第22学时　使用CGI程序发送电子邮件

毫无疑问，在你进行Web冲浪时，要填写一个窗体，以便在以后用来发送电子邮件。这些窗体常常用作信址列表、故障报告、客户支持、爱好者邮件和其他各种可以想像到的用途。

在本学时中，我们将要介绍如何用 Perl程序发送邮件，并且讲述一个简短的 Web页示例，你可以用它来生成电子邮件。我们将使你能够创造性地使用这个 Web页。

在本学时中，你将要学习：

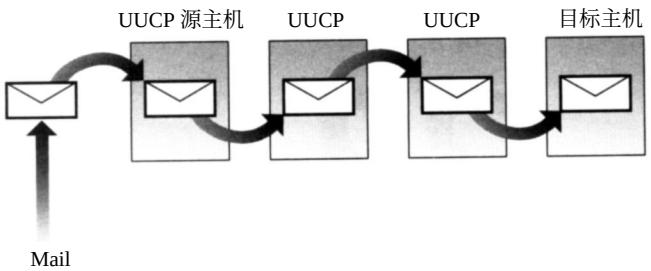
- 关于如何运行Internet电子邮件特性的简单介绍。
- 如何在UNIX和非UNIX系统下发送邮件。
- 如何建立发送邮件的Web窗体。

22.1　Internet邮件入门

在你将编程技巧用于以Perl来发送电子邮件之前，首先必须学习一些关于电子邮件特性如何在Internet上运行的一些知识。

在Perl问世之前，在美国的国家计算机安全委员会（ NCSA）尚未注意到Web的远大前景并且调制解调器的速度还比较慢的时候，全球的许多人就已经在使用电子邮件在所谓的 UNIX至UNIX拷贝（UNIX-to-UNIX copy, UUCP）的系统上进行通信了。当你在这个老式系统上发送电子邮件时，本地系统把你的电子邮件封装好，然后转发给系统链中的下一个系统，下一个系统又将电子邮件封装好，转发给下一个系统，如此传递下去。线路上的每个系统都要给邮件添加一点信息，表示它对邮件进行了处理，然后传递下去，如图 22-1所示。

图22-1　将邮件从一个系统传递到下一个系统



很明显，这种邮件传递的方法可以称为存储与转发法。后来UUCP系统被别的方法所取代，不过存储与转发的基本方法仍然没有变。当你从你的 PC发送电子邮件时，另一个系统负责接收该邮件，再将它转发给另一个系统，然后该系统又将邮件转发给下一个系统，直到最后由目标系统接收到邮件为止。

不过，如今这些协议完全发生了变化。目前最常用的方法是使用简单邮件传输协议（Simple Mail Transport Protocol, SMTP）将邮件发送到系统链上（见图22-2）。若要检索邮件，连接的目标端通常使用邮局协议（ Post Office Portocol, POP）或Internet邮件访问协议（Internet Message Access Protocol, IMAP）。下面用于发送电子邮件的协议是 SMTP。

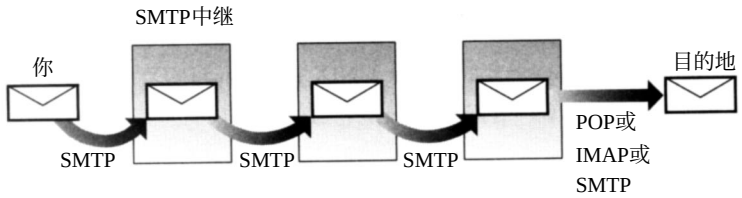


图22-2 发送电子邮件时使用的不同协议

22.1.1 发送电子邮件

若要发送电子邮件，需要两样东西，即邮件传输代理或 SMTP中继主机。

遗憾的是，它们都是很难理解的术语，不过下面将对它们加以解释。

邮件传输代理（Mail Transport Agent, MTA）是驻留在你的计算机上的一个程序，它通常是你的操作系统所配备的一个程序，负责接收电子邮件并正确地将它们转发。当你的操作系统安装时，MTA通常已经作好正确的配置。UNIX系统上的常用MTA称为sendmail。semdmail程序负责取出一个电子邮件并确定如何将它传递到目的地。

若要在UNIX下发送电子邮件，请在命令行上使用下面这个语句：

```
$ /bin/echo "Subject:Test\n\nHello, World!" | sendmail foo@bar.com
```

