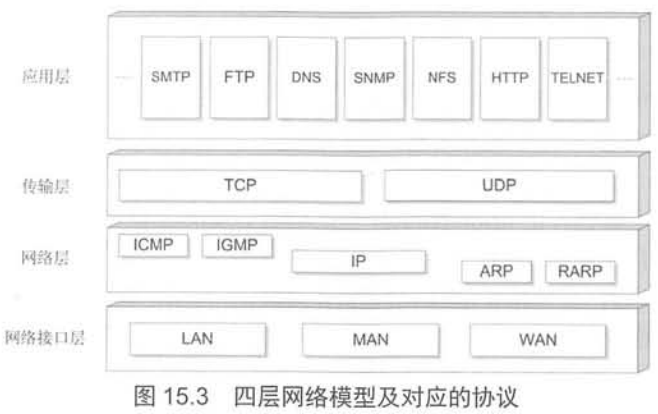
apt-get update

apt-get install apache2

systemctl status apache2

在客户端浏览器输入：http://192.168.56.121/

# python模块概述





# urllib.parse

## urlparse

from urllib.parse import \*

result = urlparse('http://www.crazyit.org:80/index.php;yeeku?name=fkit#frag')

print(result)

print(type(result))

print('scheme:', result.scheme, result[0])

print('主机和端口:', result.netloc, result[1])

print('host:', result.hostname)

print('port', result.port)

print('资源路径:', result.path, result[2])

print('param:', result.params, result[3])

print('query string:', result.query, result[4])

print('fragment:', result.fragment, result[5])

print(result.geturl())

root@2004a:/home/test# python3 test.py

ParseResult(scheme='http', netloc='www.crazyit.org:80', path='/index.php', params='yeeku', query='name=fkit', fragment='frag')

<class 'urllib.parse.ParseResult'>

scheme: http http

主机和端口: www.crazyit.org:80 www.crazyit.org:80

host: www.crazyit.org

port 80

资源路径: /index.php /index.php

param: yeeku yeeku

query string: name=fkit name=fkit

fragment: frag frag

http://www.crazyit.org:80/index.php;yeeku?name=fkit#frag

root@2004a:/home/test#

## url编码

parse\_qs

parse\_qsl

urlencode

from urllib.parse import \*

result = parse\_qs('name=zhangsan&age=123')

print(type(result))

print(result)

result = parse\_qsl('name=zhangsan&age=123')

print(type(result))

print(result)

print(urlencode(result))

root@2004a:/home/test# python3 test1.py

<class 'dict'>

{'name': ['zhangsan'], 'age': ['123']}

<class 'list'>

[('name', 'zhangsan'), ('age', '123')]

name=zhangsan&age=123

root@2004a:/home/test#

## urljoin

root@2004a:/home/test# cat test.py

from urllib.parse import \*

result = urljoin('http://www.crazyit.org/users/login.html', 'help.html')

print(result)

result = urljoin('http://www.crazyit.org/users/login.html', 'book/list.html')

print(result)

result = urljoin('http://www.crazyit.org/users/login.html', '/help.html') #虚拟机主机的根路径

print(result)

result = urljoin('http://www.crazyit.org/users/login.html', '//help.html') #绝对路径

print(result)

root@2004a:/home/test# python3 test.py

http://www.crazyit.org/users/help.html

http://www.crazyit.org/users/book/list.html

http://www.crazyit.org/help.html

http://help.html

root@2004a:/home/test#

## urllib.request

root@2004b:/home/test# cat test.py

from urllib.request import \*

result = urlopen('http://192.168.56.121/')

data = result.read(60)

print(data.decode('utf-8'))

with urlopen('http://192.168.56.121/') as f:

data = f.read(60)

print(data.decode('utf-8'))

root@2004b:/home/test#

root@2004b:/home/test# python3 test.py

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//

root@2004b:/home/test#

## POST请求

代码片段1：

import urllib.parse

params = urllib.parse.urlencode({'name':'crazy it', 'password':'123456'})

params = params.encode('utf-8')

with urlopen("http://localhost:8888/test/post.jsp", data=params) as f:

print(f.read().decode('utf-8'))

