**武汉大学计算机学院**

**本科生实验报告**

**网络工程与编程实践**

专 业 名 称 ：软件工程

课 程 名 称 ：网络工程与编程实践

团 队 名 称 ：不知道叫什么队

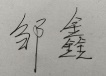
指 导 教 师 ：吕慧 讲师

团 队 成 员 一：邹鑫（2017302580198）

二○二○年三月

**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： 2020.03.15

摘 要

本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FTP（文件传输协议）协议。

实验内容主要包括：

* 实现FTP客户端的基本功能：文件上传和下载
* 实现FTP客户端扩展功能——
  + 当前工作路径的显示
  + 获取当前工作路径下的文件目录列表
  + 实现改变工作路径的功能（即文件夹直接的跳转）
  + 实现删除文件和删除文件夹的功能
  + 实现新建文件夹的功能
  + 实现文件/文件夹重命名的功能
  + 实现文件上传/下载的断点续传功能

**关键词：** FTP；socket；WinSocket；断点续传

**目 录**

[1 实验目的和意义](#_Toc35205471)

[1.1 实验目的 6](#_Toc35205472)

[1.2 实验意义 6](#_Toc35205473)

[2 实验设计](#_Toc35205474)

[2.1 概述 7](#_Toc35205475)

[2.2 实验原理 7](#_Toc35205476)

[2.3 实验基本方案 7](#_Toc35205477)

[2.3.1 Linux服务器上的vsftpd（FTP服务器端）安装与配置[1] 8](#_Toc35205478)

[2.3.2 客户端与服务器建立连接[2] 8](#_Toc35205479)

[2.3.3 客户端登录到服务器 9](#_Toc35205480)

[2.3.4 进入被动模式[4] 10](#_Toc35205481)

[2.3.5 客户端从服务器下载文件 12](#_Toc35205482)

[2.3.6 客户端上传文件到服务器 13](#_Toc35205483)

[2.4 扩展功能 14](#_Toc35205484)

[2.4.1 获取工作路径下的文件列表 15](#_Toc35205485)

[2.4.2 显示当前工作路径 15](#_Toc35205486)

[2.4.3 改变工作路径 15](#_Toc35205487)

[2.4.4 删除文件 15](#_Toc35205488)

[2.4.5 删除文件夹 16](#_Toc35205489)

[2.4.6 新建文件夹 16](#_Toc35205490)

[2.4.7 文件重命名 16](#_Toc35205491)

[2.4.8 上传功能的断点续传实现[5] 16](#_Toc35205492)

[2.4.9 下载功能的断点续传实现 17](#_Toc35205493)

[3 实验中遇到的问题](#_Toc35205494)

[3.1 vsftpd的配置问题 18](#_Toc35205495)

[3.2 文件目录返回结果中判断文件是否是文件夹的问题 18](#_Toc35205496)

[3.3 延迟问题 18](#_Toc35205497)

[4 反思与改进](#_Toc35205498)

**结论** ……………………………………………………………………………… 21

**参考文献 ……………………………………………………………………… 47**

# 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VC++完成FTP客户端系统程序。

**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，结合FTP协议标准，从用户界面设计、接口函数设计，到最终完成系统的编程。本实验将用FTP客户端作为例子，要求实现文件的上传和下载的基本功能，当然，鼓励完成断点续传等扩展功能。

## 1.2 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、开发平台的使用方法等。

# 2 实验设计

## 2.1 概述

FTP（File Transfer Protocol，文件传输协议） 是 TCP/IP 协议组中的协议之一。FTP协议包括两个组成部分，其一为FTP服务器，其二为FTP客户端。其中FTP服务器用来存储文件，用户可以使用FTP客户端通过FTP协议访问位于FTP服务器上的资源。在开发网站的时候，通常利用FTP协议把网页或程序传到Web服务器上。此外，由于FTP传输效率非常高，在网络上传输大的文件时，一般也采用该协议。

本文根据FTP协议制定的标准，进行windows操作系统下的FTP客户端编程，实现了登录、刷新文件列表、新建文件夹、删除文件、文件与文件夹的重命名、上传文件（支持断点续传）、下载文件（支持断点续传）等功能。

## 2.2 实验原理

本系统分为两部分——前端和后端接口。

前端采用的是VC++中的MFC框架，微软基础类库（英语：Microsoft Foundation Classes，简称MFC）是微软公司提供的一个类库（class libraries），以C++类的形式封装了Windows API，并且包含一个应用程序框架，以减少应用程序开发人员的工作量。其中包含大量Windows句柄封装类和很多Windows的内建控件和组件的封装类。

后端则是以WinSocket为基础，根据FTP协议标准规范，简单封装的一个基于FTP协议规范的类库，里面包含了本次FTP客户端所需要的一些函数。而本系统则是通过前端界面调用此FTP函数类库实现的。

## 2.3 实验基本方案

对于本系统的前端，本人认为没有过多可以介绍的东西，故后文的重点将会放在FTP函数类库的基本要求的实现上（当然，运行截图中可以看到前端界面的截图）。

### 2.3.1 Linux服务器上的vsftpd（FTP服务器端）安装与配置[1]

本次实验的主要任务是编程实现FTP客户端，为了便于调试，我们需要一个FTP服务器端。除此之外，我还下载了一个FTP客户端WinSCP进行对照检验。为此，我先到腾讯云购买了服务器学生套餐，拥有了一个属于自己的服务器。然后，就要针对该服务器进行vsftpd的安装与配置了。

首先，打开服务器，在终端输入指令“sudo apt-get install vsftpd”安装vsftpd服务器。安装完成后，进入/etc目录，找到vsftpd.conf文件，修改anonymous\_enable项为true，表示允许匿名登录，然后修改write\_enable项为true，表示允许任何形式的FTP写入指令（事实上，我在写完上传文件的API函数，进行测试的时候发现返回信息为“550 Permission denied”，经过查阅资料后才知道需要将此项设置为true），然后将anon\_upload\_enable、anon\_mkdir\_write\_enable设为true，表示允许匿名用户上传文件和创建文件夹。此类的配置还做了很多，就不一一介绍了。

配置文件修改后，保存（注意：保存时需要root权限）。然后使用“service vsftpd restart”命令重启vsftpd服务器。

### 2.3.2 客户端与服务器建立连接[2]

客户端连接到服务器端的基本过程如图2.3.1所示。

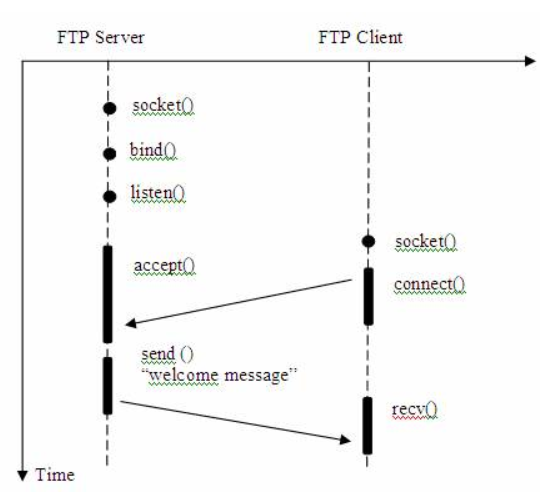


图2.3.1 客户端连接服务器流程图

从图中可以看到，服务器始终处于监听阶段，等待客户端的连接。在客户端程序获得服务器的IP地址、端口号之后，就可以使用connect函数连接到服务器，然后使用recv函数接受服务器传回的消息。如果连接成功，服务器会发送欢迎消息。

### 2.3.3 客户端登录到服务器

客户端登录服务器的过程如图2.3.2所示。

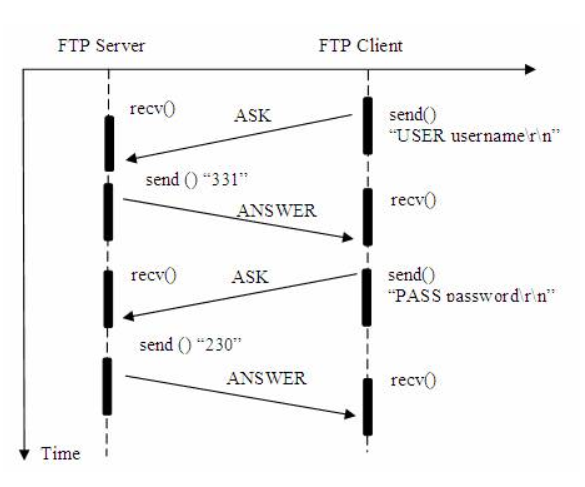


图2.3.2 客户端登录过程

在客户端接收到2.3.1中所示的欢迎信息之后，便可以开始连接到服务器了。根据FTP协议——首先，客户端需要发送的是USER指令[3]，具体的格式为“USER username\r\n”，其中，username为登录的用户名。正常情况下，客户端会接收到服务器回复的信息“331 xxxxx”，其中xxxxx是一些说明性文字，331表示上一步发送的username已经被服务器接受，接下来应该发送password。然后客户端发送PASS指令，具体格式为“PASS password\r\n”，其中，password是登录所需的密码，如果登录成功，服务器会返回信息“230 xxxxx”，其中，230则表示登录成功。

### 2.3.4 进入被动模式[4]

FTP在进行文件传输时，FTP的客户端和服务器之间要建立两个并行的TCP连接：“控制连接”和“数据连接”。

控制连接在整个会话期间一直保持打开，FTP客户所发出的传送请求，通过控制连接发送给服务器的控制进程，但是控制连接并不用来传送文件。实际用来传输文件的是“数据连接”。服务器端的控制进程在接受到FTP客户发送来文件传输请求后就创建“数据传送进程”和“数据连接”，用来连接客户端和服务器端的数据传送进程。数据传送进程实际完成文件的传送，在传送完毕后关闭“数据传送连接”，并结束运行。

由于FTP使用了一个分离的控制连接，因此FTP的控制信息是带外(out of band）传送的。而主动模式和被动模式的区别是数据连接建立的方式不同。命令连接建立的方式是相同的：客户端向服务器的FTP端口(默认是21号端口)发送连接请求，服务器接受连接，建立一条命令连接。

主动模式(port方式)：当需要传送数据时， 客户端在命令链路上用PORT命令告诉服务器：“采用主动模式建立数据连接，我打开了X端口，你过来连接我”。同时客户端在X端口监听（socket bind listen）等待服务器的连接……服务器收到消息后，从20端口向客户端的X端口发送连接请求（connect）建立一条数据链路来传送数据。

被动模式 (pasv方式)：当需要传送数据时， 客户端在命令链路上用PASV命令告诉服务器：“采用被动模式建立数据连接” ，服务器在命令链路上回复客户端：“我打开了X端口，你过来连接我”。于是客户端向服务器的X端口发送连接请求，建立一条数据链路来传送数据。

本文采用的方式是被动模式。通过被动模式建立数据连接的过程如图2.3.3所示。

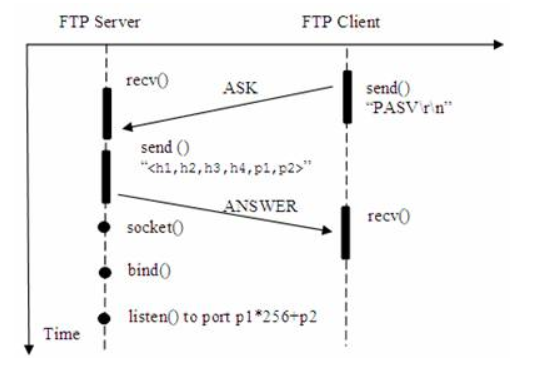


图2.3.3 进入被动模式的过程

要进入被动模式，首先，客户端发送PASV指令，指令格式为“PASV\r\n”，此处没有需要传入的参数。服务器接收到消息后，会响应消息“227 Entering passive mode (<h1,h2,h3,h4,p1,p2>)”，其中，h1，h2，h3，h4组成的h1.h2.h3.h4即为服务器IP地址，而p1\*256+p2则可以得到服务器打开的数据端口，我们此时需要在服务器的数据端口建立数据连接（要获得目录或上传、下载文件，必须通过数据端口传输数据）。

