第十七章 网络客户端编程

16章介绍的使用套接字的低级别的网络通讯协议是当今互联网中大部分客户端/服务器协议的核心。这些网络协议包括FTP, SCP, 阅读Usenet 新闻组(NNTP)，SMTP, POP3/IMAP，等等，这些协议的工作方式与之前的套接字中介绍的客户端/服务器的例子很像。

唯一的不同在于，我们已经使用过 TCP/IP 等低级别的协议，并基于此创建了新的，更具体的协议来实现我们刚刚描述的服务。

> 关于Socket编程中，Socket实现了对底层TCP/IP，UDP/IP的封装。

因特网客户端？

把因特网简化成一个数据交换中心，数据交换的参与者是一个服务提供者和一个服务的使用者或者生产者–消费者。

主要了解下三个因特网协议–FTP,NNTP,POP3，通过这3个程序你将会发现这些协议的 API 是多么的相似。同时还能学会如何写出这些协议与其它协
议实用的客户端程序来。

文件传输协议FTP

因特网中最流行的事情就是文件的交换。
有很多协议可以供因特网上传输文件使用。如文件传输协议FTP，Unix-to-Unix赋值协议(UUCP)，以及超文本传输协议HTTP。
还有(Unix 下的)远程文件复制指令 rcp (以及更安全， 更灵活的 scp和 rsync)。

HTTP 主要用于网页文件的下载和访问 Web服务上。它一般不要求用户输入登录的用户名密码就可以访问服务器上的文件和服务。
HTTP 文件传输请求主要是用于获取网页(文件下载)。

17.2.2 文件传输协议FTP

文件传输协议记录在RFC959中，主要用于匿名下载公共文件，也可以用于在两台电脑之间传输文件，尤其是在使用Unix系统作为文件存储系统。

FTP 要求输入用户名和密码才能访问远程的 FTP 服务器， 但它也允许没有帐号的用户以匿名用户
登录。不过，管理员要先设置 FTP 服务器允许匿名用户登录。这时，匿名用户的用户名是“anonymous”，密码一般是用户的 e-mail 地址。

> FTP协议本质上还是需要用户名和密码进行访问的，只不过在FTP服务器端设置了一个特殊的用户名”anonymous”和密码用户e-mail地址来进行访问的。

FTP的工作流程

1. 客户端连接远程的 FTP 服务器
2. 客户端输入用户名和密码（或“anonymous”和 e-mail 地址）
3. 客户端做各种文件传输和信息查询操作
4. 客户端登出远程 FTP 服务器，结束通讯

> 一般，在客户端超过 15 分钟（ 900 秒）不活动之后，连接就会被关闭。

在底层上只是有TCP。FTP 是客户端 / 服务器编程中很“与众不同”的例子。客户端和服务器都使用两个套接字来通讯：一个是控制和命令端口(21 号端口)，另一个是数据端口(有时是 20 号端口)。

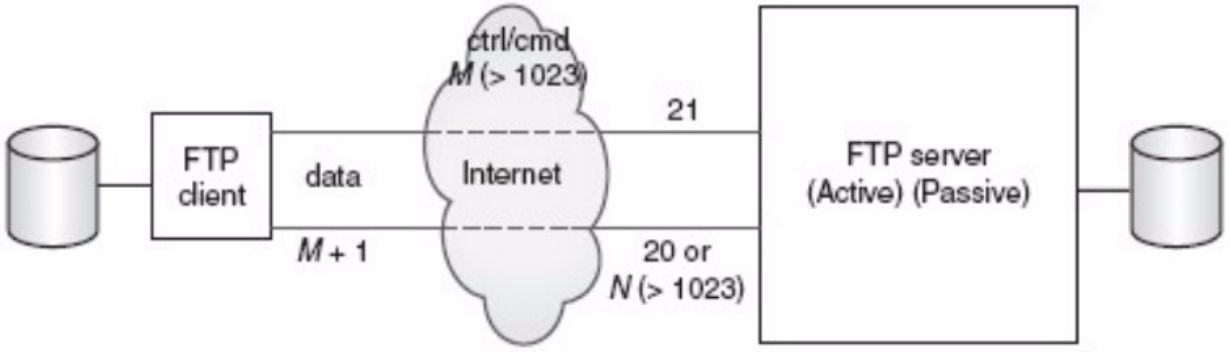

图 17-1 因特网上的 FTP 客户端和服务器。客户端和服务器使用指令和控制端口发送 FTP 协议，而数据通过数据端口传输。

FTP的两种模式:主动和被动

“有时”是因为 FTP 有两种模式：主动和被动。只有在主动模式服务器才使用数据端口。
在服务器把 20 号端口设置为数据端口后，它“主动”连接客户端的数据端口。
而被动模式中，服务器只是告诉客户端它的随机端口的号码，客户端必须主动建立数据连接。在这种模式下，你会看到，FTP 服务器在建立数据连接时是“被动”的。

