简单云盘系统的实现

姓名：冯冠玺 学号：15051415

一、简要说明

该结课报告设计了一个简单的云盘，其主要功能有：1.用户登录与密码验证 2.查看当前网盘里的文件 3.用户可上传文件到云盘 4.用户可从云盘中下载文件。其中在上传文件功能中，增加了MD5码的内容，从而实现“极速秒传”的效果。

二、运行环境

> PC系统：ubuntu-18.04-desktop
> IDE :VS code
> 开发语言：Python2.7

三、运行环境的准备工作

安装Python2.7:

> 安装Python2.7 dev包：
> `sudo apt-get install python2.7 python2.7-dev`
> 安装build依赖包：
> `sudo apt-get install build-essential libssl-dev libevent-dev libjpeg-dev libxml2-dev libxslt-dev`
> 安装pip
> `sudo apt-get install python-pip`

安装Tkinter:

> `sudo apt-get install python-tk`

四、设计分析

1.整体要求

搭建一个简单的云盘系统，完成客户端与服务器端的实现，客户端能够向服务器端发送一系列的文件操作请求和下载上传请求，服务器端接收客户端发出的请求，并经过一系列的处理，反馈信息给客户端。

2.服务端要求

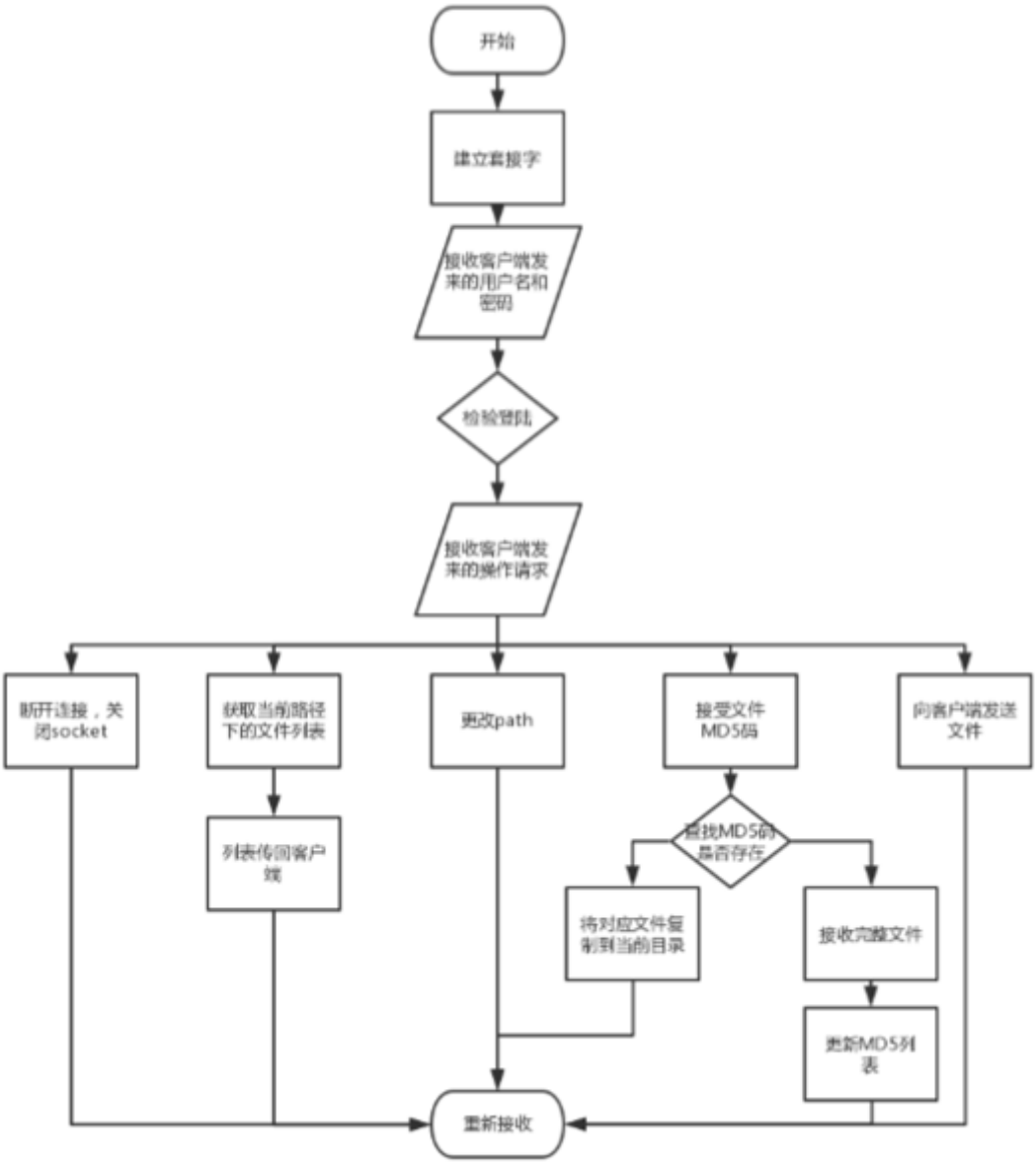
```
1）建立一个套接字，并给其分配一个端口，准备接收客户端的连接请求
2）接收客户端发来的加密用户名密码，解密并判断是否正确
3）接收客户端发来的一系列请求（ls，cd，download，upload，exit），处理这些请求并反馈信息给客户端
4）上传过程中与客户端合作，实现,MD5简单的极速秒传功能
```