### 2.3.5 客户端从服务器下载文件

客户端从服务器下载文件的过程如图2.3.4所示（此处暂时未考虑断点续传问题）。

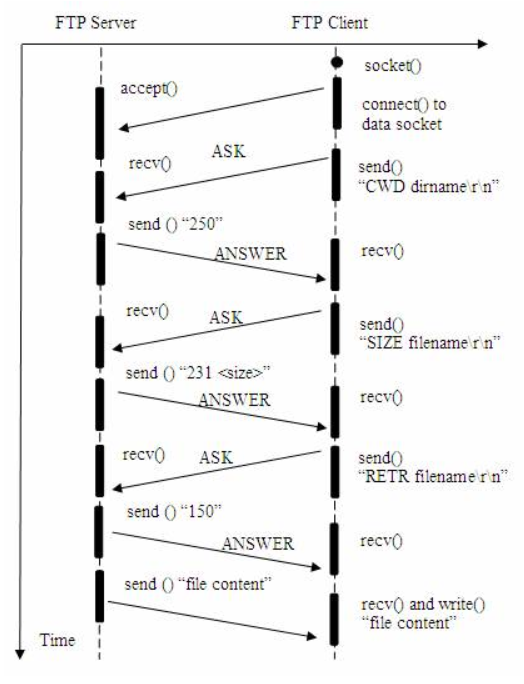


图2.3.4 客户端从服务器下载文件的过程

从图中可以看出，首先，进入某一工作目录，然后，客户端发送SIZE指令，具体格式为“SIZE filename\r\n”，表示客户端向服务器请求获得文件名为filename的文件的大小，此处，filename可以是绝对路径，也可以是相对路径，当为相对路径时，服务器会到当前工作目录下去寻找。正常情况下，若文件存在，则服务器会返回“231 size\r\n”，其中size则是文件的大小。然后，客户端可以准备一个大小大于等于size的buffer，准备接受数据。此时，若处于被动模式，则可以直接进行下一步，若未处于被动模式，则要先进入被动模式，然后开始下一步。接下来，客户端从控制端口向服务器发送RETR指令，指令格式为“RETR filename\r\n”，其中，filename为要下载的文件名，如果服务器在数据端口准备好了，那么将会返回“150 xxxxx”，此时，便可以从数据端口循环使用recv函数接受数据并写入本地文件，直到不再有数据为止。至此，文件下载功能就完成了。

### 2.3.6 客户端上传文件到服务器

与文件下载类似，客户端上传文件到服务器的过程如图2.3.5所示（此处也暂时未考虑断点续传问题）。

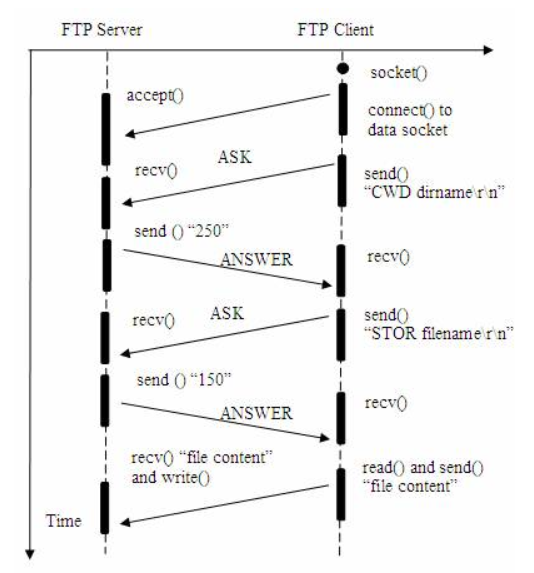


图2.3.5 客户端上传文件过程

如图所示，当进入某一目录之后，和下载文件一样，必须保证此时进入了被动模式，如果没有，则进入被动模式。然后，客户端向服务器端发送STOR指令（考虑断点续传的情况下，可能会使用APPE指令），指令格式为“STOR filename\r\n”，其中，filename是要保存到服务器端的文件名。然后，如果服务器返回“150 xxx”，则表示服务器端在数据端口已经准备好接收了，那么只需要在数据端口不断发送需要发送的文件数据即可。

## 2.4 扩展功能

2.3节讲述了基本要求的实现思路，接下来则讲述一些我做的扩展功能的实现思路。

### 2.4.1 获取工作路径下的文件列表

要想做出一个比较人性化的FTP客户端，只实现基本功能是不够的。要将文件以图形化的形式展现出来，首先则需要从服务器获得当前目录下的文件列表。要获取文件列表，首先要进入被动模式，然后从客户端发送LIST指令，指令格式为“LIST\r\n”，此指令没有参数。然后，当服务器返回“150 xxx”之后，表示服务器已经将数据从数据端口发出，此时，只需要从数据端口获取数据即可得到该工作路径下的文件列表。文件列表中的某一项如下所示：

-rw------- 1 500 500 7 Mar 13 03:53 abc.txt

其中，“-rw------”是与读写权限有关的东西，“1”则可以表示出文件是普通文件还是文件夹，当此值为1时表示它是文件，当值不为1时，表示它是文件夹。

获取了文件名且知道他们是否是文件夹之后，就可以在界面上将文件和文件夹区分开并且显示出来。

### 2.4.2 显示当前工作路径

要显示当前工作路径，则需要获得当前的工作路径。要获得当前的工作路径比较简单，只需要发送PWD指令即可。客户端从命令端口发送PWD指令，PWD指令的格式为“PWD\r\n”，没有参数。然后，从命令端口接收到响应“257 xxx is ‘curDir’”，其中，curDir即为当前的工作路径。

然后，只需要每次改变工作路径之后重新获取当前工作路径并显示即可。

### 2.4.3 改变工作路径

要实现改变工作路径也非常简单，只需要从客户端命令端口发送CWD指令即可，CWD指令的格式为“CWD workspace\r\n”，其中，workspace即为需要指定的工作路径。然后，如果服务器返回“250 xxx”则表示改变工作路径成功。

### 2.4.4 删除文件

要删除文件也比较简单，只需要从客户端命令端口发送DELE指令即可，DELE指令的格式为“DELE filename\r\n”，其中，filename为要删除的文件的路径，可以采用绝对路径，也可以采用相对路径，如果采用相对路径，则只会在当前工作路径中寻找。如果成功，则服务器会返回“250 xxx”。

### 2.4.5 删除文件夹

删除文件夹与删除文件类似，要从客户端命令端口发送RMD指令，RMD指令格式为“RMD foldername\r\n”，其中foldername为要删除的文件夹的名字。如果删除成功，则返回“250 xxx”。

### 2.4.6 新建文件夹

新建文件夹与删除文件夹类似，要从客户端命令窗口发送MKD指令，MKD指令的格式为“MKD foldername\r\n”，其中，foldername为要创建的文件夹的名字。如果创建成功，则返回“257 xxx”。

### 2.4.7 文件重命名

文件重命名的对象可以是文件，也可以是文件夹，需要两条指令，RNFR和RNTO。首先，客户端通过控制端口向服务器发送指令“RNFR oldname\r\n”，其中，oldname为需要被重命名的文件的名字，正常情况下服务器会返回“350 xxx”。紧接着，客户端继续通过控制端口向服务器发送指令“RNTO newname\r\n”，其中，newname是新的名字。如果重命名成功，服务器会返回代码“250 xxx”。

### 2.4.8 上传功能的断点续传实现[5]

要实现上传功能的断点续传，则需要在原有的过程中做一些改变。在原过程中，直接通过STOR指令告诉服务器要上传一个文件，然后从数据端口传入从本地读入的文件。如果要实现断点续传，则需要获取已经传输的数据长度，所以，首先要使用SIZE指令获取服务器端的该文件的大小，即从客户端控制端口发送指令“SIZE filename\r\n”，如果存在该文件，则返回“213 size\r\n”，如果不存在该文件，则表示此次是第一次上传该文件，不需要用到断点续传功能，服务器返回“550 xxx”。

所以，如果服务器返回码是550，则按照2.3.6中剩下的过程进行文件传输。如果返回码是213，则获取已经传送的字节数size，此时不再使用STOR指令进行存储，而是使用APPE指令，格式为“APPE filename\r\n”，表示接下来打开数据端口（前提仍然是已经进入被动模式），准备接收数据，并把接收到的数据添加到名为filename的文件中。如果返回“150 xxx”，则表示操作成功，数据端口已经准备好接收数据。接下来只需要往数据端口传入剩下的数据即可。

### 2.4.9 下载功能的断点续传实现

要实现下载功能的断点续传，首先要打开本地文件filename，得到该文件内容的长度len。然后，由FTP客户端向服务器发送指令SIZE来获得服务器上该文件内容的长度size。如果len size，则表示不再需要续传，直接完成。如果len <= 0，则表示本地无此文件，则不需要断点续传，直接按照2.3.5中的下载文件步骤进行下载。

若len > 0，则客户端从控制端口发送指令REST，格式为“REST len\r\n”，表示标识出文件内的数据点，跳到文件中指定的数据检验点上，后面需要传送数据的时候将从这个点开始继续传送文件，len即为偏移量，如果返回码为350，表示REST操作成功，然后，客户端需要再发送一条TYPE指令，具体格式为“TYPE I\r\n”，表示接下来的数据传输采用二进制传输[6]的方式进行传输，如果服务器返回码为200，则表示设置成功，接下来可以从偏移量的位置开始传输数据。然后发送指令RETR，格式为“RETR filename\r\n”，表示服务器将向客户端传送文件filename的内容（以len为偏移量开始）。接下来，客户端只需要从数据端口不断接受服务器传来的数据并添加到本地的名为filename的文件中即可。这样就完成了断点续传。

值得一提的是，在此过程中，我们选择了发送指令“TYPE I\r\n”，让接下来的传输以二进制方式进行传输。主要的原因是使用了REST之后，必须采用二进制传输方式，否则在RETR指令的时候回返回一个错误码，无法进行传输。

# 3 实验中遇到的问题

## 3.1 vsftpd的配置问题

在2.3.1中，我曾简单介绍了vsftpd的配置问题，可实际上远没有这么简单。首先，最开始我并没有安装vsftpd，在连接服务器的时候总是失败。后来查阅资料才发现linux并没有自带FTP服务器端，于是便安装了vsftpd。然后，在我完成了文件上传的代码编写之后，在测试效果时，服务器返回了“550 Permission denied”，在网上查询资料之后，才发现配置文件中有很多东西都需要修改，然后再将其一一修改。

## 3.2 文件目录返回结果中判断文件是否是文件夹的问题

在2.4.1中曾经提到，当向服务器请求文件列表时，服务器会返回一个字符串来表示所有文件的信息，其中一个文件的信息如下：

-rw------- 1 500 500 7 Mar 13 03:53 abc.txt

当时我并不知道如何判断此项到底是文件还是文件夹。经过网上大量的查找，仍然没有解决问题。网络上说明了unix系统和windows中的文件是如何表示，如何判断是否是文件夹的，但是却没提到linux系统中怎么判断。于是，我只能通过大量的实验，寻找规律，发现第二项（空白将每一项隔开）为1的时候才是文件，其他时候都是文件夹，于是便得到了解决的方法。

## 3.3 延迟问题

如果用户频繁地进行服务请求，则可能会遇到接受不到正常返回信息的问题。比如：当用户使用LIST指令后，实际上会返回两个消息，他们分别都有返回码。如果请求得比较快，在第二个返回码到达之前，用户就使用recv接收到了第一个返回码，那么第二个返回码会在到达后留在接收缓冲区。当用户发送下一个指令并快速接受服务器返回的消息，则此时由于接受缓冲区有数据，且如果该指令得回复信息到达得比较慢，那么recv函数会接收到上一个指令遗留下的一个信息，从而导致无法正确地通过返回码来判断指令是否正确执行。