17.2.3 Python 和 FTP

在Python中，模块ftplib用于支持FTP操作,FTP的所有操作都在FTP类对象中完成。

> 关于回调函数:是进程/线程在完成某个时间后调用的函数。比如要登录某一系统，用户名和密码要去查询数据库，由于查询数据库的操作非常耗时，而当前线程不可能一直等待数据库查询操作完成后在往下执行，因为那样太浪费进程/线程了，这时当数据库查询用户名和密码的操作完成时当前进程/线程回过来再调用的函数，这个函数叫做回调函数。

Python 写 FTP 客户端程序的步骤:

1. 连接到服务器
2. 登录
3. 发出服务请求(有可能有返回信息)
4. 退出

17.2.4 ftplib.FTP类方法

FTP对象的方法

方法	描述
`login(user='anonymous',passwd='', acct='')`	登录FTP服务器
`pwd()`	得到当前工作目录
`cwd(path)`	把当前工作目录设置为path
`dir([path[,...[,cb]]])`	显示path目录里的内容，可选的参数cb是一个回调函数，他会被传给`retrlines()`方法
`nlst([path[,...])`	与 dir()类似，但返回一个文件名的列表，而不是显示这些文件名
`retelines(cmd[,cb])`	给定 FTP 命令(如“RETR filename”)，用于下载文本文件。可选的回调函数cb用于处

	理文件的每一行
`retrbinary(cmd, cb[,bs=8192[,ra]])`	与 `retrlines()` 类似，只是这个指令处理二进制文件。回调函数cb用于处理每一块8k下载的数据
`storlines(cmd,f)`	给定 FTP 命令(如“STOR filename”)，以上传文本文件。
`storbinary(cmd,f[,bs=8192])`	与 `storlines()` 方法类似，只是这个命令用于处理二进制文件，上传块大小默认为8k
`rename(old,new)`	把远程文件old改名为new
`delete(path)`	删除位于path的远程文件
`mkd(directory)`	创建远程目录
`rmd(directory)`	删除远程目录
`quit()`	关闭日结并退出

17.2.6 客户端FTP例子

```
import ftplib
import os
import socket

# 17.2.6 FTP下载示例
HOST = 'ftp.debian.org'
DIR = 'debian'
FILE = 'README' # 下载的文件名

def ftpdownload():
    try:
        f = ftplib.FTP(HOST) # 创建FTP对象，连接FTP服务器
    except (socket.errno, socket.gaierror), e:
        print 'EERROR: cannot reach "%s"'  % HOST
        return
    print '*** Connected to host "%s"' % HOST

    try:
        f.login() # 以anonymous登录
    except ftplib.error_perm:
        print 'ERROR: cannot login anonymously'
        f.quit()
        return

    print '*** Logged in as "anonymous"'

    try:
        f.cwd(DIR) # 转到发布目录 debian
    except ftplib.error_perm:
        print 'ERROR: cannot CD to "%s"' % DIR
        f.quit()
```

```
            return

        print '*** Changed to "%s" folder' % DIR

        # 下载文件
        try:
            '''
            虽然这样方便，但最好还是不要这样做，做为一个程序员，
            要尽量做到在资源不再被使用的时候就直接释放， 而不是依赖其它代码来做释放操作。

            应该把文件对象保存到一个变量 中 ，
            如变量loc,然 后 把 loc.write传给ftp.retrbinary()方法
            '''
            f.retrbinary('RETR %s' % FILE, open(FILE,'wb').write) #回调函数为open().write()它在每接收到一块二进制数据的时候都会被调用
        except ftplib.error_perm:
            print 'ERROR:cannot read file "%s"' % FILE
            os.unlink(FILE) # 把存在的空文件给删掉
        else:
            print '*** Downloaded "%s" to CWD' % FILE
            f.quit()
            return
```

12.2.7 FTP的其他方面

一下经典的FTP客户端类型:

