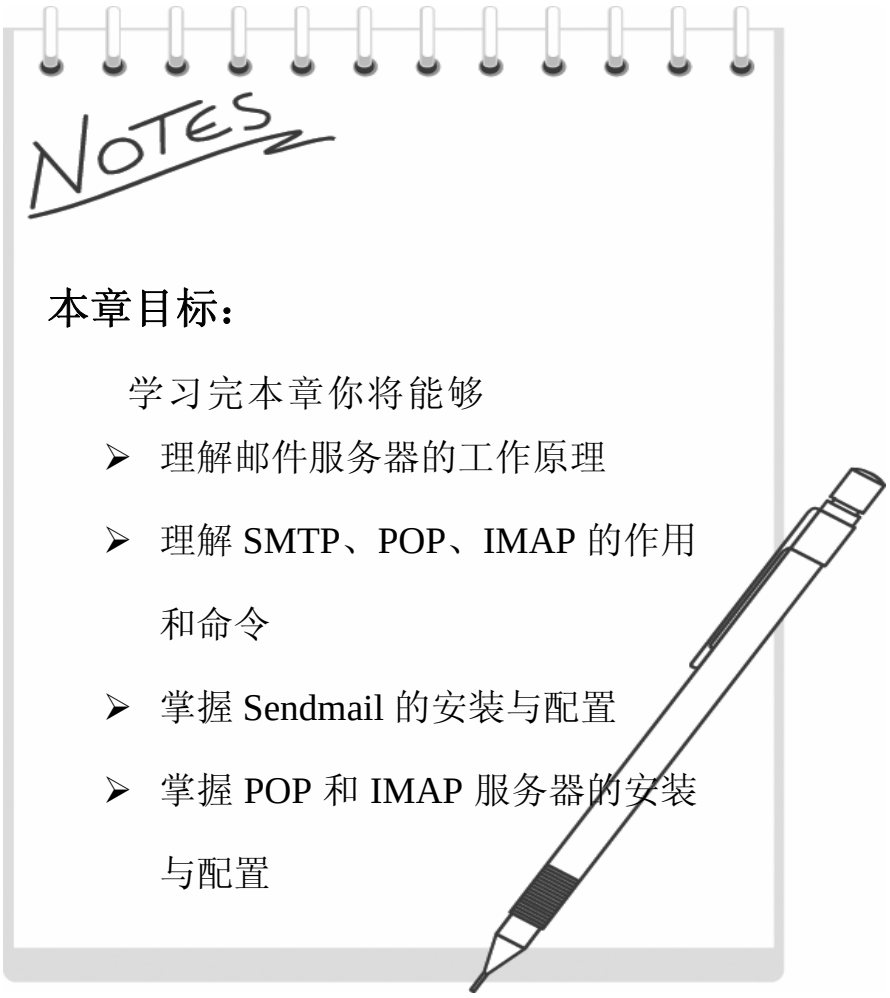
第四章　邮件服务器的配置

本章导读

设想您曾经使用笔和纸，给远方的朋友写信，您写完信后签上自己的名字、然后折起信纸塞入信封，接着您在信封左上处写上您朋友的地址，左下角写上您的地址，在右上角贴上邮票，这时您就可以将其寄出去了。电子邮件 E-Mail 的整个使用过程与写信十分相似，只是使用电脑替代了纸和笔。

电子邮件是当今网络生活不可缺少的一部份。在很多公司，人们通过电子邮件来交流工作，交换资料。在这一章中，我们将讲述邮件服务器的安装与配置。



本章目标：

学习完本章你将能够

- 理解邮件服务器的工作原理
- 理解 SMTP、POP、IMAP 的作用和命令
- 掌握 Sendmail 的安装与配置
- 掌握 POP 和 IMAP 服务器的安装与配置

1. 邮件服务器的基本原理

邮局负责运送装在真实信封里的真实信件，而邮件服务器则用来传送装在电子信封里的电子信件。如果您的朋友（邮件接收者）离您很近（在同一台主机上），那么整个邮件传递过程只与一个邮局有关（运行在本地的邮件服务器上）。如果您的朋友离您很远，那么邮件会从本地邮局（运行在本地的邮件服务器上）传送到远端邮局（运行在远处的邮件服务器）。

在此我们将邮件服务器与邮局作了一个类比，这个类比并不一定能用在所有的细节中。但这个类比将会较好地在本章中帮助我们阐述邮件服务器。

1.1 MUA、MTA 与 MDA 的概念

MUA（Mail User Agent）邮件用户代理，是一个让用户接收、阅读、回复、处理电子邮件的程序。例如，UNIX 中的/bin/mail 程序、mailx 程序、自由软件中的 musb、elm、mb 以及商业软件中的 Zmail，还有 Windows 平台下的 Outlook express 及 foxmail 等就是一个 MUA。绝大多数的 MUA 运行在单机上，一般都仅仅处理邮件传输。

MTA（Mail Transport agent）邮件传输代理，是一个专门用于在主机间传递邮件的程序，类似于邮局。通常，主机中仅有一个 MTA 程序存在。Sendmail 就是一个 MTA 程序。其他的还有：MMDF、Smail 3.x 和 Zmailer 等。

MDA（Mail Delivery Agent）邮件投递代理，例如 sendmail 自己并不完成最终的邮件发送，它要调作其他的程序来完成最后的投递服务。在 Linux 系统中一般是/bin/mail.。

下面，我们举一个简单例子来说明 MUA、MTA 及 MDA 之间的关系。当 host1.koorka.com 上的用户 zhang 给 host2.koorka.com 上的用户 wang 发送一封邮件的时候，将按照下面的步骤进行（如图 4-1 所示）：