代码片段2：

from urllib.request import \*

params = 'put 请求数据'.encode('utf-8')

req = Request(url='http://localhost:8888/test/put', data=params, method='PUT')

with urlopen(req) as f:

print(f.status)

print(f.read().decode('utf-8'))

代码片段3：

from urllib.request import \*

req = Request('http://localhost:8888/test/header.jsp')

req.add\_header('Referer', 'http://www.crazyit.org/')

with urlopen(req) as f:

print(f.status)

print(f.read().decode('utf-8'))

## download

### 代码片段

from urllib.request import \*

f = urlopen("http://192.168.56.121/")

print(type(f))

print(f.headers)

print(dict(f.headers).get('Content-Length', 0))

输出：

<class 'http.client.HTTPResponse'>

Date: Thu, 24 Feb 2022 07:23:43 GMT

Server: Apache/2.4.41 (Ubuntu)

Last-Modified: Thu, 24 Feb 2022 05:54:12 GMT

ETag: "2aa6-5d8bd362c410f"

Accept-Ranges: bytes

Content-Length: 10918

Vary: Accept-Encoding

Connection: close

Content-Type: text/html

10918

### 可以正确运行的下载程序

from urllib.request import \*

import threading

class DownUtil:

def \_\_init\_\_(self, path, target\_file, thread\_num):

self.path = path

self.thread\_num = thread\_num

self.target\_file = target\_file

self.threads = []

def download(self):

req = Request(url=self.path, method='GET')

req.add\_header('Accept', '\*/\*')

req.add\_header('Charset', 'UTF-8')

req.add\_header('Connection', 'Keep-Alive')

f = urlopen(req)

self.file\_size = int(dict(f.headers).get('Content-Length', 0))

f.close()

#分配每个线程的任务

current\_part\_size = self.file\_size // self.thread\_num + 1

for i in range(self.thread\_num):

start\_pos = i\*current\_part\_size

t = open(self.target\_file, 'wb')

t.seek(start\_pos, 0)

tid = DownThread(self.path, start\_pos, current\_part\_size, t)

self.threads.append(tid)

tid.start()

def get\_complete\_rate(self):

sum\_size = 0

for i in range(self.thread\_num):

sum\_size += self.threads[i].length

return sum\_size/self.file\_size

class DownThread(threading.Thread):

def \_\_init\_\_(self, path, start\_pos, currrent\_part\_size, current\_part):

super().\_\_init\_\_()

self.path = path

self.start\_pos = start\_pos

self.current\_part\_size = currrent\_part\_size

self.current\_part = current\_part

self.length = 0

def run(self):

req = Request(url=self.path, method='GET')

req.add\_header('Accept', '\*/\*')

req.add\_header('Charset', 'UTF-8')

req.add\_header('Connection', 'Keep-Alive')

f = urlopen(req)

#跳过self.start\_pos个字节

for i in range(self.start\_pos):

f.read(1)

while self.length < self.current\_part\_size:

data = f.read(1024)

if data is None or len(data) <= 0:

break

self.current\_part.write(data)

self.current\_part.close()

f.close()

dt = DownUtil("http://192.168.56.121/", "res.txt", 2)

dt.download()

## 管理cookie

当客户端第一次向服务器发送请求时，服务器会为客户端分配session id

当客户端第二次向服务器发送请求时，如果服务器的session id还未过期，

服务器就知道客户端与前一次发送请求的客户端是同一个

MozillaCookieJar是cookiejar的子类

from urllib.request import \*

import http.cookiejar, urllib.parse

#以指定文件创建CookieJar对象，该对象可以把cookie信息保存在文件中

cookie\_jar = http.cookiejar.MozillaCookieJar('a.txt')