上面这个代码段将一个短邮件发送到 foo@bar.com。sendmail程序负责为你解决所有难以处理的工作，比如决定使用哪个邮件中继主机，处理被拒绝的返回邮件等。

如果你使用Microsoft Windows或Macintosh操作系统，那么你将不具备内置的 MTA。不过Perl模块使你能够直接发送邮件。 Net::SMTP模块可以在没有介入的 MTA的情况下发送邮件，但是你必须知道你的SMTP中继主机的名字。这个名字是用于发送邮件的“ 邮件主机 ”的主机名，当你用你的帐户进行登录时，你将被赋予该主机名。请索取中继主机的名字，并将它写在某个地方，以后你会用到它。

你可以使用不同的“ 邮件主机 ”，以便发送和接收邮件。本学时中你需要发送邮件的主机名。

请记住，依靠SMTP中继的程序必须将正确的中继主机内置于软件之中，否则该进程将不能运行。

正确的“ SMTP中继主机名 ”取决于你从何处发送你的邮件。如果你从家中发送邮件，那么你的家庭 Internet服务提供商（ISP）帐户为你赋予一个SMTP中继主机名。如果你用租用的 Web服务器上的帐户发送邮件，那么就需要该服务器的中继主机的名字。当邮件从中继主机并不知道的一个系统发送过来，邮件中继主机便拒绝转发该邮件。

22.1.2 发送邮件时首先应该注意的问题

在下一节中，我们将要介绍一个新函数，即 send_mail，使用这个函数，你就能够用 Perl程序发送电子邮件。这个函数虽然非常有用，但同时它也有很大的危险性。将邮件发送给某个人，将会在一定程序上侵犯他的隐私权。你会要求邮件的收件人在你的邮件上耗费一定的

时间和磁盘空间，还会要求你与收件人之间的每个系统为你中继该邮件。对于一个完全陌生的人来说，这样做是很不合适的。

下面是你在使用Perl或任何其他工具发送电子邮件时应该注意的问题：

- 首先使用众所周知的地址（比如你自己的地址）测试你的代码并发送一些短邮件。这时，随时都可能产生一些问题，你应该设法避免发生问题。
- 不要发送有人主动提供的商业性电子邮件。这类商业性电子邮件通常称为垃圾邮件，这类邮件已经成为Internet上的一个令人头痛的大问题。少数人喜欢接收这类邮件，而其他人的反应则不同，他们有的对垃圾邮件非常反感，有的则痛恨之极。发送此类邮件的企业将会成为许多人唾骂的对象。当你得到一个邮件地址后，应该问一问是否可以在以后向它发送电子邮件。如果有人要求从你的邮件地址列表中删除他的地址，那么你应该尊重他的要求。
- 无论对方要求还是没有要求，都不要一次就发送很长的邮件，要按适当的速度来发送。首先，你的本地邮件中继主机会因为急匆匆发送邮件而不堪重负，你的本地 ISP将会终止你的帐户，以控制受损害的程度。其次，如果目标 ISP因为你的邮件太大而无法承受，该ISP就会阻塞从你的域发送过来的全部邮件。如果根本无法向较大的域（如 aol.com、hotmail.com等）发送邮件，那么你的日子一定不会好过，并且很可能使你的帐户与你的ISP之间的联系被中断，结果造成人们对你的指控。
- 应该提供很好的返回邮件的地址，尤其是在邮件报头中要写明这个地址。应该确保你的电子邮件的From：(或Reply To：)地址正确无误，尤其是当邮件是从一台计算机发送时更应保证地址的正确性。你可以使用 Perl伪造电子邮件，但是伪造的邮件包含一个返回给你的指针。伪造的邮件会使你陷入巨大的麻烦之中。
- 请始终都使用你自己的邮件中继主机。滥用其他系统的邮件中继主机会使你的帐户迅速停用，并使你遭人指控，甚至出现更糟糕的问题。
- 不要将很长的电子邮件或者许多很短的邮件发送给靠不住的人，这称为邮件炸弹，可能导致你的帐户被停用，并引起法律上的麻烦。

上面这些建议并非全部仅仅是一些好的网上礼仪。如果违背这些原则， ISP可能将你从它的服务对象列表中删除掉，而且ISP和邮件的收件人会指控你。当注册你的 ISP帐户时，ISP会告诉你，上述原则会成为中断对你提供服务的理由，并且可能让你对系统受到的损害负责。

