# 目 录

**[引 言](#_Toc7919_WPSOffice_Level1)** **[1](#_Toc7919_WPSOffice_Level1)**

**[第1章 需求分析](#_Toc16356_WPSOffice_Level1)** **[2](#_Toc16356_WPSOffice_Level1)**

[1.1课题背景](#_Toc16356_WPSOffice_Level2) [2](#_Toc16356_WPSOffice_Level2)

[1.2工作原理](#_Toc14471_WPSOffice_Level2) [2](#_Toc14471_WPSOffice_Level2)

[1.3需求分析](#_Toc19626_WPSOffice_Level2) [2](#_Toc19626_WPSOffice_Level2)

[1.4用例图](#_Toc16648_WPSOffice_Level2) [3](#_Toc16648_WPSOffice_Level2)

[1.5系统功能](#_Toc4299_WPSOffice_Level2) [3](#_Toc4299_WPSOffice_Level2)

**[第2章 概要设计](#_Toc14471_WPSOffice_Level1)** **[5](#_Toc14471_WPSOffice_Level1)**

[2.1文件属性](#_Toc7514_WPSOffice_Level2) [5](#_Toc7514_WPSOffice_Level2)

[2.2实现方法](#_Toc15576_WPSOffice_Level2) [5](#_Toc15576_WPSOffice_Level2)

[2.2开发环境](#_Toc9416_WPSOffice_Level2) [5](#_Toc9416_WPSOffice_Level2)

[2.3运行环境](#_Toc6124_WPSOffice_Level2) [5](#_Toc6124_WPSOffice_Level2)

**[第3章 详细设计](#_Toc19626_WPSOffice_Level1)** **[6](#_Toc19626_WPSOffice_Level1)**

[3.1登录连接](#_Toc792_WPSOffice_Level2) [6](#_Toc792_WPSOffice_Level2)

[3.2返回目录](#_Toc25468_WPSOffice_Level2) [6](#_Toc25468_WPSOffice_Level2)

[3.3解析数据端口号](#_Toc19850_WPSOffice_Level2) [6](#_Toc19850_WPSOffice_Level2)

[3.4返回文件列表](#_Toc23265_WPSOffice_Level2) [7](#_Toc23265_WPSOffice_Level2)

[3.5下载](#_Toc18627_WPSOffice_Level2) [8](#_Toc18627_WPSOffice_Level2)

[3.6上传](#_Toc5147_WPSOffice_Level2) [9](#_Toc5147_WPSOffice_Level2)

[3.7重命名](#_Toc12042_WPSOffice_Level2) [9](#_Toc12042_WPSOffice_Level2)

[3.8删除](#_Toc29580_WPSOffice_Level2) [10](#_Toc29580_WPSOffice_Level2)

[3.9新建文件夹](#_Toc390_WPSOffice_Level2) [10](#_Toc390_WPSOffice_Level2)

[3.10移动文件夹](#_Toc12999_WPSOffice_Level2) [10](#_Toc12999_WPSOffice_Level2)

[3.11退出](#_Toc1122_WPSOffice_Level2) [10](#_Toc1122_WPSOffice_Level2)

[3.12发送数据](#_Toc15018_WPSOffice_Level2) [10](#_Toc15018_WPSOffice_Level2)

[3.13读取](#_Toc17655_WPSOffice_Level2) [11](#_Toc17655_WPSOffice_Level2)

**[第四章 运行截图](#_Toc16648_WPSOffice_Level1)** **[12](#_Toc16648_WPSOffice_Level1)**

[4.1登录](#_Toc21329_WPSOffice_Level2) [12](#_Toc21329_WPSOffice_Level2)

[4.2打开文件夹](#_Toc17011_WPSOffice_Level2) [12](#_Toc17011_WPSOffice_Level2)

[4.3上传文件](#_Toc171_WPSOffice_Level2) [13](#_Toc171_WPSOffice_Level2)

[4.4删除文件](#_Toc21657_WPSOffice_Level2) [13](#_Toc21657_WPSOffice_Level2)

[5.5下载文件](#_Toc22715_WPSOffice_Level2) [14](#_Toc22715_WPSOffice_Level2)

[5.6刷新文件列表](#_Toc2438_WPSOffice_Level2) [14](#_Toc2438_WPSOffice_Level2)

[5.7修改文件名](#_Toc18626_WPSOffice_Level2) [14](#_Toc18626_WPSOffice_Level2)

[5.8新建文件夹](#_Toc15013_WPSOffice_Level2) [15](#_Toc15013_WPSOffice_Level2)

