****

Python网络编程课程设计

实 验 报 告

|  |  |
| --- | --- |
| **学 院** | 网络空间安全学院 |
| **专 业** | 网络工程 |
| **班 级** | 17272412 |
| **学 号** | 17272231 |
| **学生姓名** | 吴挺锋 |
| **教师姓名** | 胡伟通 |
| **完成日期** | 2019/11/7 |
| **成 绩** |  |

**实验二 基于UDP的C/S程序设计**

1. **实验目的**
2. 熟悉UDP协议的原理和特点
3. 掌握基于UDP协议的应用层协议设计方法。
4. 掌握基于UDP协议的C/S网络应用程序设计和实现方法。
5. **实验内容**

编写基于UDP的文件传输程序（可以参考TFTP协议，不能照抄），实现上传文件到服务器和从服务器下载文件，要求实现以下功能：

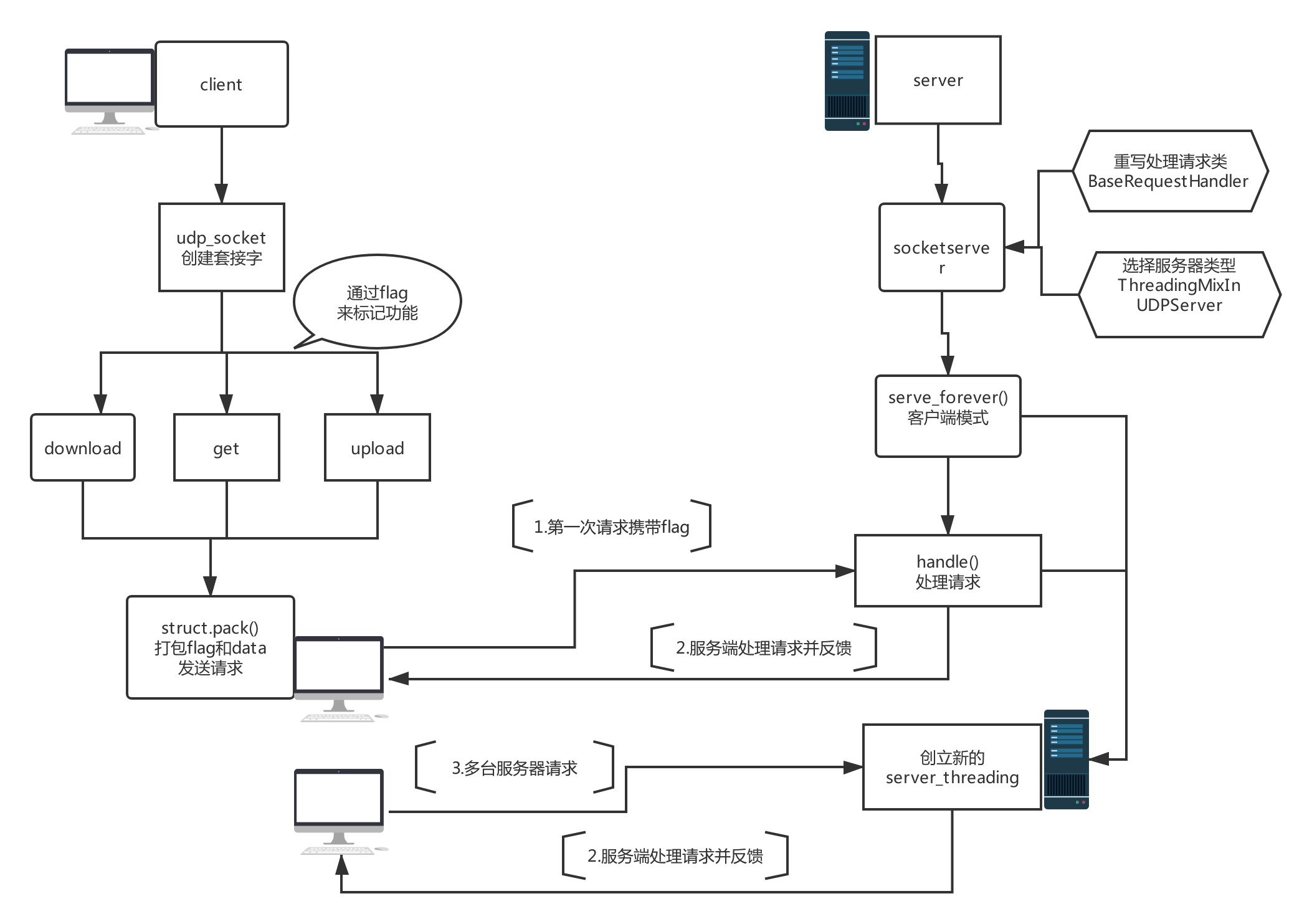
1. 获取文件列表
2. 上传文件
3. 下载文件

注意测试两个客户端同时进行上传和下载的情况

1. **程序设计思路**
2. **所选题目说明：**

基于udp协议的文件传输，客户端有三个功能，获取文件列表，upload，和download。故服务器端需要选定存取文件的路径，供客户端上传下载。文件上传和下载需要指定文件路径。然后涉及到文件的打开（open）、读取（read）、写入（write）操作。数据在传输层中以bytes流形式传播，故有时数据需要编码。文件列表利用OS模块解决。同时进行上传下载，故在传统的单循环socket套接字之上，考虑并发性。经查询，可以采用socketserver模块帮我们解决并发问题。

1. **网络应用拓扑结构**

****

1. **应用层协议设计**

* 基于UDP传输
* 第一次请求发送的固定格式:name size flag
* Flag：down\_flag,get\_flag,up\_flag
* 一次传输有最大字节1024，大文件会依次读取发送
* 双方发送文件数据时，发送方发送数据会等待对方ok，再继续发送

1. **所选用的Python库介绍**

①OS：用来对文件和目录进行操作

例如os.listdir()会帮我们返回当前目录下的所有文件以列表形式；os.chdir()则是改变当前路径，存取文件很有用

②Socket：套接字编程，不多赘述

③struct:打包不同类型的变量整合成比特流的形式

pack()打包，unpack()解包

④socketserver：支持tcp/udp分别以threading/forking两种形式的并发server

选用udp协议以多线程来处理并发，用到BaseRequestServer类，服务器类型是ThreadingMixin和UdpServer。

