**武汉大学计算机学院**

**本科生实验报告**

**FTP客户端的设计与实现**

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

指 导 教 师 ：周浩

团 队 成 员 一：龙晓怡（2018302100026）

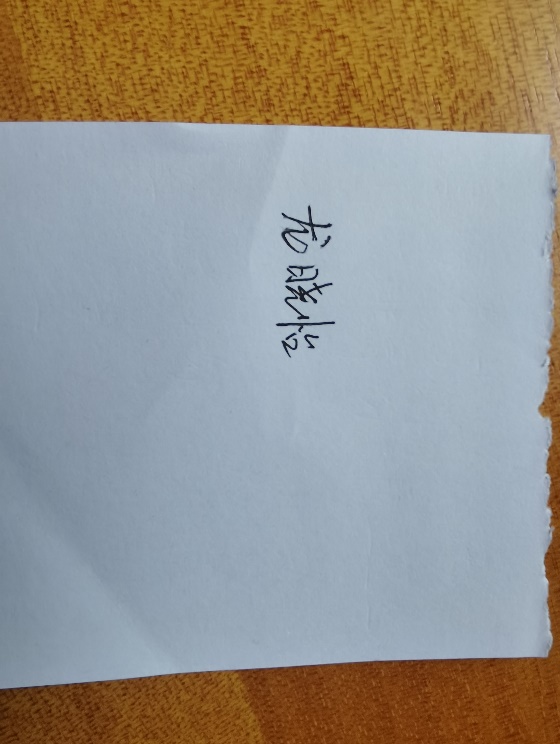
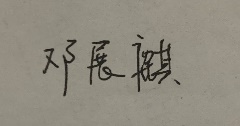
团 队 成 员 二：邓展麒（2018302060078）

团 队 成 员 三：杨景博（2018302060060）

二○二〇年七月

**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： 2020.7.30

摘 要

FTP客户端实验的实验目的是使学生掌握FTP（文件传输协议）的相关原理和实践，Soket(套接字)原理及编程，网络系统软件的开发方法等。

实验设计主要遵循socket编程，FTP原理，Winform编程规范。

实验内容主要包括：Winform设计FTP客户端图形化界面设计，C#语言实现客户端登陆连接，刷新服务器端文件列表，上传下载，断点续传功能。

实验结论为：经测试，所开发的FTP客户端能够完成应实现的操作。

**关键词：**FTP client；Socket；C#

**目录**

[1 实验目的与意义 5](#_Toc47032421)

[1.1 实验目的 5](#_Toc47032422)

[1.2 实验意义 5](#_Toc47032423)

[2 实验设计 5](#_Toc47032424)

[2.1 实验概述 5](#_Toc47032425)

[2.2 实验分工 6](#_Toc47032426)

[2.3 实验原理 6](#_Toc47032427)

[2.3.1 Socket原理 6](#_Toc47032428)

[2.3.2 System.Net.Sockets 6](#_Toc47032429)

[2.3.3 FTP 原理 6](#_Toc47032430)

[3 实验方案（详细设计） 9](#_Toc47032431)

[3.1 邓展麒部分 9](#_Toc47032432)

[3.1.1 连接 9](#_Toc47032433)

[3.1.2 登陆 9](#_Toc47032434)

[3.1.3 进入被动模式 10](#_Toc47032435)

[3.1.4 获取服务器端文件列表 11](#_Toc47032436)

[3.2 杨景博部分 12](#_Toc47032437)

[3.2.1 上传 12](#_Toc47032438)

[3.2.2 下载 13](#_Toc47032439)

[3.2.3 断点续传 14](#_Toc47032440)

[3.3 龙晓怡部分 15](#_Toc47032441)

[3.3.1 本地文件树形展示 15](#_Toc47032442)

[3.3.2 服务器端文件展示 17](#_Toc47032443)

[3.3.3 服务器端文件夹的访问 18](#_Toc47032444)

[3.3.4 断点续传交互 19](#_Toc47032445)

[4 程序主要界面 20](#_Toc47032446)

[4.1 登陆连接 20](#_Toc47032447)

[4.2 退出 20](#_Toc47032448)

[4.3 刷新操作 21](#_Toc47032449)

[4.4 上传 21](#_Toc47032450)

[4.5 下载 22](#_Toc47032451)

[4.6 断点 23](#_Toc47032452)

[4.7 续传 24](#_Toc47032453)

[附录-程序源代码 25](#_Toc47032454)

[IftpOperation.cs 25](#_Toc47032455)

[FtpFileInto.cs 26](#_Toc47032456)

[FtpOperation.cs 26](#_Toc47032457)

[Mainform.cs 37](#_Toc47032458)

1 实验目的与意义

1.1 实验目的

本实验的目的是：使学生深刻理解并实践计算机网络课程中的FTP协议，Socket编程；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，能将所学的知识集成到一起，开发实际软件系统。本实验主要采用了C#中System.Net.Sockets库和Winform设计界面，实现FTP客户端的文件传输。

1.2 实验意义

本实验的实验意义是，通过实际编程，掌握计算机理论课程的相关通信协议和Socket套接字的编程的应用，同时也锻炼了我们对程序的调试能力和学习新知识的能力。

2 实验设计

2.1 实验概述

本实验开发了一个FTP客户端应用程序，在Windows IIS（Internet信息服务管理器上搭建FTP服务器），实现了客户端登陆、连接服务器，获取服务器文件列表，刷新服务器文件列表，文件的上传、下载、断点续传功能。

本实验采用C#语言开发，使用.Net框架提供的Socket接口实现FTP协议内容。客户端图形界面用Winform设计开发。

2.2 实验分工

邓展麒：连接登陆；获取服务器文件列表功能实现；实验报告3.1部分

杨景博：上传；下载；断点续传功能的实现，实验报告3.2部分

龙晓怡：Winform界面设计；前后端交互；代码调试；代码的整理和优化；实验报告其余部分

2.3 实验原理

### 2.3.1 Socket原理

Socket(套接字)可以看成是两个网络应用程序进行通信时，各自通信连接中的端点，这是一个逻辑上的概念。它是网络环境中进程间通信的API(应用程序编程接口)，也是可以被命名和寻址的通信端点，使用中的每一个套接字都有其类型和一个与之相连进程。通信时其中一个网络应用程序将要传输的一段信息写入它所在主机的 Socket中，该 Socket通过与网络接口卡(NIC)相连的传输介质将这段信息送到另外一台主机的 Socket中，使对方能够接收到这段信息。 Socket是由IP地址和端口结合的，提供向应用层进程传送数据包的机制

### 2.3.2 System.Net.Sockets

System.Net.Sockets命名空间为需要严格控制对网络的访问的开发人员提供了 Windows 套接字 （Winsock） 接口的托管实现。Socket 类为网络通信提供了一组丰富的方法和属性。套接字类允许使用协议类型枚举中列出的任何通信协议执行同步和异步数据传输。

### 2.3.3 FTP 原理

#### 2.3.3.1 FTP概述

文件传输协议（FTP）作为网络共享文件的传输协议，在网络应用软件中具有广泛的应用。FTP的目标是提高文件的共享性和可靠高效地传送数据。

在传输文件时，FTP 客户端程序先与服务器建立连接，然后向服务器发送命令。服务器收到命令后给予响应，并执行命令。FTP 协议与操作系统无关，任何操作系统上的程序只要符合 FTP 协议，就可以相互传输数据。本实验基于 LINUX Windows平台，对 FTP 客户端的实现原理进行详尽的解释并阐述如何使用C#语言编写一个简单的 FTP 客户端。

#### 2.3.3.2 FTP协议

相比其他协议，如 HTTP 协议，FTP 协议要复杂一些。与一般的 C/S 应用不同点在于一般的C/S 应用程序一般只会建立一个 Socket 连接，这个连接同时处理服务器端和客户端的连接命令和数据传输。而FTP协议中将命令与数据分开传送的方法提高了效率。

**FTP端口：**

FTP 使用 2 个端口，一个数据端口和一个命令端口（也叫做控制端口）。这两个端口一般是21 （命令端口）和 20数据端口）。控制 Socket 用来传送命令，数据 Socket 是用于传送数据。每一个 FTP 命令发送之后，FTP 服务器都会返回一个字符串，其中包括一个响应代码和一些说明信息。其中的返回码主要是用于判断命令是否被成功执行了。

（1）命令端口

一般来说，客户端有一个 Socket 用来连接 FTP 服务器的相关端口，它负责 FTP 命令的发送和接收返回的响应信息。一些操作如“登录”、“改变目录”、“删除文件”，依靠这个连接发送命令就可完成。

（2）数据端口

对于有数据传输的操作，主要是显示目录列表，上传、下载文件，我们需要依靠另一个 Socket来完成。

**FTP模式：**

如果使用被动模式，通常服务器端会返回一个端口号。客户端需要用另开一个 Socket 来连接这个端口，然后我们可根据操作来发送命令，数据会通过新开的一个端口传输。

如果使用主动模式，通常客户端会发送一个端口号给服务器端，并在这个端口监听。服务器需要连接到客户端开启的这个数据端口，并进行数据的传输。

（1）主动模式 (PORT)

主动模式下，客户端随机打开一个大于 1024 的端口向服务器的命令端口 P，即 21 端口，发起连接，同时开放N +1 端口监听，并向服务器发出 “port N+1” 命令，由服务器从它自己的数据端口 (20) 主动连接到客户端指定的数据端口 (N+1)。

FTP 的客户端只是告诉服务器自己的端口号，让服务器来连接客户端指定的端口。对于客户端的防火墙来说，这是从外部到内部的连接，可能会被阻塞。

（2）被动模式 (PASV)

为了解决服务器发起到客户的连接问题，有了另一种 FTP 连接方式，即被动方式。命令连接和数据连接都由客户端发起，这样就解决了从服务器到客户端的数据端口的连接被防火墙过滤的问题。

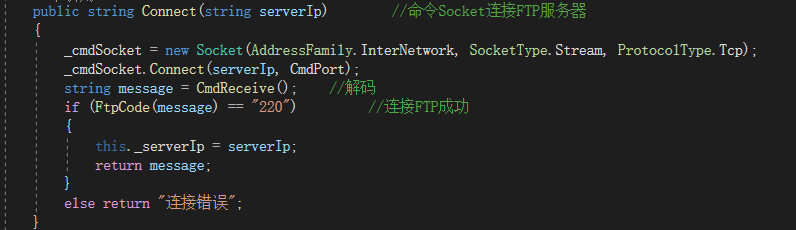
被动模式下，当开启一个 FTP 连接时，客户端打开两个任意的本地端口 (N > 1024 和 N+1) 。

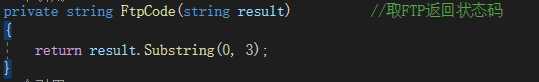
第一个端口连接服务器的 21 端口，提交 PASV 命令。然后，服务器会开启一个任意的端口 (P > 1024 )，返回如“227 entering passive mode (127,0,0,1,4,18)”。 它返回了 227 开头的信息，在括号中有以逗号隔开的六个数字，前四个指服务器的地址，最后两个，将倒数第二个乘 256 再加上最后一个数字，这就是 FTP 服务器开放的用来进行数据传输的端口。如得到 227 entering passive mode (h1,h2,h3,h4,p1,p2)，那么端口号是 p1\*256+p2，ip 地址为h1.h2.h3.h4。这意味着在服务器上有一个端口被开放。客户端收到命令取得端口号之后, 会通过 N+1 号端口连接服务器的端口 P，然后在两个端口之间进行数据传输。

