基于TCP的服务器/客户端 设计文档

180110723-沈文心

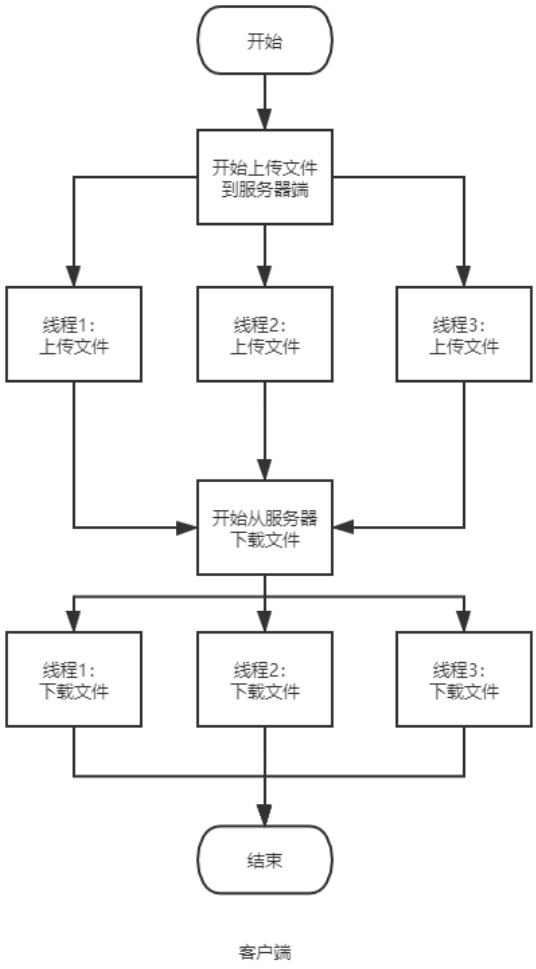
一 设计思路

1 总体思路

该题目要求构建能够实现文件传输的服务器和客户端。

还要采取多线程的方式。

设计思路如下：



客户端

按照该思路搭建服务跟客户端即可。

2 服务器端

首先，需要在服务器端创建一个基于IPv4和TCP协议的Socket

```
s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
```

然后，要绑定监听地址和端口。可以用127.0.0.1绑定带本机地址。因为这是一个特殊的IP地址，可以表示本机地址。端口号需要预先指定，因为这个服务不是标准服务，选用9999这个端口号。

```
s.bind(('127.0.0.1', 9999))
```

调用litsen()方法监听端口，设置等待连接的最大数量为5：

```
s.listen(5)
print('Waiting for connection...')
```

接下来，服务器程序通过一个永久循环来接受来自客户端的连接，`accept()`会等待并返回一个客户端的连接:

```
while True:
    # 接受一个新连接:
    sock, addr = s.accept()
    # 创建新线程来处理TCP连接:
    t = threading.Thread(target=tcplink, args=(sock, addr))
    t.start()
```

每个连接都必须创建新线程（或进程）来处理，否则，单线程在处理连接的过程中，无法接受其他客户端的连接。根据要求，对每个线程要打印出其线程号和线程名。

```
def tcplink(sock, addr):
    t = threading.currentThread()
    print('thread id : %d' % t.ident)
    print('thread name : %s'% t.getName())
    print('Accept new connection from %s:%s...' % addr)
    sock.send(b'Welcome!')
    ...
```

然后服务器等待客户端给其发送操作名`"upload/download"`

然后根据操作不同进行不同的操作。

2.1 upload

如果操作是`upload`，服务器等待客户端给其发送`文件大小/文件名`。

```
data = sock.recv(1024)
filesize = int(data.decode('utf-8'))
data = sock.recv(1024)
filename = str(data.decode('utf-8'))
```

然后根据文件名，进行写操作,写完之后将其关闭：

```
path = str(filename + "_s_receved" + ".txt")
f = open(path,'wb')
while recsize < filesize:
    data = sock.recv(1024)
    f.write(data)
    ecsize += len(data)
    print('%s receving %d byte'% (filename,recsize))
f.close()
print('finished,%s revieved %s bytes'%(filename,recsize))
sock.send(b'ok')
ock.close()
```

2.2 download

如果操作时`download`，服务器给客户端发送`文件大小/文件名`

```
sock.send(filesize.encode('utf-8'))
sock.send(filename.encode('utf-8'))
```

然后向客户端发送文件内容：

```
f = open(path,'rb')
start = time.time()
for line in f:
    sock.send(line)
    f.close()
    end = time.time()
    print('file %s upload finished ,cost %s s,upload %d bytes' % (filename,str(round(end - start, 2)),int(filesize)))
sock.close()
print ('Connection from %s:%s closed.' % addr)
```

3 客户端

先创3个线程，每个线程都用来上传文件：

```
print('Try to upload file to server\n')
for i in range(1,filenum + 1):
    t = threading.Thread(target = file_upload,args = (i,))
    t.start()
    t.join()
```