3.客户端要求

```
1）建立一个套接字，同服务器端建立连接
2）输入用户名和密码，发送给服务器端，其中密码加密后发送，接收服务器返回信息，判断是否登陆成功
3）发送一系列请求(ls，cd，download，upload，exit)给服务器，等待反馈信息
4）上传过程中与服务器端合作，实现MD5简单的极速秒传功能
```

五、流程图设计

服务端流程图设计：



客户端流程图与服务端流程图类似，所以不再附图。

六、核心模块设计

文件`FileServer.py`含有以下主要函数：

> `GetsLs(path)`：
> 功能：获取当前路径下文件和文件夹列表
> 输入参数：当前路径
> 返回值：文件和文件夹列表
> 操作：使用OS模块进行相关文件操作
>
> `IfExit(sock,addr)`：
> 功能：收到客户端的exit请求，断开连接并返回信息
> 输入值：TCP连接使用的socket，介入客户端的地址
> 返回值：无
>
> `IfLs(sock,path)`：
> 功能：收到客户端的ls请求，获取当前文件夹的文件列表并返回给客户端

输入值：TCP连接使用的socket，当前路径
返回值：无

`IfCd(sock,pat)`：
功能：收到客户端的Cd请求，更改全局变量path的值，在其后加入cd的目录，以便ls命令使用
输入值：TCP连接使用的socket，当前路径
返回值：无

`IfUpload(sock,path)`：
功能：收到客户端的upload请求，首先接收文件MD5码，如果有直接执行cp即可；如果没有，服务器接收客户端发来的文件，存在path目录，并更新md5list
输入值：TCP连接使用的socket，当前路径
返回值：无

`IfDownload(sock,path)`：
功能：收到客户端的download请求，向客户端发送文件
输入值：TCP连接使用的socket，当前路径
返回值：无

`IsLogin(sock)`：
功能：收到客户端发来的用户名密码，验证是否正确，返回登陆信息
输入值：TCP连接使用的socket，当前路径
返回值：无

文件FileClient.py含有如下主要函数：

`IfLs(s,se)`：
功能：向服务器发起ls请求，收到返回信息表示当前文件目录结构
输入值：TCP连接使用的socket，输入的命令

`IfCd(s,se)`：
功能：向服务器发起cd请求，收到返回信息表示cd执行结果，若失败，输出Error Dir!
输入值：TCP连接使用的socket，输入的命令