**[总结](#_Toc4299_WPSOffice_Level1)** **[16](#_Toc4299_WPSOffice_Level1)**

**[参考文献](#_Toc7514_WPSOffice_Level1)** **[17](#_Toc7514_WPSOffice_Level1)**

**[附录：](#_Toc15576_WPSOffice_Level1)** **[18](#_Toc15576_WPSOffice_Level1)**

# 引 言

随着互联网技术的飞速发展,信息共享的时代已经到来，不再是从前那种把资料全部保存在本地文件夹，而是把数据集中的保存在一个数据中心，根据需求进行下载和上传，FTP就是实现这一功能的一个重要部分，现在FTP已成为数据传输与信息共享的诸多方式中最广的一种方式。FTP协议是计算机网络中的应用层协议,提供了可靠的文件传输功能,并且具有跨平台特性,两个不同类型的操作系统只要都支持FTP协议就可以进行文件传输工作。此课程设计的目标是做一个FTP客户端，用户能通过一个支持FTP协议的客户机程序，连接到在远程主机上的FTP 服务器程序，然后发出命令，服务器程序执行，并将执行的结果返回到客户机，实现用户通过客户机程序向(从) 远程主机上载(下载)文件的功能。

第1章 需求分析

需求分析是软件计划阶段的重要活动，也是软件生存周期中的一个重要环节，该阶段是分析系统在功能上需要“实现什么”，而不是考虑如何去“实现”。需求分析的目标是把用户对待开发软件提出的“要求”或“需要”进行分析与整理，确认后形成描述完整、清晰与规范的文档，确定软件需要实现哪些功能，完成哪些工作。

1.1课题背景

随着计算机技术和底层芯片技术的飞速发展，网络技术得到了质的飞跃，成为了现代互联网不可或缺的一部分，在这个数据爆炸的互联网时代，本地的数据存储已经无法满足人们的工作需求，出现了集中式的数据中心，FTP的出现就是为了能够从远程的数据中心服务器下载需要的数据和上传需要上传的数据。

FTP工作在TCP/IP模型的应用层，基于的传输协议是TCP，FTP客户端和服务器之间的连接是可靠的，面向连接的，为数据的传输提供了可靠的保证。

1.2工作原理

FTP使用客户端/服务器模式，涉及到五个进程和两个相关。服务器首先运行一个本地FTP主服务进程，当客户端用户输入open或ftp命令后，客户端指定一个远程计算机时，建立一个客户端控制进程，本地应用成为一个FTP客户端，建立了与自己的半相关。然后，客户端用TCP建立与指定计算机上的FTP服务器之间的控制连接。服务器接收到客户端的请求后，派生一个服务器控制进程，该进程与客户端控制进程建立起控制连接，即相关，双方进入会话状态。 客户端要进行数据传输时，派生一个客户端数据传输进程，并在一个非默端口上被动打开，通过控制连接，客户端用port将端口号发送给服务器控制进程。服务器控制进程接收到后，派生一个服务器数据传输进程，该进程建立一个自己的半相关（20），并向客户端数据进程发出连接请求，与客户端数据传输进程建立相关，双方建立数据传输。数据传输结束后，关闭数据连接，双方数据传输进程结束，但控制进程仍然存在，然后可以再重新发起新的数据传输服务。

1.3需求分析

**登录功能：**

能够根据服务器的IP地址，端口号，账户和密码远程连接服务器；

**文件管理功能：**

能够浏览登陆账户文件夹下的所有文件；

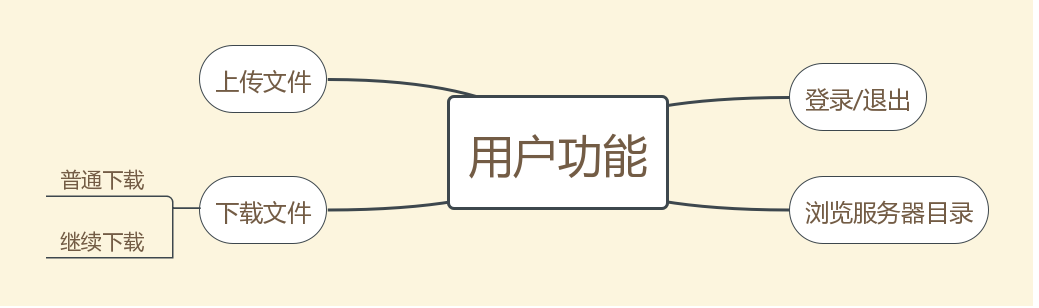
能够实现刷新服务端的文件列表；

能够实现上传本地文件到指定的服务器文件路径下；

实现从服务器下载指定文件到本地客户端；

1.4用例图

根据对用户需求，用例图如下：



1.5系统功能

实现类似于XFTP的FTP客户端，完成客户端和服务端的远程连接，实现对本地和远程文件的浏览、刷新、上传、下载等。

# 第2章 概要设计

概要设计是一个设计师根据用户交互过程和用户需求来形成交互框架和视觉框架的过程，其结果往往以反映交互控件布置、界面元素分组以及界面整体板式的页面框架图的形式来呈现。这是一个在用户研究和设计之间架起桥梁，使用户研究和设计无缝结合，将对用户目标与需求转换成具体界面设计解决方案的重要阶段。