⑤argparse：处理cmd键入值，给函数传参，命令行完成程序功能

1. **程序源代码**

**注意源代码要有详细的注释。 同学们提交的每个程序都应该遵循Honor Code（诚实代码保证）的要求。**

**请大家特别注意一定要在每个程序首部的注释中加上以下保证：**

**# 我真诚地保证：**

**# 我自己独立地完成了整个程序从分析、设计到编码的所有工作。**

**# 如果在上述过程中，我遇到了什么困难而求教于人，那么，我将在程序实习报告中**

**# 详细地列举我所遇到的问题，以及别人给我的提示。**

**# 在此，我感谢 XXX, …, XXX对我的启发和帮助。下面的报告中，我还会具体地提到**

**# 他们在各个方法对我的帮助。**

**# 我的程序里中凡是引用到其他程序或文档之处，**

**# 例如教材、课堂笔记、网上的源代码以及其他参考书上的代码段,**

**# 我都已经在程序的注释里很清楚地注明了引用的出处。**

**# 我从未没抄袭过别人的程序，也没有盗用别人的程序，**

**# 不管是修改式的抄袭还是原封不动的抄袭。**

**# 我编写这个程序，从来没有想过要去破坏或妨碍其他计算机系统的正常运转。**

**# <学生姓名> 吴挺锋**

**服务器端源码**

import struct

import os,time

import socketserver

BUF\_SIZE = 1024#一次接受1024bytes

DOWN\_FLAG=0

UP\_FLAG=1

GET\_FLAG=2

PORT=8888

def unpack(data):

    name,size,flag=struct.unpack('20sII',data)

    return name,size,flag

#1.服务器该怎么处理请求

class UdpHandller(socketserver.BaseRequestHandler):

    def handle(self):

        count=0

        print('the task initiate at:',time.strftime('%Y-%m-%d %H:%M:%S',time.localtime(time.time())))

        if count==0:

            data=self.request[0]#取出bytes数据,其实是名字

            Fname,size,flag=unpack(data)#拆分结构体

            print("addr is :",self.client\_address)