我参考了很多网上的代码，他们几乎都是直接执行后面的代码，根本不去判断上一个指令返回的返回码是否告诉我们指令是否正确执行。虽然可以使用这种方法解决问题，但不到万不得已，我还是不想使用这种方式。

于是，我想到了使用进程休眠的方法，即使用Sleep函数让进程休眠一定的时间，让返回码有足够的时间进入接收缓冲区。这样就可以正确判断执行状态了。

# 4 反思与改进

这次的FTP客户端制作过程中，我查询了很多资料，我发现我的作品比较好的一个亮点就是对服务器的返回码进行了解析，能更好地反馈给用户。当然，我的作品也有很多可以改进的地方。

第一，项目架构问题，我为Socket封装的类库中，一个函数代表一个方法，但是我没有很好地对各种指令进行定义，这样会导致项目的扩展性不那么强，逻辑结构不那么清晰明了。在我看到的一个文章中，作者对每一种指令进行了宏定义，并使用一个函数parseCommand(cmdCode,param)来解析函数，其中cmdCode即该指令的宏定义码，param是需要传入的参数。如果这样做，需要增加或减少指令的时候就可以减少代码的修改量，减少维护成本。

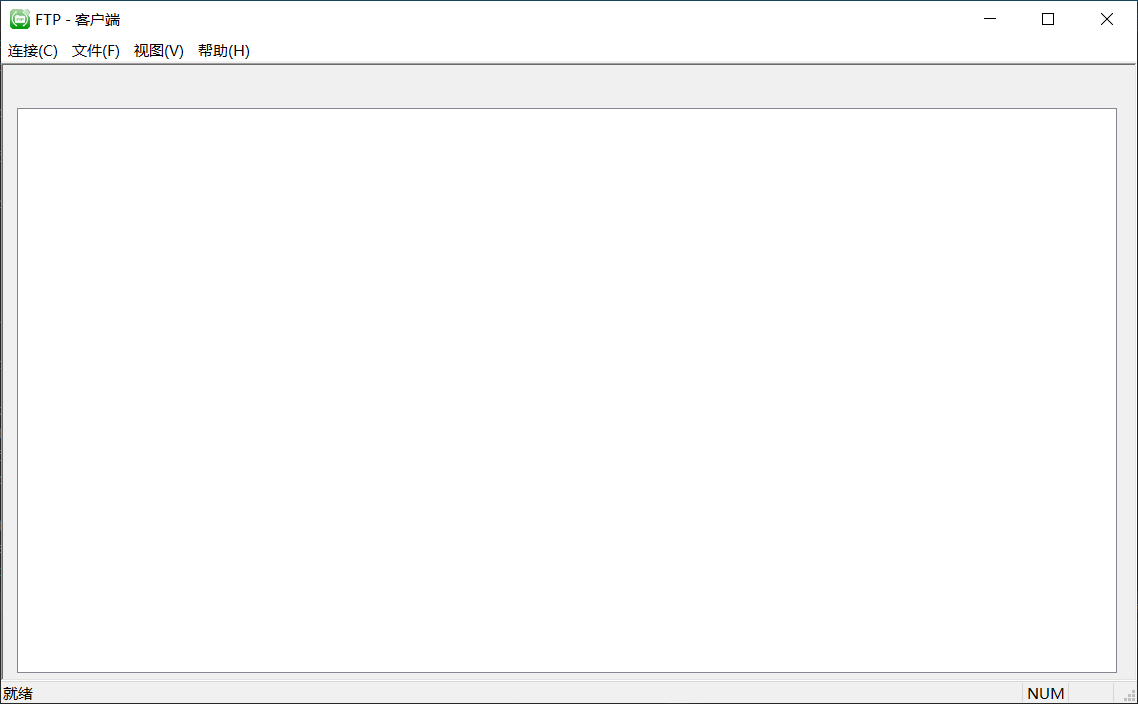
第二，其实还可以为此项目设计一个编辑器，用于打开服务器端的文件，那样就可以直接在FTP客户端中查看服务器端文件，并进行一定的修改。现在很多流行的FTP客户端都有自己的编辑器（比如WinSCP），这将会大大地增强用户的体验。