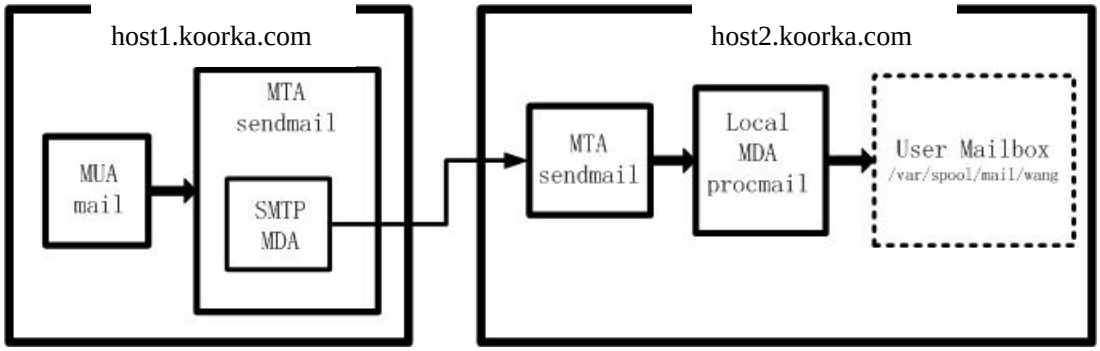


图 4-1

1、用户 zhang 在 host1.koorka.com 上通过 MUA（mail）发送消息（邮件）给本机的 MTA（sendmail）。

2、MTA 将注意到邮件是发送给 host2.koorka.com 上的用户 wang，从 sendmall 的配置中，sendmail 知道可以通过 SMTP 到达 host2.koorka.com，于是 sendmail 将邮件发送给 MDA。

3、MDA 将连接到 host2.koorka.com 的 MTA，并发送该邮件。

4、host2.koorka.com 的 MTA 将注意到邮件是发送给本地（Local）的用户 wang，所以它将邮件发送给本地的 MDA（Local MDA）。

5、Local MDA（procmail）将该邮件存储到用户 wang 的邮箱（/var/spool/mail/wang）。

6、用户直接登录到 Host2.koorka.com 读取邮件。

1.2 电子邮件的组成

电子邮件主要由信头、信体和信封三部分组成。

- **信头**

下面的内容是用 outlook express 收到的一封邮件的信头内容：

```
Received: from sm212.163.com (mta6 [127.0.0.1])
    by mta6.x263.net (Postfix) with SMTP id 497122B7E2
    for <bearzhang@263.net>; Thu, 10 Jul 2003 12:03:25 +0800 (CST)
Received: from sm212.163.com (unknown [202.108.44.212])
    by antispam-5 (Coremail:www.263.net) with SMTP id
N5FLAI3lDD+MHCzU.1
    for <bearzhang@263.net>; Thu, 10 Jul 2003 12:03:25 +0800 (CST)
X-Originating-IP: [202.108.44.212]
Received: from localhost (localhost [127.0.0.1])
    by sm212.163.com (Postfix) with SMTP id 3112D1D101922
    for <bearzhang@263.net>; Thu, 10 Jul 2003 12:03:24 +0800 (CST)
Received: from feiyan (unknown [220.163.32.50])
    by 192.168.1.212 (Coremail:163.com) with SMTP id tAsAAIvlDD8ZAyAy.1
    for <bearzhang@263.net>; Thu, 10 Jul 2003 12:03:24 +0800 (CST)
X-Originating-IP: [220.163.32.50]
From: "sn-xwj" <sn-xwj@163.com>
To: bearzhang@263.net <bearzhang@263.net>
Subject:
X-mailer: Foxmail 4.2 [cn]
Mime-Version: 1.0
```

```
Content-Type: text/plain;
        charset="GB2312"
Content-Transfer-Encoding: quoted-printable
Date: Thu, 10 Jul 2003 12:3:31 +0800
X-Virus: 1
Message-Id: <20030710040324.3112D1D101922@sm212.163.com>
```

我们可以看到，在信头中大多数行都是以一个词后跟一个冒号开始的，每一个词都表明了这一行其他信息的类别。信头中可以有许多种类别信息，有些是必须的，有些是可选的，有些还可以出现多次。

每一台接收到邮件的机器都会自动添加以“Received:”开始的那一行，如果你收的邮件有多个以它开头的行，表明这封邮件可能是被转发或被退回的错误邮件，缩进的那一行是本行的继续，以“Date:”开始的那一行给出了这封邮件寄出时的日期和时间，以“Form:”开始的那一行显示了寄信人的全名和 E-Mail 地址，以“Message-ID:”开始的那一行给出了唯一确认和辨别 E-Mail 的标识号，以“To:”开始的行则列出了所有接收者列表（每个接收者之间用逗号隔开），“X-mailer:”表示发信者使用的 MUA。关于信头的更多细节，我们这里不再细说，希望读者自己参考其它资料。

- **信体**

就是邮件的内容，这个比较好理解。

- **信封**

为了处理发送给多个不同的收信人，邮件服务程序通常会使用了一个“信封”的概念。这个“信封”与通过邮局发送物理信件时使用的信封类似。假设当我们要将一份文档发送给一个纽约和一个东京的朋友：
To:friend1@ny.com,friend2@tokyo.com

首先需将这份文档复印两份，接着将它们装进不同的信封，然后将信投入邮筒，接着邮局根据上面的地址将信送到你朋友的手中。邮件服务程序在发送信件时也是如此。

1.3 邮件信封

下面是用户 john@student.koorka.com 从 student.koorka.com 给 mary@mail.koorka.com 发送一封邮件的过程：

1、client:连接到 mail 服务器的 SMTP 端口（25）。

2、Server: 220 mail.koorka.com ESMTP Sendmail 8.12.9/8.12.9 ready; Mon, 13

July 2006 14:54:08 –0600

3、Client: helo students.koorka.com

4、Server: 250 mail.koorka.com Hello root@students.koorka.com [128.174.5.62], pleased to meet you.

5、Client: mail from: john@students.koorka.com

6、Server: 250 2.1.0 john@students.koorka.com... Sender ok

7、Client: rcpt to: mary@mail.koorka.com

8、Server: 250 2.1.5 mary@mail.koorka.com... Recipient ok

9、Client: data

10、Server: 354 Enter mail, end with "." on a line by itself

11、Client:

Received: (from john@localhost)

by students.koorka.com (8.12.9/8.12.9) id LAA05394;

Mon, 5 Jul 2003 23:46:18 -0500

Date: Mon, 5 Jul 2003 23:46:18 -0500

From: John <john@students.koorka.com>

To: mary <mary@mail.koorka.com>

Message-Id: <200307052346.LAA05394@students.koorka.com>

Subject: 这里是邮件的主题（header）

这里是邮件的内容，可以有多行。

12、Server: 250 2.0.0 e2DKuDw34528 Message accepted for delivery

13、Client: quit

14、Server: 221 2.0.0 mail.koorka.com closing connection

上面的例子说明了 SMTP 在发送一封邮件时的动作（结合图 4-1 来理解更容易），其中，在 SMTP 传送邮件的过程中使用的发送者地址（sender）和接收者地址（recipient）叫做信封（message envelope）。在邮件头中，sender 和 recipient 不是必须的，这一点一定要清除。就像我们通过邮局发送邮件一样，在信封上必须写明发信人和收信人地址，在信件的开头和结尾处不用写，因为邮局工作人员只看我们的信封，他不关心信件的内容。