上传文件的过程和服务器端基本相同，不在叙述，具体代码如下：

```
filenum = 3
def file_upload(fid):
    upsize = 0
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)#对每一个文件传输启用一个client
    s.connect(('127.0.0.1', 9999))# 建立连接:
    data = s.recv(1024)
    # print(s.recv(1024).decode('utf-8'))# 接收欢迎消息
    s.send(b'upload')

    filename = "file"+str(fid) # 拼接文件名字
    path =filename +".txt"
    filesize = str(os.path.getsize(path))#文件大小

    print('begin to upload file %s to server,size : %s bytes' % (filename,filesize))
    s.send(filesize.encode('utf-8'))#传文件大小
    f = open(path,'rb')
    s.send(filename.encode('utf-8'))#传文件名字
    start = time.time()
    for line in f:# 传文件
        s.send(line)
        upsize += len(line)
    data = s.recv(1024)
    if data.decode('utf-8') == 'ok':
        f.close()
```

```
    s.close()
    end = time.time()
    print('%s upload finished ,cost %s s,upload %d bytes' % (filename,str(round(end - start, 2)),int(upsize)))
```

阻塞主线程，等到3各自进程全部执行结束在继续执行。

然后继续创建3个线程，每个线程都用来下载文件：

```
print('Try to download file from server\n')
for i in range(1,filenum + 1):
    t = threading.Thread(target = file_download,args = (i,))
    t.start()
    t.join()
```

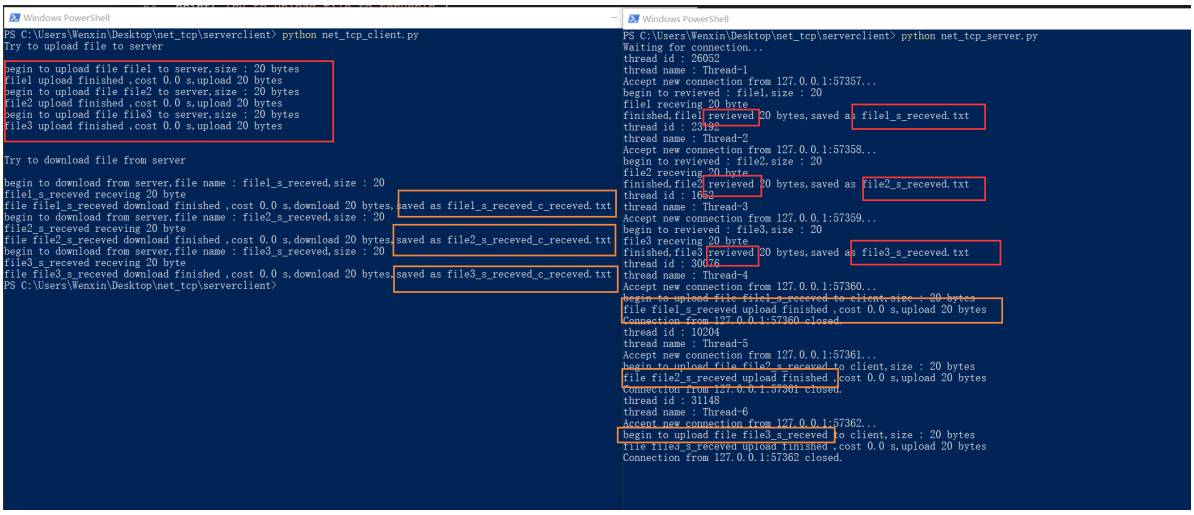
下载文件部分也不再叙述，具体实现如下：

```
def file_download(fid):
    recsize = 0
    sock = socket.socket(socket.AF_INET, socket.SOCK_STREAM)#对每一个文件传输启用一个client
    sock.connect(('127.0.0.1', 9999))# 建立连接:
    data = sock.recv(1024)
    sock.send(b'download')#下载
    sock.send(str(fid).encode('utf-8'))
    data = sock.recv(1024)
    filesize = int(data.decode('utf-8'))#文件大小
    data = sock.recv(1024)
    filename = str(data.decode('utf-8'))#文件名
    print('begin to download from server,file name : %s,size : %d'%(filename,filesize))
    path = str(filename + "_c_receved" + ".txt")
    f = open(path,'wb')
    start = time.time()
    while recsize < filesize:
        data = sock.recv(1024)
        f.write(data)
        recsize += len(data)
        print('%s receving %d byte'% (filename,recsize))
    f.close()
    end = time.time()
    print('file %s upload finished ,cost %s s,download %d bytes' % (filename,str(round(end - start, 2)),int(recsize)))
    sock.close()
```

二、实验结果

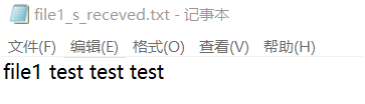
先在文件夹里新创3个文件给客户端用以发送：

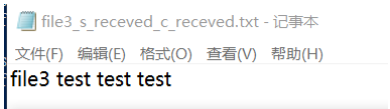
然后启动客户端和服务器端。可得如下结果：

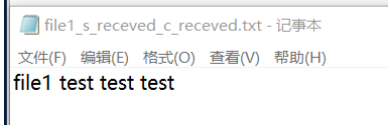


可知，客户端先向服务器发送了`file1.txt，file2.txt,file3.txt`，服务器端接收到后分别存为`file1_s_recieved.txt,file2_s_recieved.txt,file3_s_recieved.txt`.

然后客户端再向服务器申请下载`file1_s_recieved.txt,file2_s_recieved.txt,file3_s_recieved.txt`，并将其存为``file1_s_recieved_c_recieved.txt,file2_s_recieved_c_recieved.txt,file3_s_recieved_c_recieved.txt`







经检查，传输内容无误。

二、心得体会

这次试验总体来说比较简单。

python用起来真舒服。

还学会了如何搭建基于tcp的客户端/服务器，收获很多。