对于你自己的行为，应该有所约束，对于你接受他们的恩惠，不要苛求。

Internet具有长期的记忆能力。真的发送过垃圾邮件的人将会被人们长久记住并遭到唾骂。一旦因为发送垃圾邮件而变得臭名昭著，要想挽回名誉是很难的。

22.2 邮件发送函数

下面各节将介绍如何编写一个 Perl短函数，供你在CGI程序中用来发送电子邮件。不过这里存在一个问题。该函数运行的方式主要取决于你是否拥有本地 MTA（如sendmail程序），或者是否来自将邮件发送到SMTP中继主机。因此请预先考虑好，确定需要将下面的哪一节中的

函数用于你的特定程序。

22.2.1 用于UNIX系统的邮件函数

如果你拥有UNIX系统，并且sendmail可能已经配置好了（也许尚未配置好），那么你阅读本节内容是对的。如果你没有 UNIX或sendmail，只是因为好奇而阅读本节内容，这也对你有好处，不过，程序清单22-1中展示的函数也许对你没有多大帮助。

即使你拥有UNIX系统，下一节“ 用于非 UNIX系统的邮件函数 ”也是值得一读的。下一节将介绍使用模块（即面向对象的模块）的新方法。

程序清单22-1 send_mail函数

```
1:  # Function for sending mail with an MTA like sendmail
2:  sub send_mail {
3:      my($to, $from, $subject, @body)=@_;
4:
5:      # Change this as necessary for your system
6:      my $sendmail="/usr/lib/sendmail -t -oi -odq";
7:
8:      open(MAIL, "|$sendmail") || die "Can't start sendmail: $!";
9:      print MAIL<<END_OF_HEADER;
10: From: $from
11: To: $to
12: Subject: $subject
13:
14: END_OF_HEADER
15:     foreach (@body) {
16:             print MAIL "$_\n";
17:     }
18:     close(MAIL);
19: }
```

第6行：sendmail 的位置和它需要的参数在这里被放到一个变量中。 sendmail程序可能位于你的系统上的不同位置，也可以带有不同的参数。

第8行：$sendmail中设定的sendmail程序启动并打开，以便对文件句柄 MAIL进行写入操作。

第9~14行：电子邮件的报头被写入MAIL。

第15~17行：邮件的正文被写入MAIL文件句柄。每行都附加了一个\n。

若要使用该函数，只要像下面这样用 4个参数调用它：

```
@body=("Lower mine, please.", "Thanks!");
send_mail('president@whitehouse.gov', 'owner@geeksalad.org',
    'Taxes', @body);
```

该函数的运行要求你在系统上正确安装和配置 sendmail。如果没有安装和配置，请阅读下一节“ 用于非UNIX系统的邮件函数 ”，那里介绍的解决方案也可以在 UNIX下使用。

必须将变量 $sendmail改为你的系统上的 sendmail程序的正确位置。它的位置通常是/usr/lib，不过它也可以是/usr/sbin,/lib，或者你的系统上的任何其他目录。你必须花一点时间才能找到它。

如果程序的运行没有按你的期望进行，请确保你的系统上的邮件程序配置正确。可以使用mail或pine之类的邮件实用程序来发送测试邮件。如果这些实用程序不能正确运行，那么说明sendmail的安装很可能不正确。你必须首先解决这个问题，或者使用下一节介绍的方法来运行这些实用程序。

在程序清单21-1中，sendmail程序是用下列选项启动的，你可以根据情况修改这些选项。

- -t 从输入数据而不是命令行中获得邮件的报头(From、To、Subject等)。
- -oi 忽略单行程序上的“.”(圆点)。如果不使用本选项，就会中断你的邮件。
- -odq 对邮件进行排队，而不是立即将它们发送出去。如果你愿意，可以不使用本选项。但是，如果有太多的邮件要立即发送，那么你的邮件系统将会应接不暇。使用 -odq是一种很礼貌的做法。

send_mail()函数的其余部分的功能是不言自明的。

22.2.2 用于非UNIX系统的邮件函数

在没有安装sendmail之类的内置MTA的Windows和其他操作系统下，你会遇到一些复杂的问题。MTA不是个简单的邮件传输工具，试图用几行 Perl代码就复制它的功能，是很不容易的事情。不过这是可能做到的。

首先，使用Perl模块Net::SMTP，你可以通过Perl运行的任何操作系统来发送邮件。使用该模块，你就能够非常容易地发送邮件而不会遇到太大的困难。

问题是在标准的Perl产品上并没有安装该模块。为了获得该模块，必须将它加载到 Web服务器所在的系统上，或者加载到你想发送邮件的任何位置上。 Net∶∶SMTP模块是libnet组件的组成部分，它包含各种非常有用的网络模块。 Libnet组件位于本书所附光盘上。