# 3 实验方案（详细设计）

## 3.1 邓展麒部分

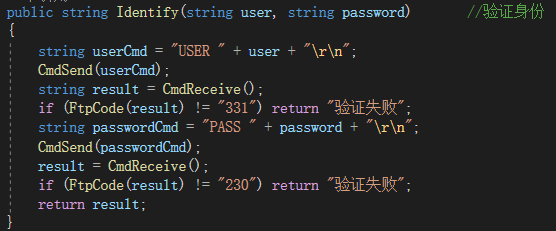
### 3.1.1 连接





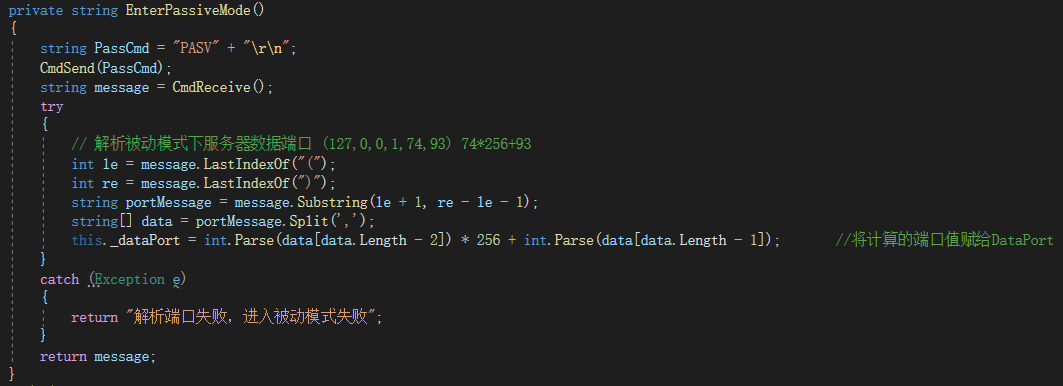
连接FTP服务器，首先将cmdSocket命令Socket初始化，用Connect函数与FTP服务器端连接，服务器IP和端口号作为参数。接着接收服务器返回状态，FtpCode函数就是取服务器返回的值的前三位为状态码，当状态码为220代表FTP连接成功，此时用一个属性serverIp将当前服务器IP存起来，准备给数据Socket使用。

### 3.1.2 登陆



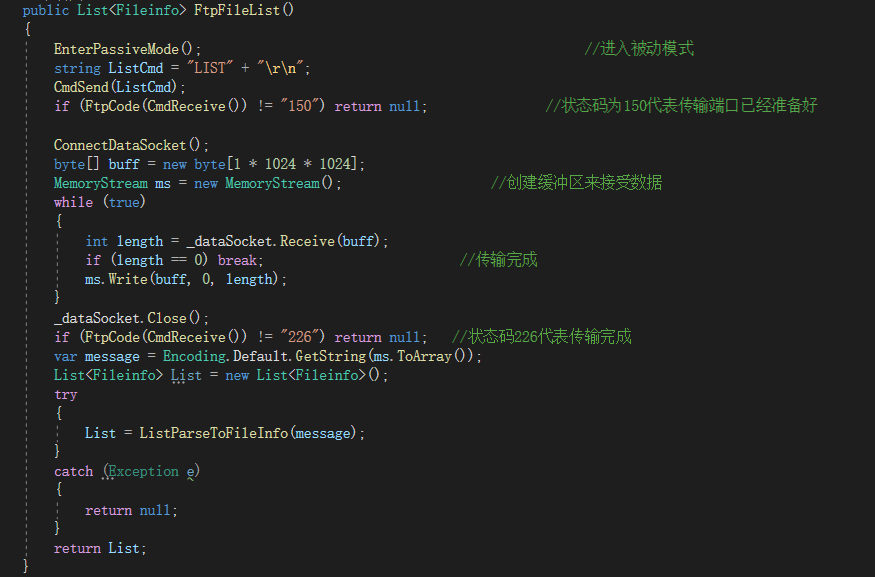
连接上服务器后，将用户名用命令Socket传到服务器，并接收回应，同样用FtpCode来获取状态码，当状态码为331代表用户名正确。同样，用命令Socket将密码传到服务器，并接收回应，当返回的状态码为230，代表密码正确，验证成功。

### 3.1.3 进入被动模式



在需要对FTP服务器传送或接收数据时，需要打开数据端口，首先需要进入被动模式。用命令Socket将进入被动模式的命令发送给服务器，然后接收服务器返回的消息。对服务器返回的消息进行解析，首先解析出服务器返回的数据端口信息（即括号内的内容），然后用Split方法获取后两个数，再用转换等式dataPort=a\*256+b (其中，a为右数第二个数，b为最右的数)，再用属性dataPort将数据端口号保存，则进入被动模式成功。如果解析失败，表示进入被动模式失败。

### 3.1.4 获取服务器端文件列表



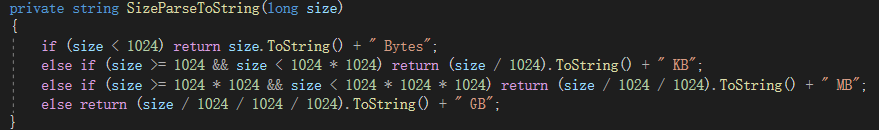
首先进入被动模式获取数据端口号，向服务器端发送LIST命令，并用数据Socket连接数据端口，创建一个缓冲区来准备接收数据。准备好之后用数据端口接收服务器的文件列表信息，并将其写入缓冲区，读入MemoryStream，接收到的数据量为0代表接收结束。关掉数据Socket后，命令Socket接收到226的状态码代表传输完成。接下来处理收到的文件列表信息，首先将MemoryStream的数据转成String类型，在对其进行正则处理，转化为文件信息类的一个List，最后返回这个List。

ListParseToFileInfo(正则处理函数)：



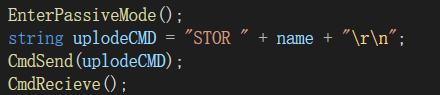
首先以“\r\n”为分隔将每个文件的信息分开放入一个字符数组，再在该数组中循环。对每一个文件信息，以“ ”为分隔将其分为4个group，分别是修改时间、是否为文件夹、文件名和文件大小，对每个group进行处理后将其赋给文件信息类的相应属性，这样就获得了一个文件信息的对象，将其添加到List中，最后返回一个文件信息List。

SizeParseToString方法：

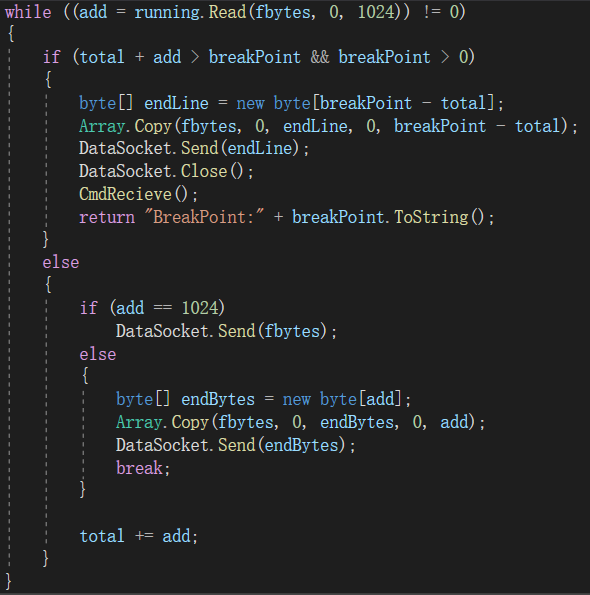


## 3.2 杨景博部分

### 3.2.1 上传

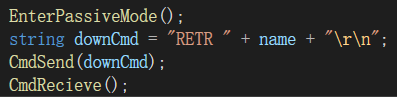


首先，进入被动模式。并且发送上传指令。

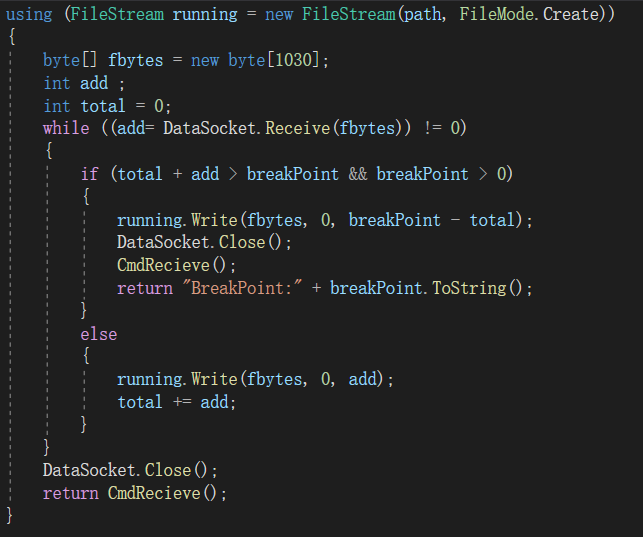


之后，设置一个大小为1024byte的缓冲区用来存放每次上传的文件数据，add变量为每次传输的文件数据量，total变量为总共传输的文件数据量。当还需要传输的文件数据不为0时，循环传输，如果有断点，则在达到断点前的位置停止数据传输，否则判断数据量发送本次是否是最后一次上传数据，若是则跳出循环，若不是，则正常上传。

### 3.2.2 下载



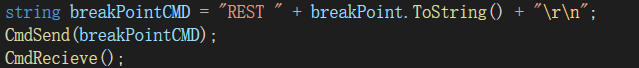
首先，进入被动模式。并且发送下载指令。



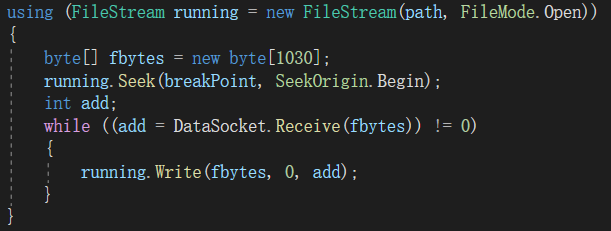
之后，设置一个大小为1030byte的缓冲区用来存放每次下载的文件数据，add变量为每次传输的文件数据量，total变量为总共传输的文件数据量。当还需要传输的文件数据不为0时，循环传输，如果有断点，则在达到断点前的位置停止数据传输，否则正常填满缓冲器传输，直到文件数据完全下载完。

