6.7.3 建立虚拟用户表数据库

虚拟用户表数据库将虚拟地址映射到真实地址。建立一个文本文件，其中每行都有一个关键字/值对，它们之间用制表键分开。例如：

超星阅览器提醒您：
您用本复制品
请尊重相关知识产权！

```
info@milkyweb.com                jgunchy
webmaster@milkyweb.com           kabir
@milkyweb.com                    jenny.gunchy@aol.com
```

在这个例子中，地址 info@milkyweb.com 将映射到本地用户 jgunchy，webmaster @milkyweb.com 将映射到本地用户 kabir，而发送到 milkyweb.com 的任何其他邮件将转到远程用户 jenny.gunchy@aol.com。如果 milkyweb.com 在其他域（如，hotmail.com）上有真正用户，那么可以用下列内容替代最后这一行：

```
@milkyweb.com                    %1@hotmail.com
```

这一行将所有 milkyweb.com 用户映射到 hostmail.com 上有相同名称的真实用户。这里使用的概念是将虚拟用户映射到真实的帐号，这些帐号或者在本地系统上，或者在远程系统上。一旦建立了这个文件，通过使用与标准 sendmail 发行版一起提供的 makemap 实用程序，就可以用来建立数据库文件：

```
makemap <database type> /etc/virtusertable < <text file
containing the virtual user map>
```

如果前面的虚拟用户表文本文件被称为/etc/virtusertable.txt，并且你在用 dbm 数据库类型，那么可以按如下方式使用 makemap 命令：

```
makemap  dbm  /etc/virtusertable < /etc/virtusertable.txt
```

实际上它建立了一个或多个非文本文件（一般情况下是/etc/virtusertable.dir 和/etc/virtusertable.pag 或/etc/virtusertable.db），但不改变/etc/virtusertable.txt 文件本身。注意在不可使用 dbm 的系统上，可以试试如下 hash 选项：

```
makemap  hash  /etc/virtusertable < /etc/virtusertable.txt
```

Tip

如果希望将向外发送的邮件逆向映射到本地用户，应该参考在 www.sendmail.org/m4/masquerading.html 上有关 sendmail 的 Masquerading 和 Relaying 页的内容。

6.7.4 配置/etc/sendmail.cf

在 /etc/sendmail.cf 文件中，找到下面这些行：

```
# Virtual user table (maps incoming users)
# Kvirtuser  dbm  /etc/virtusertable
```

通过删除第二行开头的#符号，使它不成为注释行；如果用 hash 方法代替 dbm，就用 hash -o 代替 dbm。对于 hash 方法，它将是：

```
Kvirtuser  hash  -o  /etc/virtusertable
```

而对于 dbm 方法，它将是：

```
Kvirtuser  dbm  /etc/virtusertable
```

现在，找到规则集 S98，并去掉这些规则的注释。所以它们看起来和下面的相似：

```
S98
#    addresses sent to foo@host.REDIRECT will give a 551 error code
R$*  < @ $+ .REDIRECT. >          $: $1 < @ $2  . REDIRECT. ><${opMode}>
R$*  < @ $+ .REDIRECT. ><i>       $: $1 < @ $2  . REDIRECT. >
R$*  < @ $+ .REDIRECT. ><$->      $# error $@ 5.1.1 $:"551 User has
moved; Please  try  "$1@$2
```

注意，在较新的 sendmail 发行版本中这些规则的编写方式有些不同，因此应该在试图对它们进行修改之前，首先取消它们的注释。

6.7.5　测试虚拟 e-mail 服务

一旦完成了前面的步骤，就能重启动 sendmail 服务器并测试所做的修改：

Note

当改变虚拟用户（或逆向映射的类属表）时，不需要重启 sendmail。只有改变/etc/sendmail.cf 或改变类似 /etc/sendmail.cw 的类文件时，才需要重启 sendmail。

sendmail 为测试配置文件中的规则提供了一个方法。使用如下命令：

```
/usr/sbin/sendmail  -bt
```

你可以进入 sendmail 的交互的规则集测试模式。在图 6-13 中，将为 e-mail 地址 webmaster@milkyweb.com 完成一次测试。注意 sendmail 进行所有必要的规则转换并最终将邮件传递到在本地系统上名为 kabir 的帐号。注意，要测试一个地址，在交互测试模式中所要键入的是：

```
0 user@host
```

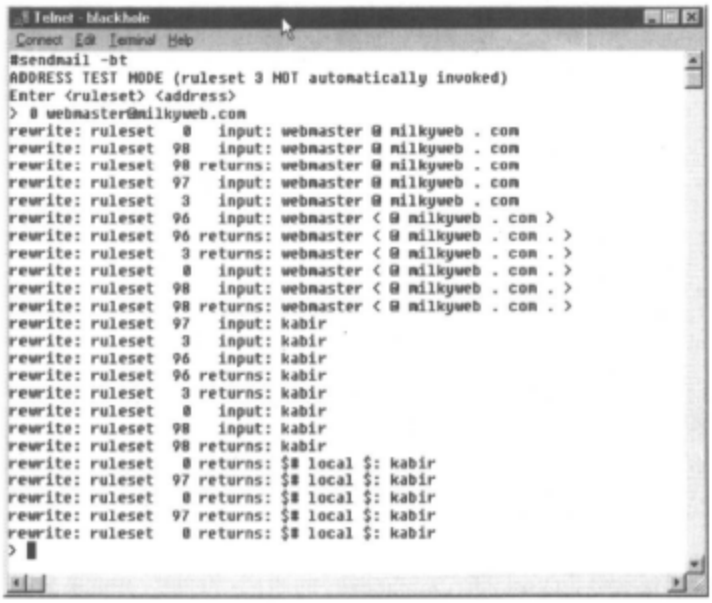

图 6-13　为虚拟域测试 sendmail 的规则集

如果在测试过程发现问题，那么确保检查了所做的改动。还应该在邮件服务器上为 sendmail 检查日志文件，看是否出现错误。如果一切正常，那么应该试试从另外某个域发送邮件并检查是否收到这一邮件。注意，在设置邮件服务时，不管对 DNS 做什么改动，在 Internet 上都需要花一定时间将你的站点的 DNS 修改传播到其他名称服务器上 。

6.8 小　　结

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

虚拟 Web 站点无处不在。今天，大多数 ISP 利用建立虚拟 Web 服务把个人和商业 Web 站点放在 Internet 上。在本章中，讨论了如何用 Apache 建立虚拟 Web 站点的所有细节。还讨论了如何为虚拟主机将 sendmail 转换成邮件服务器的必备知识。同时提供虚拟 Web 站点和邮件服务能大大提高任何 Apache 管理员的市场适应能力（marketability），不用说他或她会有多满意。