本书的附录“安装模块”提供了相当详细的如何安装 Perl模块的指南。它讲述了如何在UNIX、Windows和Macintosh操作系统下，安装各个Perl模块。此外，如果你的系统管理员没有安装模块的公用拷贝，你还会在附录中找到如何安装模块的专用拷贝的说明。

程序清单22-2显示了用于不带MTA的操作系统的send_mail函数。它包含某些非常奇特的新语句，你可能对它们不太熟悉。请务必要阅读后面的说明。

程序清单22-2 用于非MTA系统的send_mail函数

```
1: # Function for sending mail for systems without an MTA
2: sub send_mail {
3:       my($to, $from, $subject, @body)=@_;
4:
5:       use Net::SMTP;
6:
7:       # You will need to change the following line
8:       # to your mail relay host
9:       my $relay="relayhost.yourisp.com";
10:      my $smtp = Net::SMTP->new($relay);
11:      die "Could not open connection: $!" if (! defined $smtp);
```

```
12:
13:     $smtp->mail($from);
14:     $smtp->to($to);
15:
16:     $smtp->data();
17:     $smtp->datasend("To: $to\n");
18:     $smtp->datasend("From: $from\n");
19:     $smtp->datasend("Subject: $subject\n");
20:     $smtp->datasend("\n");
21:     foreach(@body) {
22:          $smtp->datasend("$_\n");
23:     }
24:     $smtp->dataend(); # Note the spelling: no "s"
25:     $smtp->quit;
26: }
```

第5行：引入Net::SMTP模块，使邮件的发送稍为容易一些。

第10行：Net::SMTP对象得以创建，并与正确的中继主机相连接，该主机是你在第 9行上设置的。

第13~23行：电子邮件的报头和正文被发送到中继主机。详细说明请参见后面的各个Net::SMTP函数。

若要使用该函数，只需使用代表电子邮件各个部分的 4个参数来调用它：

```
@body=("Lower mine, please.", "Thanks!");
send_mail('president@whitehouse.gov', 'owner@geeksalad.org',
    'Taxes', @body);
```

这个函数令你感到奇怪的第一件事情是 $smtp=Net::SMTP->new（$relay）；这行代码。这行代码用于创建一个称为“对象”的东西。“对象”实际上并不是一个标量，也不是哈希结构或者数组，它是个稍有不同的东西。 $smtp中的值现在代表一个到达邮件程序的连接，你可以对这个连接进行各种操作，请将它视为一个特殊种类的值，可以用它来调用与该值相关的函数。

你感到奇怪的下一件事情是 $smtp->mail($from)；这行代码。->用于将一个对象连接到一个对它进行调用的函数，因此， mail是个使用上一行创建的$smtp对象来调用的函数。

为了使用Net::SMTP模块，你并不需要理解对象语句的全部特征，只需顺便了解一下就够了。对于Net::SMTP对象，可以使用的函数包括下列几个：

- $smtp->mail(addr) mail函数用于指明你发送邮件时使用的是什么身份。当然，有时你可以就你的身份问题撒点儿谎。
- $smtp->to(addr) to函数用于指明你要将邮件发送给谁。如果你调用的 to函数带有一个名字列表，那么每人都会收到一个邮件拷贝。这些人的名字列表不一定出现在邮件正文中，除非你亲自将这些名字明确放入邮件正文中，比如发送 BCC。
- $smtp->data()； data函数用于指明你准备发送邮件正文。
- $smtp->datasend(data) 这个函数用于发送邮件的实际文本。你必须输出你自己的报头域（To:、From：等）。报头域，比如Date:和Received:，是自动生成的。在报头与正文之间，还必须输出一个空行——$smtp->datasend(“\n”)。你的邮件正文跟随在这个空行的后面，并且也用$smtp->datasend()来发送。
- $smtp->dataend() dataend函数用于指明你已完成邮件正文的发送，在运行这个函数之

前，邮件并未发送。

- $smtp->quit() 本函数用于断开与SMTP服务器的连接。

22.3 从Web页发送邮件

既然你有了一个邮件发送函数send_mail()，那么从Web页来发送邮件的其余工作就非常简单了。只要设计一个Web页，编写一个CGI程序与它配合运行。程序清单22-3显示了一个电子邮件示例的HTML窗体。该窗体并非完美无缺，你可以随意使用自己的设计风格来改进这个窗体。