2.1文件属性

服务端的文件大致分为文件和文件夹两种属性，在文件列表中，我们舍弃了对文件夹的显示，将文件以相对路径的形式体现了文件的归属。

2.2实现方法

2.2.1服务器连接

远程连接服务器，需要根据发生登录请求指令和根据返回的响应码，再次发生需要的指令和指令内容完成和服务端的指令交互，登陆部分需要在和服务端的第一次连接完成socket的建立，并设置读取和写入缓冲区，用于数据的读写。

2.2.2数据发送

根据用户不同的请求，把指令内容和相对于的需要发送到服务端的指令拼接并假如”\r\n”结束指令，发送到服务端。

2.2.3数据端口解析

解析服务端返回的数据端口的端口号响应码，解析出正确的端口号，客户端根据这个端口号建立于服务端的连接。

2.2开发环境

Windows +Pycharm+python3.6

2.3运行环境

Python3.6, PyQT5

# 第3章 详细设计

3.1定义连接对象

**class** FTPClient:

**# 初始化**  
 **def** \_\_init\_\_(self, host, user=**'IUSR'**, passwd=**'123'**):  
 self.ftp = FTP()  
 self.ftp.connect(host, port=21)  
 self.ftp.login(user,passwd)  
 self.ftp.encoding = **'gbk'  
 # 发送指令  
 def** sendCmd(self, cmd):  
 **return** self.ftp.sendcmd(cmd)  
 **# 返回文件大小**

**def** size(self, filename):  
 resp = self.ftp.sendcmd(**'SIZE '** + filename)  
 s = -1  
 **if** resp[:3] == **'213'**:  
 s = resp[3:].strip()  
 **return** int(s)  
 **# 得到文件夹详细目录**  
 **def** getDir(self):  
 **return** self.ftp.dir()  
 **# 得到当前路径目录**  
 **def** getList(self):  
 **return** self.ftp.nlst() *# retrlines('LIST')* **# 下载****def** download(self, name, path):  
 **if** os.path.exists(path):  
 lsize = os.path.getsize(path)  
 fsize = self.size(name)  
 **if** lsize >= fsize:  
 **return**

**# 断点重连** blocksize = 1024 \* 1024  
 cmpsize = lsize  
 **try**:  
 self.ftp.sendcmd(**'TYPE I'**)  
 lwrite = open(path, **'ab'**)  
 print(name)  
 conn = self.ftp.retrbinary(**'RETR {}'**.format(name), lwrite, lsize)  
  
 **while True**:  
 data = conn.recv(blocksize)  
 **if not** data:  
 **break** lwrite.write(data)  
 cmpsize += len(data)  
 *# print '\b'\*30,'download process:%.2f%%'%(float(cmpsize)/fsize\*100),* print(**'download process:%.2f%%'** % (float(cmpsize) / fsize \* 100))  
 lwrite.close()  
 self.ftp.voidcmd(**'NOOP'**)  
 self.ftp.voidresp()  
 conn.close()  
 *# self.ftp.quit()* **return True  
 except** Exception **as** e:  
 print(e)  
 **return False  
 else**:**# 常规下载**  
 f = open(path, **'wb'**)  
 conn = self.ftp.retrbinary(**'RETR {}'**.format(name), f.write)  
  
 **# 文件上传**  
 **def** upload(self, path):  
 **try**:  
 name = path.split(**'/'**)[-1]  
 **except**:  
 name = path  
 print(name)  
 print(path)  
 **try**:  
 self.ftp.storbinary(**'STOR {}'**.format(name), open(path, **'rb'**))  
 **return True  
 except** Exception **as** e:  
 print(e)  
 **return False** *# 更改名字* **def** rename(self, oldname, newname):  
 back = **None  
 try**:  
 back = self.ftp.rename(oldname, newname)  
 **except** Exception **as** e:  
 print(e)  
 **return** back  
 **# 删除文件**  
 **def** delete\_file(self, filename):  
 back = **None  
 try**:  
 back = self.ftp.delete(filename)  
 **except** Exception **as** e:  
 print(e)  
 **return** back  
 **# 排序好文件目录**  
 **def** sort\_files(self):  
 files = []  
 fileList = self.getList()  
 **for** folder **in** fileList:  
 **if** folder.find(**"."**) == -1:  
 n = folder  
 fileList.remove(folder)  
 fileList.append(n)  
 **for** folder **in** fileList:  
 **if** folder.find(**"."**) != -1:  
 self.ftp.cwd(**"../"**)  
 files.append((folder, self.size(folder)))  
 *# pass* **else**:  
 self.ftp.cwd(**"../"**)  
 self.ftp.cwd(folder)  
 tempList = self.getList()  
 **for** temp **in** tempList:  
 files.append((folder + **'/'** + temp, self.size(temp)))  
 self.ftp.cwd(**"../"**)  
 **return** files