`IfExit(s,se)`：
功能：向服务器发起exit请求，断开连接并退出当前客户端
输入值：TCP连接使用的socket，输入的命令

`IfUpload(s,se)`：
功能：向服务器发起upload请求，使用tk库，打开图形化文件管理器选择文件并发送
输入值：TCP连接使用的socket，输入的命令

`IfDownload(s,se)`：
功能：向服务器发出download请求，收到服务器发来的文件，并下载到当前目录
输入值：TCP连接使用的socket，输入的命令

`Login(s)`：
功能：向服务器发送用户名密码，返回是否登陆成功

> 输入值：TCP连接使用的socket
> 返回值：是否登陆成功

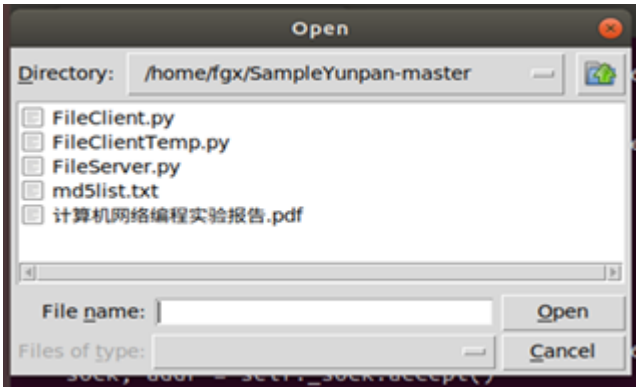
七、运行效果展示

1.文件上传

输入upload指令：

```
fgx@ubuntu:~/SampleYunpan-master$ python FileClient.py
Please Login First
Your Username: admin
Your Password:
Please Input Next Choose:
    1) ls                -- list all files and dirs
    2) cd <dir>          -- change dir
    3) download <file>   -- download your files
    4) upload            -- upload your files
    5) exit              -- exit our client
>>upload
```

进入图形界面开始选择待上传的文件：

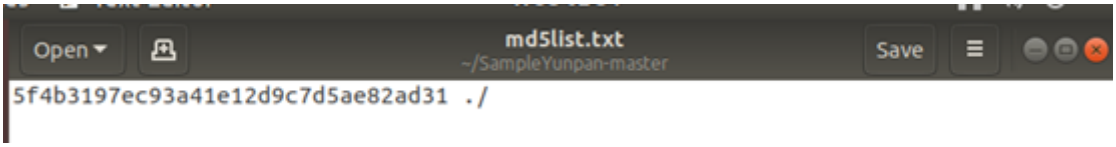


选择文件后进行上传（客户端输出上传成功，服务器端输出接收成功）：

```
>>upload
upload file success!
```

```
Accept new connection from 127.0.0.1:41574
starting reve file!
recv file success!
```

上传文件的MD5也已经写入到MD5表中：



2.进行文件的查看

之前上传的是222.pdf文件，我们在客户端向服务器端发送ls查看当前文件夹下的文件指令，可以看到222.pdf已经保存在服务器端

```
>>ls
222.pdf md5list.txt 计算机网络编程实验报告.pdf FileServer.py FileClient.py File
ClientTemp.py
```

3.cd命令的测试

在服务器端建立了一个test空文件夹，然后客户端输入 cd ./test进入到该文件夹下：

```
>>cd ./test
```

然后我们向该空文件夹上传同样的222.pdf，可以看到服务器端的输出：

```
starting reve file!
MD5码匹配成功，进行极速秒传
recv file success!
```