1.4 邮件队列

在常规的配置中，sendmail 站点在后台等待新的邮件。当一个新的连接到达时，sendmail 将产生一个子进程来处理该连接，父进程将返回继续监听并等待新连接。

当接收到一封信邮件之后，sendmail 子进程将把它放入邮件队列中（通常存放在/var/spool/mqueue），如果该邮件可以被立即处理，它将被递交并从邮件队列中删除。如果不能立即处理，它将被留在邮件对列中，并且子进程将自动结束。

Sendmail 将定期产生子进程对邮件队列进行处理，留在队列中的邮件将会一直保留到被发送出去为止。

对于处理失败的邮件（例如找不到收件人），经过一定的重试次数后，将从邮件队列中删除，并退回邮件。

邮件对列的处理过程如图 4-2 所示：

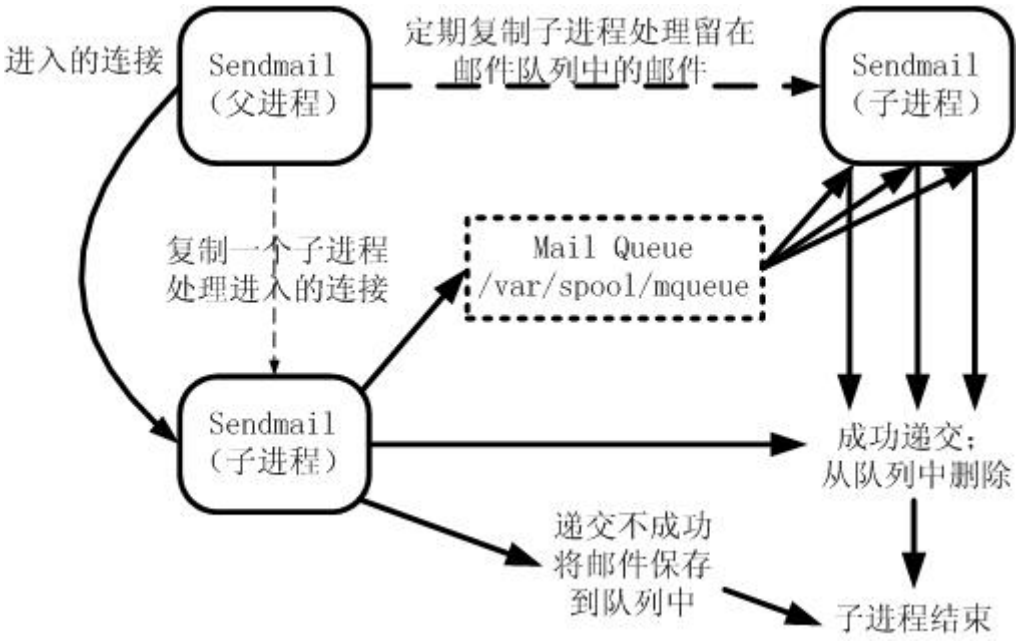
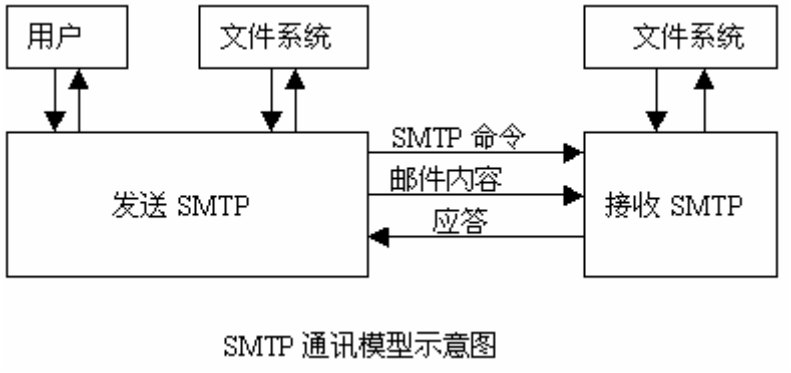


图 4-2

1.5 简单邮件传输协议 SMTP

SMTP 协议是 TCP/IP 协议族中的一员，主要对如何将电子邮件从发送方地址传送到接收方地址，也即是对传输的规则做了规定。SMTP 协议的通信模型并不复杂，主要工作集中在发送 SMTP 和接收 SMTP 上：首先针对用户发出的邮件请求，由发送 SMTP 建立一条连接到接收 SMTP 的双工通讯链路，这里的接收 SMTP 是相对于发送 SMTP 而言的，实际上它既可以是最终的接收者也可以是中间传送者。发送 SMTP 负责向接收 SMTP 发送 SMTP 命令，而接收 SMTP 则负责接收并反馈应答。可大致用下面的通讯模型示意图（图 4-3）来表示：



SMTP 通讯模型示意图

图 4-3

1.5.1 SMTP 命令

从前面的通讯模型可以看出 SMTP 协议在发送 SMTP 和接收 SMTP 之间的会话是靠发送 SMTP 的 SMTP 命令和接收 SMTP 反馈的应答来完成的。在通讯链路建立后，发送 SMTP 发送 MAIL 命令指令邮件发送者，若接收 SMTP 此时可以接收邮件则作出 OK 的应答，然后发送 SMTP 继续发出 RCPT 命令以确认邮件是否收到，如果接收到就作出 OK 的应答，否则就发出拒绝接收应答，但这并不会对整个邮件操作造成影响。双方如此反复多次，直至邮件处理完毕。SMTP 协议共包含 10 个 SMTP 命令，列表如下：

SMTP 命令	命令说明
HELLO ＜domain＞ ＜CRLF＞	识别发送方到接收 SMTP 的一个 HELLO 命令
MAIL FROM: ＜ reverse-path ＞ ＜ CRLF＞	＜reverse-path＞为发送者地址。此命令告诉接收方一个新邮件发送的开始，并对所有的状态和缓冲区进行初始化。此命令开始一个邮件传输处理，最终完成将邮件数据传送到一个或多个邮箱中。
RCPT TO:＜forward-path＞＜CRLF＞	＜forward-path＞标识各个邮件接收者的地址
DATA ＜CRLF＞	接收 SMTP 将把其后的行为看作邮件数据去处理，以＜CRLF＞.＜CRLF＞标识数据的结尾。
REST ＜CRLF＞	退出/复位当前的邮件传输
NOOP ＜CRLF＞	要求接收 SMTP 仅做 OK 应答。（用于测试）
QUIT ＜CRLF＞	要求接收 SMTP 返回一个 OK 应答并关闭传输。
VRFY ＜string＞ ＜CRLF＞	验证指定的邮箱是否存在，由于安全因素，服务器多禁止此命令。
EXPN ＜string＞ ＜CRLF＞	验证给定的邮箱列表是否存在，扩充邮箱列表，也常禁止使用。
HELP ＜CRLF＞	查询服务器支持什么命令