- 命令行客户端程序:shell,bash等
- GUI客户端程序
- 网页浏览器
- 定制程序–自己写的

这四种客户端类型都可以用 Python 来写。

怎样制作一个命令行客户端？

在命令行的基础上，你可以使用一些界面工具包，如 Tk，wxWidgets，GTK+，Qt，MFC，甚至 Swing（要导入相应的 Python[或 Jython]的接口模块）来创建一个完整的 GUI 程序。

可以使用 Python 的 urllib 模块来解析 FTP 的 URL 并进行 FTP 传输。

FTP 不仅可以用在下载应用程序上，还可以用在系统之间文件的转移上。

`scp`和`rsync`命令

如果你想要做的只是写一个 FTP 程序来帮助你在下班后自动移动文件，那用 Python是一个非常好的主意。

FTP协议的定义/规范在**RFC959**中
ftp://ftp.isi.edu/in-notes/rfc959.txt和

http://www.networksorcery.com/enp/protocol/ftp.htm
其他的RFC文档有2228,2389,2428,2577,2640,4217。

Python对FTP的支持
https://docs.python.org/2/library/ftplib.html

17.3 网络新闻

Usenet 新闻系统是一个全球存档的 “电子公告板”。

Usenet中包含了各种各样的新闻组，其他的主机可以订阅自己感兴趣的新闻组，这样就可以只收取感兴趣的新闻了。

每个系统都有一个它已经“订阅” 的新闻组的列表，它只接收它感兴趣的新闻组里的帖——而不是服务器上所有新闻组的帖子。

17.3.2 网络新闻传输协议NNTP

供用户在新闻组中下载或发表帖子的方法叫网络新闻传输协议（ NNTP ）。
和FTP的客户端/服务端的操作方式类似，而且NNTP还简单得多。
FTP需要使用20端口登录，21端口进行数据传输和控制，而NNTP只是用一个标准端口119做通讯。
NNTP通讯的示例图

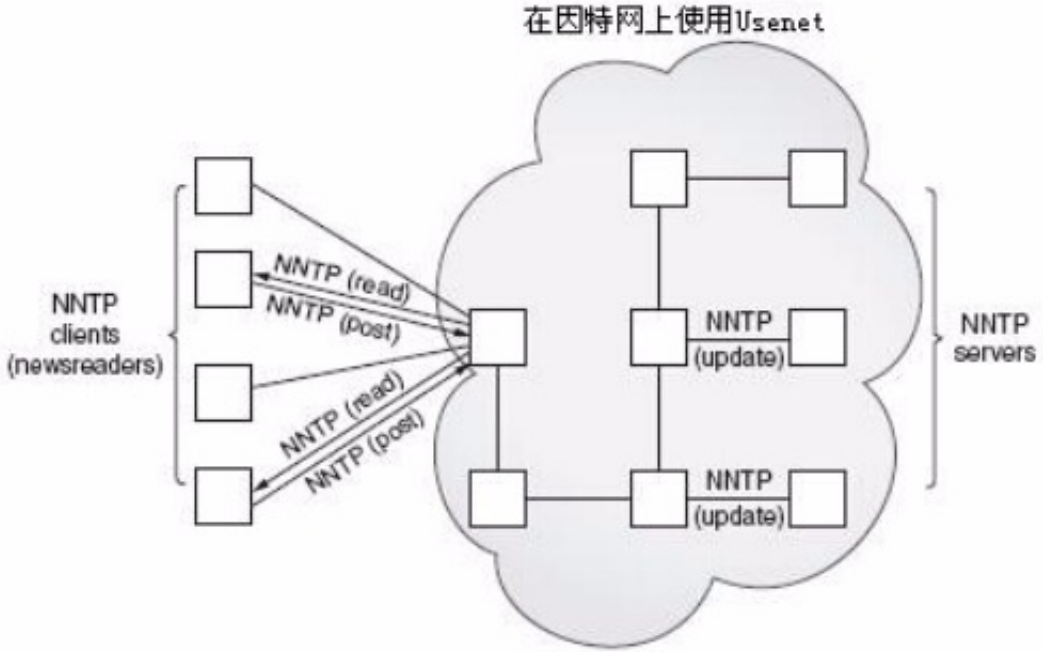

图 17-2 因特网上的 NNTP 客户端和服务器。客户端主要阅读新闻，有时也发帖子。文章会在服务器之间做同步。

NNTP的流程(与FTP类似):