在下一章中，将讨论服务器端包含命令（SSI）。

第 7 章　Apache 的服务器端包含命令

您的浏览器提醒您：
请尊重相关知识产权！

本章内容：

- 什么是 SSI
- 如何为 SSI 配置 Apache
- 怎样使用 SSI 命令

在第 8 章中，将讨论可以使用 CGI 程序来创建动态 Web 内容。然而，有一些任务虽然不需要开发完整的 CGI，但仍然需要进行动态插入。例如，假如想在 HTML 页上添加标准的版权信息；如何做到这一点呢？有两种方案：在每个 HTML 页上添加版权信息的内容；或者写一段 CGI 程序，它能将此信息添加到每个 HTML 页上。这两种选择都不是最好的。每当更新版权信息时，第一种选择需要手工修改所有的文件。第二种选择需要在每一页发送给 Web 浏览器之前，有办法使 CGI 程序运行起来，这也就表示每页上的链接需要调用 CGI 程序，使它能将信息附加到下一页。像这种情况需要有简单的解决方案：SSI（服务器端包含命令）。它正是本章讨论的主题。

7.1　什么是 SSI

一般说来，SSI 页是嵌入了用于 Web 服务器命令的 HTML 页。在通常情况下，Web 服务器在向 Web 浏览器（或任何其他 Web 客户）传递之前不分析 HTML 页。然而，允许使用 SSI 命令的 HTML 页在传递前由 Web 服务器进行分析。并且如果在页中找到任何特殊的 SSI 命令，它就被执行。图 7-1 显示普通的 HTML 页和允许使用 SSI 的 HTML 页从 Web 服务器的传递过程。

正如从图中所看到的，首先对 SSI 版的 HTML 页分析和执行 SSI 命令，并将输出结果传递给 Web 浏览器（即 Web 客户）。

Apache 支持 SSI 已有很长一段时间。用来支持 SSI 的模块被称作 mod_include，并且它是标准 Apache 发行版的一部分。不久，称为扩展的服务器端包含命令（eXtended Server Side Includes，XSSI）的 Apache 可选模块提供一组增强的功能。由 Howard Fear 所写的 XSSI 软件包现在已经被包含在标准的 Apache 发行版中，而且这个标准模块现在为所有 SSI 和 XSSI 命令提供支持。

Note　如果你仍在使用非常旧的 Apache 版本（即 1.2x 以前的版本），那么需要安装 XSSI 模块，它可以在 www.pageplus.com/~hsf/xssi/ 中找到。

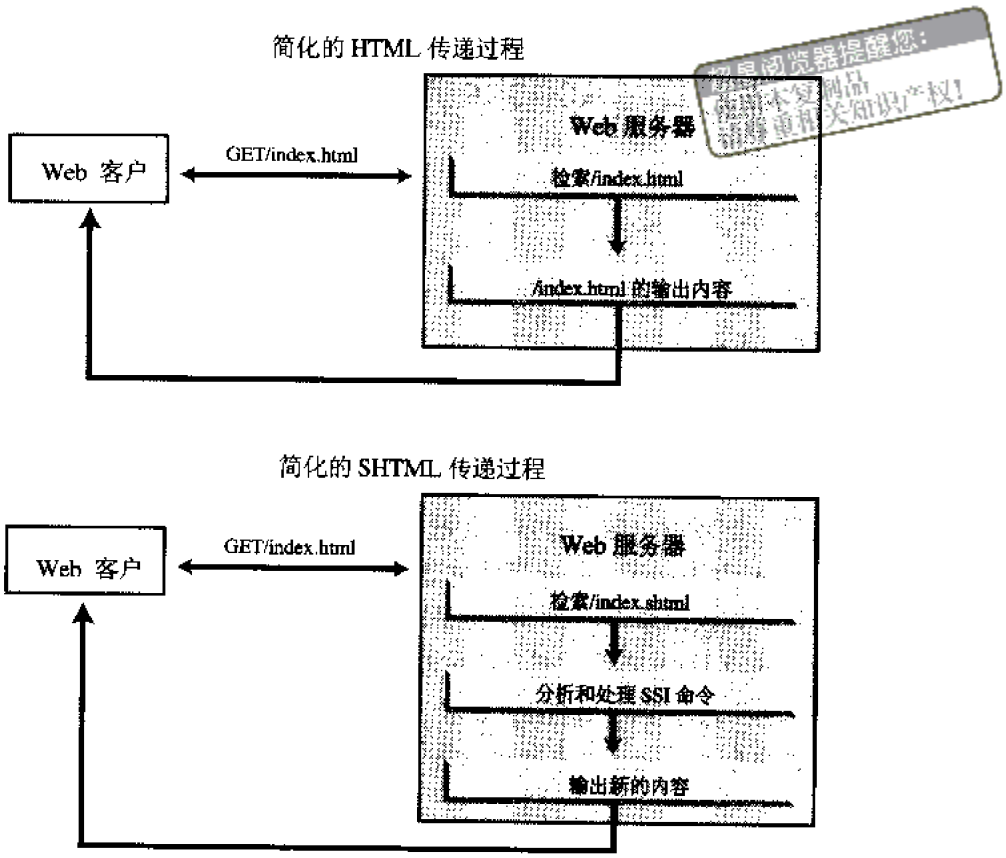

图 7-1 HTML 页和允许使用 SSI 的 HTML 页的简化传递过程图

Fear 的 XSSI 代码将更多编程语言的功能添加到 Apache 服务器的标准 SSI 命令集中，如变量和流程控制等。然而，在 HTML 页上使用 SSI 之前，需要首先配置 Apache 服务器。

7.2 为 SSI 配置 Apache

在为 SSI 配置 Apache 之前，必须检查当前 Apache 的可执行程序（httpd）以确保它包含了 mod_include 模块。要做到这点，运行下面这条命令：

```
httpd -l
```

这使你能看到在建立 Apache 可执行程序时所使用的所有模块的列表。在缺省情况下，应该编译此模块。如果不是这样，需要使用下列命令行将它添加到 Configuration 文件中：

```
AddModule modules/standard/mod_include.o
```

然后，需要分别运行 Configure 和 make 实用程序。

尽管在缺省情况下在标准的 Apache 发行版中编译了 mod_include 模块。但是在缺省情况下不允许进行 HTML 页的分析。要使 SSI 支持 Apache（版本 1.2 或更高），需要完成下列步骤：

1. 为 SSI/IXSSI HTML 页添加新的处理程序。

2. 为 SSI/IXSSI HTML 页添加新的文件扩展名。

3. 允许 SSI 对目录进行分析。

超星阅览器提醒您：

在名为 apache.nitec.com 的虚拟主机上对名为 chapter_07 的目录完成上述步骤后，它的配置如下所示：

```
<VirtualHost 206.171.50.50>

ServerName apache.nitec.com
DocumentRoot /data/web/apache/public/htdocs
ScriptAlias /data/web/apache/public/cgi-bin
<Directory /data/web/apache/public/htdocs/chapter_07>
AddHandler  server-parsed  .shtml
AddType  text/html  .shtml
Options  +Include
</Directory>

</VirtualHost>
```