1.5.2 SMTP 的应答码

SMTP 协议的每一个命令都会返回一个应答码，应答码的每一个数字都是有特定含义的，如第一位数字为 2 时表示命令成功；为 5 表失败；3 表没有完成。一些较复杂的邮件程序利用该特点，首先检查应答码的首数字，并根据其值来决定下一步的动作。下面将 SMTP 的应答码列表如下：

应答码	说明
501	参数格式错误
502	命令不可实现
503	错误的命令序列
504	命令参数不可实现
211	系统状态或系统帮助响应
214	帮助信息
220	＜domain＞服务就绪
221	＜domain＞服务关闭
421	＜domain＞服务未就绪，关闭传输信道
250	要求的邮件操作完成
251	用户非本地，将转发向＜forward-path＞
450	要求的邮件操作未完成，邮箱不可用
550	要求的邮件操作未完成，邮箱不可用
451	放弃要求的操作；处理过程中出错
551	用户非本地，请尝试＜forward-path＞
452	系统存储不足，要求的操作未执行
552	过量的存储分配，要求的操作未执行
553	邮箱名不可用，要求的操作未执行
354	开始邮件输入，以"."结束
554	操作失败

对于一次普通的邮件发送，其过程大致为：先建立 TCP 连接，随后客户端发出 HELLO 命令以标识发件人自己的身份，并继续由客户端发送 MAIL 命令，如服务器应答为"OK"，可继续发送 RCPT 命令来标识电子邮件的收件人，在这里可以有多个 RCPT 行，而服务器端则表示是否愿意为收件人接受该邮件。在双方协商结束后，用命令 DATA 将邮件发送出去，其中对表示结束的"."也一并发送出去。随后结束本次发送过程，以 QUIT 命令退出。在前面的邮件信封一节中已经列出了 SMTP 的一次会话过程，再此不再举例。

1.6 邮局协议 POP3

对于在网络上的比较小的结点，支持消息传输系统（MTS）是不实际的。例如，一台工作站可能不具有充足的资源允许 SMTP 服务器和相当的本地邮件传送系统保持序长期连接（例如拨号上网），并持续运行。同样的，将一台个人计算机长时间连

接在 IP 类型网络上的费用也是可观的。

虽然如此，在这样的小结点上允许管理邮件是十分有用的，并且这些结点经常支持一个用户代理来管理邮件。为解决这一问题，能够支持 MTS 的结点就为这些不能支持的结点提供了邮件存储功能。邮局协议-版本 3（POP3）就是使这样的工作站可以用一种比较实用的方法来访问存储于服务器上的储存邮件。通常，这意味着工作站可以从服务器上取得邮件，而服务器为它暂时保存邮件。

在下文中，客户主机指的是利用 POP3 服务的主机，而服务器主机指的是提供 POP3 服务的主机。

当我们从一台 POP3 服务器上读取邮件时（例如使用 Outlook 从 263 的邮件服务器收取邮件），将进行下面的过程：

（1）服务器通过侦听 TCP 端口 110 开始 POP3 服务。

（2）客户端发出 TCP 连接请求。

（3）当连接建立后，POP3 发送确认消息。

（4）客户机和 POP3 服务器相互交换命令和响应，这一过程一直要持续到连接终止。

1.6.1 POP3 的通信

POP3 命令由一个命令和一些参数组成。所有命令以一个 CRLF 对结束。命令和参数由可打印的 ASCII 字符组成，它们之间由空格间隔。命令一般是三到四个字母，每个参数却可达 40 个字符长。

POP3 响应由一个状态码和一个可能跟有附加信息的命令组成。所有响应也是由 CRLF 对结束。现在有两种状态码，“确定” ("+OK")和“失败” ("-ERR")。

对于特定命令的响应是由许多字符组成的。在这些情况中，下面一一表述：在发送第一行响应和一个 CRLF 之后，任何的附加信息行发送，他们也由 CRLF 对结束。当所有信息发送结束时，发送最后一行，包括一个结束字符（十进制码 46，也就是“.”）和一个 CRLF 对。如果信息中的任何一行以结束字符开始，此行就是通过在那一行预先装入结束而进行字符填充的。因此，多行响应由五个 CRLF 结束。当检测多行响应时，客户检测以确认此行是否以结束字符开始。如果是的，而且其后的字符不是 CRLF，此行的第一个字符（结束字符）将被抛弃；如果其后紧跟 CRLF，从 POP 服务器来的响应终止，包括“.CRLF” 的行也不被认为是多行响应的一部分了。

在生命周期中，POP3 会话有几个不同的状态。一旦 TCP 连接被打开，而且 POP3 服务器发送了确认信息，此过程就进入了“确认”状态。在此状态中，客户必须向 POP3 服务器确认自己是其的客户。一旦确认成功，服务器就获取与客户邮件相关的资源，此时这一过程进入了“操作”状态。在此状态中，客户提出服务，当客户发

出 QUIT 命令时，此过程进入了“更新”状态。在此状态中，POP3 服务器释放在“操作”状态中取得的资源，并发送消息，终止连接。

POP3 服务器可以拥有一个自动退出登录的记时器。此记时器必须至少可以记录 10 分钟。这样从客户发送的消息才可能刷新此记时器。当记时器失效时，POP3 会话并不进入“更新”状态，而是关闭 TCP 连接，而且不删除任何消息，不向客户发送任何响应。

1.6.2 “确认”状态

一个 TCP 连接由 POP3 客户打开，POP3 服务器发送一个单行的确认。这个消息可以是由 CRLF 结束的任何字符。例如，它可以是：

S: +OK POP3 server ready

注意：这个消息是一个 POP3 应答。POP3 服务器应该给出一个“确定”响应作为确认。

此时 POP3 会话就进入了“确认”状态。此时，客户必须向服务器证明它的身份。在此介绍两种可能的处理机制，一种是 USER 和 PASS 命令，另一种是在后面要介绍的 APOP 命令。