#创建HTTPCookieProcessor对象

cookie\_processor = HTTPCookieProcessor(cookie\_jar)

#创建OpenerDirector对象

opener = build\_opener(cookie\_processor)

user\_agent = r'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36' \

r' (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36'

headers = {'User-Agent':user\_agent, 'Connection':'keep-alive'}

params = {'name':'test', 'pass':'123'}

postdata = urllib.parse.urlencode(params).encode()

request = Request('http://localhost:8888/test/login.jsp',

data = postdata, headers=headers)

response = opener.open(request)

print(response.read().decode('utf-8'))

#将cookie信息写入文件中

#cookie\_jar.save(ignore\_discard=True, ignore\_expire=True) … (1)

print('------------------------------')

#创建向被保护页面发送GET请求的Request

request = Request('http://localhost:8888/test/secret.jsp', headers=headers)

response = opener.open(request)

print(response.read().decode())

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

将上面程序(1)语句取消屏蔽，程序就会把cookie信息写入a.txt文件中。

这意味着该程序会将response的session id等cookie等持久化保存在a.txt中

下面的程序可以读取该cookie文件信息，这样程序就可以模拟前面登录过的客户端，从而直接访问被保护的页面了

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

from urllib.request import \*

import http.cookiejar, urllib.parse

#以指定文件创建CookieJar对象，该对象可以把cookie信息保存在文件中

cookie\_jar = http.cookiejar.MozillaCookieJar('a.txt')

cookie\_jar.load(‘a.txt’, ignore\_discard=True, ignore\_expire=True)

for item in cookie\_jra: #遍历a.txt中保存的信息

print('Name=' + item.name)

print('Value=' + item.value)

#创建HTTPCookieProcessor对象

cookie\_processor = HTTPCookieProcessor(cookie\_jar)

#创建OpenerDirector对象

opener = build\_opener(cookie\_processor)

user\_agent = r'Mozilla/5.0 (Windows NT 6.1; WOW64) AppleWebKit/537.36' \

r' (KHTML, like Gecko) Chrome/56.0.2924.87 Safari/537.36'

headers = {'User-Agent':user\_agent, 'Connection':'keep-alive'}

print('------------------------------')

#创建向被保护页面发送GET请求的Request

request = Request('http://localhost:8888/test/secret.jsp', headers=headers)

response = opener.open(request)

print(response.read().decode())

# socket

默认使用tcp通信

|  |  |
| --- | --- |
| server | client |
| import socket  s = socket.socket()  s.bind(('192.168.56.121', 30000))  s.listen()  while True:  c,addr = s.accept()  print(c)  print('recv connection from ', addr)  c.send('hello, server send greet to you!'.encode('utf-8'))  c.close() | import socket  s= socket.socket()  s.connect(('192.168.56.121', 30000))  print('%s' % s.recv(1600).decode('utf-8'))  s.close() |
| <socket.socket fd=4, family=AddressFamily.AF\_INET, type=SocketKind.SOCK\_STREAM, proto=0, laddr=('192.168.56.121', 30000), raddr=('192.168.56.123', 46922)>  recv connection from ('192.168.56.123', 46922) | hello, server send greet to you! |

多线程socket

|  |  |
| --- | --- |
| server | client |
| import socket  import threading  li=[]  s = socket.socket()  s.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)  s.bind(('192.168.56.121', 30001))  s.listen()  def read\_from\_client(c):  try:  return c.recv(2048).decode('utf-8')  except:  li.remove(c)  def client\_loop(c):  try:  while True:  content = read\_from\_client(c)  print(content)  if content is None:  break  for c in li:  c.send(content.encode('utf-8'))  except e:  print(e.strerror)  while True:  c,addr = s.accept()  li.append(c)  threading.Thread(target=client\_loop, args=(c,)).start() | import socket  import threading  s= socket.socket()  s.connect(('192.168.56.121', 30001))  def read\_from\_server(s):  while True:  print(s.recv(2048).decode('utf-8'))  threading.Thread(target=read\_from\_server, args=(s,)).start()  while True:  line = input('')  if line is None or line == 'quit':  break  s.send(line.encode('utf-8'))  print('QUIT NOW')  s.close() |
| client ctrl+c关闭后，server死循环 | client程序无法退出 |

