows.Forms.TextBox == this.IPBox)

{

this.line3.BackColor = System.Drawing.SystemColors.AppWorkspace;

}

}

private void label5\_Click(object sender, EventArgs e)

{

this.hideUser(false);

}

private void button2\_Click(object sender, EventArgs e)

{

this.hideUser(true);

}

}

}

### 服务器端代码：

csftpserver/Program.cs

using System;

using System.Drawing;

using System.Collections;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Net;

using System.Net.Sockets;

using System.IO;

using System.Threading;

namespace csftpserver

{

class UserInfo//用户类

{

internal String user;//用户名

internal String password;//密码

internal String workDir;//主目录

}

class FtpState//FTP的各个状态标志

{

internal const int FS\_WAIT\_LOGIN = 0;

internal const int FS\_WAIT\_PASS = 1;

internal const int FS\_LOGIN = 2;

internal const int FTYPE\_ASCII = 0;

internal const int FTYPE\_IMAGE = 1;

internal const int FMODE\_STREAM = 0;

internal const int FMODE\_COMPRESSED = 1;

internal const int FSTRU\_FILE = 0;

internal const int FSTRU\_PAGE = 1;

}

class FtpHandler//处理用户在客户端请求的命令，然后通过服务器处理返回操作的响应码

{

// 初始化参数

TcpClient csocket;

TcpClient dsocket;

int id;

String cmd = "";

String param = "";

String user;

String remoteHost = "";

int remotePort = 0;

String dir = FtpServer.initDir;

String rootdir = "c:/";

String currentFileName = "";

String newFileName = "";

int state = 0;

long offset = 0;

String reply;

StreamWriter out\_Renamed;

int type = 0;

String requestfile = "";

bool isrest = false;

/// <summary>

/// 解析输入命令字符串：解析输入的字符串得到命令名，返回相应的响应码

/// </summary>

/// <param name="s">输入的字符串</param>

/// <returns>响应码</returns>

int parseInput(String s)

{

Console.WriteLine(s);

int p = 0;

int i = -1;

p = s.IndexOf(" ");

if (p==-1)

{

cmd = s;

}

else

{

cmd = s.Substring(0, (p) - (0));

}

if (p >= s.Length || p == -1)

{

param = "";

}

else

{

param = s.Substring(p + 1, (s.Length) - (p + 1));

}

cmd = cmd.ToUpper();

if (cmd.Equals("USER"))

i = 1;

if (cmd.Equals("PASS"))

i = 2;

if (cmd.Equals("ACCT"))

i = 3;

if (cmd.Equals("CDUP"))

i = 4;

if (cmd.Equals("SMNT"))

i = 5;

if (cmd.Equals("CWD"))

i = 6;

if (cmd.Equals("QUIT"))

i = 7;

if (cmd.Equals("REIN"))

i = 8;

if (cmd.Equals("PORT"))

i = 9;

if (cmd.Equals("PASV"))

i = 10;

if (cmd.Equals("TYPE"))

i = 11;

if (cmd.Equals("STRU"))

i = 12;

if (cmd.Equals("MODE"))

i = 13;

if (cmd.Equals("RETR"))

i = 14;

if (cmd.Equals("STOR"))

i = 15;

if (cmd.Equals("STOU"))

i = 16;

if (cmd.Equals("APPE"))

i = 17;

if (cmd.Equals("ALLO"))

i = 18;

if (cmd.Equals("REST"))

i = 19;

if (cmd.Equals("RNFR"))

i = 20;

if (cmd.Equals("RNTO"))

i = 21;

if (cmd.Equals("ABOR"))

i = 22;

if (cmd.Equals("DELE"))

i = 23;

if (cmd.Equals("RMD"))

i = 24;

if (cmd.Equals("XMKD"))

i = 25;

if (cmd.Equals("MKD"))

i = 25;

if (cmd.Equals("PWD"))

i = 26;

if (cmd.Equals("LIST"))

i = 27;

if (cmd.Equals("NLST"))

i = 28;

if (cmd.Equals("SITE"))

i = 29;

if (cmd.Equals("SYST"))

i = 30;

if (cmd.Equals("HELP"))

i = 31;

if (cmd.Equals("NOOP"))

i = 32;

if (cmd.Equals("XPWD"))

i = 33;

if (cmd.Equals("OPTS"))

i = 34;

if (cmd.Equals("SIZE"))

i = 35;

return i;

} // parseInput() end

/// <summary>

/// 传入路径，验证是否合法

/// </summary>

/// <param name="s"></param>

/// <returns></returns>

internal int validatePath(String s)