用 USER 和 PASS 命令进行确认过程，客户必须首先发送 USER 命令，如果 POP3 服务器以“确认”状态码响应，客户就可以发送 PASS 命令以完成确认，或者发送 QUIT 命令终止 POP3 会话。如果 POP3 服务器返回“失败”状态码，客户可以再发送确认命令，或者发送 QUIT 命令。

当客户发送了 PASS 命令后，服务器根据 USER 和 PASS 命令的附加信息决定是否允许访问相应的存储邮件。

一旦服务器通过这些数据决定允许客户访问储存邮件，服务器会在邮件上加上排它锁，以防止在进入“更新”状态前对邮件的改变（防止其它客户端修改邮件）。如果成功获得了排它锁，服务器返回一个“确认”状态码。会话进入“操作状态”，同时没有任何邮件被标记为删除。如果邮件因为某种原因不能打开（例如，排它锁不能获得，客户不能访问相应的邮件或者邮件不能进行语法分析），服务器将返回“失败”状态码。在返回“失败”状态码后，服务器会关闭连接。如果服务器没有关闭连接，客户可以重新发送确认命令，重新开始，或者发送 QUIT 命令。

在服务器打开邮件后，它为每个消息指定一个消息号，并以八进制表示每个消息的长度。第一个消息被指定为 1，第二个消息被指定为 2，以此类推，第 N 个消息被指定为 N。在 POP3 命令和响应中，所有的消息号和长度以十进制表示。

下面是对上述三条命令（USER、PASS、QUIT）的总结：

USER name ：指定邮箱的字符串，这对服务器至关重要 仅在 USER 和 PASS 命令失败后或在“确认”状态中使用。 入果返回“+OK”，表示是有效邮箱；如果返回“-ERR”，表示无效邮箱

例如：

```
C: USER bearzhang
S: +OK bearzhang is a real hoopy frood
...
C: USER koorka
S: -ERR sorry, no mailbox for koorka here
```

PASS string ：“string”是邮箱的口令。仅在“确认”状态中 USER 命令成功后使用（因为此命令只有一个参数，因此空格不再作为分隔符，而作为口令的一部分）。如果返回“+OK”，邮件锁住并已经准备好；如果返回“-ERR”，表示无效口令或无法锁住邮件

例如：

```
C: USER bearzhang
S: +OK mrose is a real hoopy frood
C: PASS secret
S: +OK mrose's maildrop has 2 messages (320 octets)
...
C: USER mrose
S: +OK mrose is a real hoopy frood
C: PASS secret
S: -ERR maildrop already locked
QUIT （无） （无） +OK
C: QUIT
S: +OK dewey POP3 server signing off
```

1.6.3 “操作”状态

一旦客户向服务器成功地确认了自己的身份，服务器将锁住并打开相应的邮件，这时 POP3 会话进入“操作”状态。现在客户可以重复下面的 POP3 命令，对于每个命令服务器都会返回应答。最后，客户发送 QUIT 命令，会话进入“更新”状态。

下面是在“操作”状态中可用的命令：

STAT ：仅在“操作”状态下可用。 服务器以包括邮件信息的响应做为“确认”。为简化语法分析，所有的服务器要求使用邮件列表的特定格式。“确认”响应由一个空格，以八进制表示的邮件数目，一个空格和邮件大小。这是最小实现，高级的实

现还需要别的信息。

注意：被标记为删除的信件不在此列。

+OK： nn mm

C: STAT

S: +OK 2 320

LIST [msg] ：msg 表示信件数目（可选），如果出现，不包括标记为删除的信件。 仅在“操作”状态下可用。 如果给出了参数，且 POP3 服务器返回包括上述信息的“确认”，此行称为信息的“扫描表”。 如果没有参数，服务器返回“确认”响应，此响应便以多行给出。在初的+OK 后，对于每个信件，服务器均给出相应的响应。

为简化语法分析，所有服务器要求使用扫描表的特定格式。它包括空格，每个邮件的确切大小。这是最小实现，高级的实现还需要别的信息。

注意：被标记为删除的信件不在此列。

+OK：其后跟扫描表；

-ERR：无扫描。

```
C: LIST
S: +OK 2 messages (320 octets)
S: 1 120
S: 2 200
S: .
...
C: LIST 2
S: +OK 2 200
...
C: LIST 3
S: -ERR no such message, only 2 messages in maildrop
```

RETR msg：不包括标记为删除的信件数目。 仅在“操作”状态下可用。 如果服务器返回“确认”，给出的响应是多行的。在初始的+OK 后，服务器发送与给定信息号对应的信息，对于多行响应，注意字节填充终止符。 +OK：消息在其后；

-ERR：其后无消息。

C: RETR 1

S: +OK 120 octets

S: <the POP3 server sends the entire message here>

S: .

DELE msg：不包括标记为删除的信件数目。 仅在“操作”状态下可用。 服务器将此信件标记为删除，以后任何关于此信件的操作就会产生错误。服务器在会话进入“更新”状态前不会真正删除此信件。 +OK：信件被删除；

-ERR：无此信件。

C: DELE 1

S: +OK message 1 deleted

...

C: DELE 2

S: -ERR message 2 already deleted

NOOP ：仅在“操作”状态下可用。 服务器仅返回“确认”。 +OK C: NOOP

S: +OK

RSET ：仅在“操作”状态下可用。 所有被标记为删除的信件复位，服务器返回“确认”。 +OK C: RSET

S: +OK maildrop has 2 messages (320 个字符)

1.6.4 “更新”状态

当客户在“操作”状态下发送 QUIT 命令后，会话进入“更新”状态。（注意：如果客户在“确认”状态下发送 QUIT 后，会话并不进入“更新”状态。）

如果会话因为 QUIT 命令以外的原因中断，会话并不进入“更新”状态，也不从服务器中删除任何信件。

命令 参数 限制 说明 响应 例子

QUIT ：服务器删除所有标记为删除的信件，然后释放排它锁，并返回这些操作的状态码。最后 TCP 连接被中断。 +OK C: QUIT

S: +OK dewey POP3 server signing off (清空标记邮件)

...