第三，本项目的菜单栏是在程序上方的，还可以添加鼠标右键的菜单栏，就像windows系统中的鼠标右键一样，这样也可以使操作更方便。

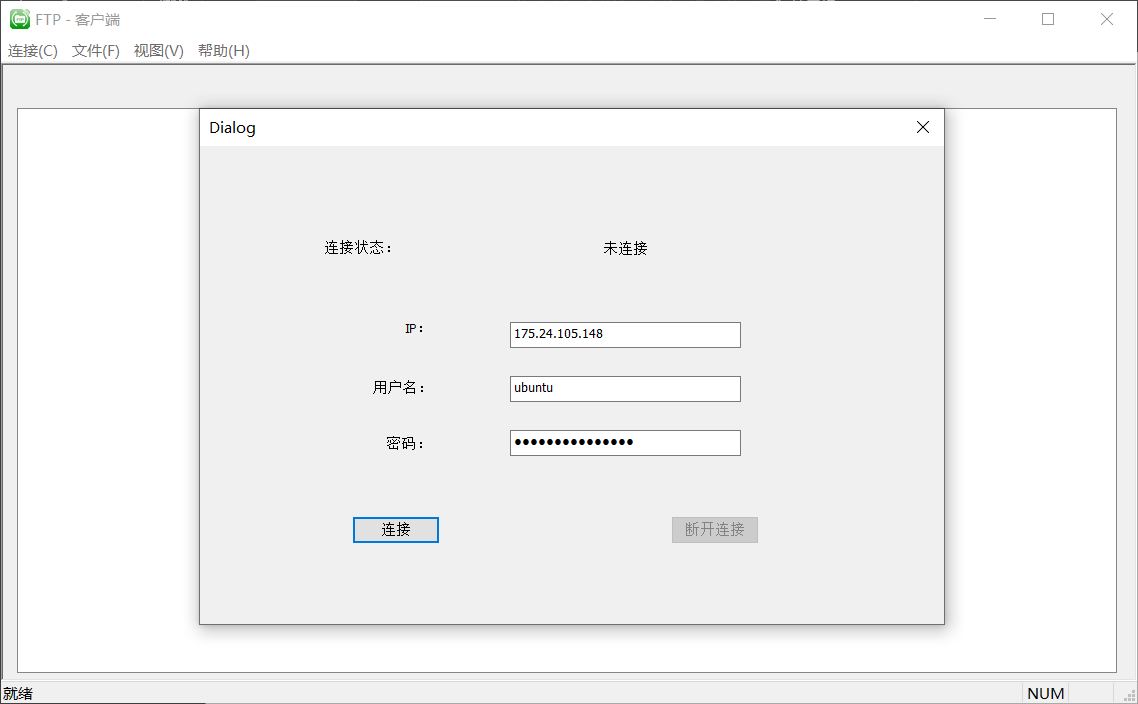
**结论**

**1. 程序主要界面及结果**

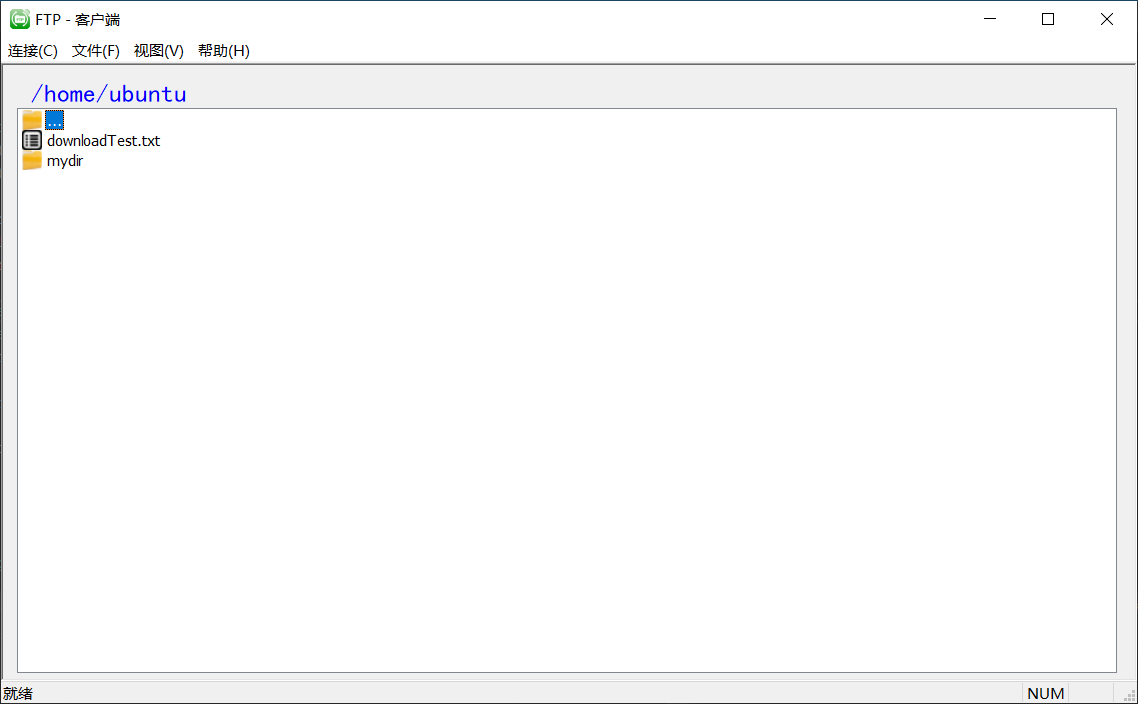
初始界面：



连接服务器的界面：

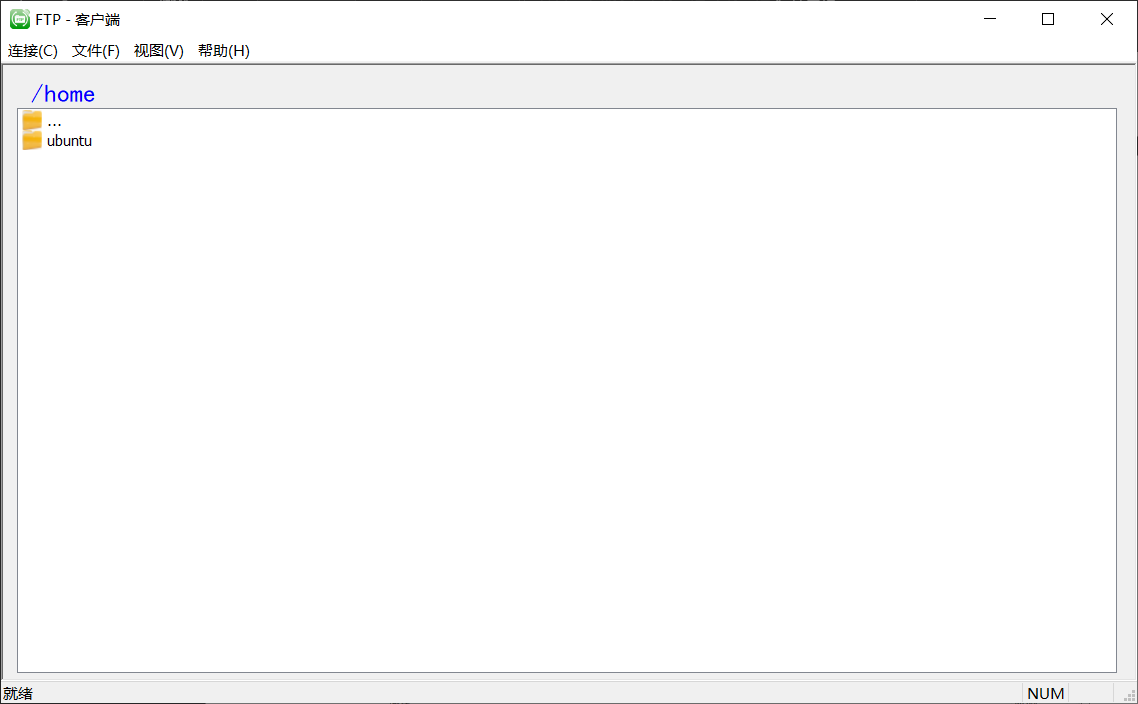


获取目录之后：

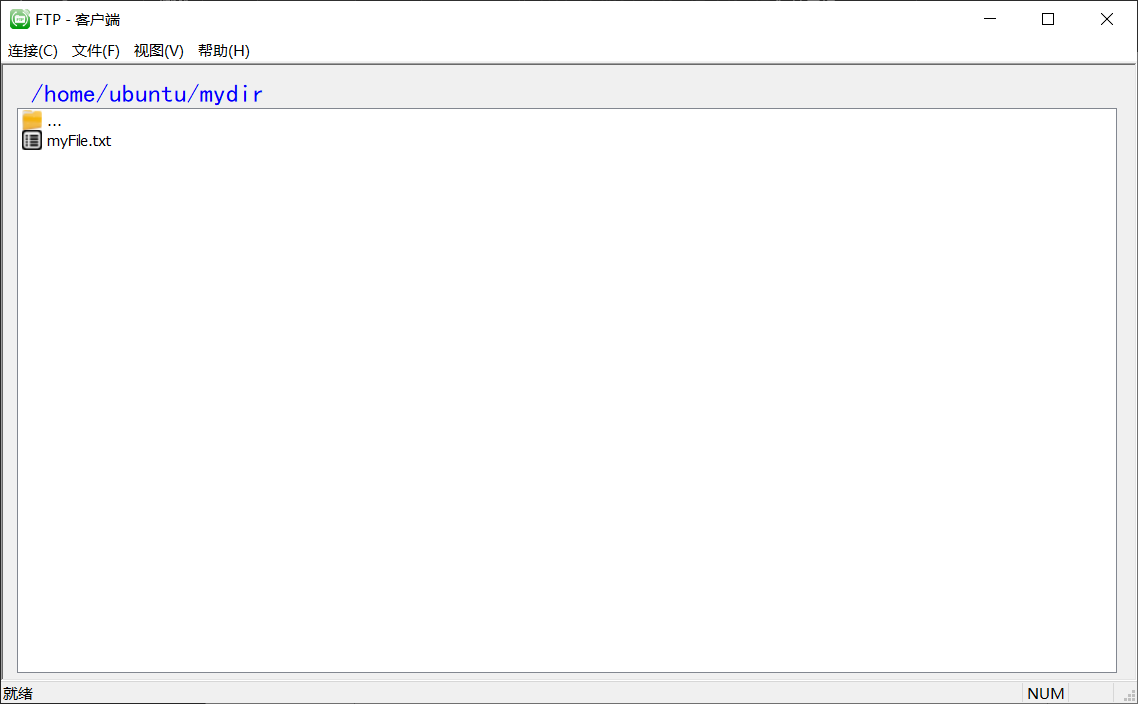


其中，上方的蓝色字体则是当前工作路径。

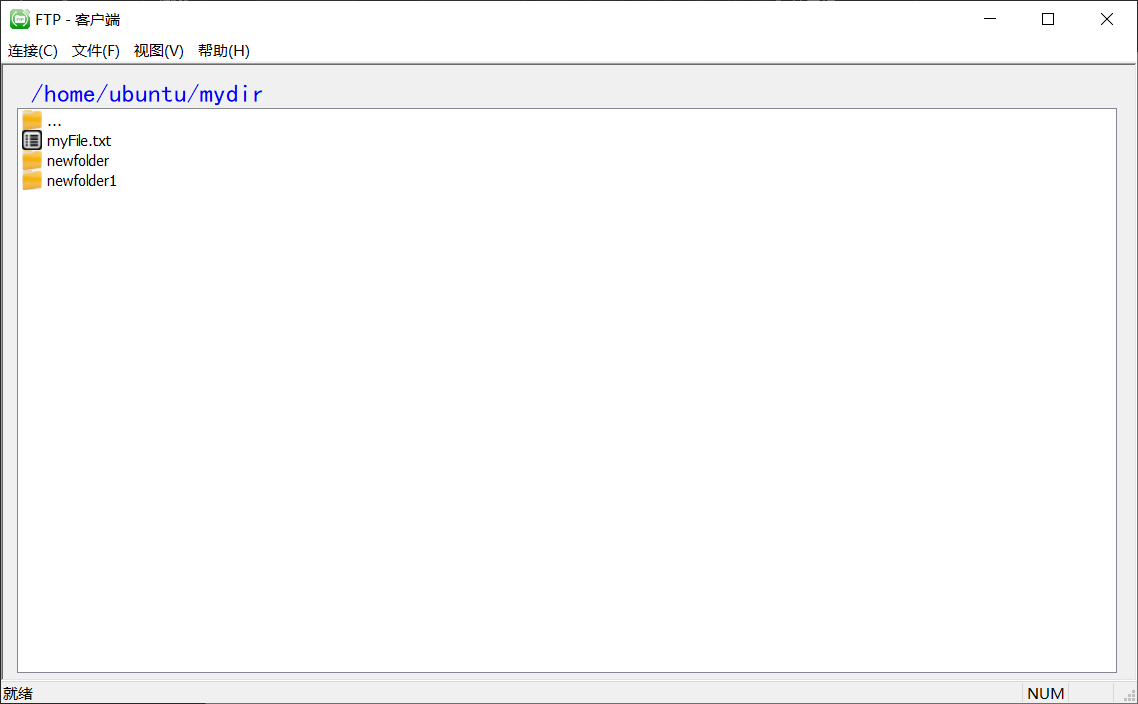
进入上层目录之后：



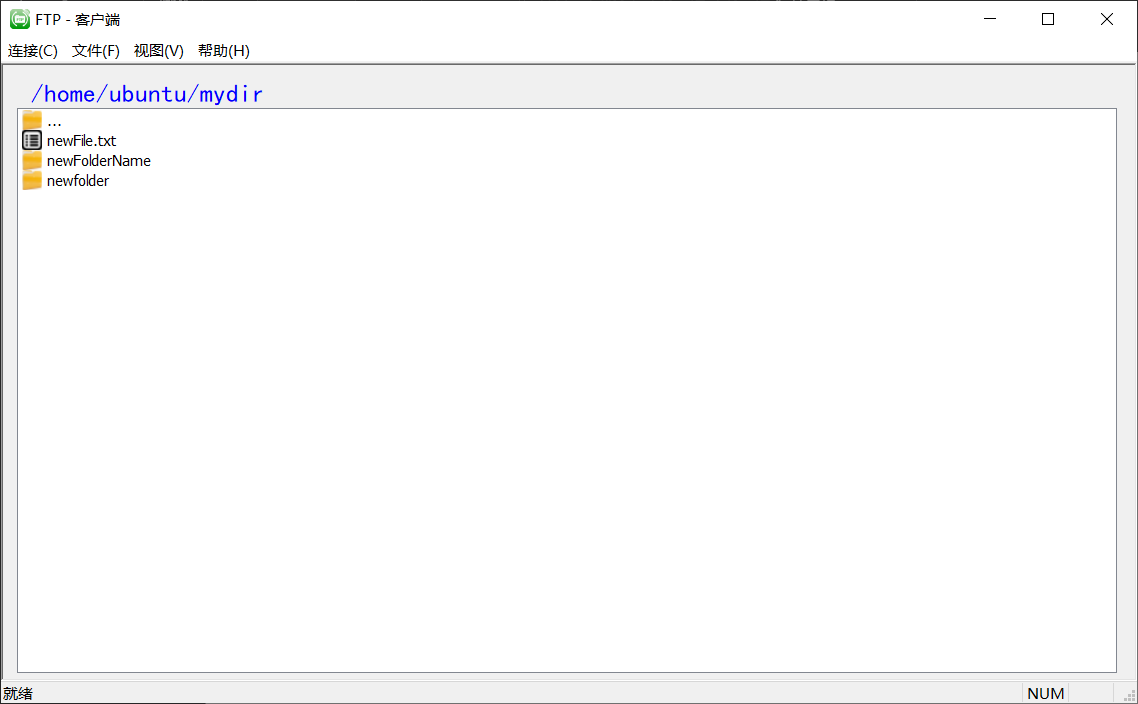
进入目录“mydir”之后：



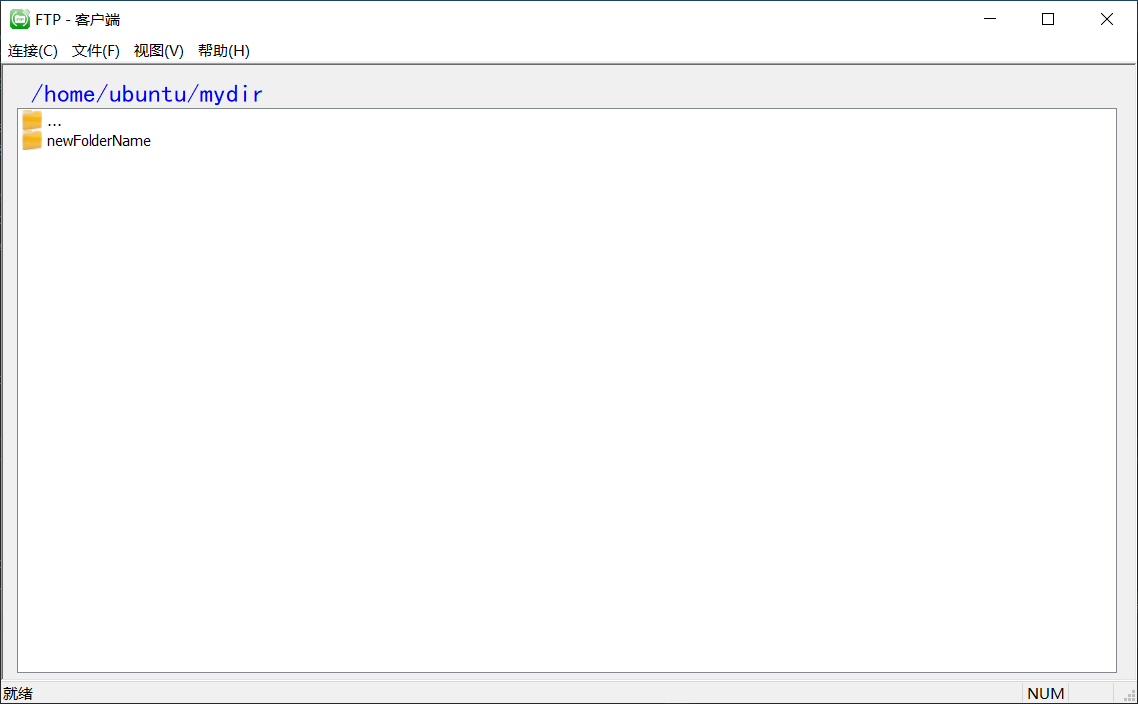
创建文件夹“newfolder”和“newfolder1”：