1. 连接到服务器
2. 登录(如果需要的话)
3. 发送亲求
4. 退出

17.3.4 nntplib.NNTP类方法

创建nntplib.NNTP客户端程序时可能用到的方法

方法	描述
`group(name)`	name为组名,服务器返回一个元组(rsp,ct,fst,lst,group)的信息,分别为文章数量，第一个和最后一个文章的号码以及组名
`stat(id)`	返回一个元组(response,number,message_id)
`next()`	发送NEXT命令,返回一个与`stat`一样的元组
`last()`	发送LAST命令,回一个与`stat`一样的元组
`head(id)`	返回元组(response, number, id, list),list为header信息
`body(id)`	与`head()`一样，返回body信息
`article(id)`	和`head()`一样,返回header和body信息

17.3.5 交互式NNTP例子

```
from nntplib import NNTP

def nntp():
    s = NNTP('news.gmane.org')
    resp,count,first,last,name = s.group('gmane.comp.python.django.announce')
    print 'Group',name,'has',count,'articles,range',first,'to',last
    # print 'Response',resp
    # resp, overviews = s.xover()
    # art_num, over = overviews[0]
    # print art_num
    #print s.getwelcome()

    # resp, number,messageid = s.stat(first) #返回一个文章号码和消息号码，不包括文章的数据
    # print resp,number,messageid
    # resp,number,messageid,data = s.body(first) # 返回文章所有行的列表
    # print number,messageid,data
    resp,number,messageid,header = s.head(first) # 返回文章标题的列表
    for ls in header:
        print ls
    #print header['Subject'] #在3.5中有decode_header()方法可以返回一个具体的头部信息

    resp,number,messageid,article = s.article(first) # 与head()类似,arcitle中包含了文章的标题和内容的列表
    print article
```

17.3.7 NNTP的其他方面

NNTP的定义–RFC977
http://www.networksorcery.com/enp/protocol/nntp.htm。
其它相关的 RFC 有 1036，2980。
Python NNTP
https://docs.python.org/2.7/library/nntplib.html

17.4 电子邮件

e-mail的确切定义RFC2822
消息由头域（合起来叫消息头）以及后面可选的消息体组成.

E-mail系统组件和协议

简单点来说，有一台发送电脑（发件人的消息从这里发送出去），和一台目的电脑（收件人的信件服务器）。

最好的解决方案是发送电脑知道如何连接到接收电脑，这样一来，它就可以直接把消息发送过去。

发送电脑要查询到某一台中间主机，这台中间主机能到达最后的收件主机。然后这台中间主机要找一台离目的主机更近一些的主机。所以，在发送主机和目的主机之间，可能会有多台叫做“跳板”的主机。

如果你仔细看看你收到的 e-mail 的邮件头， 你会看到一个“passport”标记，其中记录了邮件寄给你这一路上都到过了哪些地方。

e-mail系统的各个组件:

- 消息传输代理MTA
    - 这是一个在邮件交换主机上运行的一个服务器程序它负责邮件的路由，队列和发送工作。它们就是邮件从源主机到目的主机所要经过的跳板。所以也被称为是“信息传输”的“代理”。

> 要让所有这些工作起来，MTA 要知道两件事情:
>
> 1. **如何找到消息应该去的下一台MTA**
>
> 第一件事由域名服务（DNS）来查找目的域名的 MX(邮件交换 Mail eXchange)来完成。
> 2. 如何与下一台MTA通讯

17.4.2 发送E-mail

要能发送 e-mail，你的邮件客户端一定要连接到一个 MTA，它们靠某种协议进行通讯。 MTA 之间通讯所使用的协议叫**消息传输系统(MTS)**。只有两个 MTA 都使用这个协议时，才能进行通讯。 **STMP** RFC821 1982,RFC2821 2001 一些已经实现的MTA包括

- Sendmail
- Postfix
- Exim
- qmail
  商业的MTA
- Microsoft Exchang
- Lotus Notes Domino Mail Server

SMTP 是在因特网上MTA之间用于消息交换的最常用的 MTS。
它被 MTA 用来把 e-mail 从一台主机传送到另一台主机。

在你发 e-mail 的时候，你必须要连接到一个外部的SMTP服务器，这时，你的邮件程序是一个SMTP客户端。
你的SMTP服务器也因此成为了你的消息的第一个跳板。

17.4.3 Python和SMTP

smtplib.SMTP负责邮件的信息交换

1. 连接到服务器
2. 登录(如果需要的话)
3. 发出服务请求
4. 退出

像NNTP一样，登录是可选的，只有在服务器打开了SMTP认证(SMTP-AUTH)时才要登录。
SMTP通讯只有一个端口25.