C: QUIT

S: +OK dewey POP3 server signing off

1.6.5 POP3 的其它命令

以上讨论的命令是对 POP3 服务的最小实现。以下说明的可选命令允许客户更方便地处理信件，这是一个比较一般的 POP3 服务实现。

TOP msg n ：一个是未被标记为删除的信件数，另一个是非负数（必须提供） 仅在“操作”状态下使用。 如果服务器返回“确认”，响应是多行的。在初始的+OK 后，服务器发送信件头，一个空行将信件头和信件体分开，对于多行响应要注意字节填充终止符。

注意：如果客户要求的行数比信件体中的行数大，服务器会发送整个信件。

+OK：其后有信件头；

-ERR：其后无类似消息。

C: TOP 1 10

S: +OK

S: <服务器发送消息头，一个空行和信件的头 10 行>

S: .

...

C: TOP 100 3

S: -ERR no such message

UIDL [msg] 信件数（可选）。如果给出信件数，不包括被标记为删除的信件。 仅在“操作”状态下使用。 如果给出了参数，且 POP3 服务器返回包括上述信息的“确认”，此行称为信息的“独立-ID 表”。

如果没有参数，服务器返回“确认”响应，此响应便以多行给出。在初的+OK 后，对于每个信件，服务器均给出相应的响应。此行叫做信件的“独立-ID 表”。

为简化语法分析，所有服务器要求使用独立-ID 表的特定格式。它包括空格和信件的独立-ID。

信件的独立-ID 由 0x21 到 0x7E 字符组成，这个符号在给定的存储邮件中不会重复。

注意：信件不包括被标记为删除的信件。

+OK：其后是独立-ID 表；

-ERR：其后无类似信件。

C: UIDL

S: +OK

S: 1 whqtswO00WBw418f9t5JxYwZ

S: 2 QhdPYR:00WBw1Ph7x7

S: .

...

C: UIDL 2

S: +OK 2 QhdPYR:00WBw1Ph7x7

...

C: UIDL 3

S: -ERR no such message, only 2 messages in maildrop

APOP name digest 指定邮箱的字串和 MD5 摘要串。 仅在 POP3 确认后的“确认”状态中使用。 通常，每个 POP3 会话均以 USER/PASS 互换开始。这导致了用户名和口令在网络上的显式传送，这不会造成什么危险。但是，许多客户经常连接到服务检查信件。通常间隔时间比较短，这就加大了泄密的可能性。

另一种提供“确认”过程的方法是使用 APOP 命令。

实现 APOP 命令的服务器包括一个标记确认的时间戳。例如：在 UNIX 上使用 APOP 命令的语法为：process-ID.clock@hostname，其中进程-ID 是进程的十进制的数，时钟是系统时钟的十进制表示，主机名与 POP3 服务器名一致。

客户记录下此时间戳，然后以送 APOP 命令。name 语法和 USER 命令一致。Digest 是采用 MD5 算法产生的包括时间戳和共享密钥的字串。此密钥是客户和服务器共知的，应该注意保护此密钥，如果泄密，任何人都能够以用户身份进入服务器。

如果服务器接到 APOP 命令，它验证 digest，如果正确，服务器返回“确认”，进入“操作”状态；否则，给出“失败”并停留在“确认”状态。

注意：共享密钥的长度增加，解读它的难度也相应增加，这个密钥应该是长字符串。

+OK：邮件锁住并准备好；

-ERR：拒绝请求。

S: +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>

C: APOP mrose c4c9334bac560ecc979e58001b3e22fb

S: +OK maildrop has 1 message (369 octets)

在此例子中，共享密钥 <1896.697170952@dbc.mtview.ca.us>tanstaaf 由 MD5 算法生成，它产生了 digest 值， c4c9334bac560ecc979e58001b3e22fb

1.6.6 POP3 命令总结

基础的 POP3 命令：

USER name 在“确认”状态有效

PASS string

QUIT

STAT 在“操作”状态有效

LIST [msg]

RETR msg

DELE msg

NOOP

RSET

QUIT 在“更新”状态有效

可选的 POP3 命令：

APOP name digest 在“确认”状态有效

TOP msg n 在“操作”状态有效

UIDL [msg]

POP3 响应：

+OK

-ERR

注意：除了 STAT，LIST 和 UIDL 的响应外，其它命令的响应均为"+OK"和"-ERR"。响应后的所有文本将被客户略去。

1.6.7 POP3 会话实例

S: <等待连接到 TCP 端口 110>

C: <打开连接>

S: +OK POP3 server ready <1896.697170952@dbc.mtview.ca.us>

C: APOP bearzhang c4c9334bac560ecc979e58001b3e22fb

S: +OK bearzhang's maildrop has 2 messages (320 octets)

C: STAT

S: +OK 2 320

C: LIST

S: +OK 2 messages (320 octets)

S: 1 120

S: 2 200

S: .

C: RETR 1

S: +OK 120 octets

S: <服务器发送信件 1>

S: .

C: DELE 1

S: +OK message 1 deleted

C: RETR 2

S: +OK 200 octets

S: <服务器发送信件 2>

S: .

C: DELE 2

S: +OK message 2 deleted

C: QUIT

S: +OK dewey POP3 server signing off (maildrop empty)

C: <关闭连接>

S: <等待下一次连接>

1.6.8 POP3 的消息格式

在会话过程中的消息格式都假定与 Internet 文本消息格式标准一致。应该注意的是，由于各个服务器对于换行符的处理不同，因此计数不一定相同。通常，在“确认”状态中，服务器能够以八进制计算信件的大小。例如，如果在打开储存邮件时服务器内部认定换行符代表一个字符，一般服务器在计算它时作为两个字符计。注意，以终止符开始的消息行不被计数两次，因为客户将在接收到多行响应后删除所有字节填充。

1.7 IMAP 协议简介

IMAP(Internet Message Access Protocol，Internet 消息访问协议)是与 POP3 对应的另一种协议，为美国斯坦福大学在 1986 年开始研发的多重邮箱电子邮件系统。它能够从邮件服务器上获取有关 E－mail 的信息或直接收取邮件，具有高性能和可扩展性的优点。IMAP 为很多客户端电子邮件软件所采纳，如 Outlook Express、Netscape Messenger 等，支持 IMAP 的服务器端的软件也越来越多，如 CriticalPath、Eudora、iPlanet、Sendmail 等。

读者不禁要问，POP3 也是接收邮件的协议，现在不是用得很好么，为何还要用 IMAP 协议呢？

POP3 协议的不足

的确，POP 作为 Internet 上邮件的第一个离线协议标准，允许用户从服务器上把邮件下载到本地主机上，同时删除保存在邮件服务器上的邮件，从而使用户不必长时间地与邮件服务器连接，很大程度上减少了服务器和网络的整体开销。