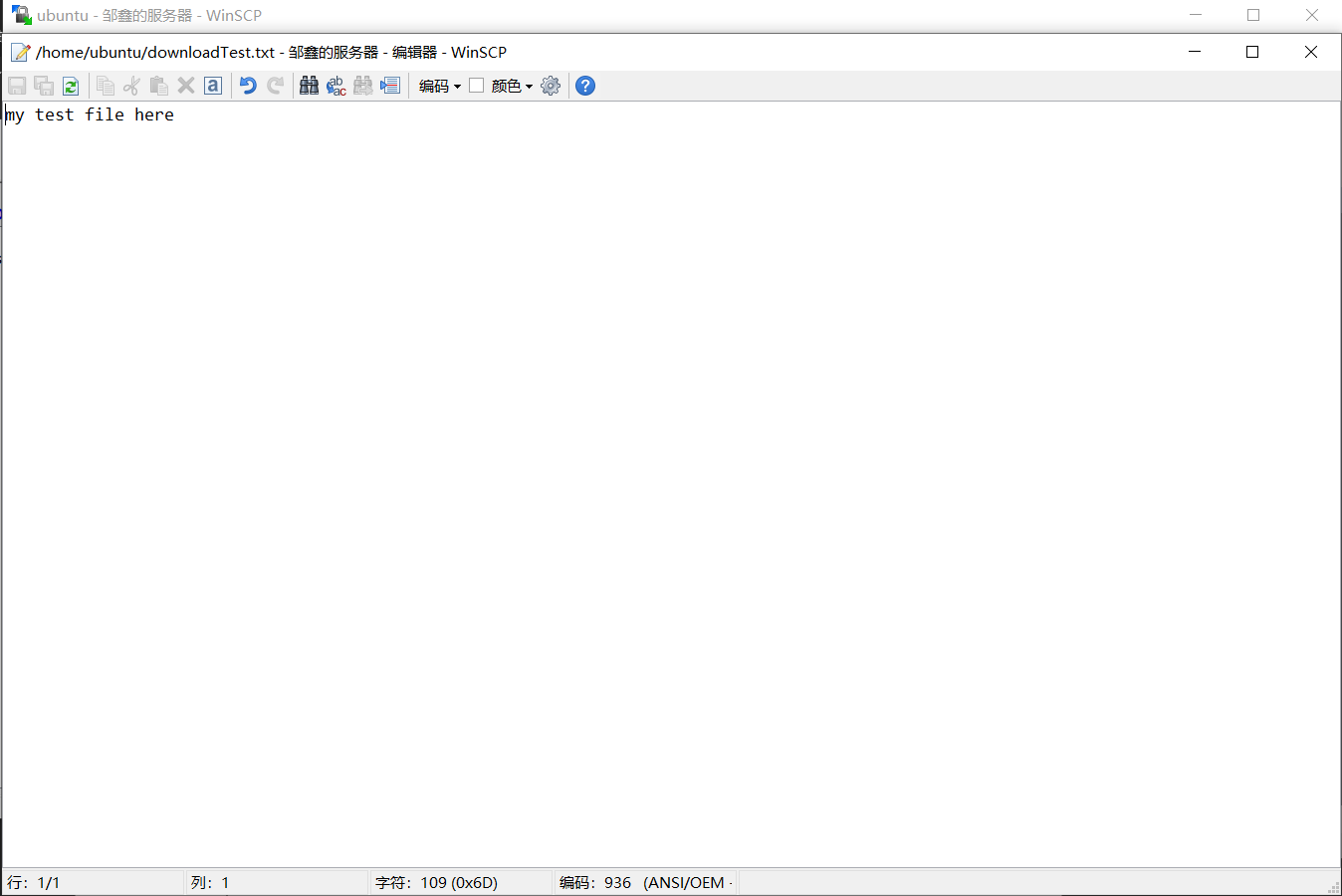
将文件夹“newfolder1”重命名为“newFolderName”，并将文件“myFile.txt”重命名为“newFile.txt”：



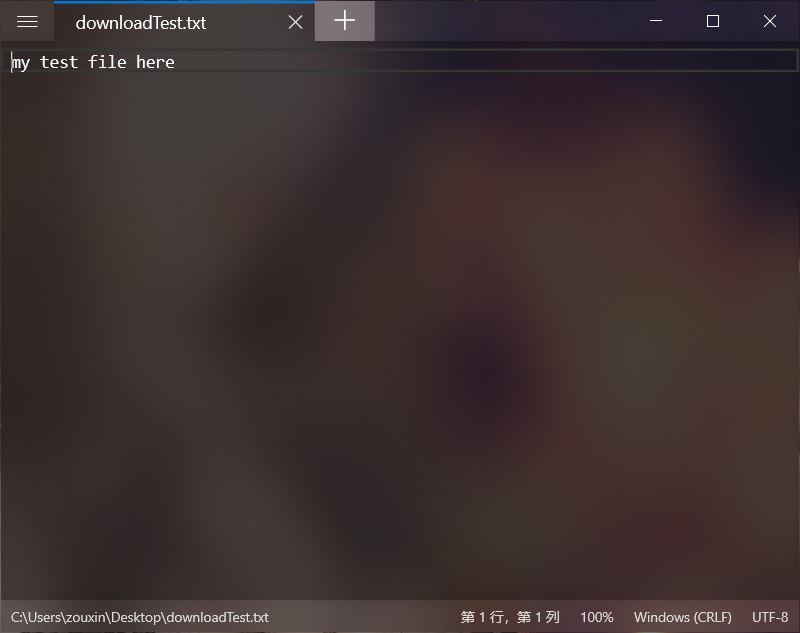
删除文件“newFile.txt”和文件夹“newfolder”：



回到初始工作界面，并用WinSCP查看文件“downloadTest.txt”的内容：

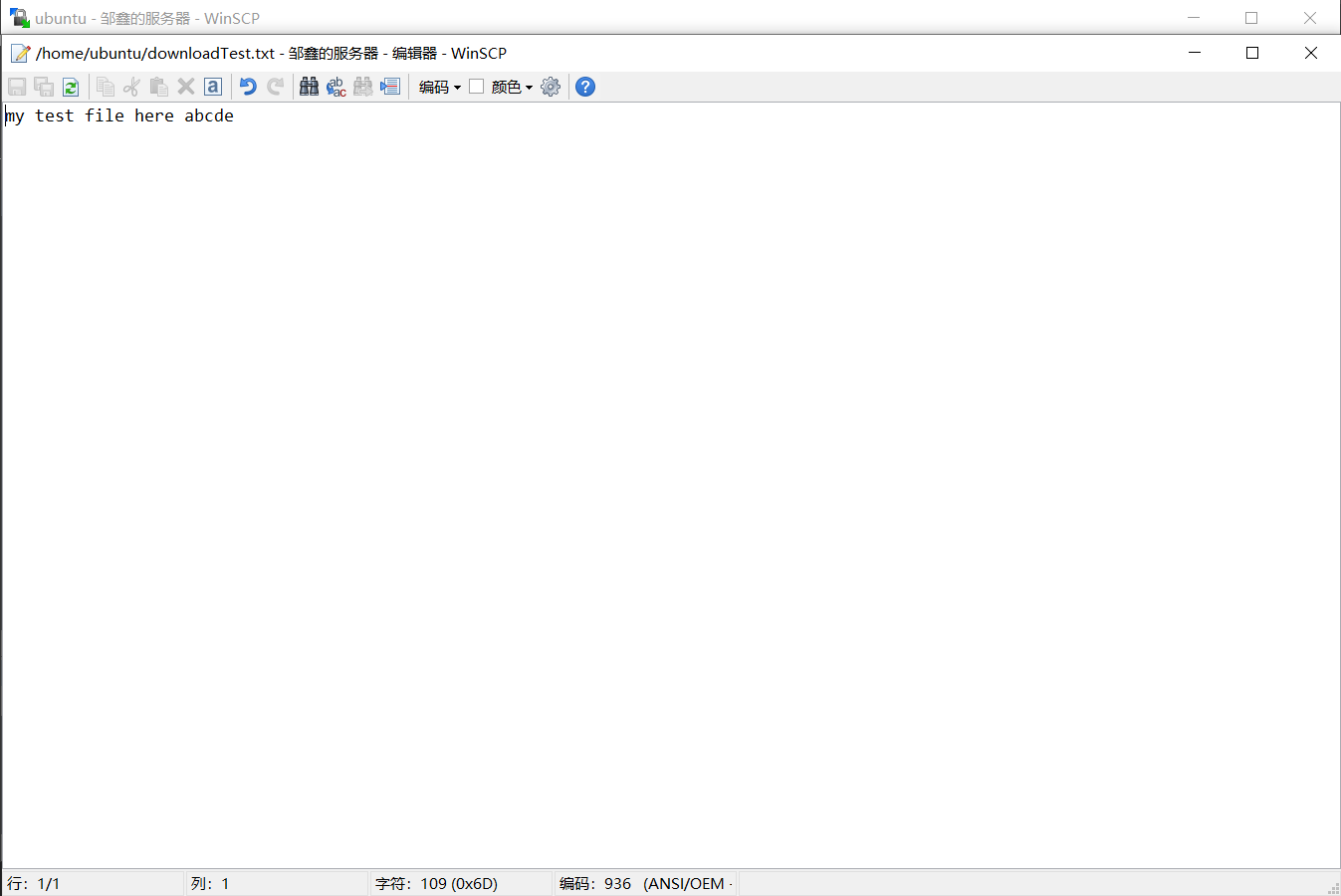


下载文件“downloadTest.txt”，并在windows系统上用记事本打开：

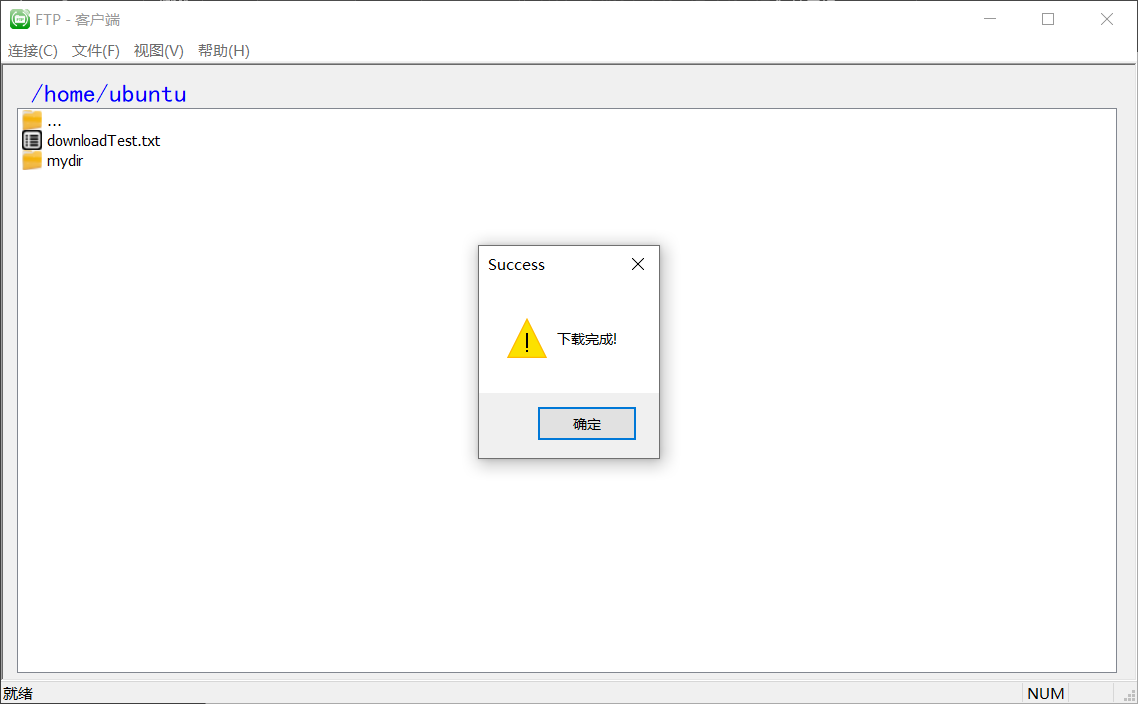


可以看到，文件成功下载。

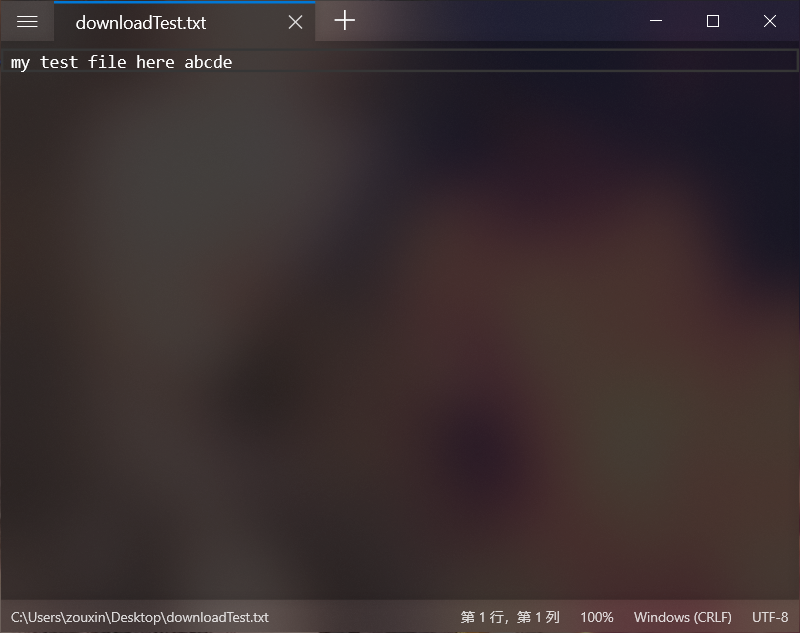
为了模拟断点续传，我们首先用WinSCP在服务器上的“downloadTest.txt”文件最后添加一段字符：



