

基于 socket 编程的文件传输协议 (FDTP)

——数据通信与计算机网络课程设计

11300240041 朱为开

11300240050 王晗谦

概述：

FDTP 是一种面向应用层的网络协议，主要用于文件传输与访问服务器目录，我们设计的 FDTP 协议，使其可以满足以下几个功能：客户端与服务器之间的简单文件传递，get 和 put，以及通过 query 来查询服务器目录，自定义存放目录在 put 中实现

协议内容：

1.请求报文：

1) 请求行：

本协议支持 3 种请求，分别对应上传文件，下载文件，以及查询服务器目录，具体形式如下：

put_filename_path：将文件 filename 放到服务器 path 目录下

get_filename_path：将文件服务器 path 目录下的 filename 文件 get 到本地

query：查看服务器存放目录

其中 filename 为文件名，path 为路径。用“_”作为分隔符

2) 请求报头：

query 时无，get，put 时需要进行文件传输，报头为

Type：“128S11I”前 128 个字节为文件名，后面接 11 个 integer

Filename：128 位以内字符串为文件名

Size：文件大小

3) 请求正文：

query 请求无请求正文

其余请求正文即所传文件信息

2.响应报文：

1) 状态行

200 query ok：查询成功

201 put ok：上传成功

202 get ok：下载成功

400 code WA：请求代码错误

401 No File：服务器上无此文件

2) 响应报头

Query 时无, get, put 时需要进行文件传输, 报头为
Type : “128S32S11I”
Filename: 128 位以内字符串为文件名
Size: 文件大小

3) 响应正文

对于 query, 成果返回服务器所有目录及各文件, 其余成功传递文件信息, 发生错误返回错误码

协议实现：

1.背景技术

本协议使用套接字(Socket)对象实现。

套接字用于在客户端与服务器端之间建立通信链。套接字的连接过程可以大致分为三个步骤：a、服务器端的套接字对网络进行监听；b、客户端的套接字通过指明服务器端的套接字的地址和端口, 向服务器端的套接字发起连接请求；c、服务器端的套接字响应客户端的套接字所发出的请求, 建立一个新的线程并确认连接, 然后转为监听状态, 等待接受其他客户端的套接字的连接请求。

2.实现思路：

实现该协议的代码分为客户端与服务器端两部分。

在客户端, 代码会获取请求报文所需要的各种信息, 创建 Socket 对象, 尝试与服务器端进行连接, 并根据用户输入的请求指令以及其他信息构造报文字符串, 再将其发送给服务器。接收到服务器的响应报文后, 再对其进行解析。

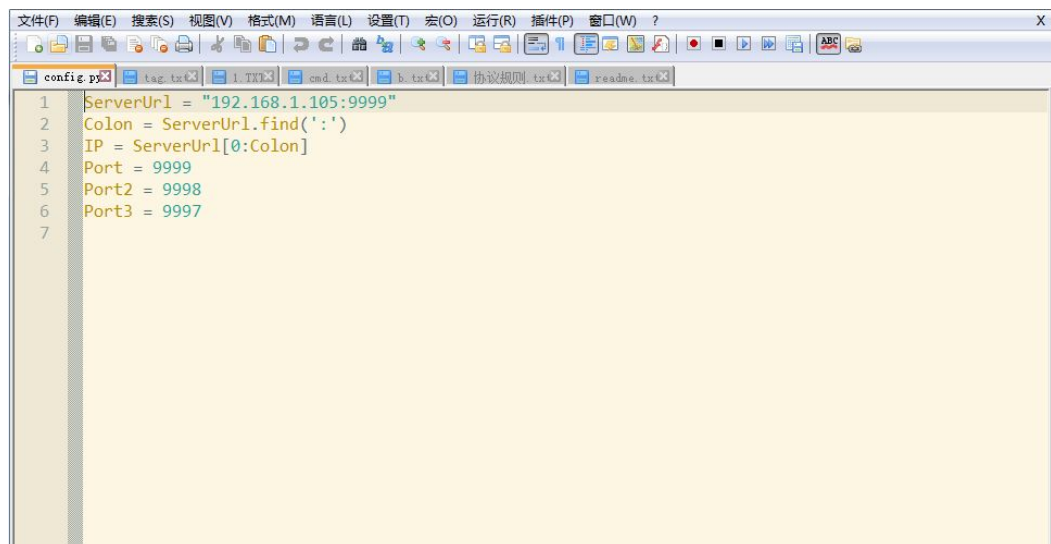
在服务器端, 利用监听进程获取客户端的请求, 随后对其进行解析, 解析过程中如发现错误, 则对错误进行相应的处理。然后根据响应的正确/错误信息和对请求指令的响应结果等构造报文字符串, 再将其返回给客户端。