Caution　在 Options 命令中，通过使用 Include 激活所有的 SSI 命令。如果打算禁止通过 SSI 命令执行外部程序，可以用 IncludesNOEXEC 代替上面的选项，这将禁止外部程序的执行。然而，它还禁止通过 SSI 的 Include 命令加载外部文件。

现在让我们进一步了解有关实现对 SSI 支持的各个步骤。

7.2.1　为 SSI HTML 页添加新的处理程序

在缺省情况下，当请求 HTML 文件时，Apache 并不分析它们。如果它这样做，没有包含 SSI 命令的 HTML 页也需要进行分析，这些不必要的分析将使页的传输变得缓慢。对这一问题的解决办法是使用不同的文件扩展名。不用典型的 HTML 文件扩展名（.html 或.htm），而用新的的文件扩展名来区分那些确实含有 SSI 命令的页。假设你想用 .shtml 作为所有包含一条或多条 SSI 命令的 HTML 页的文件扩展名。需要告诉 Apache，该文件扩展名 .shtml 应该作为允许使用 SSI 的页来对待。可以按如下方式使用 AddHandler 命令来做到这一点：

```
AddHandler server-parsed  .shtml
```

AddHandler 命令通知 Apache，带后缀.shtml 的文件需要由 server-parsed 处理程序来处理，该处理程序可以在 mod_include 模块中找到。

如果由于某种原因，不得不使用 .html 和 .htm 扩展名作为 SSI 扩展名，那么别使用：

```
AddHandler server-parsed  .html
AddType text/html  .html
AddHandler server-parsed  .htm
AddType text/html  .htm
```

这样做会降低服务器的性能。Apache 会处理所有的 .html 和.htm 文件，它表示不包含 SSI 命令的文件也将被分析，因此增加了文件传输中的延迟。应该尽量避免使用.html

或.htm 作为 SSI 的扩展名；然而，如果必须使用它们，那么应该使用 XBitHack 命令。它可以在 mod_include 模块中找到。

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

XBitHack 命令：

XBitHack 命令控制服务器分析与 MIME 类型 text/html 有关的文件。

```
Syntax: XBitHack  on | off | full
Default: XBitHack  off
Context: server config, virtual host, directory, .per-directory
access control file (.htaccess)
Override: Options
Module: mod_include
```

一般讲，只有.html 和.htm 文件与 text/html 有关。缺省值“off”通知服务器不要分析这些文件。当被设成“on”时，任何对该文件拥有者具有可执行权限的 HTML 文件都被认为是 SSI 文件并被分析。当该命令被设置为“full”时，它使服务器检查文件的拥有者和组的可执行位的权限设置。如果设置了组可执行位，那么 Apache 返回文件的最新更改日期，将它设为文件的最新修改时间。如果没有设置，那么就没有最新更改日期要发送。设置这个位使客户和代理服务器能缓存请求结果。对于那些在分析和处理时会产生出不同输出结果的 SSI 页，建议不要使用值“full”。

Note

当使用这个命令时，仍须完成第三步（Options+Includes）。

7.2.2 为 SSI HTML 页添加新的文件扩展名

尽管 Apache 现在知道怎样处理 .shtml 文件，但是它仍然需要知道将哪些有关文件的信息通知 Web 浏览器。Web 服务器为每个请求发送头标消息来告诉 Web 浏览器哪种类型的内容要作为响应发送。因此，你需要告诉 Apache，当对 .shtml 文件的请求作出响应时，它应该通过设置内容的类型告诉浏览器被发送的消息仍是 HTML 文档。这样，Web 浏览器和通常情况一样将取回的内容显示在屏屏幕上。HTML 内容的 MIME 类型是 text/html。下面这行命令表示怎样告诉 Apache，当传送.shtml 页的输出结果时，产生 text/html 内容类型的头标：

```
AddType  text/html  .shtml
```

Note

出于向后兼容的考虑，具有 MIME 类型 text/x-server-parsed-html 或 text/x-server-parsed-html3 的文档仍将被分析（并且输出结果是按 MIME 类型 text/html 给出）。

7.2.3 使 SSI 为目录进行语义分析

现在，Apache 和 Web 浏览器都知道怎样处理新的 .shtml 文件。然而，Apache 仍然还没有为分析 .shtml 页作好准备。

你需要使用 Options 命令告诉 Apache 你想使用 Includes 支持。首先，需要决定将 Options 命令放在哪儿。

如果想在整个（主）Web 站点范围内使用 SSI 支持，在一个全局配置文件（如 access.conf）中添加以下命令：

```
Options +Includes
```

温馨提醒您：使用本复制品请尊重相关知识产权！

如果想在虚拟 Web 站点范围内实现 SSI 支持，需要将前面的命令放在相应的 <VirualHost...>容器中。或者，如果想从目录到目录控制这个选项，可以将这个命令放在 <Directory..>容器中，或者放在针对目录（per-directory）的访问控制文件（.htaccess）中。

如果使用针对目录的访问控制文件（.htaccess）来实现 SSI 支持，要确保拥有该目录的站点的 AllowOverride 命令允许此类操作。用于此站点的 AllowOverride 命令允许 Includes 选项可以被覆盖掉。例如，如果某个站点的 AllowOverride 命令被设置成 None，那么就不进行任何 SSI 分析。

Note　在前面例子中，如果在 Options 命令行中不使用 + 符号，那么除了 Includes 以外其他所有选项都不能使用。

了解在 Apache 中如何实现 SSI 支持后，让我们详细讨论 SSI 命令。

7.3　使用 SSI 命令

SSI 命令以注释形式嵌入在 HTML 页中。基本的命令结构就像下面这样：

```
<!-#command argument1=value argument2=value argument3=value ...>
```

参数的值通常被包括在双引号中。许多命令只允许使用一对 attribut-value。注意注释的结束符（→），它应该放在一个空格的后面以确保不被认为是 SSI 命令的一部分。

现在，让我们讨论所有可用的 SSI 命令。

7.3.1　config

config 命令能配置分析错误消息的表示方式，以及配置用于显示时间与文件大小信息的格式。这是由以下几行代码完成的：

```
♦config errmsg="error message"
♦config sizefmt=["bytes" | "abbrev"]
♦config timefmt=format string
```

第一条语法表示怎样创建定制的错误消息：

```
config errmsg="error message"
```

当发生分析错误时，错误消息“error message”就会显示出来。例如，列表 7-1 列出了称为 config_errmsg.shtml 的文件。

列表 7-1：　config_errmsg.shtml

```
<HTML>
```

```
<HEAD><TITLE> Apache Server Bible - Chapter 7 </TITLE></HEAD>
<BODY BGCOLOR="white">

<FONT SIZE=+1 FACE="Arial"> Simple SSI Example #1</FONT>
<HR SIZE=1>

<P> Example of the SSI <STRONG>config errmsg</STRONG> command:
</P>
<P> Embedded commands: <BR><BR>

<CODE>
     &lt;!-#config errmsg="This is a custom SSI error
message."-&gt; <BR>
     &lt;!-#config errmsg_typo="This is a custom SSI error
message."-&gt;
</CODE>

</P>

<P> Result: <BR>

<!-#config errmsg="This is a custom SSI error message." -->
<BR>
<!-#config errmsg_typo="This is a custom error message." -->

</P>

</BODY>
</HTML>
```

超星阅览器提醒您
使用本复制品
请尊重相关知识产权!