3.2返回目录

**# 得到文件夹详细目录**  
 **def** getDir(self):  
 **return** self.ftp.dir()  
 **# 得到当前路径目录**  
 **def** getList(self):  
 **return** self.ftp.nlst() *# retrlines('LIST')*

3.3返回文件列表

1. 将文件按照文件夹的顺序排序

**# 排序好文件目录**  
 **def** sort\_files(self):  
 files = []  
 fileList = self.getList()  
 **for** folder **in** fileList:  
 **if** folder.find(**"."**) == -1:  
 n = folder  
 fileList.remove(folder)  
 fileList.append(n)  
 **for** folder **in** fileList:  
 **if** folder.find(**"."**) != -1:  
 self.ftp.cwd(**"../"**)  
 files.append((folder, self.size(folder)))  
 *# pass* **else**:  
 self.ftp.cwd(**"../"**)  
 self.ftp.cwd(folder)  
 tempList = self.getList()  
 **for** temp **in** tempList:  
 files.append((folder + **'/'** + temp, self.size(temp)))  
 self.ftp.cwd(**"../"**)  
 **return** files

② 将文件按照文件夹的顺序排序，按照文件名、状态、进度、大小的格式将文件打印输出

**def** getRemoteList(self):  
 **try**:  
 **if** self.FTP:  
 files = self.FTP.sort\_files()  
 i = 0  
 self.file\_length = len(files)  
 **for** f **in** files:  
 ck = QCheckBox()  
 *# self.ck.stateChanged.connect(self.changeCk)* h = QHBoxLayout()  
 h.setAlignment(Qt.AlignCenter)  
 h.addWidget(ck)  
 textbox = QLineEdit()  
 textbox.setText(f[0])  
 **try**:  
 filename = f[0].split()[-1]  
 **except**:  
 filename = f[0]  
 self.tableWidget.setCellWidget(i, 0, ck)  
 self.tableWidget.setCellWidget(i, 1, textbox)  
 textbox1 = QLineEdit()  
 **if** os.path.exists(os.path.join(self.baseSavePath, filename)):  
 **if** os.path.getsize(os.path.join(self.baseSavePath, filename)) >= f[1]:  
 textbox1.setText(**"已下载"**)  
 **else**:  
 textbox1.setText(**"下载中，可断点重传"**)  
 **else**:  
 textbox1.setText(**"未下载"**)  
 textbox2 = QLineEdit()  
 textbox2.setText(**"zero"**)  
 textbox3 = QLineEdit()  
 textbox3.setText(str(f[1]))  
 self.tableWidget.setCellWidget(i, 2, textbox1)  
 self.tableWidget.setCellWidget(i, 3, textbox2)  
 self.tableWidget.setCellWidget(i, 4, textbox3)  
 i += 1  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget.setHorizontalHeaderItem(i - 1, item)  
 self.MessageList.addItem(**"已成功获取到List..."**)  
 **else**:  
 msg = QMessageBox()  
 msg.setWindowTitle(**"Warning"**)  
 msg.setStandardButtons(QMessageBox.Ok | QMessageBox.Cancel)  
 msg.setText(**"请先连接..."**)  
 self.MessageList.addItem(**"请先连接..."**)  
 btn = msg.exec()  
 **except** Exception **as** e:  
 print(e)

3.4下载

**def** downloadFile(self):  
 info = **"确定下载选择的文件吗？"** msg = QMessageBox()  
 msg.setWindowTitle(**"Warning"**)  
 msg.setStandardButtons(QMessageBox.Ok | QMessageBox.Cancel)  
 msg.setText(info)  
 btn = msg.exec()  
 **if** btn == QMessageBox.Ok:  
 **try**:  
 **for** i **in** range(self.file\_length):  
 **if** self.tableWidget.cellWidget(i, 0).isChecked():  
 filename = self.tableWidget.cellWidget(i, 1).text()  
 **if** filename.find(**"/"**) == -1:  
 self.MessageList.addItem(**"正在下载..."**)  
 *# self.FTP.download(filename, os.path.join(self.baseSavePath, filename))* t = threading.Thread(target=self.FTP.download,  
 args=(filename, os.path.join(self.baseSavePath, filename),))  
 t.start()  
 self.MessageList.addItem(**"下载完成..."**)  
 **else**:  
 s = filename.split()  
 self.FTP.cwd(s[0])  
 *# self.FTP.download(s[-1], os.path.join(self.baseSavePath, s[-1]))* t = threading.Thread(target=self.FTP.download,  
 args=(filename, os.path.join(self.baseSavePath, filename),))  
 t.start()  
  