{

FileInfo f = new FileInfo(s);

bool tmpBool;

if (File.Exists(f.FullName))

tmpBool = true;

else

tmpBool = Directory.Exists(f.FullName);

if (tmpBool && !Directory.Exists(f.FullName))

{

String s1 = s.ToLower();

String s2 = rootdir.ToLower();

if (s1.StartsWith(s2))

return 1;

else

return 0;

}

f = new FileInfo(addTail(dir) + s);

bool tmpBool2;

if (File.Exists(f.FullName))

tmpBool2 = true;

else

tmpBool2 = Directory.Exists(f.FullName);

if (tmpBool2 && !Directory.Exists(f.FullName))

{

String s1 = (addTail(dir) + s).ToLower();

String s2 = rootdir.ToLower();

if (s1.StartsWith(s2))

return 2;

else

return 0;

}

return 0;

}

/// <summary>

/// 校验密码

/// </summary>

/// <param name="s"></param>

/// <returns></returns>

internal bool checkPass(String s)

{

for (int i = 0; i < FtpServer.usersInfo.Count; i++)

{

if(((UserInfo)FtpServer.usersInfo[i]).user.Equals(user) &&

((UserInfo)FtpServer.usersInfo[i]).password.Equals(s))

{

rootdir = ((UserInfo)FtpServer.usersInfo[i]).workDir;

dir = ((UserInfo)FtpServer.usersInfo[i]).workDir;

return true;

}

}

return false;

} // checkPass() end

internal string getUSER()

{

return user;

}

internal bool commandUSER()//USER命令的业务逻辑函数

{// 当用户输入USER命令后显示相应消息，置状态位

if(cmd.Equals("USER"))

{

reply = "331 User name okay, need password";

user = param;

state = FtpState.FS\_WAIT\_PASS;

return false;

}

else

{

reply = "501 Syntax error in parameters or arguments";

return true;

}

} // commandUSER() end

internal bool commandPASS()//登录

{// 当用户输入PASS命令后，判断密码是否合法并执行相应的登录动作

if (cmd.Equals("PASS"))

{

if (checkPass(param))

{

reply = "230 User logged in, proceed";

state = FtpState.FS\_LOGIN;

Console.Out.WriteLine("Message: user "+ user + " Form " + remoteHost +

"Login");

Console.Out.Write("->");

return false;

}

else

{

reply = "530 Not logged in";

return true;

}

}

else

{

reply = "501 Syntax error in parameters or arguments";

return true;

}

} // commandPASS() end

internal void errCMD()//显示出错信息

{

reply = "500 Syntax error, command unrecognized";

}

internal bool commandCDUP()//寻找当前目录

{

dir = FtpServer.initDir;

FileInfo f = new FileInfo(dir);

if (f.DirectoryName != null && (!dir.Equals(rootdir)))

{

dir = f.DirectoryName;

reply = "200 Command okay";

}

else

{

reply = "550 Current directory has no parent";

}

return false;

} // commandCDUP() end

internal bool commandCWD()//判别是否存在所对应的工作目录

{

FileInfo f = new FileInfo(param);

String s = "";

String s1 = "";

if(dir.EndsWith("/"))

s = dir;

else

s = dir + "/";

FileInfo f1 = new FileInfo(s + param);

bool tmpBool;

if(File.Exists(f.FullName))

tmpBool = true;

else

tmpBool = Directory.Exists(f.FullName);

if (Directory.Exists(f.FullName) && tmpBool)

{

if (param.Equals("..") || param.Equals("..\\"))

{

if (String.Compare(dir, rootdir, true) == 0)

{

reply = "550 The directory dose not exists";

}

else

{

s1 = new FileInfo(dir).DirectoryName;

if (s1 != null)

{

dir = s1;

reply = "250 Requested file action okay, directory change to " + dir;

}

else{

reply = "550 The directory does not exists";

}

}

}

else if(param.Equals(".") || param.Equals(".\\"))

{

}

else

{

dir = param;

reply = "250 Requested file action okay, directory change to " + dir;

}

}

else

{

bool tmpBool2;

if(File.Exists(f1.FullName))

tmpBool2 = true;

else

tmpBool2 = Directory.Exists(f1.FullName);

if (Directory.Exists(f1.FullName) && tmpBool2)

{

dir = s + param;

reply = "250 Requested file action okay, directory change to " + dir;

}

else

reply = "501 Syntax error in parameters or arguments";

}

return false;