在这个例子文件中，有两个 SSI 命令。它们是：

```
<!-#config errmsg="This is a custom SSI error message."  -->
```

和

```
<!-#config errmsg_typo="This is a custom error message."  -->
```

第一条是有效的 config errmsg 命令，它将错误消息设置成“This is a custom SSI error message.”（这是定制的 SSI 错误消息）。第二条是无效的 SSI 命令，有意将它输入到此文件中，用以显示所发生事情。图 7-2 显示，当向称为 apache.nitec.com 的 Apache 服务器请求此页时，返回该浏览器的内容。

正如所看到的，第二条命令产生语义错误，并且将错误消息作为结果显示出来。在找到这条命令的地方出现这则消息。可以插入 HTML 标记，或甚至在这则错误消息的字符串中插入客户端脚本。例如，下面显示了带有错误消息的弹出 JavaScript 警告窗口：

```
<!- config errmsg="<SCRIPT LANGUAGE=JavaScript> alert('An
error occurred. \n  Please report to webmaster@nitec.com');
```

图 7-2　config errmsg 命令的例子

```
</SCRIPT>"  -->
```

第二条语法可选择文件容量的输出格式：

```
config sizefmt= ['bytes" | 'abbrev']
```

可接受的格式限定符是“bytes”和“abbrev”。例如：

```
<!- config sizefmt="bytes"  -->
```

这将以字节显示文件的容量。要以千字节或兆字节显示文件，使用：

```
<!- config sizefmt="abbrev"  -->
```

最后这条语法可选择时间的显示格式：

```
config timefmt=format string
```

一般使用的格式串值可以包含以下标识符：

%a　根据当前当地的缩写周日名称

%A　根据当前当地的完整周日名称

%b　根据当前当地的缩写月份名称

%B　根据当前当地的完整月份名称

%c　当地所喜欢的当前日期和时间表示

%d　该月份的十进制数表示的日（从 01 到 31）

%H　使用 24 小时时钟的十进制数表示的小时（从 00 到 23）

%I　使用 12 小时时钟的十进制数表示的小时（从 01 到 12）

%j　该年的十进制数表示的日（从 001 到 366）

%m　十进制数表示的月（从 01 到 12）

%M　十进制数表示的分

%p　根据给出的当地时间值或者 a.m.，或者 p.m.

%S　十进制数表示的秒

%w　十进制数表示的该周每一天，星期天是 0

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

%x　当地所喜欢的当前不带时间的日期表示法
%X　当地所喜欢的当前不带日期的时间表示法
%y　不带世纪的十进制表示的年（从 00 到 99）
%Y　包含世纪的十进制表示的年
%Z　时间区名称或缩写
%%　%字符的表示

考虑下面这个例子：

```
<!-#config timefmt="%c"  -->
```

这将显示时间为：Tue May 20 00:54:07 1998（假设为当前时间）。

考虑下面这个例子：

```
<!-#config timefmt="%m/%d/%Y" -->
```

这将显示时间为：05/20/1998（假设为当前日期）。

7.3.2 echo

echo 命令打印一个 Include 变量（稍后定义）或任何 CGI 环境变量。语法是：

```
echo var="variable name"
```

如果该变量的值不可用，它就打印（none）作为值。任何打印的日期都依据当前配置的 timefmt。例如：

```
<!-#config timefmt="%m/%d/%Y" -->
<!-#echo var="DATE_LOCAL" -->
```

按照指定的 timefmt 字符串，打印出如 05/20/1998 一样的日期。

7.3.3 exec

exec 命令允许执行外部程序。这个外部程序可以是 CGI 程序或任何其他可执行程序，如：shell 脚本或二进制文件。对 CGI 程序的语法是：

```
exec cgi="path/to/cgi/program"
```

对其他程序的语法是：

```
exec cmd="path/to/other/programs"icon-note
```

Note　如果将 Options 命令设置为 IncludesNOEXEC 值，就禁止使用 exec 命令。

让我们讨论如何使用每个选项。

7.3.3.1 cgi

cgi 值指定到 CGI 脚本的（%-encoded）URL 相对路径。如果这个路径不以斜杠（/）开始，那么它就相对于当前文档的路径。由此路径引用的文档被作为 CGI 脚本调用，即使服务器通常不这样识别它。然而，必须允许在含有 CGI 脚本的目录中执行 CGI 脚本（使用 ScriptAlias 或 ExecCGI 选项）。

给 CGI 脚本提供来自客户原始请求的 PATH_INFO 和查询字符串（QUERY_ STRING），这些不能在 URL 路径中被指定。Include 变量和标准的 CGI 环境均可用于该脚本。

如果这个脚本返回 Location:头标作为输出结果，它就被翻译成 HTML 锚。例如，下面的代码表示简单的 Perl CGI 脚本，它打印 Location:头标作为输出。

```
#!/usr/local/bin/perl
print "Location: http://apache.nitec.com/\\n\\n\";
exit 0;
```

例如，当 Web 浏览器请求在列表 7-2 中所示的 exec_cgi.shtml 文件时，服务器将该 Location:头标翻译成 HTML 锚，而不是将该浏览器重定向到 http://apache.nitec.com 站点。

列表 7-2：　exec_cgi.shtml

```
<HTML>
<HEAD><TITLE> Apache Server Bible - Chapter 7 </TITLE></HEAD>

<BODY BGCOLOR="white">
<FONT SIZE=+1 FACE="Arial"> Simple SSI Example #2</FONT>
<HR SIZE=1>

<P> Example of the SSI <STRONG>exec cgi</STRONG> command: </P>
<P> Embedded commands: <BR><BR>

<CODE> &lt;!-#exec cgi="/cgi-bin/loc.cgi" -&gt;<BR></CODE>
</P>
<P> Result: <BR>  <!-#exec cgi="/cgi-bin/loc.cgi" --></P>
</BODY>
</HTML>
```