程序清单22-3 用于发送电子邮件的HTML窗体

```
1.  <!--assumes a program called /cgi-bin/mailer.cgi exists-->
2.  <FORM METHOD=POST ACTION="/cgi-bin/mailer.cgi">
3.  Your address: <INPUT TYPE=text NAME=return_addr><BR>
4.  Subject: <INPUT TYPE=text NAME=subject><BR>
5.  <BR>
6.  Message:<BR>
7.  <TEXTAREA NAME=body ROWS=20 COLS=60 WRAP=hard>
8.  Type your message here
9.  </TEXTAREA>
10. <BR>
11. <INPUT TYPE=SUBMIT VALUE="Send Message">
12. </FORM>
```

用于发送邮件的CGI程序并不比它大多少。下面显示了这个CGI程序：

```
#!/usr/bin/perl -w
use strict;
use CGI qw(:all);
use CGI::Carp qw(fatalsToBrowser);
#
# Insert the send_mail function
# from Listing 22.1 or 22.2 here!
#

print header;
my $return=param("return_addr");
if (! defined $return or ! $return) {
        print "You must supply an e-mail address<P>";
        exit;
}
my $subject=param("subject");
if (! define $subject or ! $subject) {
        print "You must supply a subject<P>";
        exit;
}

# Change this address to wherever you want your
# mail sent
send_mail('webmaster@myhost.com',
          param($return),
          param($subject),
          param("body"));

print "Mail sent.";
```

在上面这个代码中的小程序中，有几个问题你应该注意。首先，必须将程序清单 22-1或22－2中的send_mail函数插入该程序，使该程序能够运行。哪个程序清单中的函数最好，并且适合于你，就使用该程序清单中的那个函数。

其次，注意To:地址是通过硬连线与程序相连接的，正如 Webmaster@myhost.com的情况那样。必须将这个地址改为你想要将邮件发送到的那个地址。该地址不是从用户那里获得的原因很简单，因为你不希望用户使用 Web窗体将邮件发往任意的地址。如果有人滥用你的窗体，将恶意邮件发送给某个人，那么你和你的系统将成为人们指责的目标。因此这不是个好主意。

如果你希望用一个窗体将邮件发送到多个目的地，请使用下拉列表（或者单选按钮），为你提供一个地址选择表：

```
<INPUT TYPE=radio NAME=target Value=1 CHECKED>Support Department
<INPUT TYPE=radio NAME=target Value=2>Sales Department
<INPUT TYPE=radio NAME=target Value=3>Legal Department
```

然后，在你的程序中，使用下面这样的代码段：

```
$formtarget=param('target');
%targets=( 1=> 'support@myhost.com',
           2=> 'sales@myhost.com',
           3=> 'legal@myhost.com');
if (exists($targets{$formtarget})) {
        $target=$targets{$formtarget};
} else {
        $target='webmaster@myhost.com';
}
print $target;
```

无论你如何进行操作，不要让实际的 To：地址从窗体传递过来并用在你的程序中。请传递一个没有问题的值（在上面的例子中是 1至3），并在你的CGI程序中对该值进行相应的转换，即使看起来不可能，也要允许传递不正确的值（上面的例子中的 else语句）。

核实电子邮件地址

也许你已经发现CGI程序并不试图确定用户输入的电子邮件地址是否有效。它这样做是很有理由的，因为它无法确定该地址是否有效。

这个原因一定会使你大吃一惊。

设计Internet上的电子邮件系统的要求之一是要能够了解目的地址是否有效。然而这是不可能的。

困难源于本学时开头介绍的程序清单 22-1和22-2。从发送邮件系统的角度来看，它无法看到邮件传输链的结尾环节。它必须将邮件全部传递给传输链上的第二个系统，第二个系统又将邮件传递给第三个系统，以此类推。这些“传递”过程的延迟时间是很重要的，更重要的是，发送邮件的系统在将邮件送出去后就无法控制邮件了。

标准的解决办法是设法清除掉显然无效的地址，无法确定是否有效的地址则属例外。电子邮件地址的 Internet标准（RFC-822）有一个标准电子邮件地址的模板。但是，有些符合RFC-822标准的有效地址实际上是无效的，而有些不符合 RFC-822标准的地址却是有效的、可以传递邮件的地址。