} // commandCWD() end

internal bool commandQUIT() // 退出

{

reply = "221 Service closing control connection";

return true;

} // commandQUIT() end

internal bool commandPORT() // 解析字符串，查看是否有格式问题

{

int p1 = 0;

int p2 = 0;

int[] a = new int[6];

int i = 0;

try

{

while((p2 = param.IndexOf(",", p1)) != -1)

{

a[i] = Int32.Parse(param.Substring(p1, (p2)-(p1)));

p2 = p2 + 1;

p1 = p2;

i++;

}

a[i] = Int32.Parse(param.Substring(p1, (param.Length) - (p1)));

}

catch (FormatException e)

{

}

finally{

remoteHost = a[0] + "." + a[1] + "." + a[2] + "." + a[3];

remotePort = a[4] \* 256 + a[5];

reply = "200 Command okay";

}

return false;

} // commandPORT() end

internal bool commandLIST() // 命令LIST，启动二进制传输模式

{

try

{

Console.WriteLine(remoteHost);

dsocket = new TcpClient(remoteHost, remotePort);

StreamWriter temp\_writer;

temp\_writer = new StreamWriter(dsocket.GetStream(), Encoding.Default);

StreamWriter dout = temp\_writer;

if (param.Equals("") || param.Equals("LIST"))

{

out\_Renamed.WriteLine("150 Opening ASCII mode data connection for/bin/ls");

FileInfo f = new FileInfo(dir);

DirectoryInfo di = new DirectoryInfo(f.FullName);

foreach (FileInfo fi in di.GetFiles())

{

//format:

//filename lastwritetime length extension

dout.WriteLine(fi.Name + "\t" + fi.LastWriteTime.ToString() +

"\t" + fi.Length.ToString() + "\t"+ fi.Extension.ToString()+"\t");

}

}

dout.Close();

dsocket.Close();

reply = "226 Transfer complete !";

}

catch (Exception e)

{

Console.WriteLine(e.ToString());

reply = "451 Requested action aborted: local error in processing";

return false;

}

finally { }

return false;

} // commandLIST() end

internal bool commandNLST() // 命令NLST，启动二进制传输模式

{

try

{

Console.WriteLine(remoteHost);

dsocket = new TcpClient(remoteHost, remotePort);

StreamWriter temp\_writer;

temp\_writer = new StreamWriter(dsocket.GetStream(), Encoding.Default);

StreamWriter dout = temp\_writer;

if (param.Equals("") || param.Equals("LIST"))

{

out\_Renamed.WriteLine("150 Opening ASCII mode data connection for/bin/ls");

FileInfo f = new FileInfo(dir);

// String[] dirStructure = Directory.GetFiles(f.FullName);

// String fileType;

DirectoryInfo di = new DirectoryInfo(f.FullName);

foreach (FileInfo fi in di.GetFiles())

{

dout.WriteLine(fi.Name);

}

}

dout.Close();

dsocket.Close();

reply = "226 Transfer complete !";

}

catch (Exception e)

{

Console.WriteLine(e.ToString());

reply = "451 Requested action aborted: local error in processing";

return false;

}

finally { }

return false;

} // commandNLST() end

internal bool commandTYPE() //命令TYPE：获取命令类型

{

if (param.Equals("a"))

{

type = FtpState.FTYPE\_ASCII;

reply = "200 Command okay Change to ASCII mode";

}

else if (param.Equals("I"))

{

type = FtpState.FTYPE\_IMAGE;

reply = "200 Command okay Change to BINARY mode";

}

else{

reply = "504 Command not implemented for that parameter";

}

return false;

} // commandTYPE() end

internal bool commandRETR() //启动二进制传输模式，初始化工作环境

{

requestfile = param;

FileInfo f = new FileInfo(requestfile);

bool tmpBool;

if (File.Exists(f.FullName))

tmpBool = true;

else

tmpBool = Directory.Exists(f.FullName);

if (!tmpBool)

{

f = new FileInfo(addTail(dir) + param);

bool tmpBool2;

if(File.Exists(f.FullName))

tmpBool2 = true;

else

tmpBool2 = Directory.Exists(f.FullName);

if (!tmpBool2)

{

reply = "550 File not found";

return false;

}

requestfile = addTail(dir) + param;

}

if (isrest)

{

}

else

{

if(type == FtpState.FTYPE\_IMAGE)

{

try

{

out\_Renamed.WriteLine("150 Opening Binary mode data connection for " +

requestfile);

dsocket = new TcpClient(remoteHost, remotePort);

BufferedStream fin = new BufferedStream(new FileStream(requestfile, FileMode.Open, FileAccess.Read));