3.具体代码实现：

本次代码使用 python 编写, 主要使用了 socket 库, 在 code 文件夹下有各部分代码:

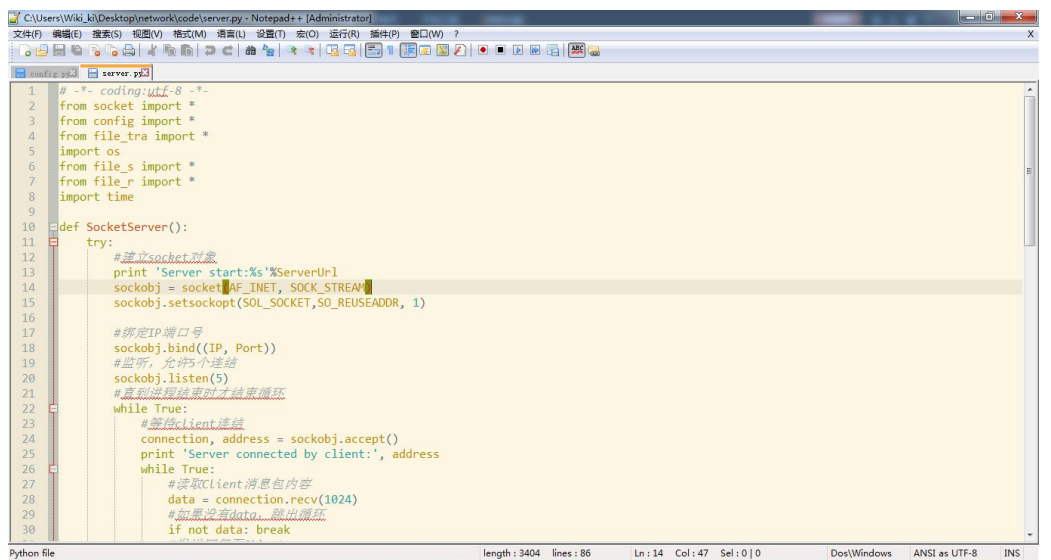
其中 config.py 为配置文件, IP, PORT 之类的, server.py 是服务器, client.py 是客户端, file_s.py 发送文件, file_r.py 接受文件, file_tra.py 遍历一遍服务器根目录。

Config.py:

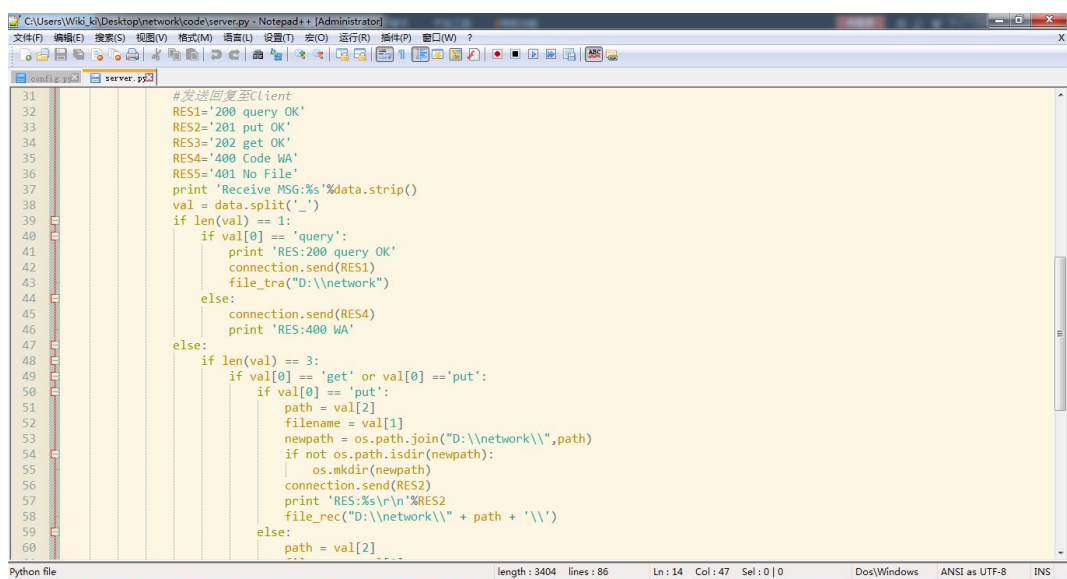


```
1 ServerUrl = "192.168.1.105:9999"
2 Colon = ServerUrl.find(':')
3 IP = ServerUrl[0:Colon]
4 Port = 9999
5 Port2 = 9998
6 Port3 = 9997
7
```

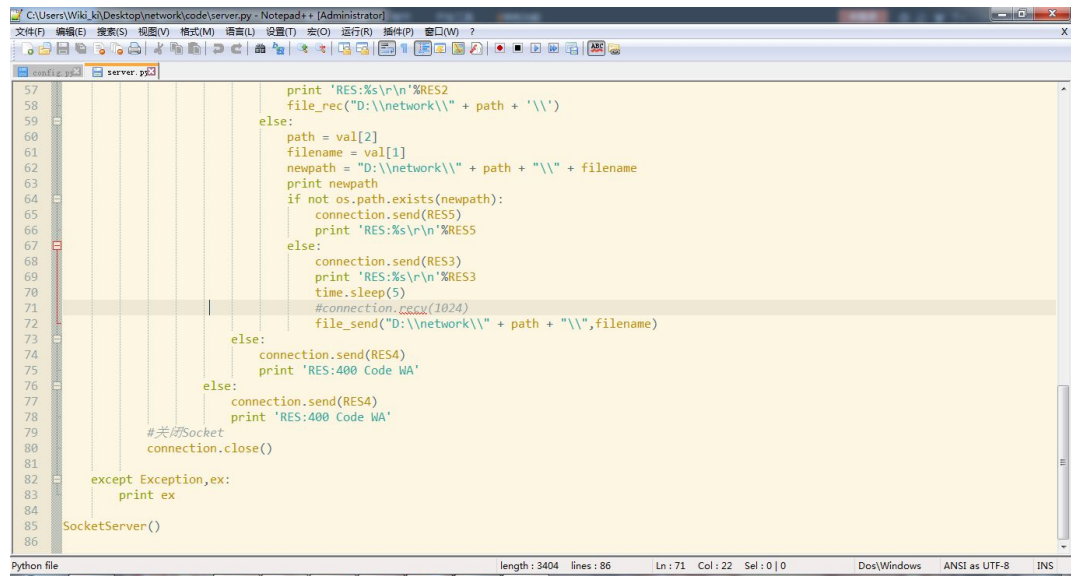
Server.py:



```
1 #-*- coding:utf-8 -*-
2 from socket import *
3 from config import *
4 from file_tra import *
5 import os
6 from file_s import *
7 from file_r import *
8 import time
9
10 def SocketServer():
11     try:
12         #建立socket对象
13         print 'Server start:%s'%ServerUrl
14         sockobj = socket(AF_INET, SOCK_STREAM)
15         sockobj.setsockopt(SOL_SOCKET, SO_REUSEADDR, 1)
16
17         #绑定IP 端口号
18         sockobj.bind((IP, Port))
19         #监听, 允许5个连接
20         sockobj.listen(5)
21         #直到进程结束才结束循环
22         while True:
23             #等待client连接
24             connection, address = sockobj.accept()
25             print 'Server connected by client:', address
26             while True:
27                 #读取Client消息包内容
28                 data = connection.recv(1024)
29                 #如果收到data...退出循环
30                 if not data: break
```