编写对电子邮件地址进行匹配的正则表达式是不行的。例如，表达式 /^[\w.-]+\@([\w,-]\.)+\w+$/看上去是可行的，它甚至与 me@somewhere.com这个地址相匹配。但是，它拒绝下面这个完全有效的电子邮件地址：

```
*@qz.az
clintp!sol2!westwood@dec.net
relay%me@host.com
"barney&fred"@flintstones.net
```

与符合RFC-822标准的电子邮件地址相匹配的一个正则表达式长达 4700个字符，因为太长，所以本书没有将它列出，你也很难键入。同时它也无法与 Internet上的每个传输邮件的地址相匹配。

那么究竟怎么办呢？

若要确定电子邮件地址是否有效，唯一的办法是将一个邮件发送到该地址，然后等待对方的答复。如果由于某个原因，你希望确保对方地址上有人（比如将来将邮件发送给他，因为他要求发送），请发送一个电子邮件，要求他回答。当对方的答复返回时，就知道你发送了一份有效的电子邮件。

22.4 课时小结

在本学时中，我们介绍了如何从Web页发送电子邮件。同时，介绍了 send_mail()函数的两个版本，它们可以用在任何 Perl程序中来发送电子邮件。我们还讲述了 Internet电子邮件的基础知识以及基本的电子邮件礼仪。

22.5 课外作业

22.5.1 专家答疑

问题：能不能使用从浏览器中搜集到的信息来获取 Web冲浪者的电子邮件地址？

解答：虽然能够这样做看起来是很好的（它可以消除获取电子邮件地址时的错误），但这是不可能的。浏览器并不包含用户的电子邮件地址。 CGI模块中的remote_host函数返回的值实际上并不是用户接收电子邮时使用的地址。如果你使用安全的 Web事务处理，那么remote_user函数也许不是用户的电子邮件地址中的“名字”部分。同时请记住，浏览器可能提供某些不准确的此类信息， Netscape和Internet Explorer的某些插件也会这样说谎。

另外，用户可能使用图书馆、朋友家、办公室或网吧中的 Web浏览器，因此浏览器的地址甚至与用户的电子邮件地址并无关系。

问题：我能核实电子邮件地址吗？

解答：你可以试试。例如，大多数最新的电子邮件地址包含 @(at符号)，你可以用它进行测试。但是，本地计算机（例如 postmaster、root）上的计算机不需要@。

问题：我试着运行CGI电子邮件程序，但在消息中出现“From nobody……(来自无人……)”这行文字，为什么？

解答：是这样的：sendmail程序记录了电子邮件发送者的用户 ID。实际上，电子邮件的发送“人”是Web服务器本身。 Web服务器常常以一个特殊用户 ID——noboby、Web、httpd或root来运行，该地址记录在电子邮件报头中。不必担心，只要你输出一个正确的 From:行，作

为邮件报头的一部分，当用户答复该邮件时，那么这就是你看到的一行信息。

问题：我应该如何将文件附加给电子邮件消息？

解答：你应该查看CPAN中的MIME模块。

22.5.2 思考题

1) $foo=Net::SMTP->new(‘ mailhost ’)这个模块有何功能？

（如果你没有阅读“ 用于非UNIX系统的邮件函数 ”这一节，请现在阅读。）

a.它会产生一个句法错误。

b.它创建一个对象，称为$foo，代表与SMTP邮件服务器的连接。

c.它将Net::SMTP模块纳入当前程序之中。

2) 下面几个电子邮件地址中哪一个可能是无效地址？

a.foo!bar!baz!quux

b.“ ”@bar.com

c.stuff%junk!“ Wowzers ”!foo.com!blat

22.5.3 解答

1) 答案是b。如果你回答是a，那么可能出现了键入错误，也可能运行了 Perl 4。选择c是不正确的，因为它实际上描述的语句是 use Net::SMTP。

2) 这是个巧妙的问题。这几个地址都可能是有效的电子邮件地址。

22.5.4 实习

- 对简单的CGI电子邮件程序进行下列简单的修改：
- 搜集用户的浏览器信息，将它附加给邮件正文。
- 给用户发送一个礼仪邮件拷贝（如果你在真实的 Web站点上发送这样的邮件，请务必告诉他这一情况）。这样做时你也要小心，因为有人不喜欢这样的邮件。
- 让用户在发送邮件之前能够“ 预览 ”邮件。必须使用第 19学时中介绍的方法之一，使第一页（电子邮件输入屏）中的数据可供第二页（电子邮件核实屏幕）使用，最后供邮件发送程序使用。