正如从列表中所看到的，在该文件中唯一的 SSI 调用是：

```
<!-#exec cgi="/cgi-bin/loc.cgi"  -->
```

其输出结果是个 HTML 锚，如图 7-3 中所示。

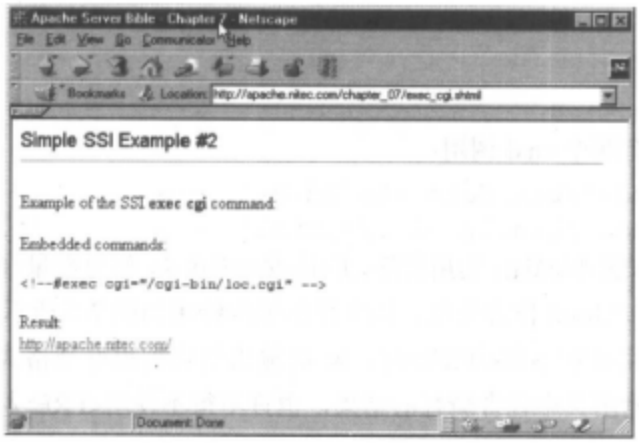

图 7-3　exec_cgi.shtml 的输出

7.3.3.2 cmd

当调用一般程序，而非 CGI 程序时，可以使用 exec 调用的 cmd 值。服务器利用大多数 UNIX 系统使用的 sh shell（/bin/sh）执行给出的字符串。Include 变量可用于这个命令。例如，列表 7-3 列出了称为 exec_cmd.shtml 的文件。

超星阅览器提醒您：
此为不复制品
请尊重相关知识产权！

列表 7-3：exec_cmd.shtml

```
<HTML>
<HEAD><TITLE> Apache Server Bible - Chapter 7 </TITLE></HEAD>
<BODY BGCOLOR="white">

<FONT SIZE=+1 FACE="Arial"> Simple SSI Example #3</FONT>
<HR SIZE=1>

<P> Example of the SSI <STRONG>exec cmd</STRONG> command: </P>
<P> Embedded commands: <BR><BR>

<CODE>
&lt;!-#exec cmd="/bin/date +%m/%d/%y" -&gt; <BR>
&lt;!-#exec cmd="/bin/ls -1 ./" -&gt; <BR>
</CODE>

</P>
<P> Result: <BR>

<!-#exec cmd="/bin/date +%m/%d/%y" --><BR>

<PRE>
<!-#exec cmd="/bin/ls -1 ./*.html" --><BR>

</PRE>

</P>
</BODY>
</HTML>
```

在这个文件中有两个 cmd 调用：

```
<!-#exec cmd="/bin/date +%m/%d/%y"  -->
<!-#exec cmd="/bin/ls -1 ./*.html"  -->
```

第一个调用 UNIX /bin/date 实用程序，并用+%m/%d/%y 作为变量。而第二个调用 UNIX ls 实用程序，并用 ./*.html 作为变量。该文件的输出结果如图 7-4 所示。

注意，通过使用<PRE>和</PRE>对，ls 的输出结果被很好地格式化。如在前面图中所显示的，如果想输出其中包含新行的结果，也许不得不使用<PRE>标记来保持输出结果可读。

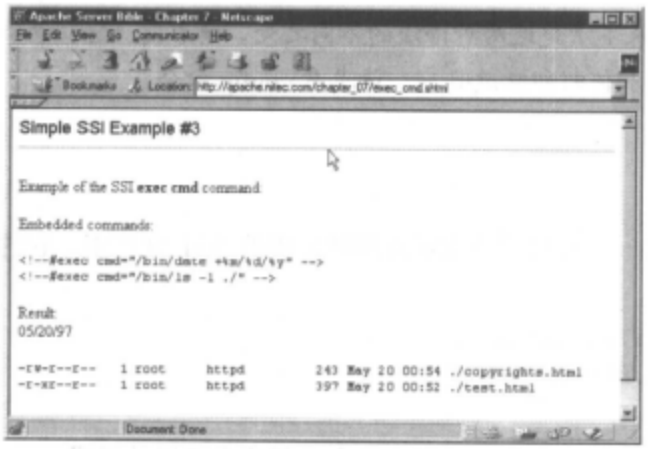

图 7-4　exec_cmd.shtml 的输出

7.3.4　fsize

这个命令打印指定文件的容量。所使用的语法取决于到目录的路径是相对的还是虚拟的：

```
♦fsize file="path"
♦fsize virtual="URL"
```

当使用第一条语法时，假定路径相对于含有当前被分析的 SSI 文档的目录。在此路径中不能使用../，也不能使用绝对路径。不能以这种方式访问 CGI 脚本。然而，能访问其他被分析的文档。例如：

```
<!-#fsize file="download.zip">
```

如果使用第二条语法时，假定虚拟路径是（%-encoded）URL 路径。如果路径不以斜杠（/）开始，那它就被当作相对于当前文档的路径。必须用这种方式访问普通文件，但不能以这种方式访问 CGI 脚本。然而，再强调一下，可以访问其他被分析的文档。例如：

```
<!-#fsize virtual="/download/free_software.zip">
```

Note　输出格式取决于指定的 sizefmt 格式。详细内容请参看 config 命令。

7.3.5　flastmod

flastmod 命令打印指定文件的最新修改日期。再次提醒，根据到目录的路径，有两种语法选项：

```
♦flastmod file="path"
♦flastmod virtual="URL"
```

该输出取决于指定的 timefmt 格式。例如：

```
<!-#flastmod file="free_software.zip">
<!-#flastmod virtual="/download/free_software.zip">
```

Note

如果不清楚这两种语法的不同之处，请参看 fsize 命令作为例子。要想控制如何打印修改日期，请参看 config 命令。

7.3.6 include

include 命令将一个文档的正文插入到被分析的 SSI 文档中。它的语法取决于到目录的路径：

```
♦include file="path"
♦include virtual="URL"
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

Note

参看 fsize 命令，了解文件和虚拟方式之间的不同。

任何被包含的文件都受通常的访问控制的支配。如果包含被分析文件的目录设置为 Option IncludesNOEXEC，那么它就不会包含引起一段程序执行的文档。这样就阻止了 CGI 脚本的执行。否则，像通常一样，CGI 脚本会被调用。它们使用在此命令中所给出的完整 URL、包含任何请求字符串。例如：

```
<!-#include file="copyrights.html"  -->
```

这将使当前文档中包含 copyrights.html 文件。这一命令对在文件中添加可重复使用的 HTML 代码非常有用。许多站点在每页上使用标准的菜单条。如果将这个菜单条放在称为 menu.html 的 HTML 文件中，通过使用类似上面的包含文件调用，它就能被所有 SSI 页调用。将来，当需要对此菜单作修改时，站点管理员只需要修改 menu.html 页就可以了。如果在此站点上存有很多文件，这样做会节省大量工作。

递归包含会被检测到，而且在第一次遇到后会出现一条错误消息。例如，如果 a.shtml 有一个 SSI 调用，如：

```
<!-#include file="b.shtml"  -->
```

并且 b.shtml 也有这样的调用：

```
<!-#include file="a.shtml"  -->
```

然后 Apache 记录并显示错误消息，说明检测到递归包含。

7.3.7 printenv

printenv 命令打印所有现有的变量和它们值的列表。

语法是：