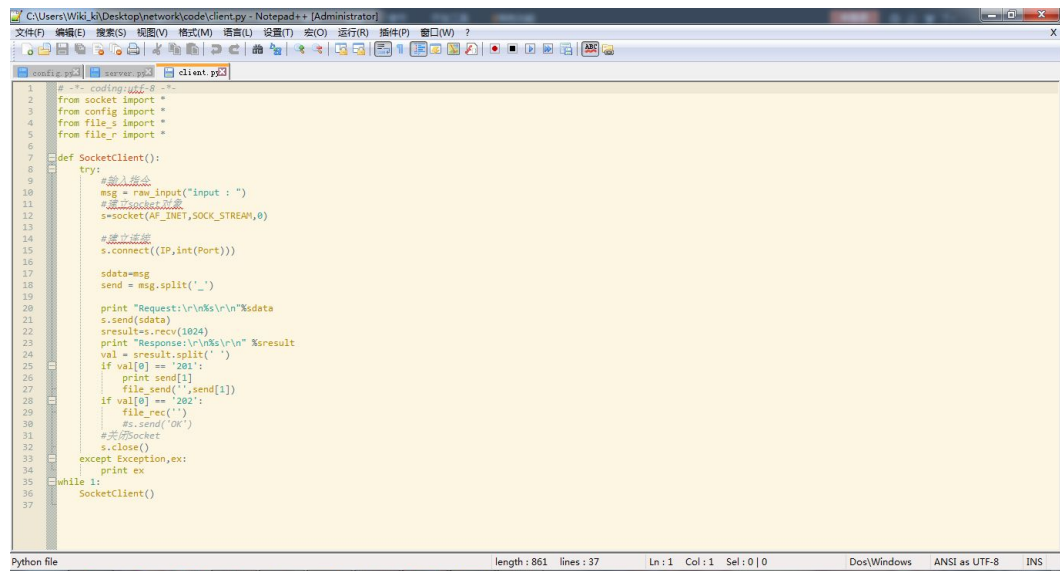
```
31 #发送回复至Client
32 RES1='200 query OK'
33 RES2='201 put OK'
34 RES3='202 get OK'
35 RES4='400 Code WA'
36 RES5='401 No File'
37 print 'Receive MSG:%s'%data.strip()
38 val = data.split('_')
39 if len(val) == 1:
40     if val[0] == 'query':
41         print 'RES:200 query OK'
42         connection.send(RES1)
43         file_tra("D:\\network")
44     else:
45         connection.send(RES4)
46         print 'RES:400 WA'
47 else:
48     if len(val) == 3:
49         if val[0] == 'get' or val[0] == 'put':
50             if val[0] == 'put':
51                 path = val[2]
52                 filename = val[1]
53                 newpath = os.path.join("D:\\network\\", path)
54                 if not os.path.isdir(newpath):
55                     os.mkdir(newpath)
56                 connection.send(RES2)
57                 print 'RES:%s\\n'%RES2
58                 file_rec("D:\\network\\" + path + '\\')
59             else:
60                 path = val[2]
```



```
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86

print 'RES:%s\r\n'%RES2
file_rec("D:\\network\\" + path + '\\')
else:
    path = val[2]
    filename = val[1]
    newpath = "D:\\network\\" + path + "\\" + filename
    print newpath
    if not os.path.exists(newpath):
        connection.send(RES5)
        print 'RES:%s\r\n'%RES5
    else:
        connection.send(RES3)
        print 'RES:%s\r\n'%RES3
        time.sleep(5)
        #connection.send(1024)
        file_send("D:\\network\\" + path + "\\," + filename)
    else:
        connection.send(RES4)
        print 'RES:400 Code WA'
    else:
        connection.send(RES4)
        print 'RES:400 Code WA'
    #关闭Socket
    connection.close()
except Exception,ex:
    print ex
SocketServer()
```