17.4.4 smtplib.SMTP 类方法

`login(user,pwd)`–登录
`sendmail()`–发送邮件

17.4.5 smtplib.SMTP 示例

```python
import smtplib
from email.mime.text import MIMEText

def sendmail2():
    host = 'smtp.163.com'  # 设置发件服务器地址
    port = 465  # 设置发件服务器端口号。注意，这里有SSL和非SSL两种形式

    # 用qq邮箱发邮件时,在登录时会产生授权码的，这是要把授权码填进去才能登录到qq邮箱
    sender = 'xxx@163.com'  # 设置发件邮箱，一定要自己注册的邮箱
    pwd = 'xxxxxx'  # 设置发件邮箱的密码，等会登陆会用到

    receiver = 'xxx@163.com' # 设置邮件接收人
    body = '<h1>Hi</h1><h3>Test邮件-10个</h3>' # 设置邮件正文，这里是支持HTML的

    msg = MIMEText(body, 'html') # 设置正文为符合邮件格式的HTML内容
    msg.set_charset("utf-8")

    msg['subject'] = 'Hello world' # 设置邮件标题
    msg['from'] = sender  # 设置发送人
    msg['to'] = receiver  # 设置接收人

    s = smtplib.SMTP_SSL(host, port)  # 注意！如果是使用SSL端口，这里就要改为SMTP_SSL
    s.login(sender, pwd)  # 登陆邮箱

    for index in range(1,11):
        s.sendmail(sender, receiver, msg.as_string())  # 发送邮件

    print 'over'  # 发送成功就会提示
```

STMP的其他方面

STMP协议的定义/规范在RFC2821中

e-mail消息详细记录在RFC 2822中
Python中队SMTP的支持
https://docs.python.org/2/library/smtplib.html

17.4.7 接收E-mail

对于家族用户来说，在家里放一个SMTP服务器来运行 SMTP 是不现实的。必须要设计一种新的系统，能够周期性地把信件下载到本地计算机，以供离线时使用。

在家用电脑中运行的应用程序叫邮件用户代理(MUA)。MUA从服务器上下载邮件。在发送邮件的时候,MTA能直接与SMTP服务器用 SMTP 进行通讯。

17.4.8 POP和IMAP

用于下载邮件的第一个协议叫邮局协议-POP,在 RFC 918中.

邮局协议(POP)的目的是让用户的工作站可以访问邮箱服务器里的邮件。邮件要能从工作站通过简单邮件传输协议(SMTP)发送到邮件服务器。
POP协议的最新版本是第3版,也叫POP3-RFC1939.

另外一个中协议是交互式邮件访问协议IMAP，RFC 1064 1998年。
现在使用的IMAP版本为IMAP4revl,RFC 3501 2003年。
Microsoft Exchange就使用IMAP作为其下载机制的。