NetworkStream dout =dsocket.GetStream();

if (offset != 0)

{

fin.Seek(offset, SeekOrigin.Begin);

Console.WriteLine("offset");

}

byte[] buf = new byte[1024];

int l = 0;

while ( (l = fin.Read(buf, 0, 1024)) != 0)

{

dout.Write(buf, 0, l);

}

fin.Close();

dout.Close();

dsocket.Close();

offset = 0;

reply = "226 Transfer complete !";

}

catch (Exception e)

{

Console.WriteLine(e.ToString());

reply = "451 Requested action aborted: local error in processing";

return false;

}

finally{}

}

if (type == FtpState.FTYPE\_ASCII)

{

try

{

out\_Renamed.WriteLine("150 Opening ASCII mode data connection for " +

requestfile);

dsocket = new TcpClient(remoteHost, remotePort);

StreamReader fin = new StreamReader(new StreamReader(requestfile,Encoding.Default).BaseStream,

new StreamReader(requestfile,Encoding.Default).CurrentEncoding);

StreamWriter temp\_writer;

temp\_writer = new StreamWriter(dsocket.GetStream(),Encoding.Default);

temp\_writer.AutoFlush = true;

StreamWriter dout = temp\_writer;

String s;

while ((s = fin.ReadLine()) != null)

{

dout.WriteLine(s);

}

fin.Close();

dout.Close();

dsocket.Close();

reply = "226 Transfer complete !";

}

catch (Exception e)

{

Console.WriteLine(e.ToString());

reply = "451 Requested action aborted: local error in processing";

return false;

}

finally{}

}

}

return false;

} // commandRETR() end

internal bool commandSTOR()//获取当前目录

{

if (param.Equals(""))

{

reply = "501 Syntax error in parameters or arguments";

return false;

}

requestfile = addTail(dir) + param;

if(type == FtpState.FTYPE\_IMAGE)

{

try

{

out\_Renamed.WriteLine("150 Opening Binary mode data connection for " +

requestfile);

dsocket = new TcpClient(remoteHost, remotePort);

BufferedStream fout = new BufferedStream(new FileStream(requestfile, FileMode.Create));

BufferedStream din = new BufferedStream(dsocket.GetStream());

byte[] buf = new byte[1024];

int l = 0;

while ((l = din.Read(buf, 0, 1024)) != 0)

{

fout.Write(buf, 0, l);

}

din.Close();

fout.Close();

dsocket.Close();

reply = "226 Transfer complete !";

}

catch (Exception e)

{

Console.WriteLine(e.ToString());

reply = "451 Requested action aborted: local error in processing";

return false;

}

finally{}

}

if (type == FtpState.FTYPE\_ASCII)

{

try

{

out\_Renamed.WriteLine("150 Opening ASCII mode data connection for " +

requestfile);

dsocket = new TcpClient(remoteHost, remotePort);

StreamWriter fout = new StreamWriter(new FileStream(requestfile,

FileMode.Create), Encoding.Default);

StreamReader din = new StreamReader(new StreamReader(dsocket.GetStream(),

Encoding.Default).BaseStream,

new StreamReader(dsocket.GetStream(),Encoding.Default).CurrentEncoding);

String line;

while ((line = din.ReadLine()) != null)

{

fout.WriteLine(line);

}

din.Close();

fout.Close();

dsocket.Close();

reply = "226 Transfer complete !";

}

catch (Exception e)

{

Console.WriteLine(e.ToString());

reply = "451 Requested action aborted: local error in processing";

return false;

}

finally{}

}

return false;

} //commandSTOR() end

internal bool commandPWD() // 显示当前目录

{

reply = "257 " + dir + " is current directory.";

return false;

} //commandPWD() end

internal bool commandNOOP() // 确认

{

reply = "200 OK.";

return false;

} //commandNOOP end

internal bool commandABOR() // 关闭套接字

{

try

{

dsocket.Close();

}

catch (Exception e)

{

Console.WriteLine(e.ToString());

reply = "451 Requested action aborted: local error in processing";

return false;

}

finally{}

reply = "421 Service not available, closing control connection";

return false;

} //commandABOR() end

internal bool commandDELE() // 删除文件

{

int i = validatePath(param);

if (i == 0)

{

reply = "550 Request action not taken";

return false;

}

if (i == 1)

{

FileInfo f = new FileInfo(param);

bool tmpBool;

if (File.Exists(f.FullName))

{

File.Delete(f.FullName);

tmpBool = true;