```
printenv
```

例如：

```
<!-#printenv  -->
```

它就打印所有可用的 Include 和 CGI 环境变量。注意，为了使输出结果更为可读，建议使用<PRE>标记对。

7.3.8 set

set 命令设定一个由用户定义的变量值。此语法是：

```
set  var="variable name"  value="value of the variable"
```

例如：

```
<!-#set  var="home"  value="index.shtml"  -->
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

7.4 SSI 变 量

SSI 模块提供了一组变量，与 CGI 环境变量（见第 8 章）一起，所有 SSI 文件都可使用这些变量。这组变量被称作包含（include）变量。这些变量能由 SSI 命令（echo、if、elif 等）使用，以及由 SSL 命令调用的任何程序使用。这些包含变量是：

DATE_GMT	以格林尼治时间表示的当前时间
DATE_LOCAL	以当地时区表示的当前时间
DOCUMENT_NAME	当前的 SSI 文件名
DOCUMENT_URI	此文档的（%-decoded）URL 路径
LAST_MODIFIED	当前文件的最新修改日期。该日期取决于 config 命令的 timefmt 格式。

包含变量及 CGI 变量被预先设置并可供使用。任何预先设置的变量都能用作其他命令的变元（argument）。使用已定义的变量的语法如下：

```
<!-#command argument1="$variable" argument2="$variable2"...  -->
```

正如所看到的，变量名前面加了$符号。下面是另外一个例子：

```
<!-#config errmsg="An error occurred in $DOCUMENT_NAME page."  -->
```

Note　在 var="variable"字段中使用变量，不需要使用$符号。例如：

```
<!-#echo var="DOCUMENT_NAME"  -->
```

如果需要将美元符号插入到变量的值中，可以通过使用反斜杠引用将此美元符号插入进去。例如：

```
<!-#set var="password" value="\$cheese"  -->
<!-#echo var="password"  -->
```

它将打印$cheese 作为变量“password”的值。

同样，如果需要在字符序列中间引用变量名称，就它自身而言可能被认为是有效的标识符，可在此变量名两边使用一对大括弧。例如：

```
<!-#set var="uniqueid" value="${DATE_LOCAL}_${REMOTE_HOST}"  -->
```

取决于 timefmt 的设置，这将 uniqueid 设置成类似 Tue May 20 06:47:48 1998_206.171.50.51。

7.5 流程控制命令

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

像编程语言一样，在 SSI 模块中也可以使用流程控制。通过使用流程控制命令，可以有条件地产生不同的输出。最简单的流程控制（即，条件）语句是：

```
<!-#if expr="test_expression"  -->
<!-#endif  -->
```

其中，判断 text_expression，如果测试结果是真，那么直至 endif 命令为止的所有文本都被包括在输出中。text_expression 可以是下面的一种：

```
string
```

如果字符串不为空，它是真。或者：

```
string1 comparison_operator string2
```

其中，comparison_operator 可能是=、!=、<、>、<=或>=。

Note 如果 string2 的形式如同/string/，那么它被当作规则表达式。有关规则表达式的详细内容，请见附录 C。

让我们看 string 是字符串的例子：

```
<!-#if expr="foobar"  -->
This test is successful.
<!-#endif -->
```

这段语句始终打印“This test is successful”（测试成功），因为当 text_expression 是个非空字符串时，这个表达式为真。然而，如果将 expr="foobar"改成 expr=""或 expr="""，那么在 if-endif 块内的文本将不会成为输出结果。

让我们看字符串比较测试的例子：

```
<!-#set  var="quicksearch"  value="yes"  -->

<!-#if  expr="$quicksearch=yes"  -->
Quick search is requested.
<!-#endif  -->
```

其中，称为 quicksearch 的变量的值设置为 yes，然后与 yes 做比较。由于设置值与比较值相等，这时就会输出“Quick search is requested”（请求快捷查询）行。

通过使用诸如 !、&&和 || 这些逻辑操作符，能创建更复杂的 test_expressions。例如：

```
<!-#if expr="${REMOTE_HOST} = /206\.171\.50/ &&
${DOCUMENT_NAME} = /${DATE_LOCAL}/"  -->
<!-#exec cmd="/usr/local/build/timesheet/timecalc.pl">
<!-#endif  -->
```

其中，test_expressions 由两个子表达式组成。第一个子表达式，用来判断 ${REMOTE_HOST}=/206\.171\.50/，检查定义服务器的变量 REMOTE_HOST 是否与网络地址 206.171.50 相匹配。注意地址用简单的规则表达式/206\.171\.50/ 书写，这里每个“.”（句号）用“\”（反斜杠）符号进行转义。抑制“.”符号在规则表达式中的特殊含义。

Explorer 浏览器使用的字符串）。如果它确实包含这个字符串，那么将浏览器变量设置成“IE”，将称为 vbscript.html 的文件插入到当前文档中。另一方面，如果 HTTP_USER_AGENT 不包含 MSIE 字符串，就假定它为另外一个领先的浏览器（Netscape Navigator），因此将浏览器变量设置成“Navigator”，而将 javascript.html 文件插入到当前文档中。通过使用 if-then-else 结构，这个例子给同一个变量赋不同值并加载不同的文件。

7.6 小　　结

在本章中，讨论了如何使用服务器端包含（SSI）命令和扩展服务器端包含（XSSI）命令向静态的 Web 页添加动态内容。一旦可供使用，SSI 命令用起来很容易。然而，在所有 Web 页上使用 exec 命令是不明智的，因为这条命令增加了服务器的负担。使用 SSI 的最佳方法是将单一的文件扩展名当作 SSI 扩展名，并且只允许在需要的目录中使用这些命令。此外，如果不需要执行外部程序或 CGI 脚本，就应该禁止使用 exec SSI 命令。还讨论了 SSI 提供的流程控制命令，以及给新的变量赋值供以后使用的功能。

在下一章中，将讨论通用网关接口 CGI 配置。

第 8 章　CGI　配　置

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

本章内容：

- CGI 基础
- 怎样为 CGI 配置 Apache
- 怎样为用户提供 cgi-bin 访问
- 常用的 CGI 应用程序
- 怎样配置 Apache 来调试 CGI 应用程序

Web 服务器的主要工作是向客户（如 Web 浏览器）传递所请求的内容。在大多数情况下，请求的内容被存储在服务器的某种类型的文件中，并已准备好以供服务器访问。服务器检索文件，并将其内容传输到发出请求的客户系统。然而，如果内容没有准备好，需要由应用程序来产生这些内容又该怎么办？Web 服务器必须运行应用程序，并返回输出结果。这就是通用网关接口（CGI）发挥作用的地方。

CGI 规范告诉 Web 服务器怎样和外部应用程序对话。运行 CGI 应用程序的 Web 服务器实际上能使任何人在此服务器上运行外部应用程序。对于许多并不熟悉 CGI 的人们来说，这是个令人吃惊的想法。