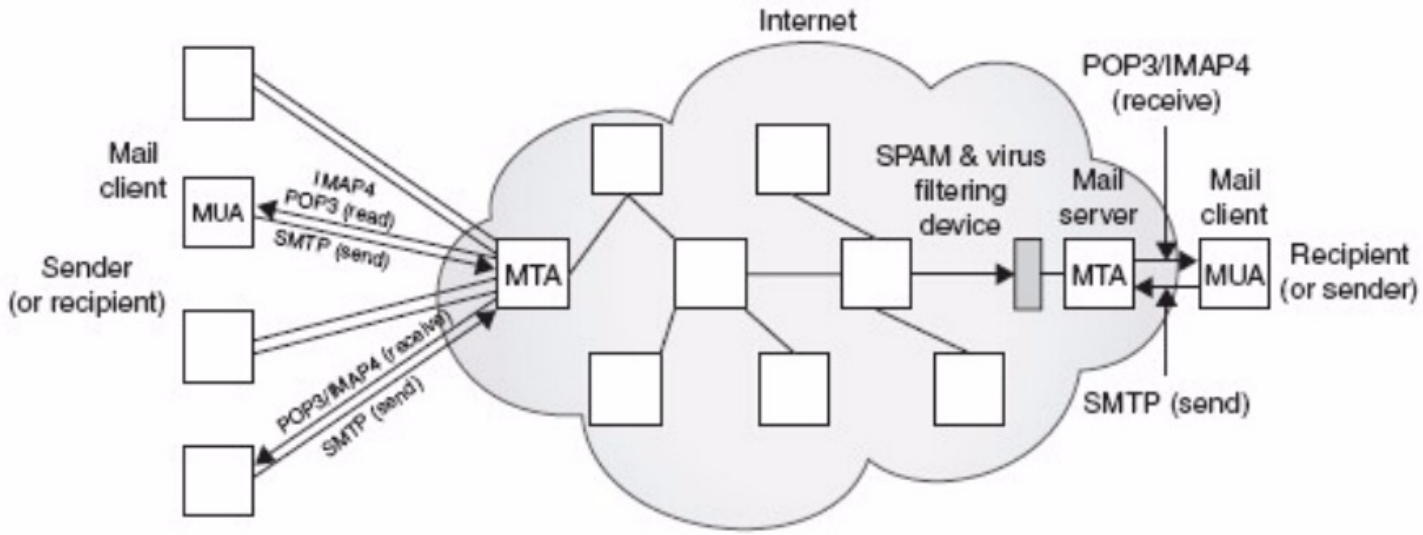

图 17-3 因特网上的 E-Mail 发件人和收件人。客户端通过他们的 MUA 和相应的 MTA 进行通讯，来下载和发送邮件。E-Mail 从一个 MTA “跳”到另一个 MTA，直到到达目的地为止。

17.4.9 Python和POP3

poplib.POP3
流程:

1. 连接到服务器
2. 登录
3. 发出服务请求
4. 退出

17.4.10 poplib.POP3类方法

POP3 类有无数的方法来帮助你下载和离线管理你的邮箱。

方法	描述

`stat`	Get mailbox status. The result is a tuple of 2 integers: (message count, mailbox size).
`retr(msgnum)`	从服务器中得到消息msgnum,并设置其“已读”标志。返回一个长度为3 的元组(rsp, msglines, msgsiz)

17.4.11 poplib.POP3示例

```python
def smtppop3():
    from smtplib import SMTP_SSL
    from poplib import POP3
    from time import sleep

    SMTPSVR = 'smtp.163.com'
    PORT = 465
    POP3SVR = 'pop.163.com'

    sender = 'xxx@163.com'
    pwd = 'xxxxxx'  # 设置发件邮箱的密码，等会登陆会用到
    receiver = sender

    body = '<h1>Hi</h1><h3>骚扰邮件10个</h3>' # 设置邮件正文，这里是支持HTML的
    msg = MIMEText(body,'html')
    msg.set_charset('utf-8')

    msg['subject'] = 'smtp'
    msg['from'] = sender
    msg['to'] = receiver

    # msg = '\r\n\r\n'.join(['\r\n'.join(headers),'\r\n'.join(body)])# 把消息头和消息体放在一起组成一个可以发送的消息

    server = SMTP_SSL(SMTPSVR,PORT)
    server.login(sender, pwd) # 必须要登录
    errs = server.sendmail(sender, receiver, msg.as_string())
    server.quit()
    assert len(errs) == 0, errs
    print 'send ok'
    sleep(8)
```

```
# 收取邮件
recvServer = POP3(POP3SVR)
recvServer.user('xxx') # 发送用户名 login 到服务器
recvServer.pass_('xxxxxx')# 发送密码 passwd
massagecnt,mailboxsiz = recvServer.stat()
print 'massage count',massagecnt,'mailbox size',mailboxsiz
resp,msg,size = recvServer.retr(recvServer.stat()[0])

# strip headers and compare to orig msg
print 'message',msg
sep = msg.index('')
recvBody = msg[sep+1:] # 去掉头部分
print 'header',msg[sep:1]
print recvBody
```

相关模块

详细的参见Page715

`email` –e-mail 处理的包(也支持 MIME)

`smtpd` –SMTP服务器

`base64` –Base 16,32和64数据编码(RFC3548)

`mailbox` –支持mailbox文件格式解析的类

`mimetypes` –在文件名或URL到相关的MIME类型之间转换的模块

其他的网络协议

`ftplib` –FTP协议客户端

`nntplib` –NNTP protocol client
`smtplib` –SMTP 协议客户端
`poplib` –POP3协议客户端
`imaplib` –IMAP4协议客户端
`httplib` –HTTP和HTTPS协议客户端
`telnetlib` –Telnet协议客户端
`gopherlib` –Gopher协议客户端