            #server=self.request[1]

            os.chdir(r'C:\Users\wtf\Desktop\python网络编程\csdn\_udp\SeverList') #服务器端路径

            #1.处理上传请求

            #Do Upload\_Request

            if flag==UP\_FLAG:

                name=Fname.split(b'\x00')[0]

                if size>0:

                    print ('client ip：',self.client\_address,' File name is:',name,"字节大小是",size)

                    f=open(name,'wb')#写模式打开

                    self.request[1].sendto(b'ok',self.client\_address) #发送就绪,准备接受bytes数据

                else:

                    print("该文件为空")

                while True:

                    #NOTICE:这里的client\_addr与self.client\_address值相同

                    data,client\_addr=self.request[1].recvfrom(BUF\_SIZE)

                    if data!=b'end': #0-9，接受bytes

                        f.write(data)

                        print (str(count)+'.',client\_addr,'客户端',client\_addr,'上传文件中')

                        count+=1

                    else:

                        break

                #回复消息

                    self.request[1].sendto('ok'.encode('utf-8'),client\_addr)

                    count+=1

                print('循环了'+str(count),'文件',name,'上传完毕')

                f.close()

            #处理获取文件列表

            #2.Do GetList\_Request

            elif flag==GET\_FLAG:

                a=os.listdir()

                flist=''

                for line in a:

                    flist+=str(line)+'%'

                data=flist.encode('utf-8')

                self.request[1].sendto(data,self.client\_address)

                print('已向',self.client\_address,'发送我方列表')

            #处理下载请求

            #3.do Download\_Request

            elif flag==DOWN\_FLAG:

                #根据名字查找打开文件

                name=Fname.split(b"\x00")[0]

                down\_count=1

                f=open(name,'rb')

                while True:

                    data=f.read(BUF\_SIZE)

                    if data!=b'':

                        self.request[1].sendto(data,self.client\_address)

                        print('server are sending',down\_count,'to',self.client\_address)

                        down\_count+=1

                    else:

                        break

                    #data,client\_addr=self.request[1].recvfrom(BUF\_SIZE)#阻塞返回的ok

                self.request[1].sendto(b'end',self.client\_address)

                print('客户端下载结束')

                f.close()

#2选择一个合适的服务类型，mixin和udpsever，决定是多线程处理并发

class ThreadingUdpServer(socketserver.ThreadingMixIn,socketserver.UDPServer):

        #daemon\_threads=True

        def \_\_init\_\_(self,sever\_addr,RequestHandler):

            socketserver.UDPServer.\_\_init\_\_(self,sever\_addr,RequestHandler)

server=ThreadingUdpServer(('127.0.0.1',PORT),UdpHandller)

try:

    server.serve\_forever()

except KeyboardInterrupt:

    sys.exit(0)

**客户端源码**

import socket,struct

import sys,os

import argparse

BUF\_SIZE = 1024

DOWN\_FLAG=0

UP\_FLAG=1

GET\_FLAG=2

#客户端启动

def init\_client():

     client=socket.socket(socket.AF\_INET,socket.SOCK\_DGRAM)

     server\_addr=('127.0.0.1',8888)

     return client,server\_addr

#关闭套接字和文件

def terminate(client,f):

     f.close()

     client.close()

#封装flag标志结构体

def pack(name,size,flag):

     data=struct.pack('20sII',name,size,flag)

     return data

#client module 1.下载文件，先对路径处理路径

def do\_path(data\_path,flag):

     name=str(data\_path).split('\\')[-1]

     try:

          size=os.path.getsize(data\_path)  #获取文件大小

     except FileNotFoundError:

          print('该文件不存在')

     else:

          name=name.encode('utf-8')

          data=pack(name,size,flag)

     print(data\_path)

     print(size)

     return data

def getlist(data\_path):

     client,server\_addr=init\_client()

     data=pack(b'',0,GET\_FLAG)#前两位为0或空

     client.sendto(data,server\_addr)

     print('client sent request')

     data,addr=client.recvfrom(BUF\_SIZE)#等待信息返回

     print('the server files as fllows:')

     data=data.decode('utf-8')

     filel=data.split('%')

     for line in filel:

          print(line)