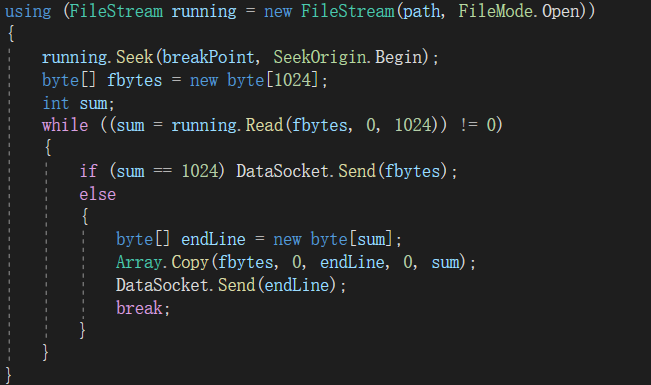
### 3.2.3 断点续传

在上传下载的基础上使用REST指令关于断点（文件数据传输起始点）的设置：



之后操作类似于无断点情况下的上传和下载操作：



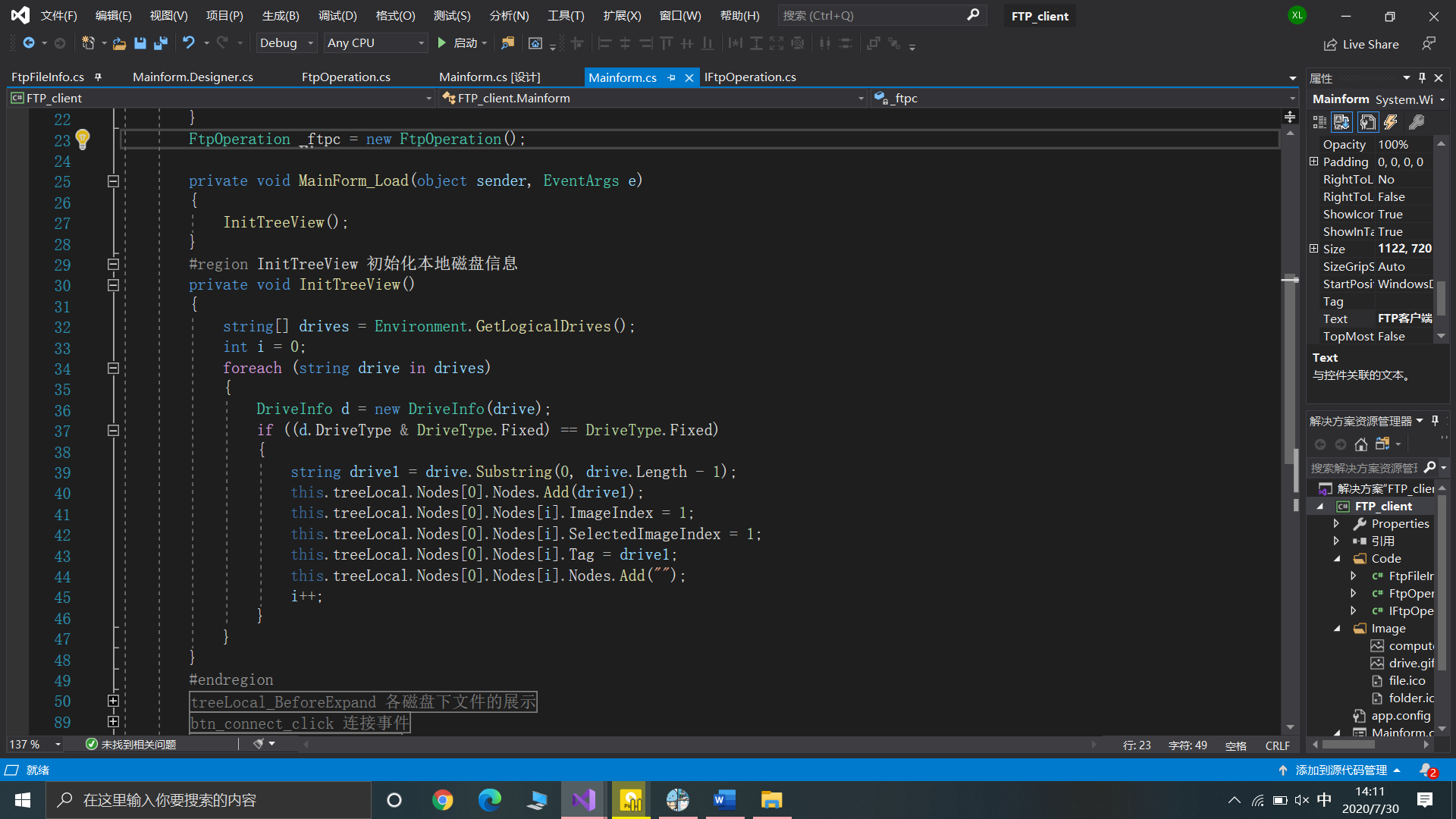


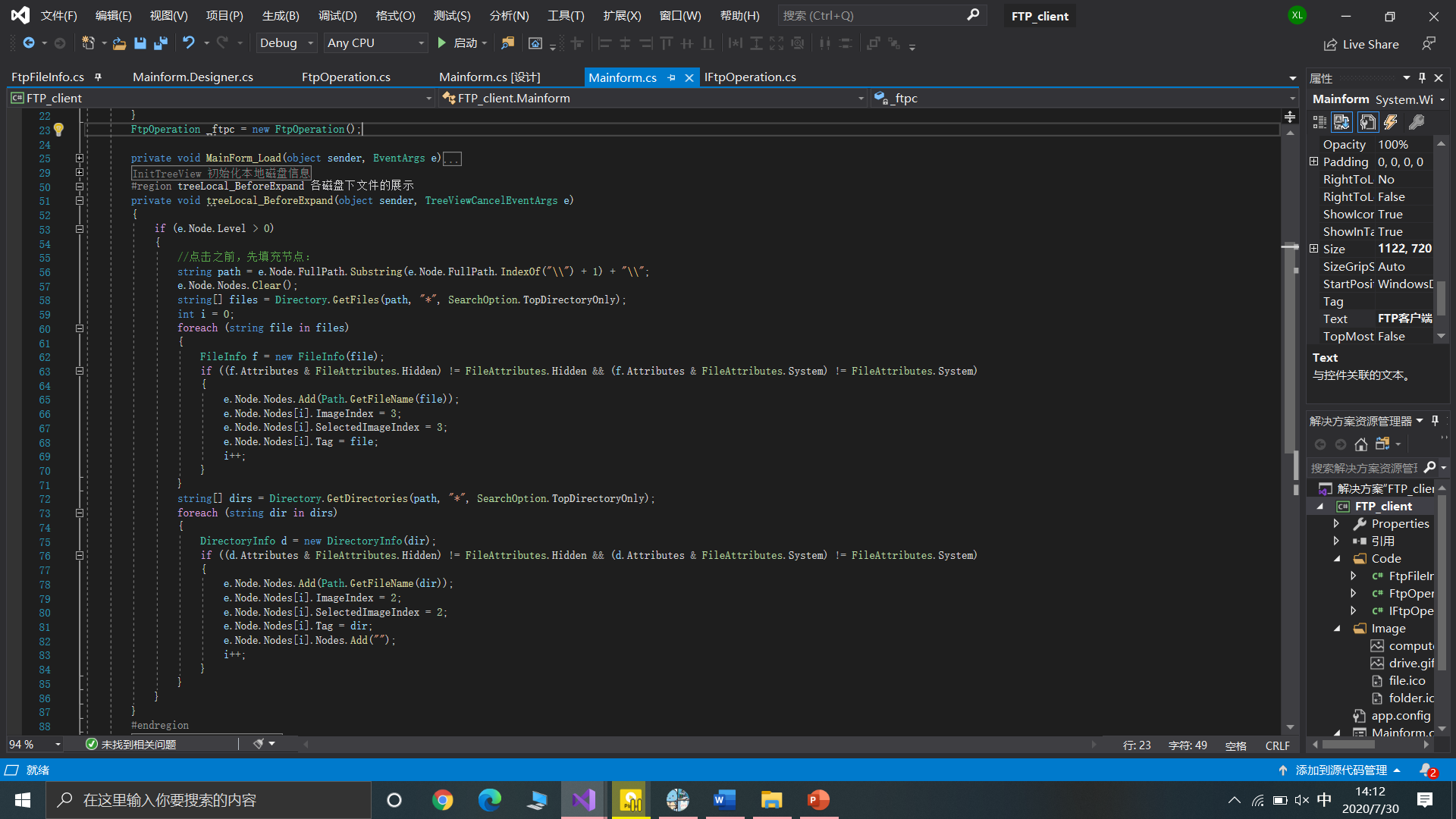
## 3.3 龙晓怡部分

### 3.3.1 本地文件树形展示

首先要初始化本地磁盘信息，变量drives是返回驱动器名称的字符串数组，比如电脑上的C盘,D盘,E盘，实例化d用来访问每个驱动器的信息，并将浙西信息存入对应节点中，每个磁盘节点显示的图像，其索引对应为inmageList中磁盘图像。

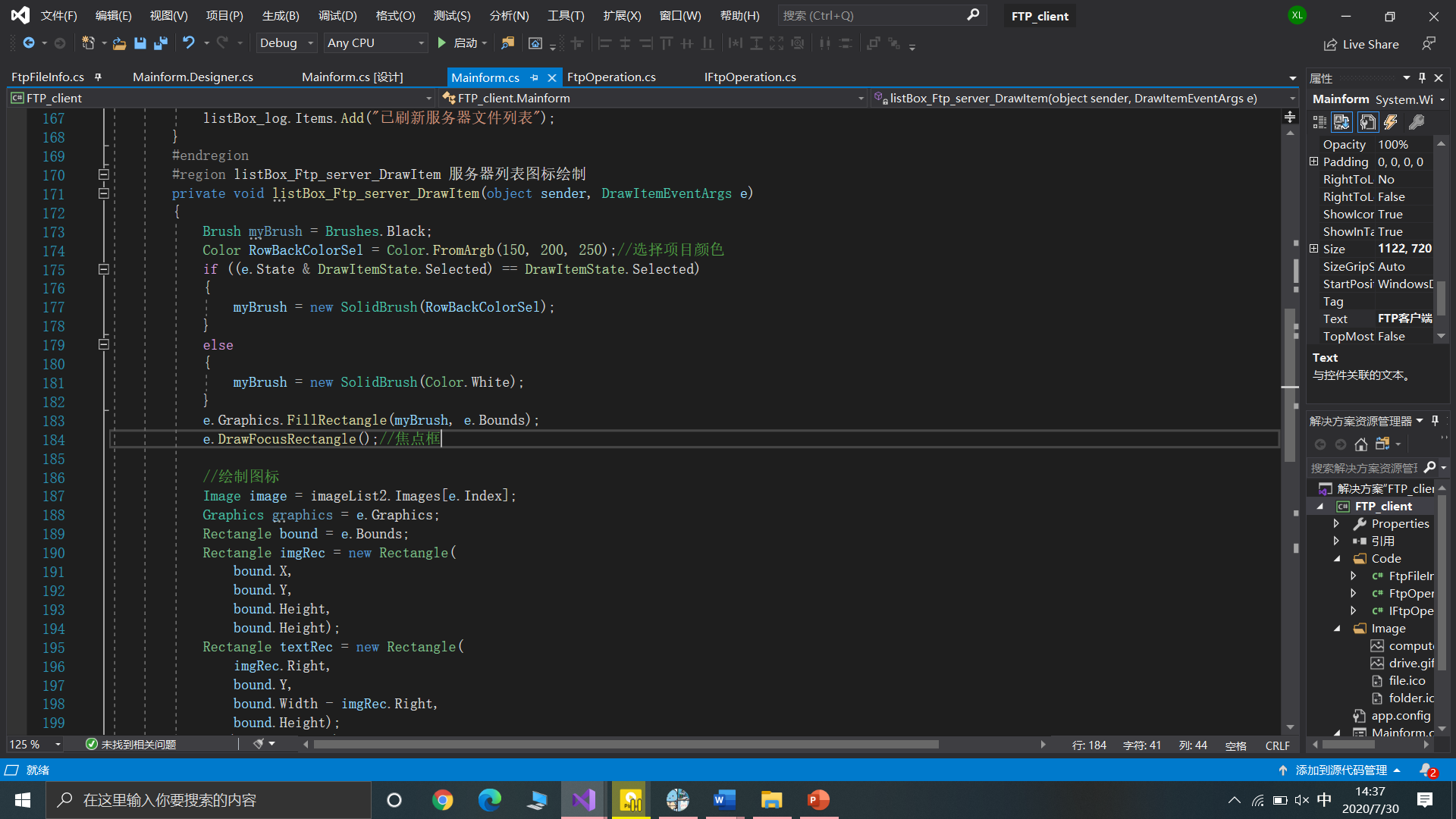
然后是各磁盘下文件信息的展示，在每次点击磁盘之前，需要填充好每个磁盘下的文件信息，依次扫描如果该文件没有被隐藏且不是系统文件，就标记为file,并且显示文件图标，获取文件名；如果是文件夹，则显示文件夹图标，并在该目录下面填充所含的文件信息。每一级的节点后加上空格e.Node.Nodes[i].Nodes.Add("")，帮助区分文件所在路径。