}

else if (Directory.Exists(f.FullName))

{

Directory.Delete(f.FullName);

tmpBool = true;

}

else

{

tmpBool = false;

}

bool generatedAux = tmpBool;

}

if (i == 2)

{

FileInfo f = new FileInfo(addTail(dir) + param);

bool tmpBool2;

if (File.Exists(f.FullName))

{

File.Delete(f.FullName);

tmpBool2 = true;

}

else if(Directory.Exists(f.FullName))

{

Directory.Delete(f.FullName);

tmpBool2 = true;

}

else

{

tmpBool2 = false;

}

bool generatedAux2 = tmpBool2;

}

reply = "250 Request file action ok, complete.";

return false;

} //commandDELE() end

internal bool commandRNFR()

{

int i = validatePath(param);

if (i == 0)

{

reply = "550 Request action not taken";

return false;

}

if (i == 1)

{

FileInfo f = new FileInfo(param);

bool tmpBool;

if (File.Exists(f.FullName))

{

currentFileName = f.FullName;

tmpBool = true;

}

//else if (Directory.Exists(f.FullName))

//{

// Directory.Delete(f.FullName);

// tmpBool = true;

//}

else

{

tmpBool = false;

}

bool generatedAux = tmpBool;

}

if (i == 2)

{

FileInfo f = new FileInfo(addTail(dir) + param);

bool tmpBool2;

if (File.Exists(f.FullName))

{

currentFileName = f.FullName;

tmpBool2 = true;

}

//else if (Directory.Exists(f.FullName))

//{

// Directory.Delete(f.FullName);

// tmpBool2 = true;

//}

else

{

tmpBool2 = false;

}

bool generatedAux2 = tmpBool2;

}

reply = "250 Request file action ok, complete.";

return false;

} //commandRNFR() end

internal bool commandRNTO()

{

if (param.Equals(""))

{

reply = "553 Request action not taken: filename is invalid";

return false;

}

newFileName = addTail(dir) + param;

try

{

File.Move(currentFileName, newFileName);

reply = "250 Request file action ok, complete.";

}

catch (Exception e)

{

Console.WriteLine(e.ToString());

reply = "553 Request action not taken: filename is invalid";

return false;

}

finally { };

return false;

} //commandRNTO() end

internal bool commandMKD() // 创建目录

{

String s1 = param.ToLower();

String s2 = rootdir.ToLower();

if (s1.StartsWith(s2))

{

FileInfo f = new FileInfo(param);

bool tmpBool;

if (File.Exists(f.FullName))

tmpBool = true;

else

tmpBool = Directory.Exists(f.FullName);

if (tmpBool)

{

reply = "550 Request action not taken";

return false;

}

else

{

Directory.CreateDirectory(f.FullName);

reply = "250 Request file action ok, complete.";

}

}

else

{

FileInfo f = new FileInfo(addTail(dir) + param);

bool tmpBool2;

if (File.Exists(f.FullName))

tmpBool2 = true;

else

tmpBool2 = Directory.Exists(f.FullName);

if (tmpBool2)

{

reply = "550 Request action not taken";

return false;

}

else

{

Directory.CreateDirectory(f.FullName);

reply = "250 Request file action ok, complete.";

}

}

return false;

} //commandMKD() end

internal bool commandREST()

{

offset = long.Parse(param);

return false;

}

internal bool commandSIZE() // 获取文件长度

{

requestfile = param;

try

{

Console.WriteLine(remoteHost);

dsocket = new TcpClient(remoteHost, remotePort);

StreamWriter temp\_writer;

temp\_writer = new StreamWriter(dsocket.GetStream(), Encoding.Default);

StreamWriter dout = temp\_writer;

out\_Renamed.WriteLine("150 Opening ASCII mode data connection for " +

requestfile);

FileInfo f = new FileInfo(requestfile);

dout.WriteLine(f.Length.ToString());

dout.Close();

dsocket.Close();

reply = "226 Transfer complete !";

}

catch (Exception e)

{

Console.WriteLine(e.ToString());

reply = "451 Requested action aborted: local error in processing";

return false;

}

finally { }

return false;

}

internal String addTail(String s) // 增加尾缀"/"

{

if (!s.EndsWith("/"))

s = s + "/";

return s;

}

public FtpHandler(TcpClient s, int i) // 构造函数

{

csocket = s;

id = i;

}

public void HandleThread() // 命令翻译

{

String str = "";

int parseResult;

try

{

StreamReader in\_Renamed = new StreamReader(csocket.GetStream(),Encoding.Default);