#上传请求

def upload(data\_path):

     #创建套接字

     client,server\_addr=init\_client()

     data=do\_path(data\_path,UP\_FLAG)#路径处理

     print(data)

     f=open(data\_path,'rb')

     count=0

     while True:

          if count==0:#计数还未开始，只执行一次

               client.sendto(data,server\_addr)

               recv\_data,addr = client.recvfrom(BUF\_SIZE)#等待server 返回就绪

               if recv\_data == b'ok':

                    print("客户端准备就绪")

          #循环执行

          data=f.read(BUF\_SIZE)#先读1024字节

          if data != b'':

          #每次按最大字节来读取发送

               client.sendto(data,server\_addr)

               count+=1

               print(str(count)+"byte")

          else:#读不出数据了

               client.sendto('end'.encode('utf-8'),server\_addr)

               break

          data, server\_addr = client.recvfrom(BUF\_SIZE)

     print('循环了'+str(count)+'次数据')#一共发送了机

     terminate(client,f)

#client module 2.实际下载

def download(name):

     #读取文件名

     os.chdir(r'C:\Users\wtf\Desktop\python网络编程\csdn\_udp\Download')

     client,server\_addr=init\_client()

     name=name.encode('utf-8')

     print(name)

     data=pack(name,0,DOWN\_FLAG)

     f=open(name,'wb')

     print('已准备创建新文件')

     client.sendto(data,server\_addr)

     print('已发送请求信息')

     count=0

     while True:

          data,server\_addr=client.recvfrom(BUF\_SIZE)

          count+=1

          if count==0:

               print('服务器正在向我们发送数据')

          elif data==b'end':

               print('接收完毕')

               break

          print('客户端正在从服务器',count,'下载')

          f.write(data)#写数据

          name=b'ok'

          data=pack(name,0,DOWN\_FLAG)

          #client.sendto(b'ok',server\_addr)#发送OK

     terminate(client,f)

def mess(mess):

     client=socket.socket(socket.AF\_INET,socket.SOCK\_DGRAM)

     server\_addr=('127.0.0.1',8888)

     mess=mess.encode('utf-8')

     client.sendto(mess,server\_addr)

     print('已发送信息')

     client.close()

if \_\_name\_\_ == '\_\_main\_\_':

     choices={'upload':upload,'get':getlist,'download':download,'test':mess}

     parser=argparse.ArgumentParser(description='Choose download or get')

     parser.add\_argument('role',choices=choices,help='which role to play')

     parser.add\_argument('path',metavar='path', type=str, default='',

      help='要上传的文件路径')

     args=parser.parse\_args()

     function=choices[args.role]

     function(args.path)

     #print(args.path)