# 半关闭的socket

如果要表示输出数据已经结束，则可以通过关闭socket来实现。但如果彻底关闭了socket，则会导致程序无法再从该socket中读取数据。

在这种情况下，socket提供了一个shutdown(how)关闭方法。该方法可以只关闭socket的输入或输出部分。

SHUT\_RD：关闭输入

SHUT\_WR：关闭输出

SHUT\_RDWR：全关闭

import socket

s = socket.socket()

s.bind(('127.0.0.1', 3003))

s.listen()

sock,addr = s.accept()

sock.send("server's fist line data".encode('utf-8'))

sock.send("server's second line data".encode('utf-8'))

sock.shutdown(socket.SHUT\_WR)

while True:

line = sock.recv(2048).decode('utf-8')

if line is None or line == '':

break

print(line)

sock.close()

# selectors

|  |  |
| --- | --- |
| server | client |
| import selectors,socket  sock\_list=[]  sel = selectors.DefaultSelector()  def read(sock, mask):  try:  data = sock.recv(1024)  if data:  for s in sock\_list:  s.send(data)  else:  print('关闭', sock)  sel.unregister(sock)  sock.close()  sock\_list.remove(sock)  except:  print('关闭', sock)  sel.unregister(sock)  sock.close()  sock\_list.remove(sock)  def accept(sock, mask):  conn,addr = sock.accept()  sock\_list.append(conn)  conn.setblocking(False)  sel.register(conn, selectors.EVENT\_READ, read)    sock = socket.socket()  sock.bind(('127.0.0.1', 3001))  sock.listen()  sock.setblocking(False)  sel.register(sock, selectors.EVENT\_READ, accept)  while True:  events = sel.select()  for key,mask in events:  callback = key.data  callback(key.fileobj, mask) | import selectors, socket, threading  sel = selectors.DefaultSelector()  def read(conn, mask):  data = conn.recv(1024)  if data:  print(data.decode('utf-8'))  else:  print('closing', conn)  sel.unregister(conn)  conn.close()  s = socket.socket()  s.connect(('127.0.0.1', 3001))  s.setblocking(False)  sel.register(s, selectors.EVENT\_READ, read)  def keyboard\_input(s):  while True:  line = input('')  if line is None or line == 'exit':  break  s.send(line.encode('utf-8'))  threading.Thread(target=keyboard\_input, args=(s,)).start()  while True:  events = sel.select()  for key,mask in events:  callback = key.data  callback(key.fileobj, mask) |

# udp编程

## 数据报

|  |  |
| --- | --- |
| server | client |
| import socket  port=7777  data\_len = 4096  books = ("crazy python", "crazy swift", "crazy android", "crazy ios")  s = socket.socket(type=socket.SOCK\_DGRAM)  s.bind(('127.0.0.1', port))  for i in range(5):  data,addr = s.recvfrom(data\_len)  print(data.decode('utf-8'))  send\_data = books[i%4].encode('utf-8')  s.sendto(send\_data, addr)  s.close() | import socket  port=7777  data\_len=4096  server\_ip="127.0.0.1"  s=socket.socket(type=socket.SOCK\_DGRAM)  while True:  line=input('')  if line is None or line == 'exit':  break  data = line.encode('utf-8')  print("sending: ", data)  s.sendto(data, (server\_ip, port))  data = s.recv(data\_len)  print(data.decode('utf-8'))  s.close() |
|  | 工作正常 |