但 POP3 有其天生的缺陷，即当用户接收电子邮件时，所有的信件都从服务器上清除并下载到客户机。在整个收信过程中，用户无法知道邮件的具体信息，只有照单全收入硬盘后，才能慢慢浏览和删除。这使用户几乎没有对邮件接收的控制决定权。一旦碰上邮箱被轰炸，或有比较大的邮件，用户不能通过分析邮件的内容及发信人地址来决定是否下载或删除，从而造成系统资源的浪费。而 IMAP 协议不但可以克服 POP3 的缺陷，而且还提供了更强大的功能。

对 IMAP 的解析

IMAP 提供操作的三种模式

在线方式：邮件保留在 Mail 服务器端，客户端可以对其进行管理。其使用方式与 WebMail 相类似。

离线方式：邮件保留在 Mail 服务器端，客户端可以对其进行管理。这与 POP 协议一样。

分离方式：邮件的一部分在 Mail 服务器端，一部分在客户端。这与一些成熟的组件包应用（如 LotusNotes/Domino）的方式类似。

IMAP 工作原理及特性

在在线方式下，IMAP 允许用户象访问和操纵本地信息一样来访问和操纵邮件服务器上的信息。IMAP 软件支持邮件在本地文件夹间和服务器文件夹间的随意拖动，以把本地硬盘上的文件存放到服务器上，或将服务器上的文件取回本地，所有的功能仅需要一次鼠标拖放的操作来实现。

在用户端可对服务器上的邮箱建立任意层次结构的文件夹，并可灵活地在文件夹间移动邮件，标出那些读过或回复过的邮件，删除对你来说无用的文件。

IMAP 提供的摘要浏览功能可以让你在阅读完所有的邮件到达时间、主题、发件人、大小等信息，同时还可以享受选择性下载附件的服务。比如一封邮件里含有 3 个附件，而其中只有 1 个附件是您需要的，则可以选择只下载这 1 个附件。你可以充分了解后才作出是否下载，是全部下载还是仅下载一部分等决定，使用户不会因下载垃圾信息而占用宝贵的空间和浪费网费。

IMAP 还提供基于服务器的邮件处理以及共享邮件信箱等功能。邮件（包括已下载邮件的副本）在手动删除前保留在服务器中，这有助于邮件档案的生成和共享。用户可在任何客户机上都可查看服务器上的邮件。这让那些漫游用户感到很方便。

同时 IMAP 也象 POP3 一样，允许用户从服务器上下载信息到他们的电脑上，这意味着他们仍然可以在离线方式下阅读邮件。

在分离状态下，本地系统上的邮件状态和服务器上的邮件状态，可能和以后再连接时不一样。此时，IMAP 的同步机制解决了这个问题。IMAP 邮件的客户端软件能够记录用户在本地的操作，当他们连上网络后会把这些操作传送给服务器，服务器也会告诉客户端软件，当用户离线的时候服务器端发生的事件，比如有新邮件到达等，以保持服务器和客户端的同步。

在 IMAP 下可定义供其他拥有特别访问权利的用户使用的共享文件夹，而使用 POP 不能实现共享邮件信箱和共享邮件，仅能通过抄送给或用手工传送邮件。共享信箱将使以使用 Internet 邮件为主的工作组的工作变得更为容易。

IMAP 还提供许多特别的功能比如建立子目录和通过 IMAP 访问 Usenet。在系统管理员方面，IMAP 也提供了一整套可用的特性。

IMAP 的监听端口为 143,消息的内在时间和日期是由服务器给出的，而不是在 RFC822 中信头给出的时间和日期，是消息最后到达的真实日期和时间。如果信息是被 IMAP 的 Copy 命令投递的，这应当是源信息的内在时间和日期；如果信息是被 IAMP 的 Append 命令投递的，这应当是由 Append 命令专门描述的时间和日期。

在 IMAP 协议中定义了很多的命令，可用 telnet 来执行，例如 Authenticate、List 和 Close 等等，此处不再详述。

实现 IMAP 的不足

在利用服务器磁盘资源方面，IMAP 不如 POP3。由于使用 POP 时服务器端的邮件被下载到客户机的同时会删除，因而不占用额外空间用以存放旧的邮件。而 IMAP 服务器将保持旧的邮件，占用了额外空间，而且需要定期地删除旧邮件。

同时，由于用户查阅信息标题和决定下载哪些附件，也需要一定时间，因此链接时间也比 POP 方式长。

在应用方面，由于 IMAP 比较复杂，给开发者开发服务器和客户机的软件带来一些难题。对于 ISP 来说，采用 IMAP 意味着要花钱购买相关商业软件，同时会付出高额技术支撑费用，因而商用的实现方案还不多。

2. sendmail

2.1 从源代码安装 sendmail

1、获取源代码。

ftp://ftp.sendmail.org/pub/sendmail/sendmail.8.12.9.tar.gz

2、安装前的准备工作：

将 sendmail.8.12.9.tar.gz 拷贝到/tmp 目录下，并解压。

tar –zxvf sendmail.8.12.9.tar.gz

阅读相关的文档：

/tmp/sendmail-8.12.9/README

/tmp/sendmail-8.12.9/INSTALL

/tmp/sendmail-8.12.9/sendmail/README

/tmp/sendmail-8.12.9/sendmail/SECURITY

/tmp/sendmail-8.12.9/devtools/README

/tmp/sendmail-8.12.9/devtools/Site/README

/tmp/sendmail-8.12.9/libmilter/README

/tmp/sendmail-8.12.9/mail.local/README

/tmp/sendmail-8.12.9/smrsh/README

/tmp/sendmail-8.12.9/cf/README

3、编译并安装 sendmail

（1）创建配置文件 site.config.m4:

cp /tmp/sendmail-8.12.9/devtools/OS/Linux /tmp/sendmail-8.12.9/devtools/Site/site.config.m4 ，并做相应的修改。

RedHat Linux 的手册缺省放在/usr/share/man/，所以我们将 site.config.m4 内的：

define(`confMANROOT', `/usr/man/man') 改为：

define(`confMANROOT', `/usr/share/man/man')

要使 SMTP 支持基于 SSL 的认证，需要在 site.config.m4 文件内增加下面几行：

dnl Stuff for TLS

APPENDDEF(`confINCDIRS', `-I/usr/local/include')

APPENDDEF(`confLIBDIRS', `-L/usr/local/lib')