这是因为上传的文件与之前的文件一样。MD5匹配成功了，于是直接在服务器端进行了复制操作。

同样的，我们使用ls对之前的空文件夹进行查看：

```
>>ls
222.pdf
```

可以看到，空目录里的确出现了222.pdf

4.download命令的测试

客户端：

```
>>download md5list.txt
downloading~~
download file success!
>>
```

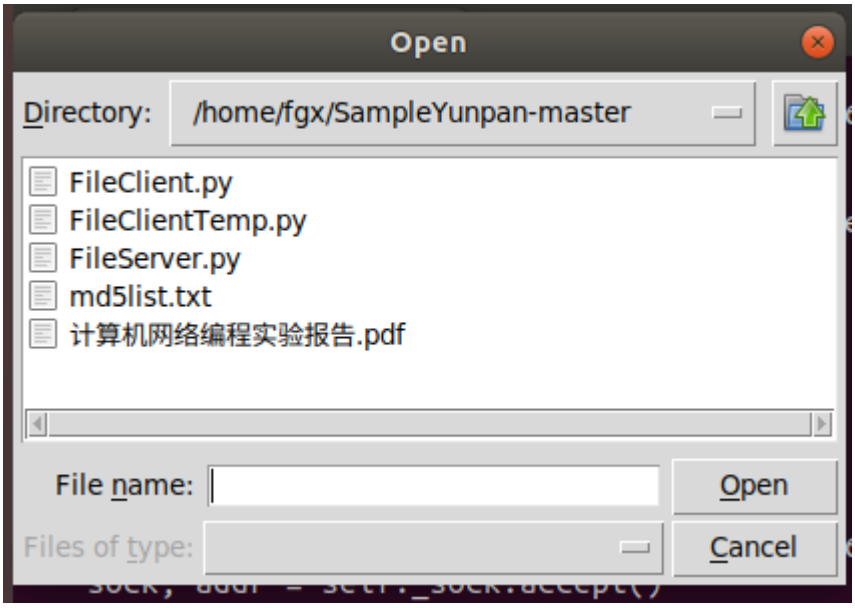
服务器端：

```
starting send file!
send file success!
```

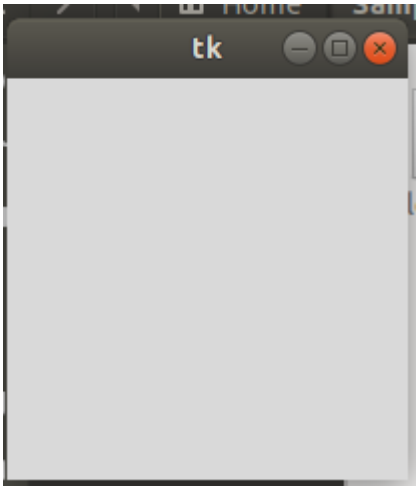
八、疑难解决

1.Tkinter图形对话框无法自动关闭

Tkinter.askopenfilename是Tkinter库内置的一个函数，它的作用是添加一图形界面的形式获取文件路径：



然而，在选中文件之后，会出现一个空白对话框，这个空白对话框必须手动关闭才行。



解决方式是，先使用Tk.tk()函数建立一个主窗口，然后Tkinter.askopenfilename()返回文件路径之后，再使用destroy函数进行销毁。

2.密码使用非明文表示

一开始我是自己写了一个非常简单的加密算法，比如把e字符在键盘上向右移动两位，变为t，但是这种加密性实在是不好，后来想到在电子邮件中经常使用的base64模块，它就是专门把那些不能使用明文传输的数据进行编码加密，而后传输到服务器端，再进行解码即可，码长仅增加了需要传输数据的1/3.实现起来也很简单，只需要`base64.encodestring()`和`base64.decodestring()`两个函数即可。

3.文件传输终止

在进行文件上传或下载时，无法确定何时终止，后来想到在文件传输结束时，再输出一个”EOF“字符串，当收到”EOF”时即进行终止。

九、优缺点

优点：

1.密码非明文表示
2.通过MD5的方式实现文件秒传

缺点:

1.缺少删除文件的功能
2.真正的秒传应该仅是分配一个共享权限（原始文件仍为一份），而我这里使用的是复制到对应的路径的方法

十、心得体会

感谢吴老师的半年来的教导，吴老师在大一时就曾教过我计算机科学导论，激起了我学习计算机的兴趣，而网络编程课也是让我认识了计算机科学的新一领域。这次结课设计让我收获颇丰，使我对这门课的认识更深了一步，同时还激起了我对网络编程的兴趣，希望以后我能继续在计算机科学道路上越走越远。

代码附录

客户端

```
#-*-coding:utf8-*-
import socket
import sys
import os
import Tkinter, tkFileDialog
import time
import getpass
import base64
import md5

# 向服务器发起ls请求，收到返回信息表示当前文件目录结构
def IfLs(s, se):
    s.send(se[0])
    print s.recv(1024)

# 向服务器发起cd请求，收到返回信息表示cd执行结果，若失败，输出Error Dir!
def IfCd(s, se):
    s.send(se[0])
    s.send(se[1])
    rec = s.recv(1024)
    if rec == 'Error Dir!':
        print rec

# 向服务器发起exit请求，断开连接并退出当前客户端
def IfExit(s, se):
    s.send(se[0])
    print s.recv(1024)
```

```
# 向服务器发起upload请求，使用tk库，打开图形化文件管理器选择文件并发送
def IfUpload(s, se):
    s.send(se[0])
    root = Tkinter.Tk() #自己加的

    fi = tkFileDialog.askopenfilename()

    root.destroy()  #自己加的

    if fi == '': return
    # print filename
    filename = fi.split('/')[-1]
    s.send(filename)
    m = md5.new()
    m.update(filename)
    with open(fi, 'rb') as f:
        while True:
            data = f.read(1024)
            if not data:
                break
            m.update(data)
    s.send(m.hexdigest())

    with open(fi, 'rb') as f:
        while True:
            data = f.read(1024)
            if not data:
                break
            s.sendall(data)
        time.sleep(1)
        s.sendall('EOF')
        print "upload file success!"

# 向服务器发出download请求，收到服务器发来的文件，并下载到当前目录
def IfDownload(s, se):
    s.send(se[0])
    filename = se[1]
    s.send(filename)
    print "downloading~~"
    with open(filename, 'wb') as f:
        while True:
            data = s.recv(1024)
            if data == 'EOF':
                print "download file success!"
                break
            f.write(data)

# 向服务器发送用户名密码，返回是否登录成功
def Login(s):
    print 'Please Login First'
    cnt = 3
```

```
    isLogin = False
    while cnt > 0:
        username = raw_input("Your Username: ")
        s.send(username)
        passwd = getpass.getpass('Your Password: ')
        passwd = base64.encodestring(passwd)
        s.send(passwd)
        isLogin = s.recv(1024)
        if isLogin == 'OK':
            return True
        cnt = cnt -1
        print 'Error Login! You have %d times.' %cnt
    return False

if __name__ == "__main__":
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(('127.0.0.1', 9999))

    if Login(s) is False:
        s.close()
        exit()

    print '''Please Input Next Choose:
    1) ls               -- list all files and dirs
    2) cd <dir>         -- change dir
    3) download <file>  -- download your files
    4) upload           -- upload your files
    5) exit             -- exit our client'''

    while True:
        data = raw_input(">>")
        se = data.split()
        if se[0] == 'ls':
            IfLs(s, se)
        elif se[0] == 'cd':
            IfCd(s, se)
        elif se[0] == 'exit':
            IfExit(s, se)
            break
        elif se[0] == 'upload':
            IfUpload(s, se)
        elif se[0] == 'download':
            IfDownload(s, se)
        else:
            print 'Error Input!'

    s.close()
```