## 组播

import time,socket, threading, os

sendip='192.168.56.121'

sendport=7777

mygroup="230.0.0.1"

s = socket.socket(type=socket.SOCK\_DGRAM)

s.bind(('0.0.0.0', sendport))

s.setsockopt(socket.IPPROTO\_IP, socket.IP\_MULTICAST\_TTL, 64)

s.setsockopt(socket.SOL\_SOCKET, socket.SO\_REUSEADDR, 1)

print(socket.inet\_aton(mygroup) + socket.inet\_aton(sendip))

ret = s.setsockopt(socket.IPPROTO\_IP, socket.IP\_ADD\_MEMBERSHIP, socket.inet\_aton(mygroup)+socket.inet\_aton(sendip))

def read\_from\_peer(sock):

while True:

data = sock.recv(2048)

print('info: ', data.decode('utf-8'))

threading.Thread(target=read\_from\_peer, args=(s,)).start()

while True:

line =input('')

if line is None or line == 'exit':

break

os.exit(0)

s.sendto(line.encode('utf-8'), (mygroup, sendport))

root@2004b:/home/test/111# python3 test.py

Traceback (most recent call last):

File "test.py", line 11, in <module>

ret = s.setsockopt(socket.IPPROTO\_IP, socket.IP\_ADD\_MEMBERSHIP, socket.inet\_aton(mygroup)+socket.inet\_aton(sendip))

OSError: [Errno 19] No such device

出错的原因在于：发送ip地址:192.168.56.121并非本地地址。

# 电子邮件

## smtplib

### #发送plain邮件

import smtplib

from email.message import EmailMessage

smtp\_server="smtp.qq.com"

from\_addr = "297234085@qq.com"

password='123'

to\_addr = 'flychengwen@126.com'

#conn = smtplib.SMTP(smtp\_server, 25)

conn = smtplib.SMTP\_SSL(smtp\_server, 465)

conn.set\_debuglevel(1)

conn.login(from\_addr, password)

msg = EmailMessage()

msg.set\_content('hello, this is email from python3', 'plain', 'utf-8')

conn.sendmail(from\_addr, [to\_addr], msg.as\_string())

conn.quit()

### #发送html邮件

import smtplib

from email.message import EmailMessage

smtp\_server="smtp.qq.com"

from\_addr = "297234085@qq.com"

password='123'

to\_addr = 'flychengwen@126.com'

#conn = smtplib.SMTP(smtp\_server, 25)

conn = smtplib.SMTP\_SSL(smtp\_server, 465)

conn.set\_debuglevel(1)

conn.login(from\_addr, password)

msg = EmailMessage()

#msg.set\_content('hello, this is email from python3', 'plain', 'utf-8')

msg.set\_content('<h2>邮件内容</h2>' + '<p>您好，这是一封来自python的邮件<p>' +

'来自<a href="http://www.crazyit.org">疯狂联盟</a>', 'html', 'utf-8')

msg['subject'] = '一封html邮件'

msg['from'] = '李刚 <%s>' % from\_addr

msg['to'] = '新用户 <%s>' % to\_addr

conn.sendmail(from\_addr, [to\_addr], msg.as\_string())

conn.quit()

### #附件

maintype #大类型，如image，application

subtype #子类型，如jpeg，gif

filename #在邮件中显示的名称，可任取

cid=img #email.utils生成的随机数，用于在邮件正文中引用附件，可选

import smtplib, email.utils

from email.message import EmailMessage

smtp\_server="smtp.qq.com"

from\_addr = "297234085@qq.com"

password='123'

to\_addr = 'flychengwen@126.com'

first\_id,second\_id = email.utils.make\_msgid(),email.utils.make\_msgid()

print(type(first\_id))

print(first\_id)

print(first\_id[1:-1])

#import sys

#sys.exit()