APPENDDEF(`conf_sendmail_ENVDEF', `-DSTARTTLS')

dnl add to previous direction APPENDDEF(`conf_sendmail_LIBS', `-lssl -lcrypto')

APPENDDEF(`conf_sendmail_LIBS', `-lsasl -lssl -lcrypto')

支持其它的功能和定义方法，请参考/tmp/sendmail-8.12.9/sendmail/README 文件。这里只要求基本的 sendmail 功能（收发电子邮件），所以对 site.config.m4 文件除修改手册的存放位置外，不作其它修改。

（2）、在/tmp/sendmail-8.12.9 下执行:

./Build –f /tmp/sendmail-8.12.9/devtools/Site/site.config.m4 （编译 sendmail）

（3）、rpm –e sendmail （卸载掉系统自带的 sendmail）。

（4）、在/tmp/sendmail-8.12.9 下执行:

./Build install （安装 sendmail）

4、编译并安装 sendmail 的配置文件（sendmail.cf 和 submit.cf）

（1）、创建 sendmail.mc 文件：

cd /tmp/sendmail-8.12.9/cf/cf/

cp generic-linux.mc sendmail.mc

缺省的配置文件不支持中文，所以我们不用对该文件作任何修改。

（2）、执行./Build sendmail.mc sendmail.cf （编译并生成 cf 文件）

（3）、执行./Build intstall-cf （安装 cf 文件）

2.2 配置并启动 sendmail

sendmail 安装完成之后，我们需要对其进行配置才能正常使用。

1、创建用户 smmsp 和用户组 smmsp，用于运行 sendmail

adduser smmsp –s /sbin/nologin –d /var/spool/mqueue

adduser mail –s /sbin/nologin -d /var/spool/mail （邮件管理用户）

2、创建队列目录

mkdir /var/spool/mqueue

chmod 700 /var/spool/mqueue

3、修改邮件队列存储目录的权限

chown smmsp.smmsp /var/spool/clientmqueue

chmod 770 /var/spool/clientmqueue

4、用户邮件存储目录的权限

chown root.mail /var/spool/mail

chmod 775 /var/spool/mail

5、创建文件/etc/mail/local-host-names，其内容为本机的拥有的域名信息，我们希望发送给 zhang@koorka.com 和 zhang@mail.koorka.com 的邮件都发送给本机（自己），故该文件应该包含入下内容：

koorka.com.

mail.koorka.com.

6、最后还要创建别名数据库。在/etc/mail/aliases 目录下创建文件 aliases，内容如下：

MAILER-DAEMON: postmaster

postmaster: root

bin: root

daemon: root

nobody: root

……

然后生成 aliases 库：

newaliases

7、启动 Sendmail：

/usr/sbin/sendmail -bd -q20m （以守护进程的方式运行，并且每 20 分钟处理一次邮件队列）。

如果出现错误提示，请参照错误提示进行修改，并查看邮件日志（/var/log/maillog）没有提示的话，用 netstat –ln 应该可以看到 25 端口已被监听。

2.3 imap 软件的安装

安装 imap 软件，以便用户可以通过 outlook express 等 MUA 软件，接收服务器上的邮件。

1、rpm –ivh imap-2001a-10.0as.i386.rpm （不同发行版版本号不同）

2、编辑/etc/xinetd.d/ipop3，将该文件内的 disable=yes 改为 disable=no

3、重新启动 xinet 服务

/etc/rc.d/init.d/xinetd restart

此时运行 netstat –ln 可以看到 110 端口被监听。

2.4 测试

接下来是测试邮件服务器的邮件发送/接收功能。

在邮件服务器上分别创建用户：u1、u2，并设置密码。

找一台 windows 客户机，分别创建邮件账户 u1@koorka.com 和 u2@koorka.com，收服务器和发件服务器都设置为 mail.koorka.com，如图 4-4 所示：

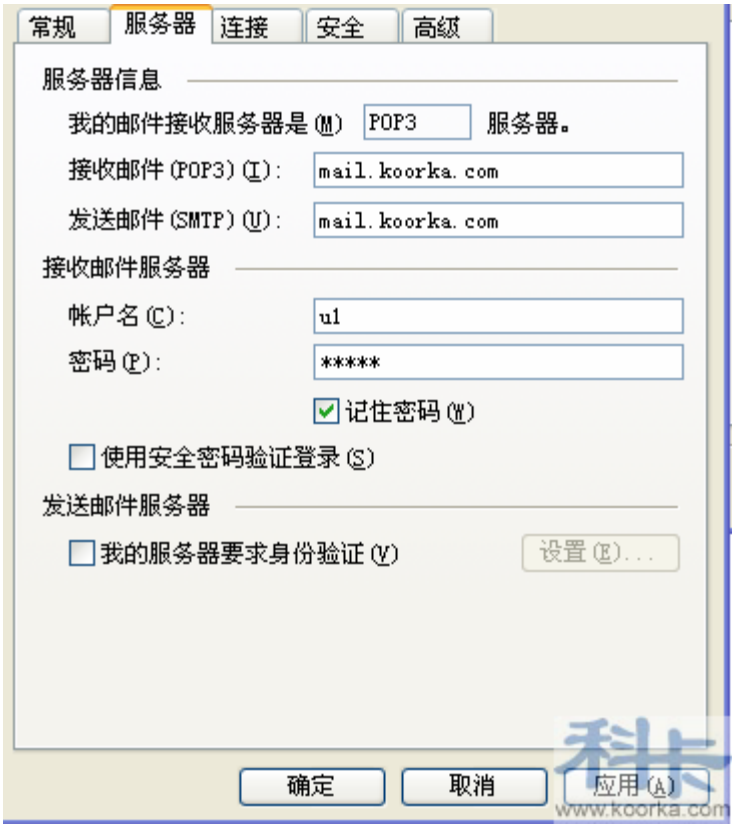


图 4-4

用 u1@koorka.com 给 u2@koorka.com 发一封邮件，并检查 u2@koorka.com 是否收到该邮件。（可以）

4、用 u1@koorka.com 给互联网络上的用户发送一封邮件（不可以），应该出现下面的提示：

由于服务器拒绝收件人之一，无法发送邮件。被拒绝的电子邮件地址是“bearzhang@263.net”。 主题 'eteast', 帐户: 'u1', 服务器: 'mail.zhang.com', 协议: SMTP, 服务器响应: '550 5.7.1 <zhangzhx@163.net>... Relaying denied', 端口: 25, 安全(SSL): 否, 服务器错误: 550, 错误号: 0x800CCC79