服务端

```python
#-*-coding:utf8-*-
import socket
import threading
import subprocess
import sys
import os
import time
import base64
import shutil

md5list = 'md5list.txt'

# 获取当前路径下文件和文件夹列表
def GetLs(path):
    fileList = os.listdir(path)
    result = []
    for f in fileList:
        if os.path.isdir(f):
            f = f + '/'
        result.append(f)
    return result

# 收到客户端的exit请求，断开连接并返回信息
def IfExit(sock, addr):
    sock.send('Bye!')
    sock.close()
    print ('Connection from %s:%s closed.' % addr)

# 收到客户端的ls请求，获取当前文件夹的文件列表并返回给客户端
def IfLs(sock, path):
    # print 'IfLs ' + path
    fileList = GetLs(path)
    list_file = ' '.join(fileList)
    # print list_file
    sock.send(list_file)

# 收到客户端的cd请求，更改全局变量path的值，在其后加入cd的目录，一遍ls命令中使用
def IfCd(sock, path):
    rec = sock.recv(1024)
    if rec == "..":
        tmp = path.split('/')
        if path[-1] == '/':
            tmp = tmp[:-2]
        else:
            tmp = tmp[:-1]
        newpath = '/'.join(tmp) + '/'
        if path == '.':
            newpath = '.'
    elif rec[0] == '.':
        if rec[-1] == '/':
            newpath = rec
```

```
            else:
                newpath = rec + '/'
        else:
            if rec[-1] == '/':
                newpath = path + '/' + rec
            else:
                newpath = path + '/' + rec + '/'
        if os.path.isdir(newpath):
            sock.send('OK')
            return newpath
        else:
            sock.send('Error Dir!')
            return path

    # 收到客户端的upload请求，首先接受文件MD5码，与md5list比较是否存在相同的MD5码，如果有直接执行cp即可；如果没有，服务器接收客户端发来的文件，存在path目录，并更新md5list
    def IfUpload(sock, path):
        print ("starting reve file!")
        filename = sock.recv(1024)
        m = sock.recv(1024)
        with open(md5list, 'rb') as f:
            while True:
                str = f.readline()
                if not str:
                    break
                list = str.split()
                if m == list[0]:
                    src = list[1] + filename
                    det = path
                    print("MD5码匹配成功，进行极速秒传")
                    shutil.copy(src, det)
            print ("recv file success!")
                    return

        with open(md5list, 'a+') as f:
            str = m + ' ' + path + '\n'
            f.write(str)
        path = path + filename
        # print path
        with open(path, 'wb') as f:
            while True:
                data = sock.recv(1024)
                if data == 'EOF':
                    print ("recv file success!")
                    break
                f.write(data)

    # 收到客户端的download请求，向客户端发送文件
    def IfDownload(sock, path):
        print ("starting send file!")
        filename = sock.recv(1024)
        path = path + filename
```

```
    with open(path, 'rb') as f:
        while True:
            data = f.read(1024)
            if not data:
                break
            sock.send(data)
    sock.send('EOF')
    print ("send file success!")

# 收到客户端发来的用户名密码，验证是否正确，返回登录信息
def isLogin(sock):
    cnt = 3
    while cnt > 0:
        username = sock.recv(1024)
        passwd = sock.recv(1024)
        passwd = base64.decodestring(passwd)
        if username == 'admin' and passwd == 'admin':
            sock.send('OK')
            return True
        else:
            sock.send('FAIL')
        cnt = cnt -1
    return False

# 多线程要执行的main函数，实现上述功能
def main(sock, addr, path):
    print ('Accept new connection from %s:%s' % addr)
    cnt = 0

    if isLogin(sock) is False:
        return

    while True:
        data = sock.recv(1024)
        if data == 'exit':
            IfExit(sock, addr)
            break
        elif data == 'ls':
            print (path)
            IfLs(sock, path)
        elif data == 'cd':
            path = IfCd(sock, path)
            # print path
        elif data == 'upload':
            IfUpload(sock, path)
        elif data == 'download':
            IfDownload(sock, path)

if __name__ == "__main__":
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.bind(('127.0.0.1', 9999))
    s.listen(5)
```

```
    print ('Waiting for connection...')

    while True:
        sock, addr = s.accept()
        path = './'
        t = threading.Thread(target=main, args=(sock, addr, path)) # 多线
程
        t.start()
```