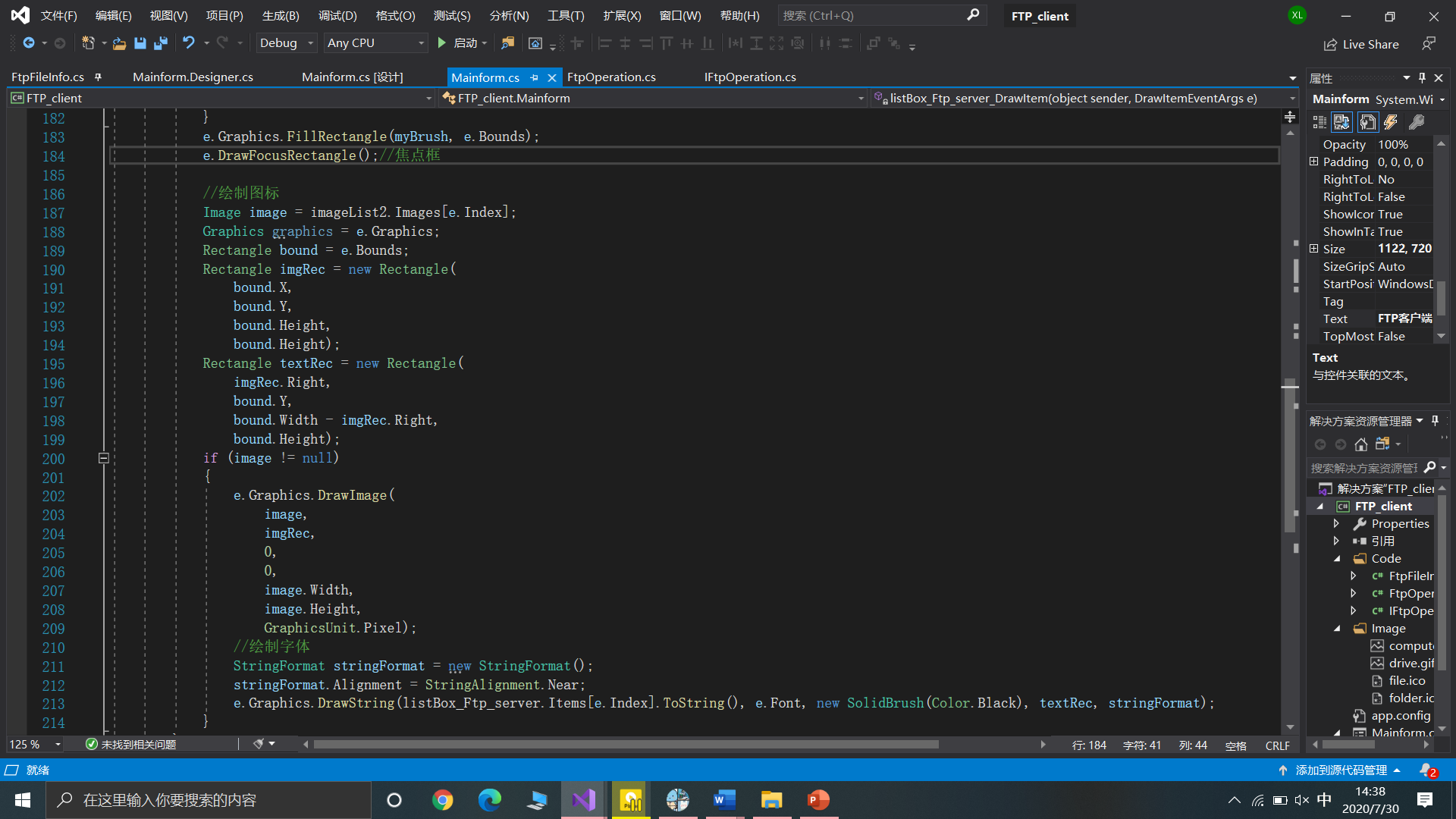




### 3.3.2 服务器端文件展示

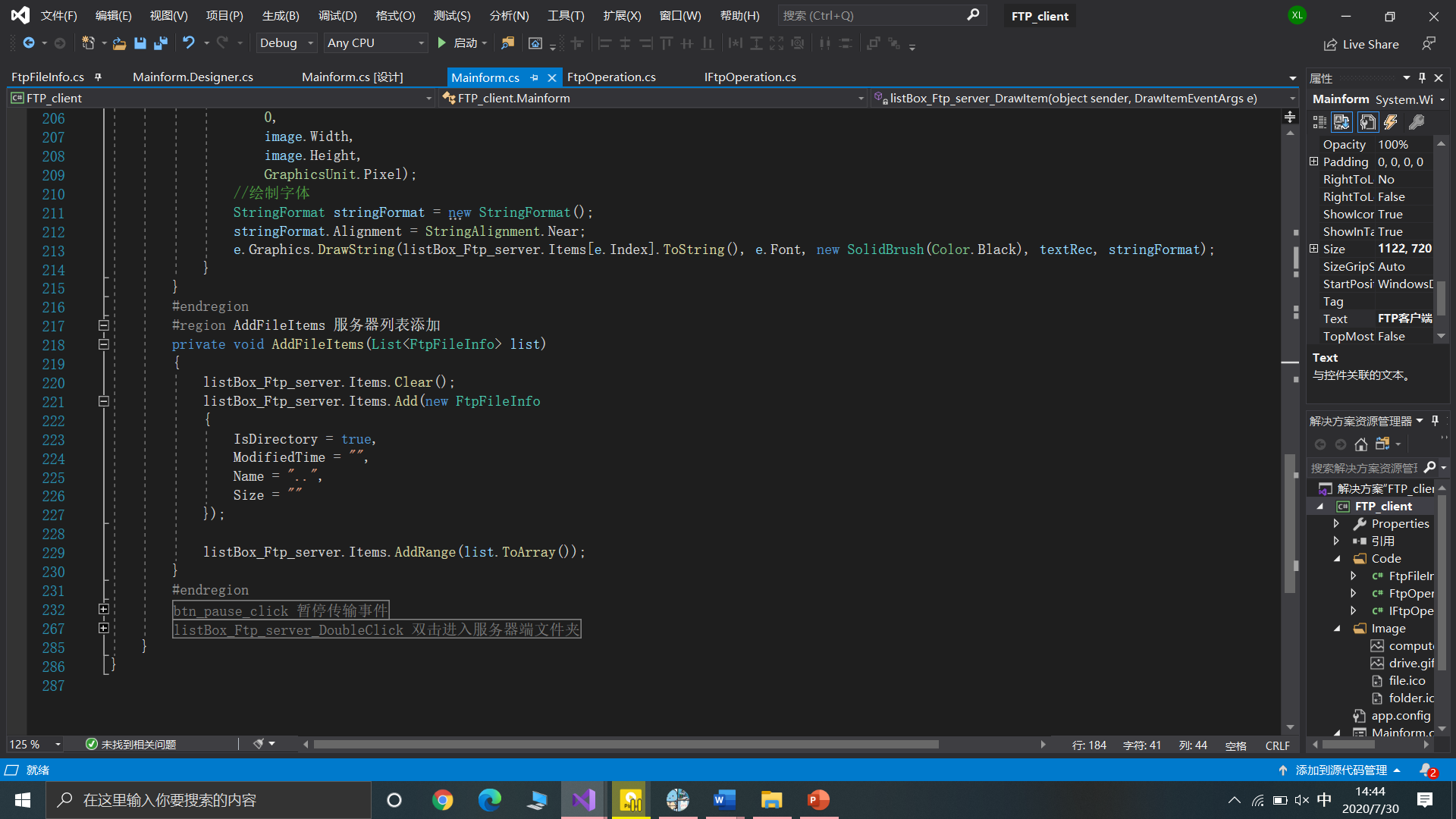
使用ListBox的DrawItem函数，在服务器端显示文件名，文件大小，修改时间之前，根据文件类型绘制文件图标，定义好颜色，大小等相关参数，填充inmagelist的图像。