再次下载文件“downloadTest.txt”：

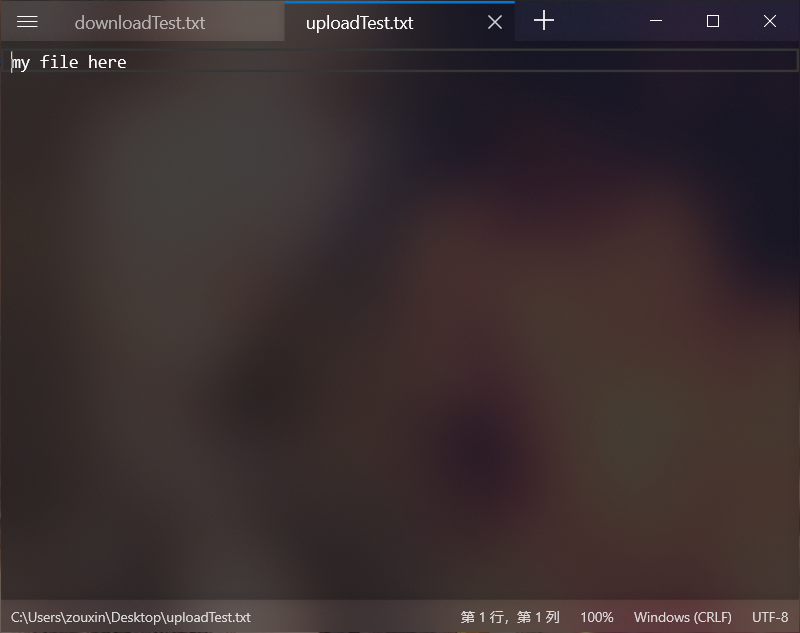


用记事本打开刚才下载的文件：

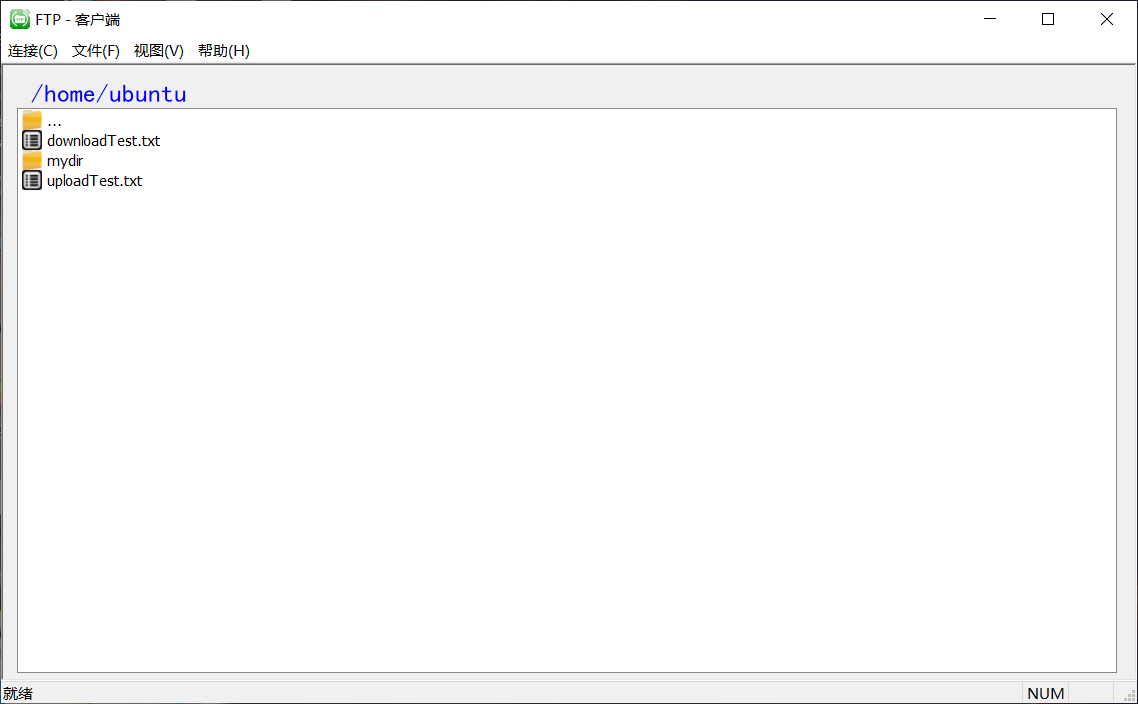


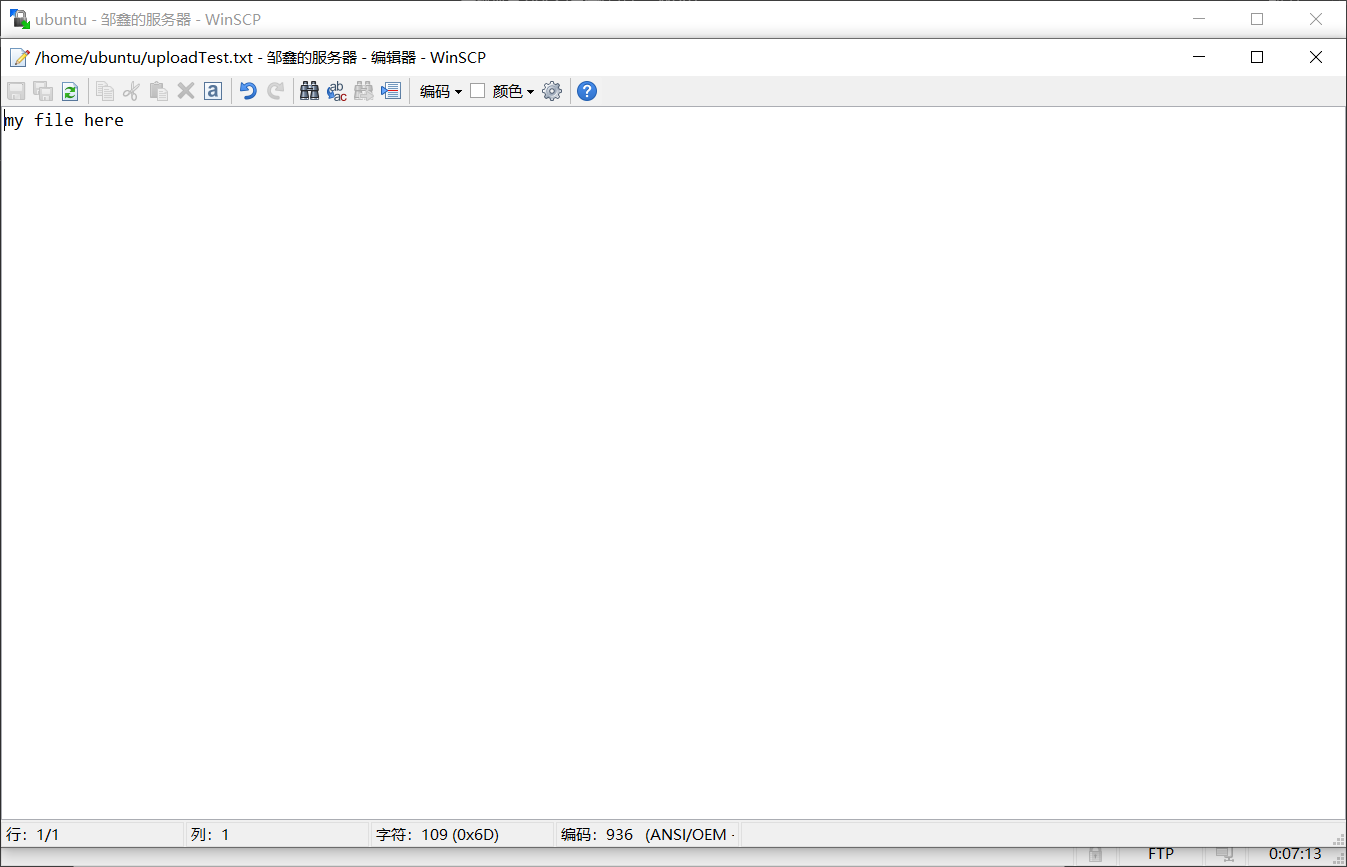
可以看到，断点续传功能是成功的。

接下来，用记事本打开windows系统下的文件“uploadTest.txt”：



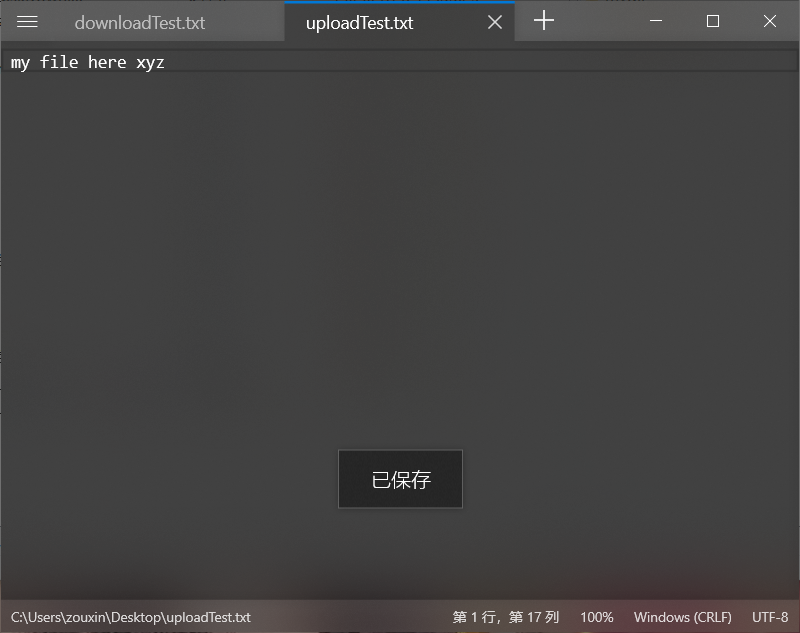
将“uploadTest.txt”上传到服务器并使用WinSCP打开查看：



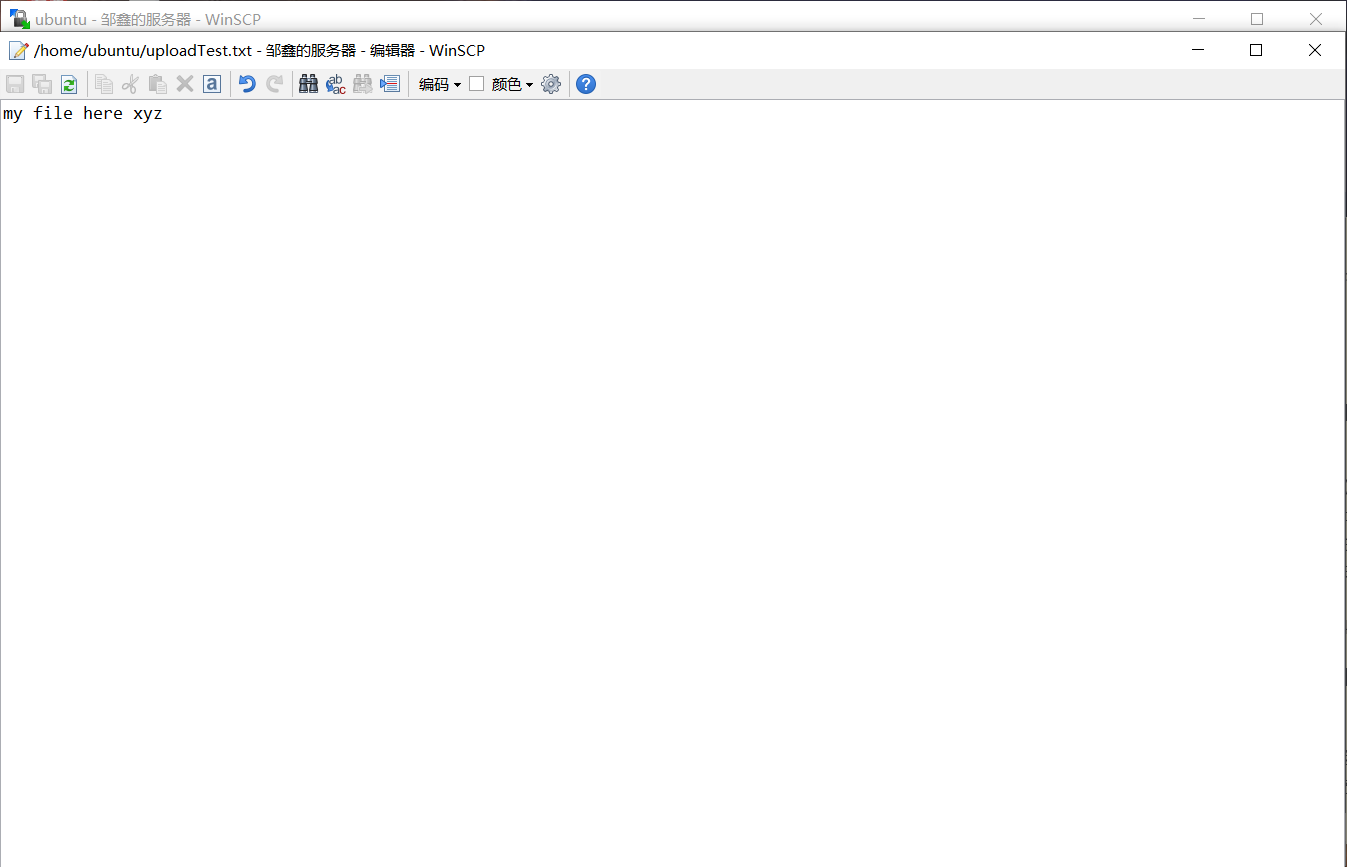


可以看到，上传成功了。

同样地，为了模拟断点续传，在本地的“uploadTest.txt”文件后加上一段字符串：



再次上传到服务器并使用WinSCP打开查看：



可以看到，断点续传功能也成功了。

**2. 程序源程序**

启动Socket服务并连接到服务器：

FTPClient::FTPClient(const std::string& host, int port)//创建MySokcet并将其连接

{

this->host = host;

this->port = port;

try

{

WSAStart();

}

catch (FTPException e)

{

throw e;

}

sock\_data = MySocket();

sock\_ctl = MySocket();

sock\_ctl.SetAddrInfo(host, port);

if (!sock\_ctl.Create())

{

throw FTPException(ExType::CreateSocketFail);

}

if (!sock\_ctl.Connect())

{

throw FTPException(ExType::ConnectFail);

}

char buf[MAX\_BUF];

int code = sock\_ctl.Receive(buf, MAX\_BUF, 0);

if (code >= 3)

{

buf[code] = '\0';

std::string str = buf;

std::string first3Str = str.substr(0, 3);

if (first3Str != "220")//没有连接成功

{

throw FTPException(ExType::ConnectFail);

}

}

else

{

throw FTPException(ExType::ConnectFail);

}

}

登录到服务器：

bool FTPClient::Login(const std::string& usr, const std::string& pwd)

{

char buf[MAX\_BUF];

sprintf(buf, "USER %s\r\n", usr.c\_str());

std::string str;

int ret = sock\_ctl.Send(buf, strlen(buf), 0);

memset(buf, 0, MAX\_BUF);

ret = sock\_ctl.Receive(buf, MAX\_BUF, 0);

str = buf;

if (!containsCode(str, "331"))

{

throw FTPException(ExType::LoginFail);

}

sprintf(buf, "PASS %s\r\n", pwd.c\_str());

ret = sock\_ctl.Send(buf, strlen(buf), 0);

memset(buf, 0, MAX\_BUF);

ret = sock\_ctl.Receive(buf, MAX\_BUF, 0);

if (ret >= 3)

{

str = buf;

if (!containsCode(str,"230"))//报错

{

if (containsCode(str, "530"))

{

throw FTPException(ExType::UsrOrPwdWrong);

}

else

{

throw FTPException(ExType::LoginFail);

}

}

}

else

{

throw FTPException(ExType::LoginFail);

}

return true;

}

进入被动模式：

bool FTPClient::EnterPasvMode()

{

char buf[MAX\_BUF];

memset(buf, 0, MAX\_BUF);

sprintf(buf, "PASV\r\n");

int ret = sock\_ctl.Send(buf, strlen(buf), 0);

ret = sock\_ctl.Receive(buf, MAX\_BUF, 0);

std::string str = buf;

std::string subStr = str.substr(0, ret);//227

if (subStr.size() < 3 || !containsCode(subStr,"227"))

{

throw FTPException(ExType::EnterPasvFail);

}

//获取服务器端的ip和其打开的端口号

std::string host = getIpByPasvRet(subStr, subStr.size());

int port = getPortByPasvRet(subStr, subStr.size());

if (port < 0)

{

throw FTPException(ExType::EnterPasvFail);

}

//sock\_data与新开的端口进行连接

sock\_data.Disconnect();

sock\_data.SetAddrInfo(host, port);

if (!sock\_data.Create())

{

CleanUp();

throw FTPException(ExType::CreateSocketFail);

}

if (!sock\_data.Connect())

{

CleanUp();

throw FTPException(ExType::ConnectFail);

}

int port\_ctl = sock\_ctl.getSocketLoaclPort();

int port\_data = sock\_data.getSocketLoaclPort();

std::cout << "control port:" << port\_ctl << std::endl;

std::cout << "data port:" << port\_data << std::endl;

return true;

}

获得目录文件信息：

std::vector<FileInfo> FTPClient::getFilesOfCurWorkDir()

{

std::string curDir = getCurWorkingDir();

EnterPasvMode();

char buf[MAX\_BUF];

memset(buf, 0, MAX\_BUF);

sprintf(buf, "LIST\r\n");//获取当前工作目录下的文件列表

int ret = sock\_ctl.Send(buf, strlen(buf), 0);

memset(buf, 0, MAX\_BUF);

ret = sock\_ctl.Receive(buf, MAX\_BUF, 0);

std::string str = buf;

std::cout << str;

if (ret < 3)

{

throw FTPException(ExType::OtherFails,str);

}

else

{

if (!containsCode(str,"150"))

{

throw FTPException(ExType::GetFileInfoFail,str);

}

}

std::string fileInfo = "";

memset(buf, 0, MAX\_BUF);

while ((ret = sock\_data.Receive(buf, MAX\_BUF, 0)) != 0)

{

str = buf;//特点：每条文件信息之间用\r\n分隔，一条信息中，各种属性用1或多个空格分隔