Client.py:

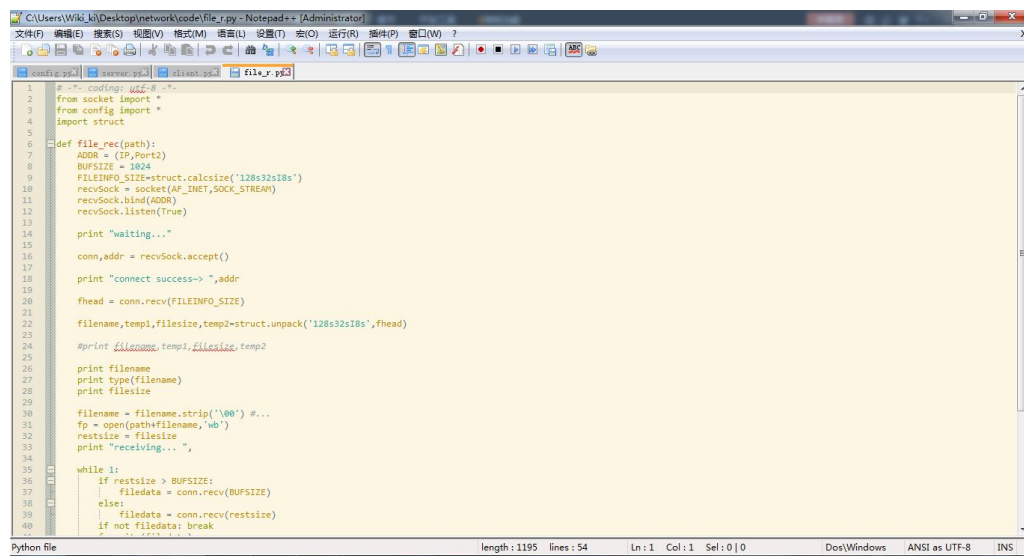


```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37

#-*- coding:utf-8 -*-
from socket import *
from config import *
import struct

def SocketClient():
    try:
        #输入IP地址
        msg = raw_input("input : ")
        #socket地址
        s=socket(AF_INET,SOCK_STREAM,0)
        #连接IP地址
        s.connect((IP,int(Port)))
        #发送数据
        sdata=msg
        send = msg.split('_')
        print "Request:\r\n%s\r\n"%sdata
        s.send(sdata)
        sresult=s.recv(1024)
        print "Response:\r\n%s\r\n"%sresult
        val = sresult.split(' ')
        if val[0] == '201':
            print send[1]
            file_send(' ',send[1])
        if val[0] == '202':
            file_rec(' ')
            #发送'OK'
            s.send('OK')
        #关闭Socket
        s.close()
    except Exception,ex:
        print ex
while 1:
    SocketClient()
```

File_s.py:



```
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40

#-*- coding:utf-8 -*-
from socket import *
from config import *
import struct

def file_rec(path):
    ADDR = (IP,Port2)
    BUFSIZE = 1024
    FILEINFO_SIZE=struct.calcsize('128s32s18s')
    recvsSock = socket(AF_INET,SOCK_STREAM)
    recvsSock.bind(ADDR)
    recvsSock.listen(True)
    print "waiting..."
    conn,addr = recvsSock.accept()
    print "connect success-> ",addr
    fhead = conn.recv(FILEINFO_SIZE)
    filename,temp1,filesize,temp2=struct.unpack('128s32s18s',fhead)
    #print filename,temp1,filesize,temp2
    print filename
    print type(filename)
    print filesize
    filename = filename.strip('\00') #...
    fp = open(path+filename,'wb')
    restsize = filesize
    print "receiving... ",
    while 1:
        if restsize > BUFSIZE:
            filedata = conn.recv(BUFSIZE)
        else:
            filedata = conn.recv(restsize)
        if not filedata: break
```

```
41         fp.write(filedata)
42         restsize = restsize-len(filedata)
43         if restsize == 0:
44             break
45
46     print "receive completed"
47
48     fp.close()
49     conn.close()
50     recvSock.close()
51     print "disconnected..."
52
53 #file_rec("")
54
```