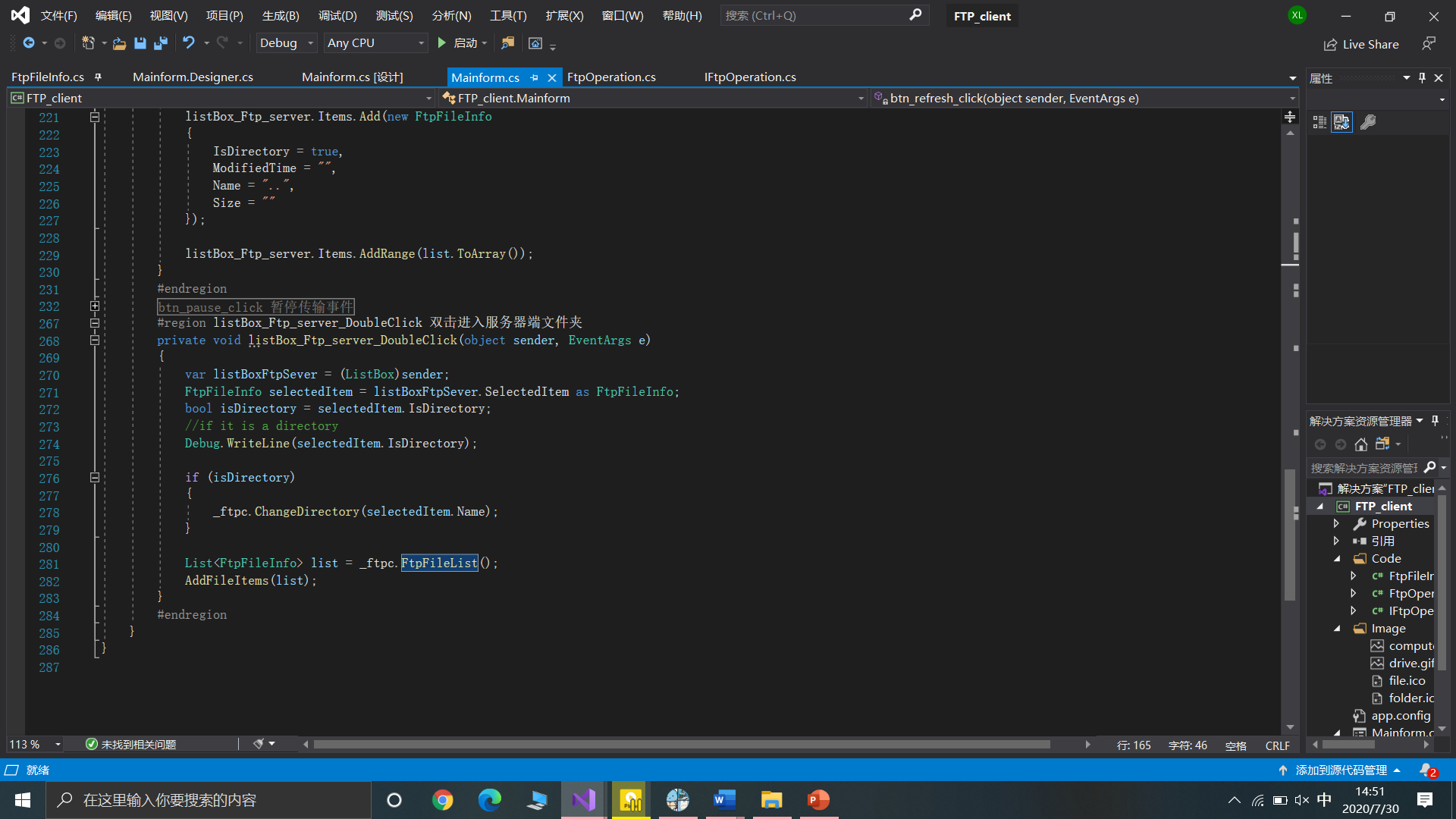


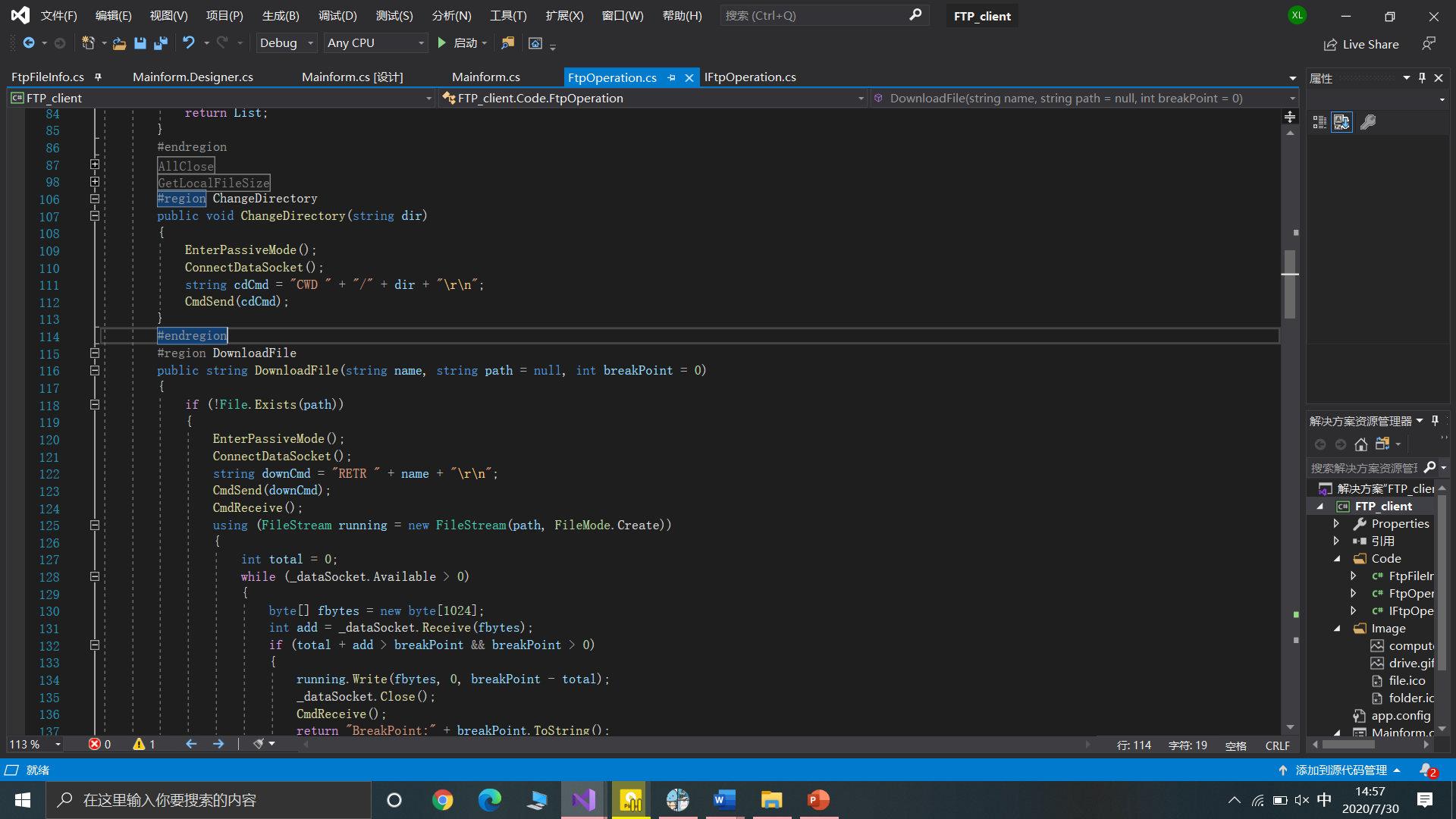


### 3.3.3 服务器端文件夹的访问

先向服务器列表添加文件信息：首先，添加一个返回上级路径的文件夹“..”，这样可以在FTP服务器端双击文件夹进入，进行下载操作，然后再返回上级目录。其余文件项再一次添加进ListBox中

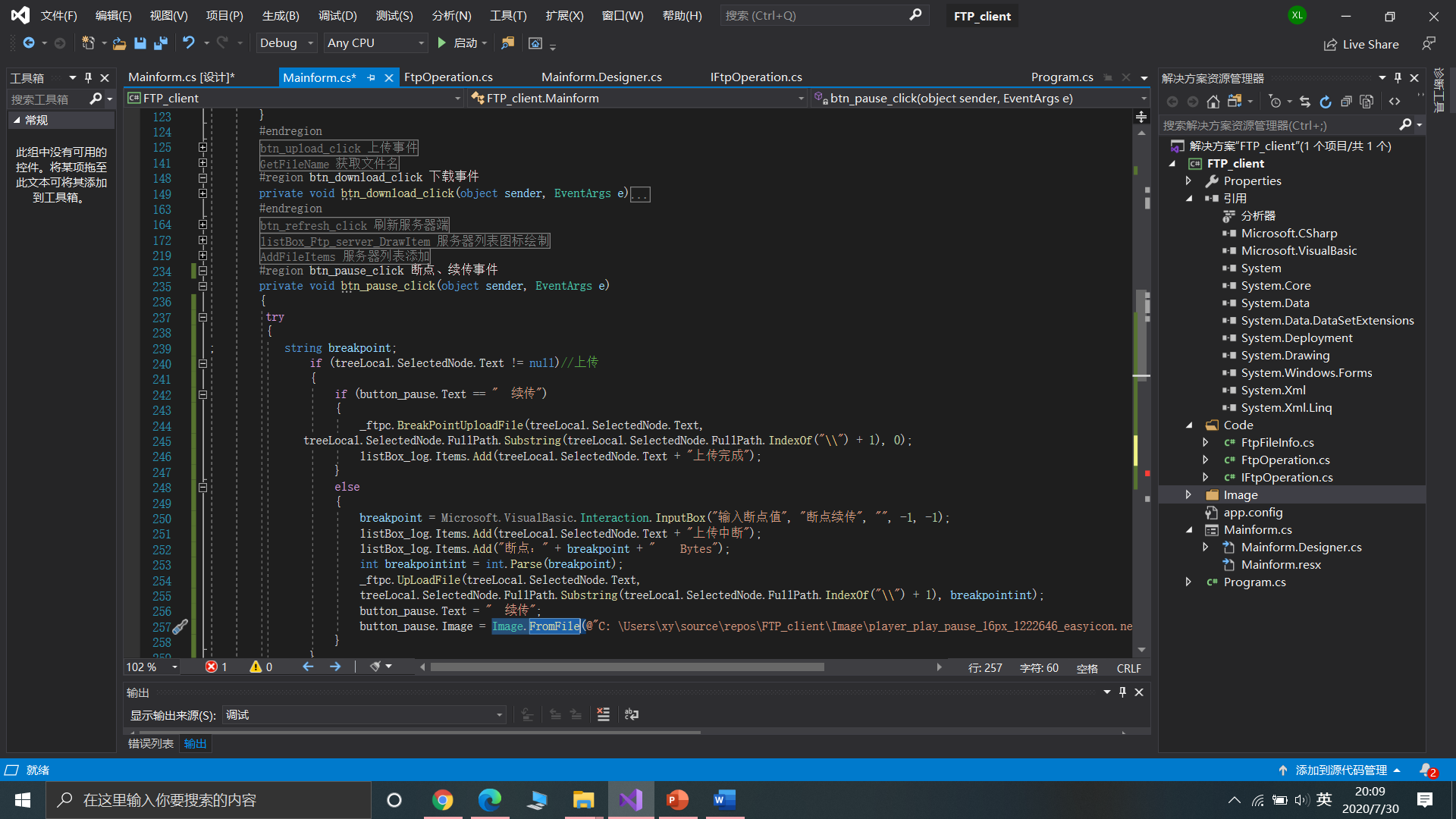


在服务器端双击访问文件夹，如果是文件夹，调用ChangeDirectoty函数，定义于FTPOperations中，注意客户端发送的是CWD命令。



### 3.3.4 断点续传交互

引用 Microsoft.VisualBasic，点击断点按钮时弹出窗体，提示输入断点值，再进行上传或下载函数的调用。点击后，通过Image.FromFile（@”图片路径）更改按钮的图标，更改文本为续传。其余调用FtpOperation中的函数。