StreamWriter temp\_writer;

temp\_writer = new StreamWriter(csocket.GetStream(),Encoding.Default);

temp\_writer.AutoFlush = true;

out\_Renamed = temp\_writer;

state = FtpState.FS\_WAIT\_LOGIN;

bool finished = false;

while (!finished)

{

str = in\_Renamed.ReadLine();

if(str == null)

finished = true;

else

{

parseResult = parseInput(str);

Console.Out.WriteLine("Command:" + cmd + " Parameter:" + param);

Console.Out.Write("->");

switch(state)

{

case FtpState.FS\_WAIT\_LOGIN:

finished = commandUSER();

break;

case FtpState.FS\_WAIT\_PASS:

finished = commandPASS();

break;

case FtpState.FS\_LOGIN:

{

switch(parseResult)

{

case -1:

errCMD();

break;

case 4:

finished = commandCDUP();

break;

case 6:

finished = commandCWD();

break;

case 7:

finished = commandQUIT();

break;

case 9:

finished = commandPORT();

break;

case 27:

finished = commandLIST();

break;

case 28:

finished = commandNLST();

break;

case 11:

finished = commandTYPE();

break;

case 14:

finished = commandRETR();

break;

case 15:

finished = commandSTOR();

break;

case 19:

finished = commandREST();

break;

case 20:

finished = commandRNFR();

break;

case 21:

finished = commandRNTO();

break;

case 26:

case 33:

finished = commandPWD();

break;

case 32:

finished = commandNOOP();

break;

case 22:

finished = commandABOR();

break;

case 23:

finished = commandDELE();

break;

case 25:

finished = commandMKD();

break;

case 35:

finished = commandSIZE();

break;

} // switch(parseResult) end

} // case FtpState.FS\_LOGIN: end

break;

} // switch(state) end

} // else

out\_Renamed.WriteLine(reply);

} // while

csocket.Close();

}

//try

catch (Exception e)

{

Console.WriteLine(e.ToString());

}

finally{}

}

}

class FtpConsole//承担在控制台上的所有业务逻辑

{

internal StreamReader cin;

internal String conCmd;

internal String conParam;

internal int consoleQUIT()//退出

{

Environment.Exit(0);

return 0;

}

internal bool consoleLISTUSER()//列出用户列表

{

Console.Out.WriteLine("username \t\t workdirectory");

for(int i=0;i<FtpServer.usersInfo.Count;i++)

{

Console.Out.Write(((UserInfo)FtpServer.usersInfo[i]).user + "\t\t\t" + ((UserInfo)FtpServer.usersInfo[i]).workDir);

}

Console.Out.WriteLine("");

return false;

}

internal bool consoleLIST()//判断是否是系统用户

{

int i = 0;

for(i=0;i<FtpServer.users.Count;i++)

{

Console.Out.WriteLine(((FtpHandler)FtpServer.users[i]).getUSER());

}

return false;

}

internal bool validateUserName(String s)//根据传入的用户名称验证是否与文件中的名称匹配

{

for(int i=0;i<FtpServer.usersInfo.Count;i++)

{

if (((UserInfo)FtpServer.usersInfo[i]).user.Equals(s))

return false;

}

return true;

}

internal bool consoleADDUSER()//增加用户

{

Console.Out.Write("please enter username:");

try

{

cin = new StreamReader(new StreamReader(Console.OpenStandardInput(), Encoding.Default).BaseStream, new StreamReader(Console.OpenStandardInput(), Encoding.Default).CurrentEncoding);

UserInfo tempUserInfo = new UserInfo();

String line = cin.ReadLine();

if((object)line!=(object)"")

{

//用户已经存在

if(!validateUserName(line))

{

Console.Out.WriteLine("user" + line + "already exsits!");

return false;

}

}

else

{

//出错用户名不能为空

Console.Out.WriteLine("username cannnot be null!");

return false;

}

tempUserInfo.user = line;

Console.Out.Write("enter password:");

line = cin.ReadLine();

if ((Object)line != (Object)"")

tempUserInfo.password = line;

else

{

Console.Out.WriteLine("password cannot be null:");

return false;

}

Console.Out.Write("enter the initial directory:");

line = cin.ReadLine();

if((object)line!=(Object)"")

{

FileInfo f = new FileInfo(line);

bool tmpBool;

if (File.Exists(f.FullName))

tmpBool = true;

else

tmpBool = Directory.Exists(f.FullName);

if(!tmpBool)

{

Directory.CreateDirectory(f.FullName);

}

tempUserInfo.workDir = line;