1. **程序测试方法及测试结果记录（不能光截图，要有相应的文字说明）**
2. **测试方法**

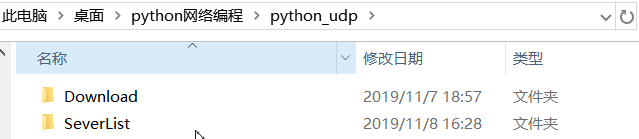
**①测试获取文件列表**

**②测试单客户端上传下载**

**③测试双客户端通上传下载**

1. **测试流程**

1.首先看下上传和下载的文件夹



2.服务器里头的文件，下载暂时为空

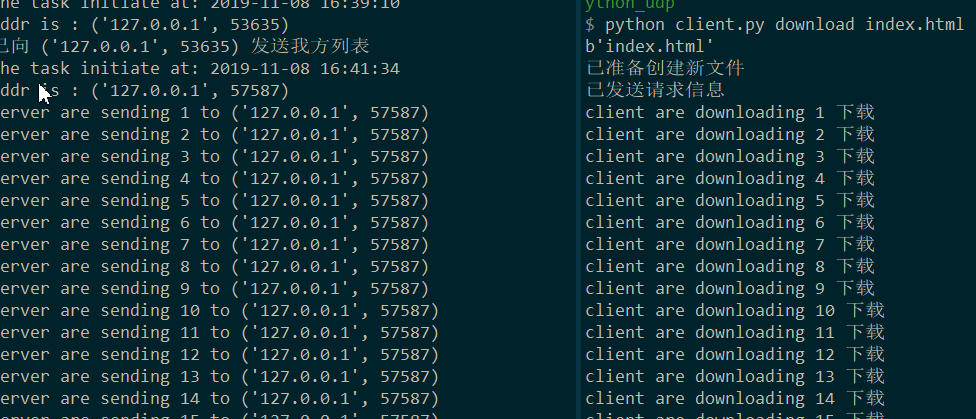


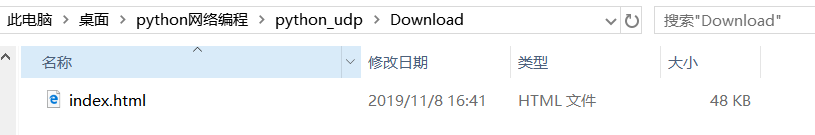


3.server开启，client请求文件列表成功