（该部分仅展示了部分代码，详见附录）

# 4 程序主要界面

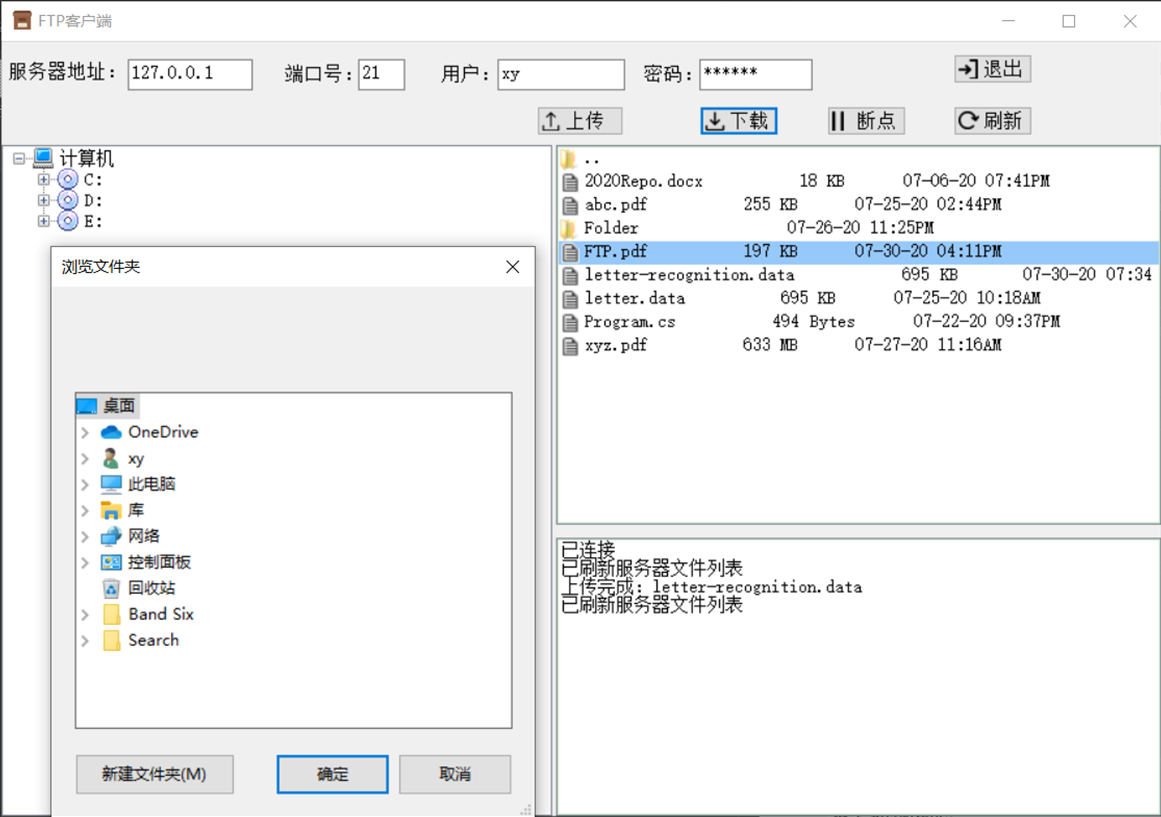
### 4.1 登陆连接

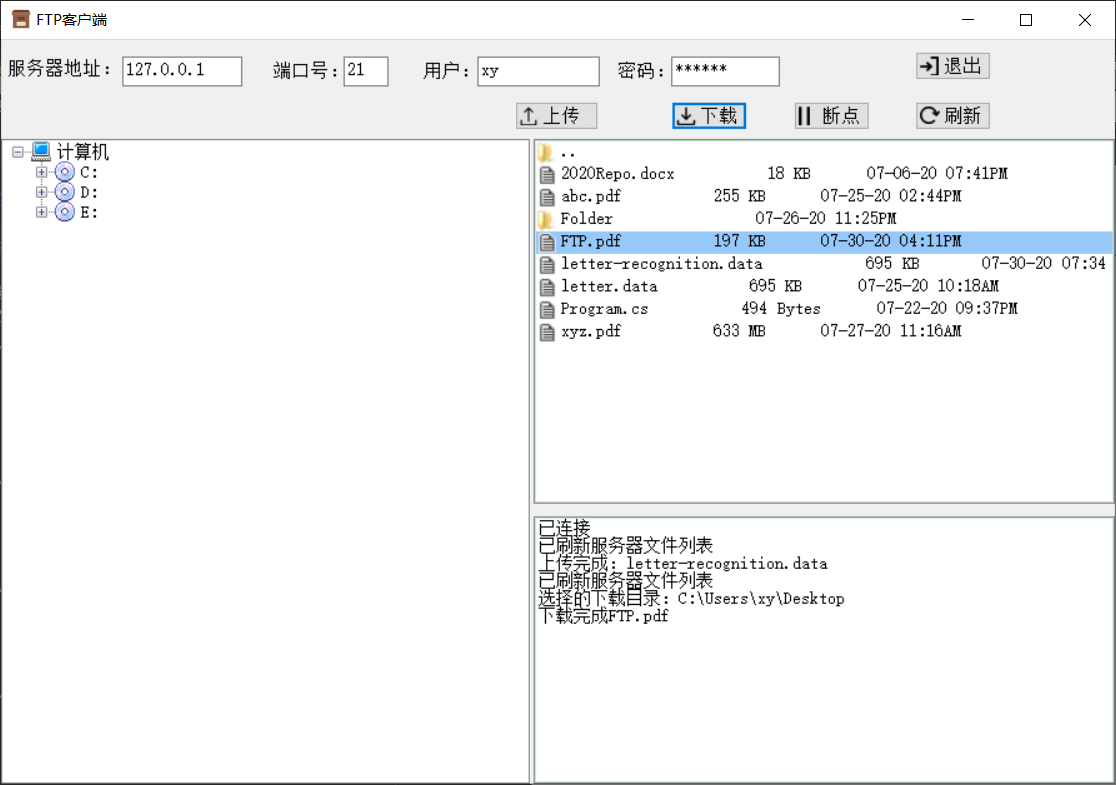
### 4.2 退出

### 4.3 刷新操作

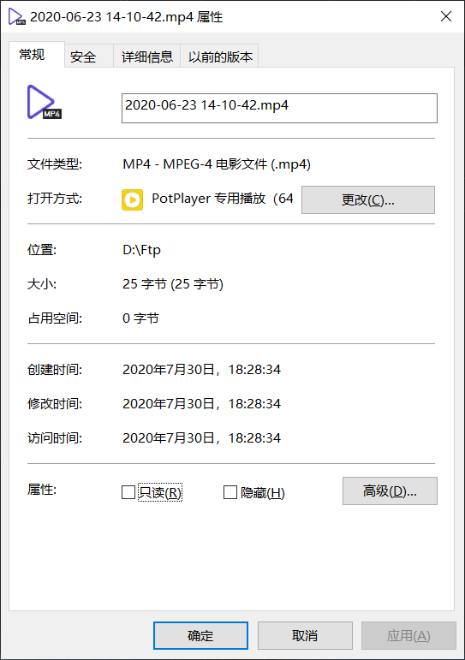
### 4.4 上传

### 4.5 下载

点击下载按钮，弹出选择文件夹对话框，选好后点击确定，开始下载，刷新后可在本地文件夹中找到。



### 4.6 断点

在IIS中FTP站点文件夹上，该文件大小为25字节

### 4.7 续传

上传完成后，点击续传，继续传输文件。下载同理。

在IIS中FTP站点文件夹中可以看到，该文件已经传输完成。

# 附录-程序源代码

## IftpOperation.cs

**using** **System**;

**using** **System.Collections.Generic**;

**using** **System.Linq**;

**using** **System.Text**;

**using** **System.Threading.Tasks**;

**namespace** **FTP\_client.Code**

{

**public** **interface** IFtpOperation

{

**string** Connect(**string** serverIp);

**string** Identify(**string** user, **string** password);

List<FtpFileInfo> FtpFileList();

**string** AllClose();

**long** GetLocalFileSize(**string** name);

**string** DownloadFile(**string** name, **string** path = **null**, **int** breakPoint = 0);

**string** UpLoadFile(**string** name, **string** path = **null**, **int** breakPoint = 0);

**string** BreakPointDownLoadFile(**string** name, **string** path, **int** breakPoint);

**string** BreakPointUploadFile(**string** name, **string** path, **int** breakPoint);

**void** ChangeDirectory(**string** dir);

}

}

## FtpFileInto.cs

**using** **System**;

**using** **System.Collections.Generic**;

**using** **System.Linq**;

**using** **System.Text**;

**using** **System.Threading.Tasks**;

**namespace** **FTP\_client.Code**

{

**public** **class** FtpFileInfo

{

**public** **bool** IsDirectory { **get**; **set**; } *//是否是文件夹*

**public** **string** Name { **get**; **set**; } *//文件名*

**public** **string** Size { **get**; **set**; } *//文件大小*

**public** **string** ModifiedTime { **get**; **set**; } *//最近修改时间*

**public** **override** **string** ToString()

{

**return** Name + " " + Size + " " + ModifiedTime;

}

}

}

## FtpOperation.cs

**using** **System**;

**using** **System.Collections.Generic**;

**using** **System.Diagnostics**;

**using** **System.IO**;

**using** **System.Net.Sockets**;

**using** **System.Text**;

**using** **System.Text.RegularExpressions**;

**using** **FTP\_client.Code**;

**namespace** **FTP\_client.Code**

{

**public** **class** FtpOperation : IFtpOperation

{

*/// <summary>命令Socket</summary>*

**private** Socket \_cmdSocket;

*/// <summary>数据Socket</summary>*

**private** Socket \_dataSocket;

*/// <summary>命令端口</summary>*

**public** **static** **readonly** **int** CmdPort = 21;

*/// <summary>数据端口</summary>*

**private** **int** \_dataPort;

*/// <summary>服务器地址</summary>*

**private** **string** \_serverIp;

*/// <summary>客户端命令端口返回的最后一条消息</summary>*

**private** **string** \_lastMessage;

#region public method

#region Connect

**public** **string** Connect(**string** serverIp) *//命令Socket连接FTP服务器*

{

\_cmdSocket = **new** Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

\_cmdSocket.Connect(serverIp, CmdPort);

**string** message = CmdReceive(); *//解码*

**if** (FtpCode(message) == "220") *//连接FTP成功*

{

**this**.\_serverIp = serverIp;

**return** message;

}

**else** **return** "连接错误";

}

#endregion

#region Identify

**public** **string** Identify(**string** user, **string** password) *//验证身份*

{

**string** userCmd = "USER " + user + "\r\n";

CmdSend(userCmd);

**string** result = CmdReceive();

**if** (FtpCode(result) != "331") **return** "验证失败";

**string** passwordCmd = "PASS " + password + "\r\n";

CmdSend(passwordCmd);

result = CmdReceive();

**if** (FtpCode(result) != "230") **return** "验证失败";

**return** result;

}

#endregion

#region List<Fileinfo> FtpFileList

**public** List<FtpFileInfo> FtpFileList()

{

EnterPassiveMode(); *//进入被动模式*

**string** ListCmd = "LIST" + "\r\n";

CmdSend(ListCmd);

**if** (FtpCode(CmdReceive()) != "150") **return** **null**; *//状态码为150代表传输端口已经准备好*

ConnectDataSocket();

**byte**[] buff = **new** **byte**[1 \* 1024 \* 1024];

MemoryStream ms = **new** MemoryStream(); *//创建缓冲区来接受数据*

**while** (**true**)

{

**int** length = \_dataSocket.Receive(buff);

**if** (length == 0) **break**; *//传输完成*

ms.Write(buff, 0, length);

}

\_dataSocket.Close();

**if** (FtpCode(CmdReceive()) != "226") **return** **null**; *//状态码226代表传输完成*

**var** message = Encoding.Default.GetString(ms.ToArray());

List<FtpFileInfo> List = **new** List<FtpFileInfo>();

**try**

{

List = ListParseToFileInfo(message);

}

**catch** (Exception)

{

**return** **null**;

}

**return** List;

}

#endregion

#region AllClose

**public** **string** AllClose() *//彻底断开连接，没什么用，因为要保持命令端口连接*

{

\_dataSocket.Close();

**string** Close = "QUIT" + "\r\n";

CmdSend(Close);

**string** msg = CmdReceive();

\_cmdSocket.Close();

**return** msg;

}

#endregion

#region GetLocalFileSize

**public** **long** GetLocalFileSize(**string** name)

{

**if** (!File.Exists(name)) **return** -1;*//文件不存在返回-1*

FileInfo fileInfo = **new** FileInfo(name);

**return** fileInfo.Length;

}

#endregion

#region ChangeDirectory

**public** **void** ChangeDirectory(**string** dir)

{

EnterPassiveMode();

ConnectDataSocket();