}

else

{

Console.Out.WriteLine("the directory cannot be null!");

return false;

}

FtpServer.usersInfo.Add(tempUserInfo);

saveUserInfo();

}

catch(Exception e)

{

Console.WriteLine(e.ToString());

return false;

}

finally

{

}

return false;//上述操作都没有出现错误

}

internal void saveUserInfo()//保存用户信息函数

{

String s = "";

try

{

StreamWriter fout = new StreamWriter(new FileStream("user.cfg", FileMode.OpenOrCreate), Encoding.Default);

for(int i=0;i<FtpServer.usersInfo.Count;i++)

{

s = ((UserInfo)FtpServer.usersInfo[i]).user + "|" + ((UserInfo)FtpServer.usersInfo[i]).password + "|" + ((UserInfo)FtpServer.usersInfo[i]).workDir + "|";

fout.Write(s);

fout.WriteLine();

}

fout.Close();

}

catch(Exception e)

{

Console.WriteLine(e.ToString());

}

finally

{

}

}

internal bool consoleDELUSER()//删除用户

{

String s = "";

if(conParam.Equals(""))

{

Console.Out.WriteLine("usage:deluser username");

return false;

}

for(int i=0;i<FtpServer.usersInfo.Count;i++)

{

s = ((UserInfo)FtpServer.usersInfo[i]).user;

if(s.Equals(conParam))

{

Console.Out.WriteLine("User" + conParam + "deleted");

FtpServer.usersInfo.RemoveAt(i);

saveUserInfo();

return false;

}

}

Console.Out.WriteLine("User" + conParam + "not exists");

return false;

}

internal bool consoleHELP()//显示帮助

{

if (conParam.Equals(""))

{

Console.Out.WriteLine("adduser:add new user");

Console.Out.WriteLine("deluser<username>:delete a user");

Console.Out.WriteLine("quit :quit");

Console.Out.WriteLine("list :list all user connect to server");

Console.Out.WriteLine("listener:list all account of this server");

Console.Out.WriteLine("help:show this help");

}

else if (conParam.Equals("adduser"))

Console.Out.WriteLine("adduser:add new user");

else if (conParam.Equals("deluser"))

Console.Out.WriteLine("deluser<username>:delete a user");

else if (conParam.Equals("qiut"))

Console.Out.WriteLine("quit :quit");

else if (conParam.Equals("list"))

Console.Out.WriteLine("list :list all user connect to server");

else if (conParam.Equals("listener"))

Console.Out.WriteLine("listener:list all account of this server");

else if (conParam.Equals("help"))

Console.Out.WriteLine("help:show this help");

else

return false;

return false;

}

internal bool consoleERR()//出错函数

{

Console.Out.WriteLine("bad command!");

return false;

}

public FtpConsole()//构造函数

{

Console.Out.WriteLine("ftp server started!");

cin = new StreamReader(new StreamReader(Console.OpenStandardInput(), Encoding.Default).BaseStream, new StreamReader(Console.OpenStandardInput(), Encoding.Default).CurrentEncoding);

}

public void ConsoleThread()//控制台线程函数

{

bool ok = false;

String input = "";

while(!ok)

{

Console.Out.Write("->");

try

{

//读入一行内容

input = cin.ReadLine();

}

catch(Exception e)

{

//读入失败，打印错误信息

Console.WriteLine(e.ToString());

}

finally

{

}

//判断输入命令内容

switch(parseInput(input))

{

case 1:

consoleQUIT();

break;

case 8:

ok = consoleLISTUSER();

break;

case 0:

ok = consoleLIST();

break;

case 2:

ok = consoleADDUSER();

break;

case 3:

ok = consoleDELUSER();

break;

case 7:

ok = consoleHELP();

break;

case -1:

ok = consoleERR();

break;

}

}

}

internal int parseInput(String s)//解析输入的函数

{

//变量初始化

//大写命令名

String upperCmd;

int p = 0;

conCmd = "";

conParam = "";

p = s.IndexOf(" ");

if (p == -1)

conCmd = s;

else

conCmd = s.Substring(0, (p) - (0));

if (p >= s.Length || p == -1)

conParam = "";

else

conParam = s.Substring(p + 1, (s.Length) - (p + 1));

upperCmd = conCmd.ToUpper();

if (upperCmd.Equals("LIST"))

return 0;

else if (upperCmd.Equals("QUIT") || upperCmd.Equals("EXIT"))

return 1;

else if (upperCmd.Equals("ADDUSER"))

return 2;

else if (upperCmd.Equals("DELUSER"))

return 3;

else if (upperCmd.Equals("EDITUSER"))

return 4;

else if (upperCmd.Equals("ADDDIR"))

return 5;

else if (upperCmd.Equals("REMOVEDIR"))

return 6;

else if (upperCmd.Equals("HELP") || upperCmd.Equals("?"))