 **except** Exception **as** e:  
 print(e)

3.5上传

将文件以二进制的IO流通过FTP传输至服务器

**def** UploadFile(self, Qmodelidx):  
 fileWholeName = self.model.filePath(Qmodelidx)  
 fileName = self.model.fileName(Qmodelidx)  
 *# print(self.model.fileInfo(Qmodelidx))* info = **"确定上传文件 {} 吗？"**.format(fileName)  
 detail = **"详情路径为:{}"**.format(fileWholeName)  
 msg = QMessageBox()  
 msg.setWindowTitle(**"Warning"**)  
 msg.setStandardButtons(QMessageBox.Ok | QMessageBox.Cancel)  
 msg.setText(info)  
 msg.setDetailedText(detail)  
 btn = msg.exec()  
 **if** btn == QMessageBox.Ok:  
 t = threading.Thread(target=self.FTP.upload, args=(fileWholeName,))  
 t.start()  
 **else**:  
 **pass**

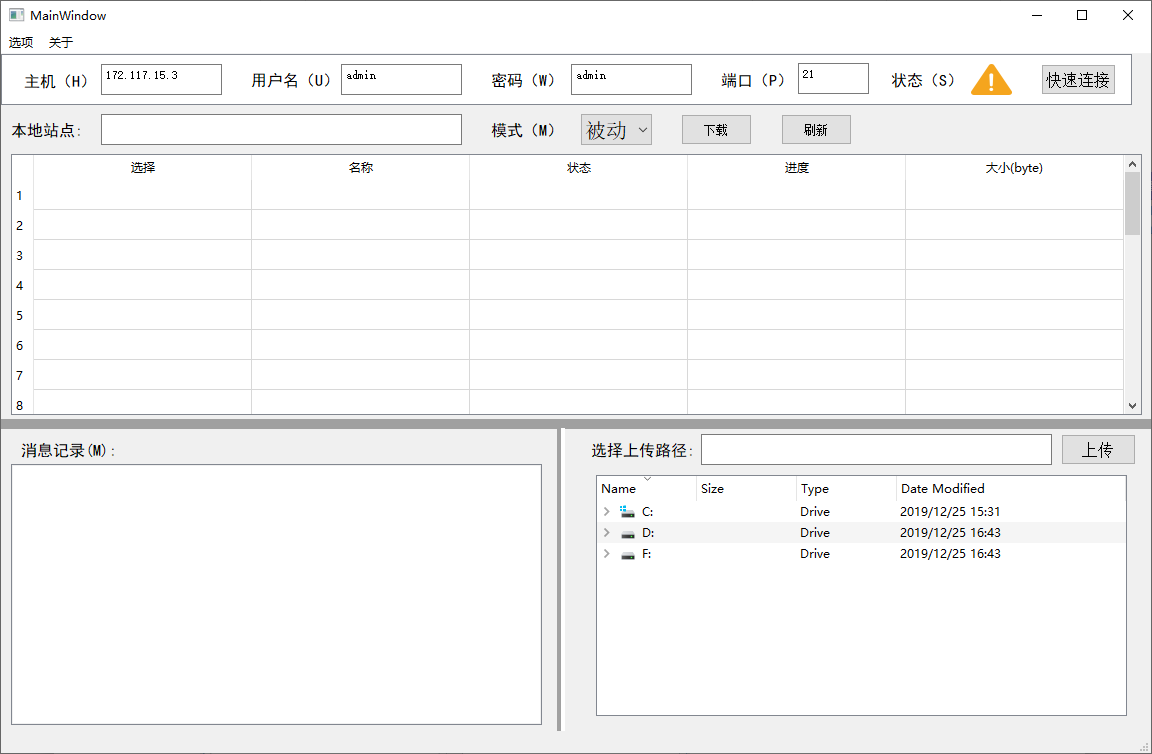
3.6刷新文件列表

**def** refreshList(self):  
 self.tableWidget.clear()  
 self.getRemoteList()

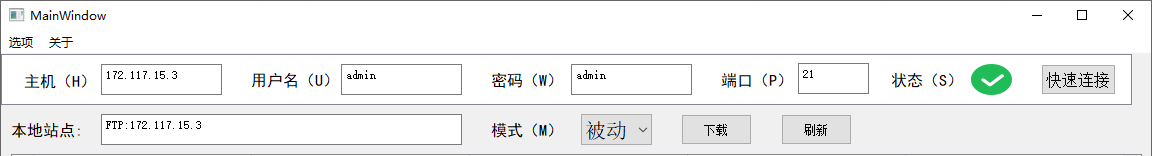
# 第四章 运行截图

4.1登录

输入登录信息，点击快速连接



状态图标为OK时，完成登录



否则，登录失败



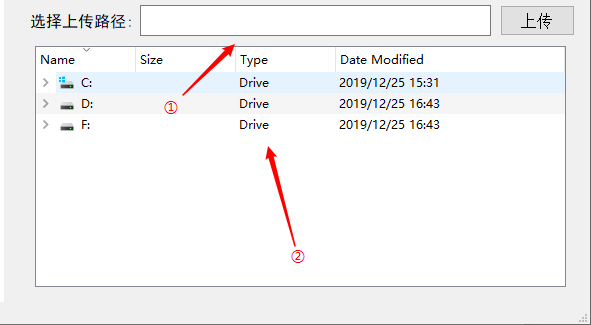
5.2远程文件目录



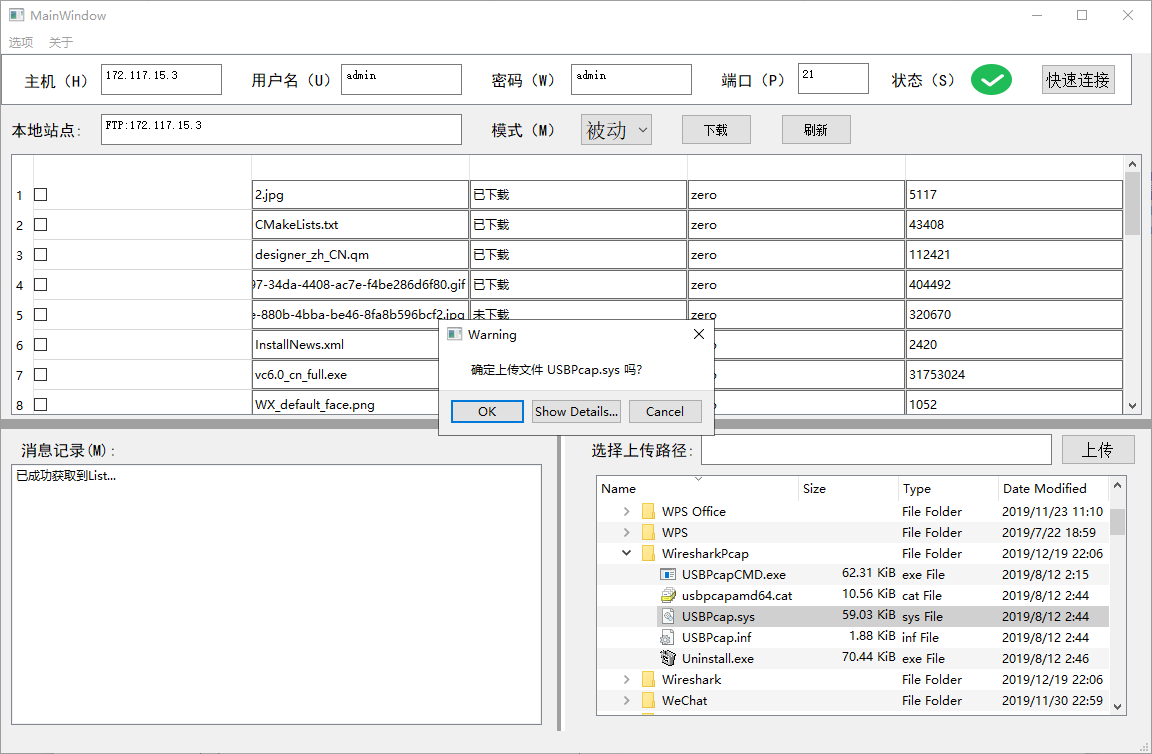
5.3上传文件

①填写文件绝对路径

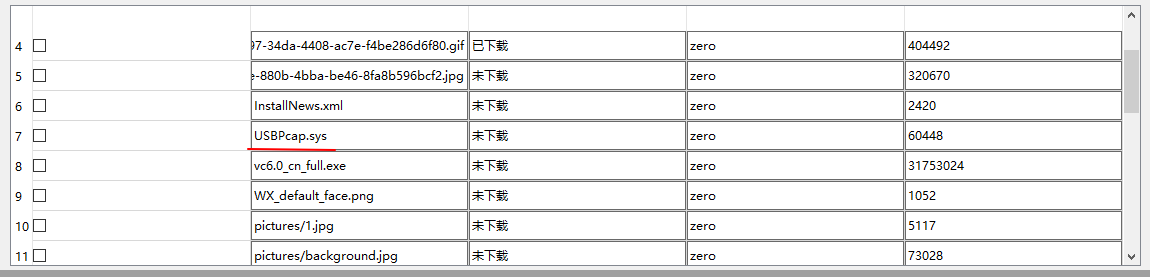
②目录的文件选择



上传选择：

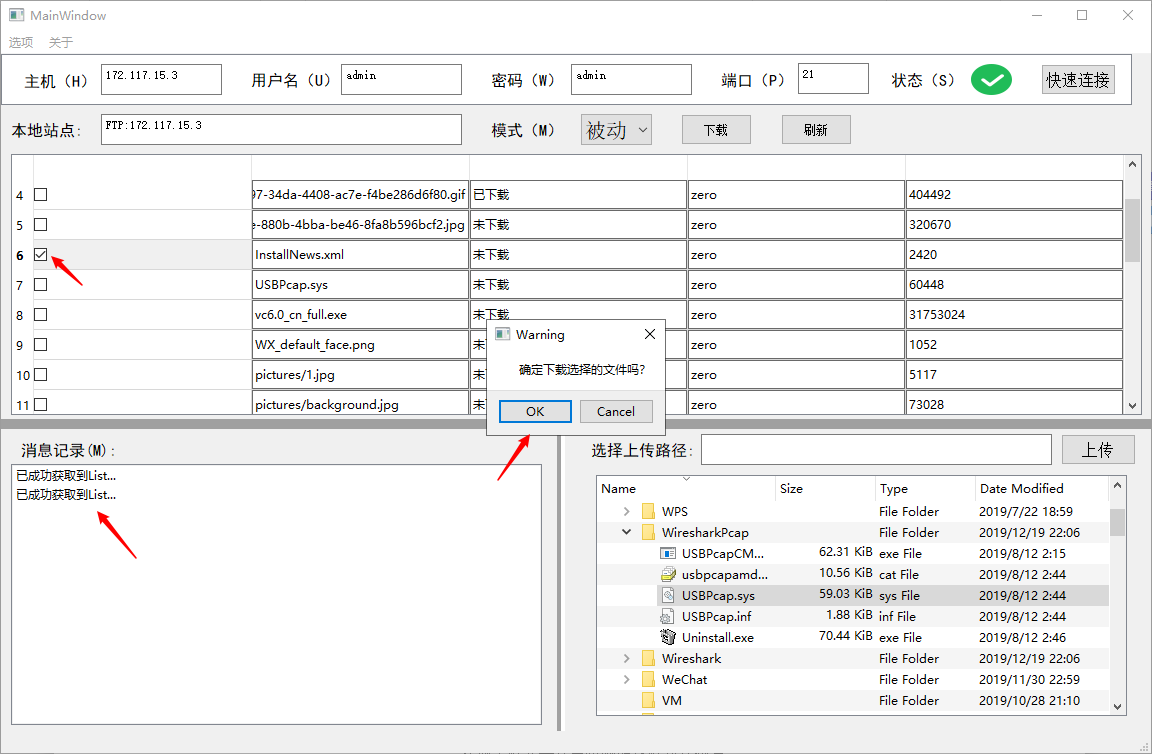


上传成功:

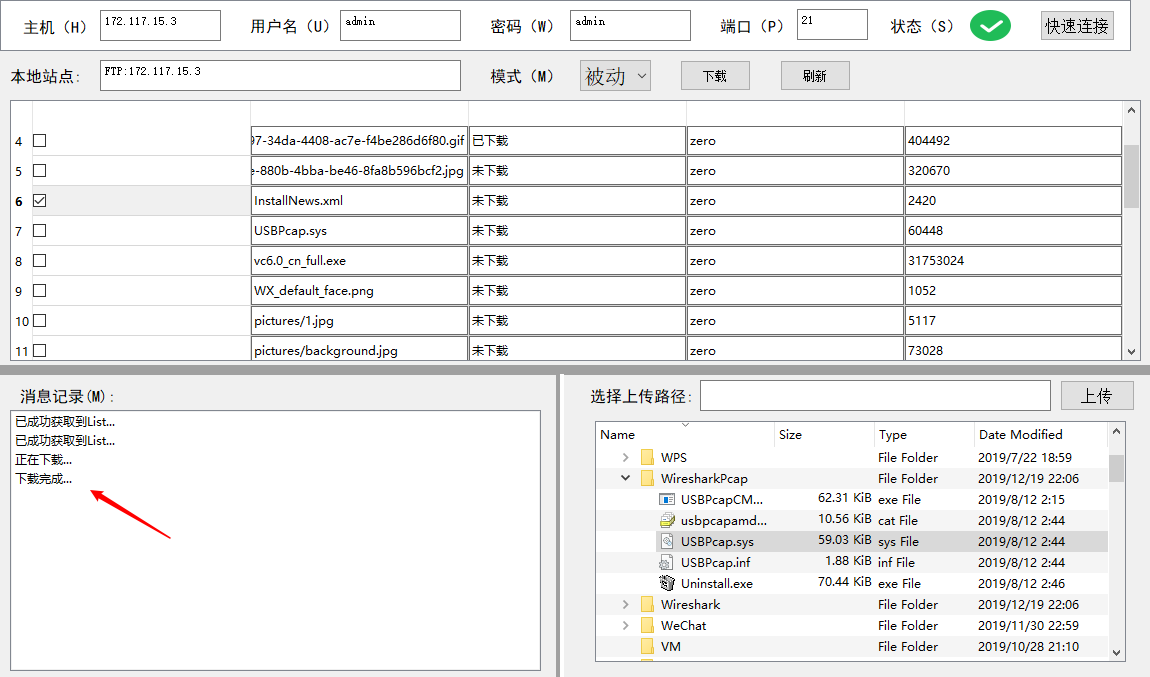


5.4下载文件

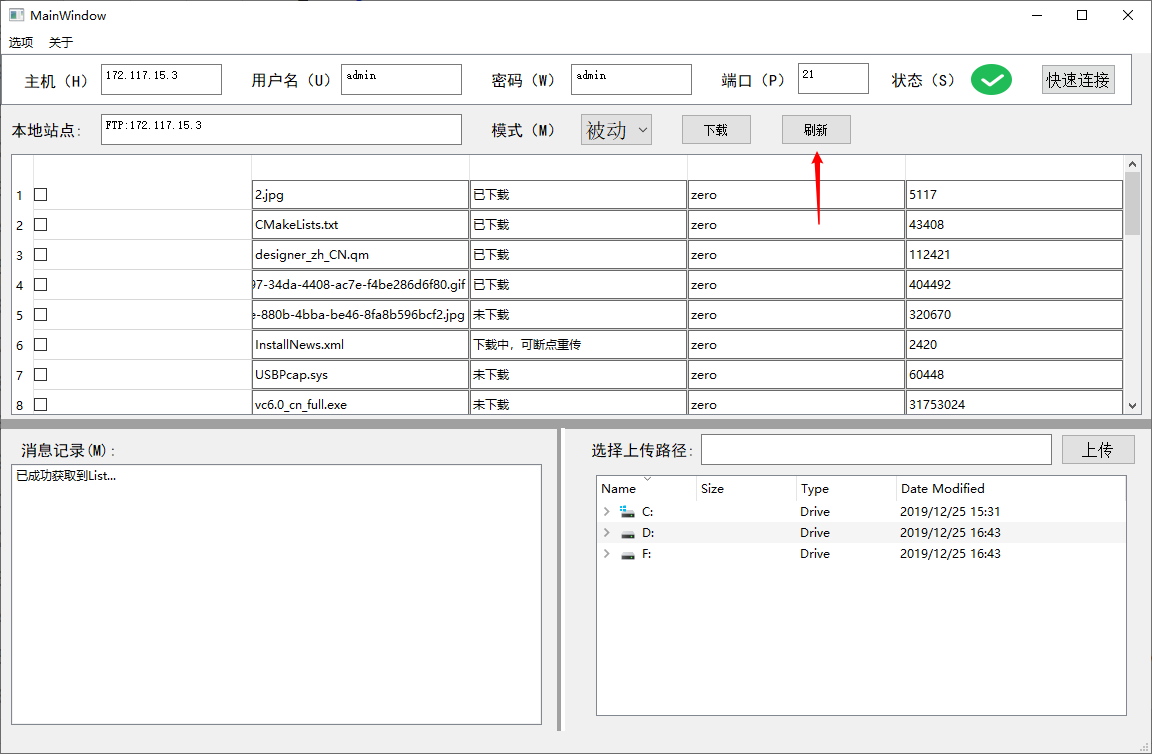
选择下载文件



完成无损下载



5.6刷新文件列表



# 

# 总结

本次课程设计我完成了基于 TCP/IP的FTP文件传输系统，使用python编写的客户端程序，输入搭建在Windows服务端的FTP服务器，完成一系列的请求和功能。本次课程设计前后花费了大概一周的时间，从确定课题，到确定编程和运行环境，在做到一半因为时间问题换客户端的实现方式，到最终完成答辩，是一个收获良多的过程，无论是在对python语言的练习，在对TCP/IP协议的理解，在对FTP服务过程的理解，都有很深入的熟悉。

在实现的过程在遇到了很多的问题，首先就是在编写UI界面的时候，因为对python语言的不熟练，导致很多功能使用起来步骤繁琐，有的甚至没有实现，例如更改服务端的文件地址;其次因为FTP协议需要客户端开启两个端口（命令端口，数据端口）与服务端进行交互，命令端口是固定的端口21，但是数据端口需要根据服务器的返回的响应码解析出数据端口地址，在最初没有理解，导致数