fileInfo += str;

memset(buf, 0, MAX\_BUF);

}

std::vector<std::string> fileInfos = getFileInfoStrs(fileInfo);

std::vector<FileInfo> retVec;

int len = fileInfos.size();

for (int i = 0; i < len; i++)

{

retVec.push\_back(FileInfo(fileInfos[i],curDir));

}

return retVec;

}

FileInfo类的构造函数：

FileInfo::FileInfo(std::string strFromServerResponse,std::string curDir)

{

int i = 0, len = strFromServerResponse.size();

//各种属性值，是按顺序声明的

std::string accessRigthStr = "";

std::string isFolder = "";

std::string num1 = "",num2 = "",num3="";

std::string createMonth = "";//创建文件的月份

std::string createDay = "";//创建文件的日期

std::string createTime = "";//创建文件的时间

std::string name = "";

while (i < len && strFromServerResponse[i] != ' ')

{

accessRigthStr += strFromServerResponse[i];

i++;

}

while (i < len && strFromServerResponse[i] == ' ') i++;

while (i < len && strFromServerResponse[i] != ' ')

{

isFolder += strFromServerResponse[i];

i++;

}

while (i < len && strFromServerResponse[i] == ' ') i++;

while (i < len && strFromServerResponse[i] != ' ')

{

num1 += strFromServerResponse[i];

i++;

}

while (i < len && strFromServerResponse[i] == ' ') i++;

while (i < len && strFromServerResponse[i] != ' ')

{

num2 += strFromServerResponse[i];

i++;

}

while (i < len && strFromServerResponse[i] == ' ') i++;

while (i < len && strFromServerResponse[i] != ' ')

{

num3 += strFromServerResponse[i];

i++;

}

while (i < len && strFromServerResponse[i] == ' ') i++;

while (i < len && strFromServerResponse[i] != ' ')

{

createMonth += strFromServerResponse[i];

i++;

}

while (i < len && strFromServerResponse[i] == ' ') i++;

while (i < len && strFromServerResponse[i] != ' ')

{

createDay += strFromServerResponse[i];

i++;

}

while (i < len && strFromServerResponse[i] == ' ') i++;

while (i < len && strFromServerResponse[i] != ' ')

{

createTime += strFromServerResponse[i];

i++;

}

while (i < len && strFromServerResponse[i] == ' ') i++;

while (i < len && strFromServerResponse[i] != ' ')

{

name += strFromServerResponse[i];

i++;

}

this->fileName = name;

this->path = curDir + '/' + name;

int num = str2UInt(isFolder);

this->folder = (num > 1);

}

获取当前工作路径：

std::string FTPClient::getCurWorkingDir()

{

char buf[MAX\_BUF];

memset(buf, 0, MAX\_BUF);

sprintf(buf, "PWD\r\n");

int ret = sock\_ctl.Send(buf, strlen(buf), 0);

memset(buf, 0, MAX\_BUF);

ret = sock\_ctl.Receive(buf, MAX\_BUF, 0);

std::string str = buf;

if (ret < 3)

{

throw FTPException(ExType::OtherFails,str);

}

else if (!containsCode(str, "257"))

{

throw FTPException(ExType::GetCurDirFail,str);

}

this->curDir = getFileNameWithTheRetOfPWD(str, str.size());

return this->curDir;

}

在当前工作路径创建一个文件夹：

bool FTPClient::createFolderAtWorkingDir(std::string folderName)//可能会产生文件夹名不符合规范的异常

{

char buf[MAX\_BUF];

memset(buf, 0, MAX\_BUF);

sprintf(buf, "MKD %s\r\n", folderName.c\_str());

int num = sock\_ctl.Send(buf, strlen(buf), 0);

memset(buf, 0, MAX\_BUF);

num = sock\_ctl.Receive(buf, MAX\_BUF, 0);

std::string str = buf;

if (num < 3)

{

throw FTPException(ExType::OtherFails);

}

else

{

if (!containsCode(str, "257"))

{

throw FTPException(ExType::CreateFolderFail, str);

}

}

return true;

}

返回上层目录：

bool FTPClient::returnToParentDir()//若当前为最上层目录，应抛出一个异常 调用之后还需要改变curDir，目前还没改

{

std::string curDir = this->curDir;//可能需要调用getCurWorkingDir函数重新获取

if (curDir.empty()) return "";

int index = curDir.find\_last\_of('/');

if (index == curDir.size() - 1)

{

if (index == 0) throw FTPException(ExType::AtTopDir);

else

{

curDir.pop\_back();

index = curDir.find\_last\_of('/');

}

}

std::string aimDir = curDir.substr(0, index);

char buf[MAX\_BUF];

memset(buf, 0, MAX\_BUF);

if (aimDir == "/")

{

sprintf(buf, "CDUP\r\n");

}

else

{

sprintf(buf, "CWD %s\r\n", aimDir.c\_str());