return 7;

else if (upperCmd.Equals("LISTENER"))

return 8;

return -1;

}

}

public class FtpServer

{

public static String initDir;

public static ArrayList users = new ArrayList();

public static ArrayList usersInfo = new ArrayList();

public FtpServer()

{

FtpConsole fc = new FtpConsole();

Thread t = new Thread(new ThreadStart(fc.ConsoleThread));

t.Start();

loadUsersInfo();

int counter = 1;

int i = 0;

try

{

TcpListener tcpListener;

tcpListener = new TcpListener(IPAddress.Any, 21);

tcpListener.Start();

TcpListener s = tcpListener;

//Console.WriteLine("I'm listening.");

while(true)

{

TcpClient incoming = s.AcceptTcpClient();

StreamReader in\_Renamed = new StreamReader(incoming.GetStream(), Encoding.Default);

StreamWriter temp\_writer = new StreamWriter(incoming.GetStream(), Encoding.Default);

temp\_writer.AutoFlush = true;

StreamWriter out\_Renamed = temp\_writer;

out\_Renamed.WriteLine("220 Service ready for new user" + counter);

//创建服务线程

FtpHandler h = new FtpHandler(incoming, i);

Thread t1 = new Thread(new ThreadStart(h.HandleThread));

t1.Start();

users.Add(h);

counter++;

i++;

}

}

catch (Exception e)

{

Console.WriteLine(e.ToString());

}

finally

{

}

}

public void loadUsersInfo()//加载用户信息

{

String s = new Uri(Path.GetFullPath("user.cfg")).ToString();//该文件保存用户名密码等信息

s = s.Substring(6, s.Length - 6);

int p1 = 0;

int p2 = 0;

bool tmpBool;

if (File.Exists(s))

{

tmpBool = true;

}

else

{

tmpBool = Directory.Exists(s);

}

if (tmpBool)

{

try

{

StreamReader fin = new StreamReader(new StreamReader(new FileStream(s, FileMode.Open, FileAccess.Read), Encoding.Default).BaseStream, new StreamReader(new FileStream(s, FileMode.Open, FileAccess.Read), Encoding.Default).CurrentEncoding);

String line;

String field;

int i = 0;

while ((line = fin.ReadLine()) != null)

{

UserInfo tempUserInfo = new UserInfo();

p1 = 0;

p2 = 0;

i = 0;

if (line.StartsWith("#"))

continue;

while ((p2 = line.IndexOf("|", p1)) != -1)

{

field = line.Substring(p1, (p2) - (p1));

p2 = p2 + 1;

p1 = p2;

switch (i)

{

case 0:

tempUserInfo.user = field;

break;

case 1:

tempUserInfo.password = field;

break;

case 2:

tempUserInfo.workDir = field;

break;

}

i++;

}

usersInfo.Add(tempUserInfo);

}

fin.Close();

}

catch (Exception e)

{

Console.WriteLine(e.ToString());

}

finally

{

}

}

}

public static void Main(string[] args)

{

initDir = "c:/";

FtpServer ftpServer = new FtpServer();

Console.WriteLine("#");

}

}

}

# 参考文献

[1] Visual C#网络编程技术与实践

|  |
| --- |
| 【结论】：完成了FTP客户端和服务器的设计及编写，对于socket编程有了更加深刻的理解，巩固了FTP协议的工作原理、知晓了各种FTP指令及其参数的含义，同时对C#语言的FTP、TCP网络编程有了进一步的了解。克服了实验过程中出现的许多困难，解决了许多问题，最终圆满的完成了实验课程的要求。 |
| 【小结】：由于本次实验正逢暑假，小组成员之间只能通过网络交流，虽然这样对理解双方的意图不利，但我们依然克服了空间上的阻隔，圆满完成了任务。在实验的过程中，我们遇到了很多困难，比如无法使用PASV指令进行信息传输，我们只好改用PORT；界面设计时出现了未选中目标时进行下载、重命名等操作引起的Bug；在实验的最后阶段，还为程序添加了断点续传功能，用到了教材中未提到的REST指令等。最终圆满完成了实验任务。 |
| 指导老师评语及成绩 |
| 【评语】：  成 绩： 指导老师签名：  批阅日期：2021.8 |