据端口持续错误，花费了一些时间；还有在上传和下载这种大数据传输的过程中，发现下载和上传之后的文件损坏，在查看传输以后的图片以后，发现问题所在，改用二进制传输，最终实现了数据的无损传输，还有许许多多小问题，虽然繁琐，但是程序的编写一定会伴随BUG的出现和修改，只有这样才能提升自己的编程能力。

虽然完成了本次课程设计，但是和我最初的设想还有一定的差距，无论是UI设计，功能的可靠性等方面，都没有达到我的要求，在寒假期间我会继续学习，完成整个项目。

# 参考文献

[1]顾彤辰. FTP客户端及服务器系统的设计与实现[D].南京邮电大学,2016.

[2]陈阳,黄宁,康锐,李瑞莹.局域网FTP业务可靠性试验与评估技术[J].北京航空航天大学学报,2011,37(01):91-94+122.

[3]郝浩.FTP原理解析[J].计算机与网络,2016,42(14):40-41.

[4]葛伟伦.FTP协议两种工作方式下消息交互的分析[J].电脑知识与技术,2016,12(08):30-31+35.

[5]易建勋，《计算机网络设计》人民邮电出版社，2007

[6]村山公保 白玉林《TCP/IP计算机网络篇》科学出版社，2005

# 附录：