**string** cdCmd = "CWD " + "/" + dir + "\r\n";

CmdSend(cdCmd);

}

#endregion

#region DownloadFile

**public** **string** DownloadFile(**string** name, **string** path = **null**, **int** breakPoint = 0)

{

**if** (!File.Exists(path))

{

EnterPassiveMode();

ConnectDataSocket();

**string** downCmd = "RETR " + name + "\r\n";

CmdSend(downCmd);

CmdReceive();

**using** (FileStream running = **new** FileStream(path, FileMode.Create))

{

**int** total = 0;

**while** (\_dataSocket.Available > 0)

{

**byte**[] fbytes = **new** **byte**[1024];

**int** **add** = \_dataSocket.Receive(fbytes);

**if** (total + **add** > breakPoint && breakPoint > 0)

{

running.Write(fbytes, 0, breakPoint - total);

\_dataSocket.Close();

CmdReceive();

**return** "BreakPoint:" + breakPoint.ToString();

}

**else**

{

running.Write(fbytes, 0, **add**);

total += **add**;

}

}

\_dataSocket.Close();

**return** CmdReceive();

}

}

**else***//本地存在该文件则查看是否需要断点续传*

{

**long** localFileSize = GetLocalFileSize(path);*//获取本地文件大小*

**long** ftpFileSize = GetFtpFileSize(name);*//获取服务器文件大小*

**if** (localFileSize < ftpFileSize)*//若服务器文件比本地文件大则启动断点续传*

{

BreakPointDownLoadFile(name, path, (**int**)localFileSize);

\_dataSocket.Close();

**return** CmdReceive();

}

**else**

**return** "已经下载完成";

}

}

#endregion

#region UpLoadFile

**public** **string** UpLoadFile(**string** name, **string** path = **null**, **int** breakPoint = 0)

{

**if** (path == **null**) path = name;

**long** ftpFileSize = GetLocalFileSize(name);

**if** (ftpFileSize == -1)

{

EnterPassiveMode();

ConnectDataSocket();

**string** uplodeCMD = "STOR " + name + "\r\n";

CmdSend(uplodeCMD);

CmdReceive();

**using** (FileStream running = **new** FileStream(path, FileMode.Open))

{

**int** total = 0;

Debug.WriteLine($@"running's length is {running.Length}");

**while** (running.Position != running.Length)

{

Debug.WriteLine(running.Position);

**byte**[] buffer = **new** **byte**[1024];

**int** **add** = running.Read(buffer, 0, 1024);

**if** (total + **add** > breakPoint && breakPoint > 0)

{

**byte**[] endLine = **new** **byte**[breakPoint - total];

Array.Copy(buffer, 0, endLine, 0, breakPoint - total);

\_dataSocket.Send(endLine);

\_dataSocket.Close();*//传输成功后关闭数据套接字*

CmdReceive();

**return** "BreakPoint:" + breakPoint.ToString();

}

**else**

{

**if** (**add** == 1024)*//当数组满直接发送*

\_dataSocket.Send(buffer);

**else***//数组不满则只发送前面一部分*

{

**byte**[] endBytes = **new** **byte**[**add**];

Array.Copy(buffer, 0, endBytes, 0, **add**);

\_dataSocket.Send(endBytes);

**break**;

}

total += **add**;

}

}

}

\_dataSocket.Close();

**return** CmdReceive();

}

**else**

{

**if** (ftpFileSize < GetLocalFileSize(path))

{

BreakPointUploadFile(name, path, (**int**)ftpFileSize);

\_dataSocket.Close();

*// Console.WriteLine("上传失败");*

**return** CmdReceive();

}

**else**

{

*// Console.WriteLine("完成上传");*

**return** "已经上传完成";

}

}

}

#endregion

#region BreakPointDownLoadFile

**public** **string** BreakPointDownLoadFile(**string** name, **string** path, **int** breakPoint)

{

**if** (path == **null**) path = name;

EnterPassiveMode();

ConnectDataSocket();

**string** breakPointCmd = "REST " + breakPoint.ToString() + "\r\n";

CmdSend(breakPointCmd);

CmdReceive();

**string** downCmd = "RETR " + name + "\r\n";

CmdSend(downCmd);

CmdReceive();

**using** (FileStream running = **new** FileStream(path, FileMode.Open))

{

**byte**[] fbytes = **new** **byte**[1024];

running.Seek(breakPoint, SeekOrigin.Begin);

**int** **add** = \_dataSocket.Receive(fbytes);

**while** (**add** != 0)

{

running.Write(fbytes, 0, **add**);

}

}

**return** "下载完成";

}

#endregion

#region BreakPointUploadFile

**public** **string** BreakPointUploadFile(**string** name, **string** path, **int** breakPoint)

{

**if** (path == **null**) path = name;

EnterPassiveMode();

ConnectDataSocket();

**string** breakPointCmd = "REST " + breakPoint.ToString() + "\r\n";

CmdSend(breakPointCmd);

CmdReceive();

**string** uploadCmd = "STOR " + name + "\r\n";

CmdSend(uploadCmd);

CmdReceive();

**using** (FileStream running = **new** FileStream(path, FileMode.Open))

{

running.Seek(breakPoint, SeekOrigin.Begin);

**byte**[] fbytes = **new** **byte**[1024];

**int** sum = running.Read(fbytes, 0, 1024);

**while** (sum != 0)

{

**if** (sum == 1024) \_dataSocket.Send(fbytes);

**else**

{

**byte**[] endLine = **new** **byte**[sum];

Array.Copy(fbytes, 0, endLine, 0, sum);

\_dataSocket.Send(endLine);

**break**;

}

}

}

**return** "上传完成";

}

#endregion

#endregion

#region private method

#region GetFtpFileSize

**private** **long** GetFtpFileSize(**string** fileName)

{

*//获取文件大小的命令*

**string** sizeCmd = "SIZE " + fileName + "\r\n";

CmdSend(sizeCmd);

**string** message = CmdReceive();

*//获取服务器返回的响应码*

**string** respond = FtpCode(\_lastMessage);

**if** (respond == "550" || respond == "451")*//响应码错误*

{

CmdReceive();*//吞入服务器报错信息*

**return** -1;

}

*//响应码正确则分割输出*

**string**[] messageSplit = Regex.Split(message, " ");

**long** fileSize = **long**.Parse(messageSplit[messageSplit.Length - 1]);

**return** fileSize;

}

#endregion

#region EnterPassiveMode

**private** **string** EnterPassiveMode()

{

**string** PassCmd = "PASV" + "\r\n";

CmdSend(PassCmd);

**string** message = CmdReceive();

**try**

{

*// 解析被动模式下服务器数据端口 (127,0,0,1,74,93) 74\*256+93*

**int** le = message.LastIndexOf("(");

**int** re = message.LastIndexOf(")");

**string** portMessage = message.Substring(le + 1, re - le - 1);

**string**[] data = portMessage.Split(',');

**this**.\_dataPort = **int**.Parse(data[data.Length - 2]) \* 256 + **int**.Parse(data[data.Length - 1]); *//将计算的端口值赋给DataPort*

}

**catch** (Exception e)

{

**return** "解析端口失败，进入被动模式失败";

}

**return** message;

}

#endregion

#region ConnectDataSocket

**private** **void** ConnectDataSocket()

{

\_dataSocket = **new** Socket(AddressFamily.InterNetwork, SocketType.Stream, ProtocolType.Tcp);

\_dataSocket.Connect(**this**.\_serverIp, \_dataPort); *//ServerIP与命令端口的IP相同，DataPort在进入被动模式时已经设好*

}

#endregion

#region CmdSend

**private** **void** CmdSend(**string** cmd) *//命令端口发出命令*

{

\_cmdSocket.Send(Encoding.UTF8.GetBytes(cmd));

}

#endregion

#region CmdReceive

**private** **string** CmdReceive() *//命令端口接受消息*

{

**byte**[] result = **new** **byte**[1024];

**int** recieveLength = \_cmdSocket.Receive(result);

\_lastMessage = Encoding.UTF8.GetString(result, 0, recieveLength);

**return** \_lastMessage;

}

#endregion

#region FtpCode

**private** **string** FtpCode(**string** result) *//取FTP返回状态码*

{

**return** result.Substring(0, 3);

}

#endregion

#region List<Fileinfo> ListParseToFileInfo

**private** List<FtpFileInfo> ListParseToFileInfo(**string** fileList)

{

**string**[] files = fileList.Split(**new** **string**[] { "\r\n" }, StringSplitOptions.RemoveEmptyEntries);

**string** pattern = @"^(?<dir>[\-ld])(?<permission>([\-r][\-w][\-xs]){3})\s+(?<filecode>\d+)\s+(?<owner>\S+)\s+(?<group>\S+)\s+(?<size>\d+)\s+(?<timestamp>((?<month>\w{3})\s+(?<day>\d{1,2})\s+(?<hour>\d{1,2}):(?<minute>\d{2}))|((?<month>\w{3})\s+(?<day>\d{1,2})\s+(?<year>\d{4})))\s+(?<name>.+)$";

Regex regexForUnix = **new** Regex(pattern, RegexOptions.Compiled);

List<FtpFileInfo> list = **new** List<FtpFileInfo>();

**foreach** (**var** file **in** files)

{

FtpFileInfo remoteFile = **new** FtpFileInfo

{

Size = "",

ModifiedTime = "",

IsDirectory = **false**

};

**string**[] groups = file.Split(**new** **string**[] { " " }, StringSplitOptions.RemoveEmptyEntries);

**if** (groups.Length < 4) **return** **null**;

**try**

{

remoteFile.ModifiedTime = groups[0] + " " + groups[1];

**if** (groups[2] == "<DIR>") remoteFile.IsDirectory = **true**;

**else** remoteFile.Size = SizeParseToString(**long**.Parse(groups[2]));

remoteFile.Name = groups[3];

**for** (**int** i = 4; i < groups.Length; i++) remoteFile.Name += " " + groups[i];

}

**catch** (Exception)

{

**return** **null**;

}

list.Add(remoteFile);

}

**return** list;

}

#endregion

#region SizeParseToString

**private** **string** SizeParseToString(**long** size)

{

**if** (size < 1024) **return** size.ToString() + " Bytes";

**else** if (size >= 1024 && size < 1024 \* 1024) **return** (size / 1024).ToString() + " KB";

**else** if (size >= 1024 \* 1024 && size < 1024 \* 1024 \* 1024) **return** (size / 1024 / 1024).ToString() + " MB";

**else** return (size / 1024 / 1024 / 1024).ToString() + " GB";

}

#endregion

#endregion

}

}

## Mainform.cs

**using** **System**;

**using** **System.Collections.Generic**;

**using** **System.ComponentModel**;

**using** **System.Data**;

**using** **System.Diagnostics**;

**using** **System.Drawing**;

**using** **System.IO**;

**using** **System.Linq**;

**using** **System.Text**;

**using** **System.Windows.Forms**;

**using** **System.Text.RegularExpressions**;

**using** **FTP\_client.Code**;

**using** **Microsoft.VisualBasic**;

**namespace** **FTP\_client**

{

**public** **partial** **class** Mainform : Form

{

**public** Mainform()

{

InitializeComponent();

listBox\_Ftp\_server.DrawItem += listBox\_Ftp\_server\_DrawItem;

}

FtpOperation \_ftpc = **new** FtpOperation();

**private** **void** MainForm\_Load(**object** sender, EventArgs e)

{

InitTreeView();