在本章中，将讨论 CGI 基础，使你对它有清楚的理解。还将讨论设置 Apache 来支持 CGI 执行的细节。

8.1　CGI 是什么

在 Web 上你曾有过使用一个或多个 CGI 应用程序的机会。为了在 Web 上提供动态的、交互式的内容，许多流行的站点使用 CGI 应用程序。当然，随着越来越多的 Web 技术的涌现，新的通过 Web 传递动态内容的手段已可供使用。但是，大多数解决方案或者是针对某种语言的，或者依赖特定的操作系统或商用软件。从另一方面讲，CGI 是独立于语言的网关接口规范，它实际上可以用任何广泛流行的应用程序开发语言来实现，包括 C、C++、Perl、shell 脚本甚至 Java。

让我们讨论 CGI 程序是如何工作的（见图 8-1）。基本的概念是 Web 服务器从客户端得到某个 URL，它魔术般地（至少现在是这样）告诉 Web 服务器：必须运行名为 helloworld.cgi 的外部应用程序。Web 服务器启动这一应用程序，等待它完成并返回输出结果。然后，它将此应用程序的输出结果传输给另一端的 Web 客户。

当想使客户和应用程序对话时，会出现什么情况呢？从客户那里输入的数据必须提供给应用程序。与此相似，当应用程序产生输出结果时，服务器或客户又需要知道返回的输

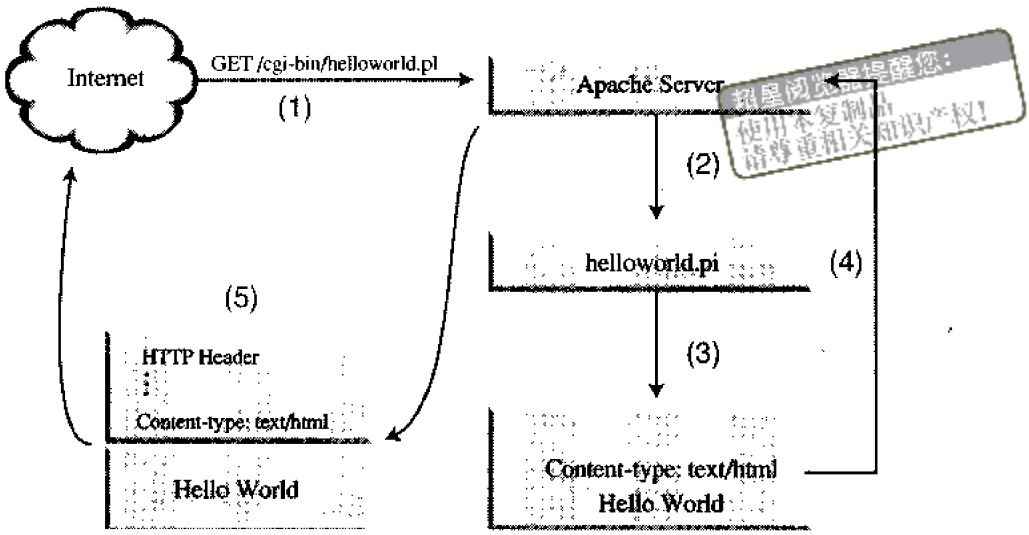

图 8-1 CGI 程序如何进行工作

出结果属于哪种类型。程序能产生文本信息、供进一步输入的 HTML 表单、影像等等。正如你所看到的，从程序到程序，输出结果会有很大的不同。所以，对应用程序来说，必须有方法通知 Web 服务器和客户有关输出结果的类型。

CGI 为服务器定义了一套标准方法把客户输入传送给外部应用程序，并且它还定义了外部应用程序如何返回输出结果。任何遵守这些标准的应用程序都可被称为 CGI 应用程序，或 CGI 程序，或 CGI 脚本。为简单起见，用 CGI 程序来表示任何这样的应用程序（如 Perl 脚本或 C 程序）。在下面这一节里，将讨论输入/输出过程是如何实现的。

8.1.1 CGI 的输入和输出

Web 服务器从客户（如 Web 浏览器）那里接收信息有多种方法。HTTP 协议定义了 Web 服务器和客户相互交换信息的方法。向 Web 服务器传输请求数据的最一般的方法是 GET 请求和 POST 请求。

8.1.1.1 GET 请求方法

GET 请求是最简单的方法。每当在 Web 浏览器中键入 Web 站点地址时，都会产生 GET 请求并将它发送给要到达的 Web 服务器。如果键入以下内容：

```
www.idgbooks.com
```

Web 浏览器就会发送：

```
GET /
```

到 www.idgbooks.com Web 服务器。这个 GET 请求要求 IDG Web 服务器返回该 Web 文档树的顶级文档。这个文档常常被称为主页，而且通常指的是在顶级 Web 目录里的 index.html 页。此外，HTTP 允许在 GET 请求中加入额外的信息。例如：

```
www.mycompany.com/cgi-bin/search.cgi?books=cgi&author=kabir
```

这里的 GET 请求如下：

```
GET www.mycompany.com/cgi-bin/search.cgi?books=cgi&author
=kabir
```

它告诉服务器执行 cgi-bin/search.cgi CGI 程序并传给它 books=cgi 和 author=kabir 输入数据。

当遵守 CGI 规范的 Web 服务器（如 Apache）接收此类请求时，它遵循 CGI 规范并将输入数据传给应用程序（在当前的情况下，在 cgi-bin 目录下的 search.cgi）。当通过 HTTP 的 GET 请求 CGI 资源时，Apache 做以下一些工作：

1. 为 CGI 程序设置环境变量。包括将 HTTP 的请求方法名称存储在名为 REQUEST_METHOD 的环境变量中，以及将从客户接收到的数据存储在名为 QUERY_STRING 的环境变量中。