}

int num = sock\_ctl.Send(buf, strlen(buf), 0);

memset(buf, 0, MAX\_BUF);

num = sock\_ctl.Receive(buf, MAX\_BUF, 0);

std::string str = buf;

std::cout << str;

if (num < 3)

{

throw FTPException(ExType::OtherFails);

}

else if(!containsCode(str, "250"))

{

throw FTPException(ExType::ChangeWorkSpaceFail, str);

}

return true;

}

删除当前工作路径下的一个文件：

bool FTPClient::deleteFileAtCurDir(std::string fileName)//删除当前工作区的一个文件-----

{

char buf[MAX\_BUF];

memset(buf, 0, MAX\_BUF);

sprintf(buf, "DELE %s\r\n",fileName.c\_str());

int num = sock\_ctl.Send(buf, strlen(buf), 0);

memset(buf, 0, MAX\_BUF);

num = sock\_ctl.Receive(buf, MAX\_BUF, 0);

std::string str = buf;

std::cout << str;

if (num < 3)

{

throw FTPException(ExType::OtherFails);

}

else if (!containsCode(str, "250"))

{

std::string tStr = str.substr(0, 3);

if (containsCode(str, "550") || containsCode(str, "450"))//文件不可用

{

throw FTPException(ExType::FileAccessFail);

}

else//其他错误

{

throw FTPException(ExType::OtherFails);

}

}

return true;

}

删除当前工作路径下的一个文件夹：

bool FTPClient::deleteFolderAtCurDir(std::string folderName)//删除当前工作区的一个文件夹

{

char buf[MAX\_BUF];

memset(buf, 0, MAX\_BUF);

sprintf(buf, "RMD %s\r\n", folderName.c\_str());

int num = sock\_ctl.Send(buf, strlen(buf), 0);

memset(buf, 0, MAX\_BUF);

num = sock\_ctl.Receive(buf, MAX\_BUF, 0);

std::string str = buf;

std::cout << str;

if (num < 3)

{

throw FTPException(ExType::OtherFails);

}

else if (!containsCode(str, "250"))

{

std::string tStr = str.substr(0, 3);

if (containsCode(str, "550") || containsCode(str, "450"))//文件不可用

{

throw FTPException(ExType::FileAccessFail,str);

}

else//其他错误

{

throw FTPException(ExType::OtherFails);

}

}

return true;

}

进入某一文件夹：

bool FTPClient::enterFolder(std::string folderName)//进入某一文件夹

{

std::string aimStr = this->curDir + '/' + folderName;

try

{

setWorkSpace(aimStr);

}

catch (FTPException e)

{

throw e;

}

return true;

}

bool FTPClient::setWorkSpace(std::string workSpace)

{

char buf[MAX\_BUF];

memset(buf, 0, MAX\_BUF);

sprintf(buf, "CWD %s\r\n", workSpace.c\_str());

int num = sock\_ctl.Send(buf, strlen(buf), 0);

memset(buf, 0, MAX\_BUF);

num = sock\_ctl.Receive(buf, MAX\_BUF, 0);

std::string str = buf;

std::cout << str;

if (num < 3)

{

throw FTPException(ExType::OtherFails);

}

else if (!containsCode(str, "250"))

{

throw FTPException(ExType::ChangeWorkSpaceFail, str);

}

this->curDir = workSpace;

return true;

}

下载文件（考虑断点续传）：

bool FTPClient::DownloadWithPos(const std::string& des, const std::string& src)

{

int len = getFileLength(des);

char buf[MAX\_BUF];

memset(buf, 0, MAX\_BUF);

sprintf(buf, "SIZE %s\r\n", src.c\_str());//是否需要完整路径?

int num = sock\_ctl.Send(buf, strlen(buf), 0);

memset(buf, 0, MAX\_BUF);

num = sock\_ctl.Receive(buf, MAX\_BUF, 0);

std::string str = buf;

if (num < 3)

{

throw FTPException(ExType::OtherFails);

}

else

{

if (!containsCode(str, "213"))

{

throw FTPException(ExType::GetFileSizeFail, str);

}

}

int index = str.find("213");

std::string sizeStr = "";

for (int i = index + 4; i < str.size() && str[i] != '\r'; i++)

{

sizeStr += str[i];

}

int size = str2UInt(sizeStr);

if (len >= size) return false;

if (len == -1) len = 0;

//设置pos

if (len > 0)

{

memset(buf, 0, MAX\_BUF);

sprintf(buf, "REST %d\r\n", len);

num = sock\_ctl.Send(buf, strlen(buf), 0);

num = sock\_ctl.Receive(buf, MAX\_BUF, 0);//350

str = buf;

if (num < 3)

{

throw FTPException(ExType::OtherFails, str);

}

else if (!containsCode(str, "350"))

{

throw FTPException(ExType::SetPosFail, str);

}

setASCIImode();

}

EnterPasvMode();

memset(buf, 0, MAX\_BUF);

sprintf(buf, "RETR %s\r\n", src.c\_str());

num = sock\_ctl.Send(buf, strlen(buf), 0);

num = sock\_ctl.Receive(buf, MAX\_BUF, 0);

str = buf;

if (num < 3)

{

throw FTPException(ExType::OtherFails);

}

else

{

if (!containsCode(str, "150"))

{

throw FTPException(ExType::FileDownloadFail, str);

}

}

FILE\* file = fopen(des.c\_str(), "a+");

memset(buf, 0, MAX\_BUF);

while ((num = sock\_data.Receive(buf, MAX\_BUF, 0)) != 0)

{

fprintf(file, buf);

memset(buf, 0, MAX\_BUF);

}

fclose(file);

return true;

}

上传文件（考虑了断点续传）：

bool FTPClient::Upload(const std::string& pathName, const std::string& fileName)//上传文件

{

FILE\* pFile = fopen(pathName.c\_str(), "rb"); // 以只读方式打开 且文件必须存在

char buf[MAX\_BUF];

//setASCIImode();

EnterPasvMode();

memset(buf, 0, MAX\_BUF);

sprintf(buf, "SIZE %s\r\n", fileName.c\_str());//是否需要完整路径?

int num = sock\_ctl.Send(buf, strlen(buf), 0);

memset(buf, 0, MAX\_BUF);

num = sock\_ctl.Receive(buf, MAX\_BUF, 0);// 成功示例：213 7\r\n 失败示例：550 Could not get the file size.\r\n

std::string str = buf;

int fileSize = -1;

//根据服务器上是否有数据选择合适的方式传输

if (num < 3)

{

throw FTPException(ExType::OtherFails, str);

}

else

{

if (containsCode(str, "550"))//不存在该目录

{

//就让fileSize = -1；

memset(buf, 0, MAX\_BUF);

sprintf(buf, "STOR %s\r\n", fileName.c\_str());//是否需要完整路径?

}

else if (containsCode(str, "213"))//存在该目录

{

fileSize = parseFileSize(str);

memset(buf, 0, MAX\_BUF);

sprintf(buf, "APPE %s\r\n", fileName.c\_str());//是否需要完整路径?

}

else

{

throw FTPException(ExType::OtherFails, str);

}

}

num = sock\_ctl.Send(buf, strlen(buf), 0);

memset(buf, 0, MAX\_BUF);

num = sock\_ctl.Receive(buf, MAX\_BUF, 0);

str = buf;

if (num < 3)

{

throw FTPException(ExType::OtherFails, str);

}

else if(!containsCode(str,"150"))

{

throw FTPException(ExType::UploadRefused,str);

}

fseek(pFile, fileSize, SEEK\_SET);

int nLen;

const unsigned long dataLen = MAX\_BUF;

char strBuf[dataLen] = { 0 };

while (!feof(pFile))

{

nLen = fread(strBuf, 1, dataLen, pFile);

if (nLen < 0)

{

break;

}

if ((num = sock\_data.Send(strBuf,strlen(buf),0)) < 0)

{

//报错

fclose(pFile);

throw FTPException(ExType::DataUploadFail);

}

}

sock\_data.Disconnect();

fclose(pFile);

memset(buf, 0, MAX\_BUF);

num = sock\_ctl.Receive(buf, MAX\_BUF, 0);//226

if (num < 3 || !containsCode(str, "150"))

{

throw FTPException(ExType::OtherFails, str);

}

return true;

}

重命名文件：

bool FTPClient::rename(std::string from, std::string to)

{

char buf[MAX\_BUF];

memset(buf, 0, MAX\_BUF);

sprintf(buf, "RNFR %s\r\n", from.c\_str());

int num = sock\_ctl.Send(buf, strlen(buf), 0);

memset(buf, 0, MAX\_BUF);

num = sock\_ctl.Receive(buf, MAX\_BUF, 0);//350

std::string str = buf;

if (num < 3)

{

throw FTPException(ExType::OtherFails,str);

}

else if (!containsCode(str, "350"))

{

throw FTPException(ExType::RNFRFail, str);

}

memset(buf, 0, MAX\_BUF);

sprintf(buf, "RNTO %s\r\n", to.c\_str());

num = sock\_ctl.Send(buf, strlen(buf), 0);

memset(buf, 0, MAX\_BUF);

num = sock\_ctl.Receive(buf, MAX\_BUF, 0);//250

str = buf;

if (num < 3)

{

throw FTPException(ExType::OtherFails, str);

}

else if (!containsCode(str, "250"))

{

throw FTPException(ExType::RNTOFail, str);

}

return true;

}

进入二进制传输模式：

bool FTPClient::setASCIImode()

{

char buf[MAX\_BUF];

memset(buf, 0, MAX\_BUF);

sprintf(buf, "TYPE I\r\n");

int num = sock\_ctl.Send(buf, strlen(buf), 0);

memset(buf, 0, MAX\_BUF);

num = sock\_ctl.Receive(buf, MAX\_BUF, 0);//200

std::string str = buf;

if (num < 3)

{

throw FTPException(ExType::OtherFails, str);

}

else if (!containsCode(str, "200"))

{

throw FTPException(ExType::SetBinModeFail, str);

}

return true;

}

**参考文献**

[1] CSDN用户：fang\_a\_kai，vsftpd安装与配置，网址：<https://blog.csdn.net/fang_a_kai/article/details/91045617> ，访问时间[2020.02.20]

[2] CSDN用户：缎锦小妹，使用Socket通信实现FTP客户端程序，网址：<https://blog.csdn.net/lijia11080117/article/details/56693378> ，访问时间[2020.02.21]

[3] 博客园用户：洪雨编程笔记，FTP协议（指令集），网址：<https://www.cnblogs.com/hongyuyingxiao/p/10486036.html> ，访问时间[2020.02.09]

[4] CSDN用户：码农诗人，FTP服务器主动被动模式详解，网址：<https://blog.csdn.net/ddazz0621/article/details/84890003> ，访问时间[2020.02.09]

[5] CSDN用户：The\_King\_Cloud，C++实现ftp客户端，网址：<https://blog.csdn.net/the_king_cloud/article/details/8090699> ，访问时间[2020.03.12]

[6] CSDN用户：lwei\_998，FTP的ASCII和Binary传输模式，网址：<https://blog.csdn.net/lwei_998/article/details/6025410> ，访问时间[2020.3.14]

|  |
| --- |
| 【结论】：此项目能成功地完成各种要求中的文件上传、下载功能，并且在其基础上，还实现了文件夹访问、删除文件及文件夹、创建新文件夹、重命名、上传及下载文件的断点续传等额外功能。 |
| 【小结】：在这次实验中，因为本组只有我一个人，我查阅了大量的资料，不仅完成了规定中的任务，还实现了很多额外的功能。当然，最重要的是，在此次实验中，我深入地了解了FTP协议的相关内容，对网络编程有了更深一步的理解。其实，我一个人组一队，一个原因是这个班上我不认识什么人，另一个原因是，我更想一个人将制作FTP客户端的整个过程走一遍，以便更深入地了解它。现在看来，我的选择也是非常正确的。 |
| 指导老师评语及成绩 |
| 【评语】：  成 绩： 指导老师签名：  批阅日期：2019.5.10 |