File_r.py:

```
C:\Users\Wiki_kh\Desktop\network\code\file_s.py - Notepad++ [Administrator]
文件(F) 编辑(E) 搜索(S) 视图(V) 格式(M) 语言(L) 设置(T) 宏(O) 运行(R) 插件(P) 窗口(W) ?
1  # -*- coding: utf-8 -*-
2  from socket import *
3  from config import *
4  import os
5  import struct
6
7  def file_send(path,filename):
8      ADDR = (IP,Port2)
9      BUFSIZE = 1024
10
11      FILEINFO_SIZE=struct.calcsize('128s32sI8s')
12      sendSock = socket(AF_INET,SOCK_STREAM)
13      sendSock.connect(ADDR)
14      fhead=struct.pack('128sIII',filename,0,0,0,0,0,0,0,0,os.stat(path+filename).st_size,0,0)
15      sendSock.send(fhead)
16      fp = open(path+filename,'rb')
17      while 1:
18          filedata = fp.read(BUFSIZE)
19          if not filedata: break
20          sendSock.send(filedata)
21          print "sending...disconnecting..."
22
23      fp.close()
24      sendSock.close()
25      print "disconnected..."
26
27 #file_send('D:\network\888\','b.txt')
28
29
Python file                                     length: 720   lines: 29   Ln: 1   Col: 1   Sel: 0 | 0   Dos/Windows   ANSI as UTF-8   INS
```

File_tra.py:

```
C:\Users\Wiki_kh\Desktop\network\code\file_tra.py - Notepad++ [Administrator]
文件(F) 编辑(E) 搜索(S) 视图(V) 格式(M) 语言(L) 设置(T) 宏(O) 运行(R) 插件(P) 窗口(W) ?
1  import os
2  import os.path
3
4  def file_tra(dir):
5      rootdir = dir
6      for parent,dirnames,filenames in os.walk(rootdir):
7          for dirname in dirnames:
8              print "parent is:" + parent
9              print "dirname is:" + dirname
10         for filename in filenames:
11             print "parent is:" + parent
12             print "filename is:" + filename
13             print "the full name of the file is:" + os.path.join(parent,filename)
14
15
16
Python file                                     length: 485   lines: 16   Ln: 1   Col: 1   Sel: 0 | 0   Dos/Windows   ANSI as UTF-8   INS
```

4、测试结果：

因为有展示了，所以就不截图了，而且文件传输，截图什么的总感觉不太真实。

实验总结：

通过这次实验，加深了对协议组成内容，层次建构，通信机理，运作方式的理解，掌握了 socket 变成的基础和一些技巧。

这次实验是由朱为开和王晗谦两人小组完成，朱为开主要负责代码的书写，文档的细节校验以及演示，王晗谦则负责文档的书写以及文档格式的规范。

我（朱为开）大二的时候上过一门网络创新实验的课，当时的课程 pj 就是做的一个文件同步系统（原理也就是 socket 文件传输），当时是做了一个简单的 Android 应用，那个时候用的 socket 模板是套用的课上给的 java 模板，所以这次由我来负责代码这边，本来想直接就拿大二的东西交差的，后来觉得实在太水了点，于是就抛弃了 java，尝试用 python 来实现了。

总的来说，python 相比 C++，java 要简洁些，很多东西都有很方便的实现模板，本来想让代码整洁美观点的，所以很多函数都直接额外放到一个 .py 里去了，但后来还是莫名其妙把 SocketServer() 这个函数写的很冗长，有个小插曲，有一个地方接收文件和传递文件如果直接写不好确定哪边先运行，按理，来说应该通信发送信息来确定先后顺序，但这样比较麻烦，我就直接用了 time.sleep 函数先在一端直接运行 5s 延迟下来实现了。