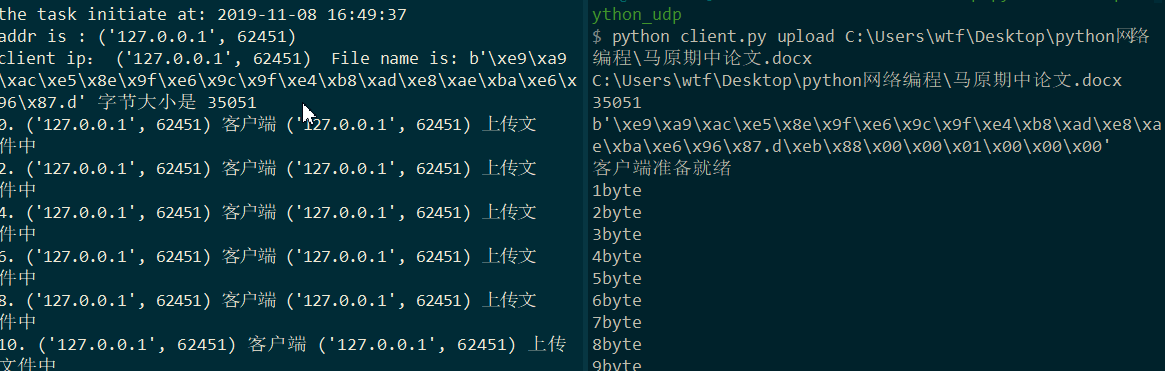


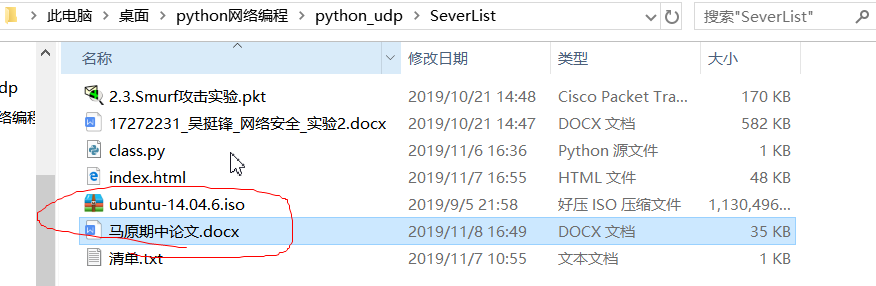
4.试试下载和上传。下载成功





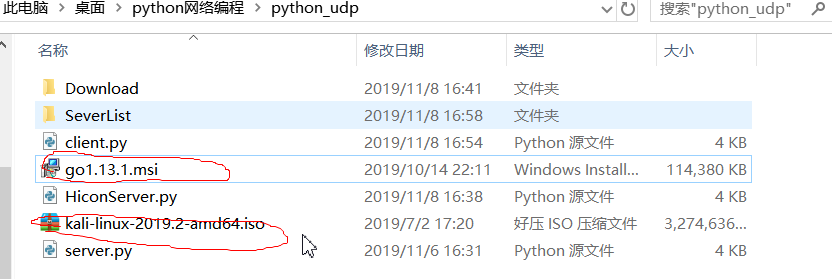
②论文上传，也没有问题

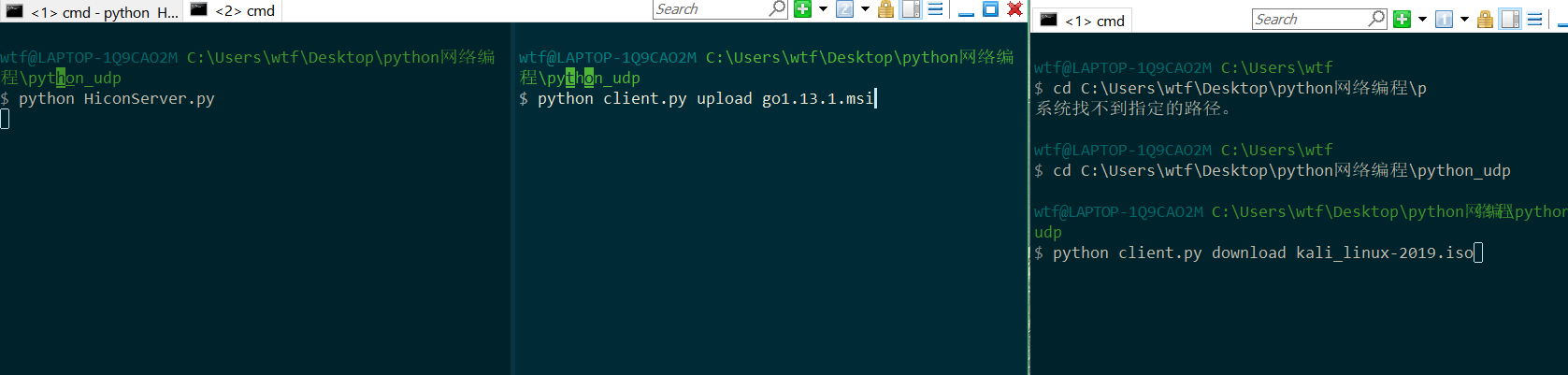


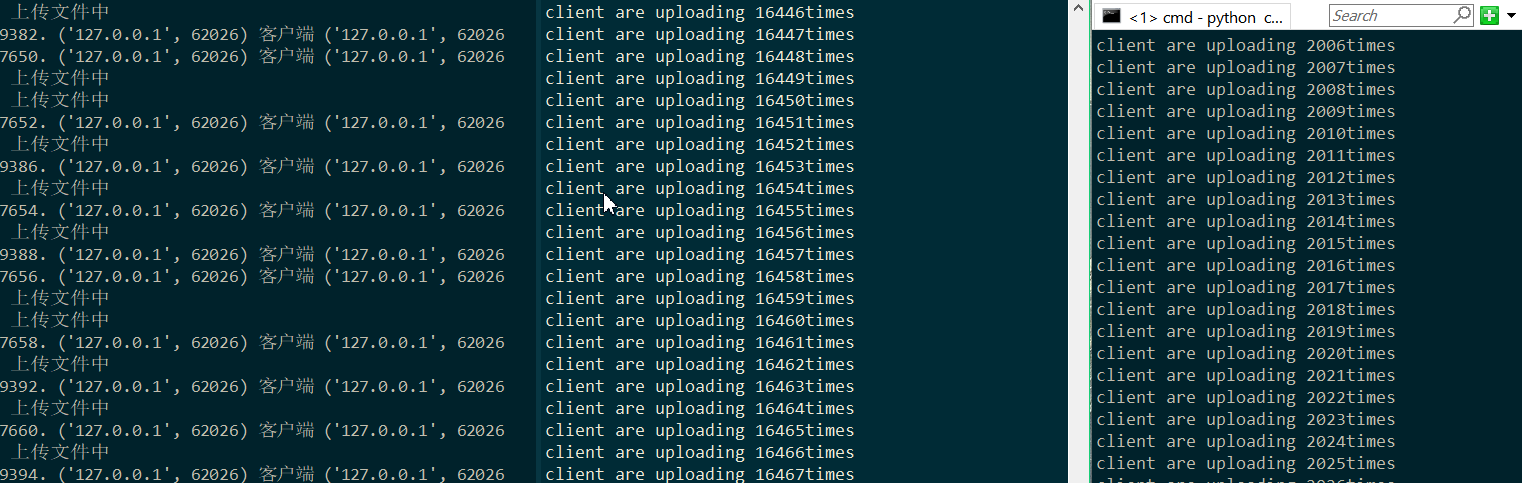


1. 两客户端同时下载上传

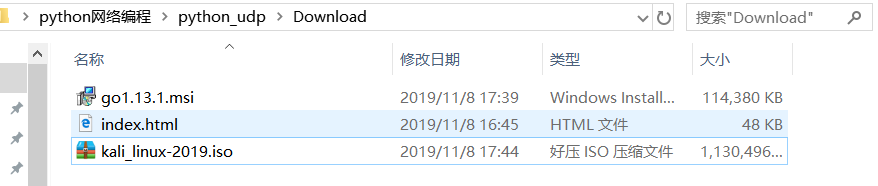
为了方便我准备好了两个文件要同时上传，看看客户端会怎么应答吧