}

#region InitTreeView 初始化本地磁盘信息

**private** **void** InitTreeView()

{

**string**[] drives = Environment.GetLogicalDrives();

**int** i = 0;

**foreach** (**string** drive **in** drives)

{

DriveInfo d = **new** DriveInfo(drive);

**if** ((d.DriveType & DriveType.Fixed) == DriveType.Fixed)

{

**string** drive1 = drive.Substring(0, drive.Length - 1);

**this**.treeLocal.Nodes[0].Nodes.Add(drive1);

**this**.treeLocal.Nodes[0].Nodes[i].ImageIndex = 1;

**this**.treeLocal.Nodes[0].Nodes[i].SelectedImageIndex = 1;

**this**.treeLocal.Nodes[0].Nodes[i].Tag = drive1;

**this**.treeLocal.Nodes[0].Nodes[i].Nodes.Add("");

i++;

}

}

}

#endregion

#region treeLocal\_BeforeExpand 各磁盘下文件的展示

**private** **void** treeLocal\_BeforeExpand(**object** sender, TreeViewCancelEventArgs e)

{

**if** (e.Node.Level > 0)

{

*//点击之前，先填充节点：*

**string** path = e.Node.FullPath.Substring(e.Node.FullPath.IndexOf("\\") + 1) + "\\";

e.Node.Nodes.Clear();

**string**[] files = Directory.GetFiles(path, "\*", SearchOption.TopDirectoryOnly);

**int** i = 0;

**foreach** (**string** file **in** files)

{

FileInfo f = **new** FileInfo(file);

**if** ((f.Attributes & FileAttributes.Hidden) != FileAttributes.Hidden && (f.Attributes & FileAttributes.System) != FileAttributes.System)

{

e.Node.Nodes.Add(Path.GetFileName(file));

e.Node.Nodes[i].ImageIndex = 3;

e.Node.Nodes[i].SelectedImageIndex = 3;

e.Node.Nodes[i].Tag = file;

i++;

}

}

**string**[] dirs = Directory.GetDirectories(path, "\*", SearchOption.TopDirectoryOnly);

**foreach** (**string** dir **in** dirs)

{

DirectoryInfo d = **new** DirectoryInfo(dir);

**if** ((d.Attributes & FileAttributes.Hidden) != FileAttributes.Hidden && (d.Attributes & FileAttributes.System) != FileAttributes.System)

{

e.Node.Nodes.Add(Path.GetFileName(dir));

e.Node.Nodes[i].ImageIndex = 2;

e.Node.Nodes[i].SelectedImageIndex = 2;

e.Node.Nodes[i].Tag = dir;

e.Node.Nodes[i].Nodes.Add("");

i++;

}

}

}

}

#endregion

#region btn\_connect\_click 连接事件

**private** **void** btn\_connect\_click(**object** sender, EventArgs e)

{

**try**

{

**if** (button\_connect.Text == " 退出")

{

\_ftpc.AllClose();

listBox\_Ftp\_server.Items.Clear();

listBox\_log.Items.Clear();

listBox\_log.Items.Add("已退出连接");

}

**else**

{

\_ftpc.Connect(textBox\_server.Text);

\_ftpc.Identify(textBox\_user.Text, textBox\_password.Text);

listBox\_log.Items.Add("已连接");

List<FtpFileInfo> list = \_ftpc.FtpFileList();

**if** (list == **null**)

Console.WriteLine("error");

**foreach** (FtpFileInfo file **in** list)

{

listBox\_Ftp\_server.DrawMode = DrawMode.OwnerDrawFixed;

AddFileItems(list);

}

button\_connect.Text = " 退出";

}

}

**catch**(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

#endregion

#region btn\_upload\_click 上传事件

**private** **void** btn\_upload\_click(**object** sender, EventArgs e)

{

**try**

{

\_ftpc.UpLoadFile(treeLocal.SelectedNode.Text,

treeLocal.SelectedNode.FullPath.Substring(treeLocal.SelectedNode.FullPath.IndexOf("\\") + 1), 0);

listBox\_log.Items.Add("上传完成：" + treeLocal.SelectedNode.Text);

}

**catch** (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

#endregion

#region GetFileName 获取文件名

**private** **string** GetFileName(**string** raw)

{

**var** strings = Regex.Split(raw, "(\\d)+\\sKB");

**return** strings[0].Trim();

}

#endregion

#region btn\_download\_click 下载事件

**private** **void** btn\_download\_click(**object** sender, EventArgs e)

{

FolderBrowserDialog fbd = **new** FolderBrowserDialog();

**if** (fbd.ShowDialog() == DialogResult.OK)

{

**var** localPath = fbd.SelectedPath;

listBox\_log.Items.Add("选择的下载目录：" + localPath);

FtpFileInfo ftpFileInfo = (FtpFileInfo)listBox\_Ftp\_server.Items[listBox\_Ftp\_server.SelectedIndex];

**string** fileName = ftpFileInfo.Name;

\_ftpc.DownloadFile(fileName, localPath + "\\" + fileName, 0);

listBox\_log.Items.Add("下载完成" + fileName);

}

}

#endregion

#region btn\_refresh\_click 刷新服务器端

**private** **void** btn\_refresh\_click(**object** sender, EventArgs e)

{

List<FtpFileInfo> list = \_ftpc.FtpFileList();

AddFileItems(list);

listBox\_log.Items.Add("已刷新服务器文件列表");

}

#endregion

#region listBox\_Ftp\_server\_DrawItem 服务器列表图标绘制

**private** **void** listBox\_Ftp\_server\_DrawItem(**object** sender, DrawItemEventArgs e)

{

Brush myBrush = Brushes.Black;

Color RowBackColorSel = Color.FromArgb(150, 200, 250);*//选择项目颜色*

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

{

myBrush = **new** SolidBrush(RowBackColorSel);

}

**else**

{

myBrush = **new** SolidBrush(Color.White);

}

e.Graphics.FillRectangle(myBrush, e.Bounds);

e.DrawFocusRectangle();*//焦点框*

*//绘制图标*

Image image = imageList2.Images[e.Index];

Graphics graphics = e.Graphics;

Rectangle bound = e.Bounds;

Rectangle imgRec = **new** Rectangle(

bound.X,

bound.Y,

bound.Height,

bound.Height);

Rectangle textRec = **new** Rectangle(

imgRec.Right,

bound.Y,

bound.Width - imgRec.Right,

bound.Height);

**if** (image != **null**)

{

e.Graphics.DrawImage(

image,

imgRec,

0,

0,

image.Width,

image.Height,

GraphicsUnit.Pixel);

*//绘制字体*

StringFormat stringFormat = **new** StringFormat();

stringFormat.Alignment = StringAlignment.Near;

e.Graphics.DrawString(listBox\_Ftp\_server.Items[e.Index].ToString(), e.Font, **new** SolidBrush(Color.Black), textRec, stringFormat);

}

}

#endregion

#region AddFileItems 服务器列表添加

**private** **void** AddFileItems(List<FtpFileInfo> list)

{

listBox\_Ftp\_server.Items.Clear();

listBox\_Ftp\_server.Items.Add(**new** FtpFileInfo

{

IsDirectory = **true**,

ModifiedTime = "",

Name = "..",

Size = ""

});

listBox\_Ftp\_server.Items.AddRange(list.ToArray());

}

#endregion

#region btn\_pause\_click 断点、续传事件

**private** **void** btn\_pause\_click(**object** sender, EventArgs e)

{

**try**

{

; **string** breakpoint;

**if** (treeLocal.SelectedNode.Text != **null**)*//上传*

{

**if** (button\_pause.Text == " 续传")

{

\_ftpc.BreakPointUploadFile(treeLocal.SelectedNode.Text,

treeLocal.SelectedNode.FullPath.Substring(treeLocal.SelectedNode.FullPath.IndexOf("\\") + 1), 0);

listBox\_log.Items.Add(treeLocal.SelectedNode.Text + "上传完成");

}

**else**

{

breakpoint = Microsoft.VisualBasic.Interaction.InputBox("输入断点值", "断点续传", "", -1, -1);

listBox\_log.Items.Add(treeLocal.SelectedNode.Text + "上传中断");

listBox\_log.Items.Add("断点：" + breakpoint + " Bytes");

**int** breakpointint = **int**.Parse(breakpoint);

\_ftpc.UpLoadFile(treeLocal.SelectedNode.Text,

treeLocal.SelectedNode.FullPath.Substring(treeLocal.SelectedNode.FullPath.IndexOf("\\") + 1), breakpointint);

button\_pause.Text = " 续传";

button\_pause.Image = Image.FromFile(@"C: \Users\xy\source\repos\FTP\_client\Image\player\_play\_pause\_16px\_1222646\_easyicon.net.ico");

}

}

**else***//下载*

{

**if** (button\_pause.Text == " 续传")

{

FolderBrowserDialog fbd = **new** FolderBrowserDialog();

**if** (fbd.ShowDialog() == DialogResult.OK)

{

**var** localPath = fbd.SelectedPath;

listBox\_log.Items.Add("选择的下载目录：" + localPath);

FtpFileInfo ftpFileInfo = (FtpFileInfo)listBox\_Ftp\_server.Items[listBox\_Ftp\_server.SelectedIndex];

**string** fileName = ftpFileInfo.Name;

\_ftpc.DownloadFile(fileName, localPath + "\\" + fileName, 0);

listBox\_log.Items.Add("下载完成" + fileName);

}

}

**else**

{

breakpoint = Microsoft.VisualBasic.Interaction.InputBox("输入断点值", "断点续传", "", -1, -1);

listBox\_log.Items.Add(treeLocal.SelectedNode.Text + "上传中断");

listBox\_log.Items.Add("断点：" + breakpoint + " Bytes");

**int** breakpointint = **int**.Parse(breakpoint);

FolderBrowserDialog fbd = **new** FolderBrowserDialog();

**if** (fbd.ShowDialog() == DialogResult.OK)

{

**var** localPath = fbd.SelectedPath;

listBox\_log.Items.Add("选择的下载目录：" + localPath);

FtpFileInfo ftpFileInfo = (FtpFileInfo)listBox\_Ftp\_server.Items[listBox\_Ftp\_server.SelectedIndex];

**string** fileName = ftpFileInfo.Name;

\_ftpc.DownloadFile(fileName, localPath + "\\" + fileName, breakpointint);

listBox\_log.Items.Add("下载完成" + fileName);

button\_pause.Text = " 续传";

button\_pause.Image = Image.FromFile(@"C: \Users\xy\source\repos\FTP\_client\Image\player\_play\_pause\_16px\_1222646\_easyicon.net.ico");

}

}

}

**catch**(Exception ex)

{

MessageBox.Show(ex.Message);

}

}

#endregion

#region listBox\_Ftp\_server\_DoubleClick 双击进入服务器端文件夹

**private** **void** listBox\_Ftp\_server\_DoubleClick(**object** sender, EventArgs e)

{

**var** listBoxFtpSever = (ListBox)sender;

FtpFileInfo selectedItem = listBoxFtpSever.SelectedItem **as** FtpFileInfo;

**bool** isDirectory = selectedItem.IsDirectory;

*//if it is a directory*

Debug.WriteLine(selectedItem.IsDirectory);

**if** (isDirectory)

{

\_ftpc.ChangeDirectory(selectedItem.Name);

}

List<FtpFileInfo> list = \_ftpc.FtpFileList();

AddFileItems(list);

}

#endregion

}

}