#输出结果为：

#<class 'str'>

#<164569518521.3068.4416140751380693855@2004b>

#164569518521.3068.4416140751380693855@2004b

#conn = smtplib.SMTP(smtp\_server, 25)

conn = smtplib.SMTP\_SSL(smtp\_server, 465)

conn.set\_debuglevel(1)

conn.login(from\_addr, password)

msg = EmailMessage()

#msg.set\_content('hello, this is email from python3', 'plain', 'utf-8')

msg.set\_content('<h2>邮件内容</h2>' + '<p>您好，这是一封来自python的邮件<p>' +

'来自<a href="http://www.crazyit.org">疯狂联盟</a>' +

'<img src="cid:' + second\_id[1:-1] + '"><p>' +

'<img src="cid:' + first\_id[1:-1] + '">', 'html', 'utf-8')

msg['subject'] = '一封html邮件'

msg['from'] = '李刚 <%s>' % from\_addr

msg['to'] = '新用户 <%s>' % to\_addr

with open('/home/test/logo.jpg', 'rb') as f:

msg.add\_attachment(f.read(), maintype='image', subtype='jpeg', filename='test.png', cid=first\_id)

with open('/home/test/fklogo.gif', 'rb') as f:

msg.add\_attachment(f.read(), maintype='image', subtype='gif', filename='test.gif', cid=second\_id)

with open('/home/test/fkit.pdf', 'rb') as f:

msg.add\_attachment(f.read(), maintype='application', subtype='pdf', filename='test.pdf',)

conn.sendmail(from\_addr, [to\_addr], msg.as\_string())

conn.quit()

## poplib

流程如下：

获取邮件总数和总大小

通过list获取邮件列表，显示每条邮件的编号

获取最大邮件编号，即最后一封邮件

通过retr命令获取该编号之邮件

遍历邮件，

判断邮件是否容器，若是则跳过

判断邮件是否正文，若是打印之

判断邮件是否附件，若是附件：

附件名是否存在，若附件名不存在，则猜测出附件类型，自动生成附件

若猜测附件类型失败，则默认bin

写入附件到本地

import poplib,os.path,mimetypes

from email.parser import BytesParser, Parser

from email.policy import default

username='flychengwen@126.com'

password='123'

pop3\_server='pop.qq.com'

conn = poplib.POP3(pop3\_server, 110)

conn = poplib.POP3\_SSL(pop3\_server, 995)

conn.set\_debuglevel(1)

print(conn.getwelcome().decode('utf-8'))

conn.user(username)

conn.pass(password)

message\_num, total\_size = conn.stat()

print("message\_num=%s total\_size=%s" % (message\_num, total\_size))

#send 'list' command

#resp是响应码，mails是编号，octets是大小

resp,mails,octets = conn.list()

print(resp, mails)

#send 'retr' command

resp,data,octets = conn.retr(len(mails)) #mails是邮件编号，len(mails)则是最大邮件编号，即最后一封邮件

msg\_data = b'\r\n'.join(data) #data是一个列表，将列表的元素拼起来

msg = ByteParser(policy=default).parsebytes(msg\_data)

print(type(msg))

print('sender: ' +msg['from'])

print('receiver: ' +msg['to'])

print('subject: ' +msg['subject'])

print('1st receiver: ' +msg['to'].addresses[0].username)

print('1st sender: ' +msg['from'].addresses[0].username)

for part in msg.walk():

counter=1

#说明是容器，用于包含正文，附件等

if part.get\_content\_maintype() == 'multipart':

continue

#说明是正文

elif part.get\_content\_maintype() == 'text':

print(part.get\_content())

#处理附件

else:

#先获取附件名，若文件名不存在，则猜测文件类型（默认为bin），自动生成一个文件名

filename = part.get\_filename()

if not filename:

ext = mimetypes.guest\_extension(part.get\_content\_types())

if not ext:

ext = '.bin'

filename = 'part-%03d%s' % (counter, ext)

counter += 1

with open(os.path.join('.', filename), 'wb') as fp:

fp.write(part.get\_payload(decode=True))

conn.quit()