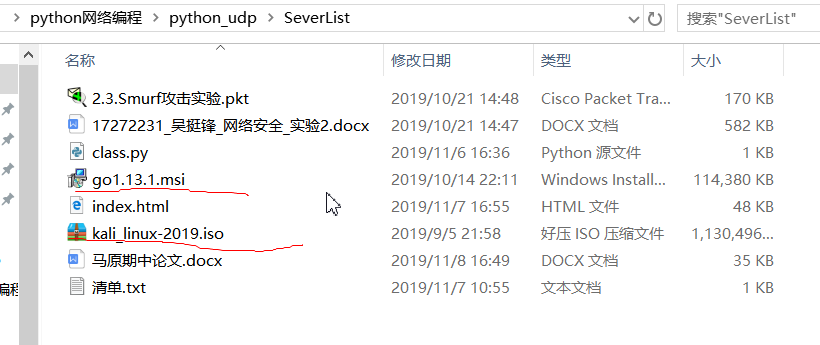




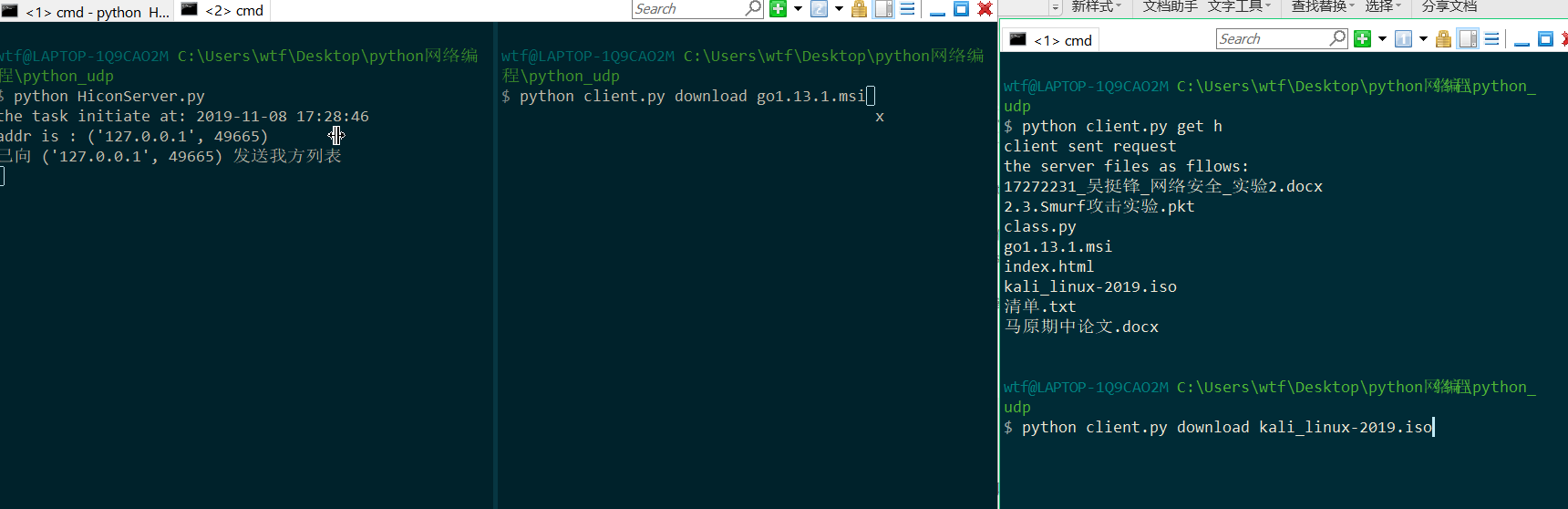


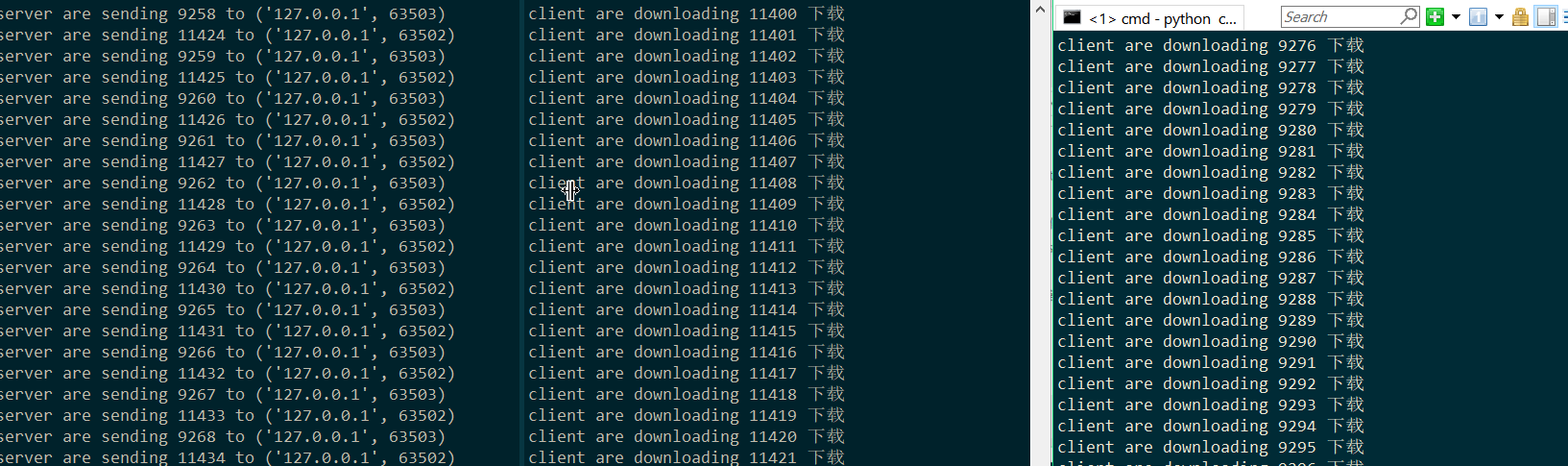
上传成功，再把这两个分别下载回来





准备同时下载





1. **实验分析总结及心得**

（结合所学知识对实验过程中观察到的实验结果进行分析总结，以便加深对知识的理解，并总结通过实验学到的知识或技术）

①客户端的功能分类，一开始想的也是在命令行操作，于是用的sys.args[]数组获取参数，但是后面发现不好通用，为了方便我参考了课本上的argparse模块，模仿着写了功能，我发现这个模块真的很实用方便，但还只是浅尝辄止。

②怎么向客户端发送我的标识码信息，请求下载肯定要把要下载的文件名和代表下载的标识符发送过去，那么该怎么发呢，于是在网上找到了struct模块，可以打包数据生成比特率=流，这样就可以直接传送，到了服务器在解包提取出标识符，就可以做出相应的处理了。

③同时性带来的并发问题，如果是小文件，我原来的代码是看不差别的，毫秒级的速度，但是遇上同时进行大文件操作，必然有个客户端被拒绝连接。SocketServer是专门支持并发服务器的模块，我适当阅读了源码和网上示例，重写了handle()方法处理请求，最终是采取UDP协议多线程模式来处理并发的。