2. 执行所请求的 CGI 程序。

3. 等待程序完成和返回输出结果。

4. 如果是需要分析头标的程序，则分析 CGI 程序的输出结果（不需要分析头标的 CGI 程序创建它自己的 HTTP 头标，服务器不必分析头标）。

5. 创建必要的 HTTP 头标。

6. 发送头标和程序的输出给提出请求的客户。

图 8-2 说明了这一过程。

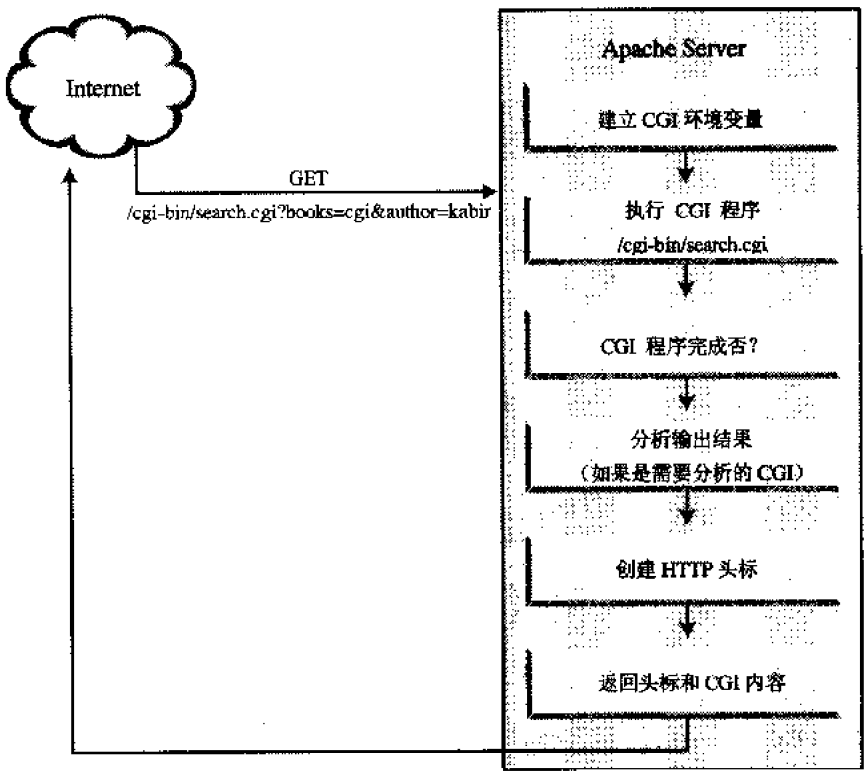

图 8-2　CGI 服务器的流程图

现在，让我们看看 CGI 程序必须做些什么来检索输入并在内部使用它。

正如从图 8-3 中所看到的，CGI 程序要做以下几方面的工作：

1. 读取 REQUEST_METHOD 环境变量。
2. 使用存储在 REQUEST_METHOD 变量中的值，确定是否使用了 GET 方法。
3. 如果是 GET 方法，提取存储在 QUERY_STRING 环境变量中的数据。
4. 对数据进行解码。
5. CGI 程序按要求处理解码的数据。
6. 在完成处理后，它将输出的 Content-Type 写入标准输出设备（STDOUT）。
7. 最后，在 STDOUT 上写出输出结果并退出。

Web 服务器读取应用程序的 STDOUT 并分析它，找出输出的 Content-Type。然后在向该客户传输此输出结果之前，传输相应的 HTTP 头标和 Content-Type。最后 CGI 程序退出，整个 CGI 的事务处理过程就完成了。

图 8-3 表示 CGI 程序的流程图

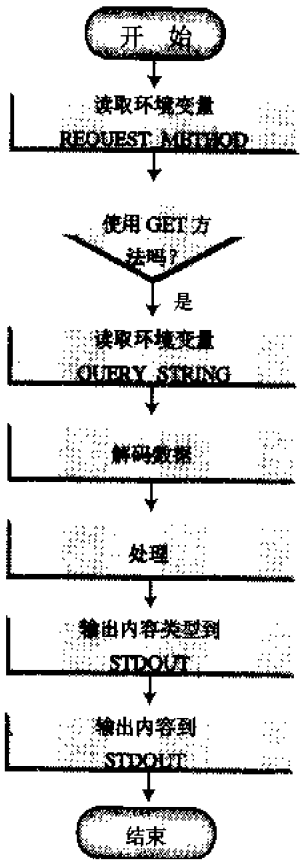

图 8-3 CGI 程序的流程图

注意，如果 CGI 程序自己提供必要的 HTTP 头标和 Content-Type 信息，那么在它的名称前面需要加前缀 nph（它表示不需要分析头标）。服务器不分析 nph CGI 程序的输出，并将它的输出结果直接传输给客户。大多数 CGI 程序让服务器输出 HTTP 头标，因而是一些需要分析头标的程序。

通过 GET 请求方法将输入数据传递给 CGI 程序，在许多方面有局限性，包括如下：

♦作为 URL 的一部分，能够传输的数据总量受到客户的 URL 长度限制的制约。许多流行的 Web 浏览器对 URL 长度有硬性限制。因而，通过编码的 URL 发送的数据总量是相当有限的。然而，在某些情况下，通过 URL 传递数据给 CGI 程序不失为一个好的方法。例如，如果有一个 HTML 表单，它使用 GET 方法给 CGI 程序发送数据，被提交的 URL 可以作为书签使用。以后使用时不必再通过数据输入表单。对数据库查询应用程序来说，它是用户友好的特性。

♦单个环境变量（QUERY_STRING）的值的长度是有限的。许多（如果不是全部）操作系统对一个环境变量值中所能容纳的字节数都有限制。这将实际限制能作为输入数据存储的字节总数。

8.1.1.2　POST 请求方法

HTTP POST 请求方法被广泛用于传递数据给 CGI 程序。在 Web 上填写的许多 HTML 表单中可以找到这种方法的典型用法。例如，列表 8-1 列出了此种表单。

列表 8-1

```
<HTML>
      <HEAD>
      <TITLE> Apache Server Bible - Chapter 8 Listing 8-1
</TITLE>
      </HEAD>

<BODY>
<H1>Listing 8-1</H1>
<H2>An Example HTML Form Using the HTTP POST Request
Method</H2>
<HR>
<FORM ACTION="/cgi-bin/search.cgi" METHOD="POST">

<PRE>
Type of Book <INPUT TYPE="TEXT" NAME="book" SIZE="10"
MAXSIZE="20"
Author's Name <INPUT TYPE="TEXT" NAME="author" SIZE="10"
MAXSIZE="20"
</PRE>
<INPUT TYPE=SUBMIT VALUE="Search Now">

</FORM>

</BODY>
</HTML>
```

注意在列表中有一个<FORM...>...</FORM>段。HTML 表单常常有一个开始的<FORM>标记，它为此表单定义 ACTION 和请求 METHOD。以上例子中，定义的操作是/cgi-bin/search.cgi CGI 程序，定义的方法是 POST。

紧接开始的<FORM>标记之后，常常有一个或多个 INPUT 实体。INPUT 实体可以包括文本输入框、下拉菜单和列表。在例子中，有三个实体。第一个使用户为 book 变量键入一个值。第二个与第一个类似，使用户为 author 变量键入一个值。第三个有些特殊，它使用户提交此表单。当用户提交此表单时，客户软件将 POST 请求传输到服务器上要求 ACTION（即/cgi-bin/search.cgi）资源，并以编码格式传输 book=<用户输入值>和 author=<用户输入值>。

8.1.1.3 比较 GET 和 POST

GET 和 POST 请求方法之间有什么区别呢？由 POST 传递的数据未被保存在 CGI 程序的 QUERY_STRING 环境变量中。改为保存在 CGI 程序的标准输入（STDIN）中。REQUEST_METHOD 变量被置成 POST，同时将编码数据存储在 CGI 程序的 STDIN 中，并且将名为 CONTENT_LENGTH 的新的环境变量设置为保存在 STDIN 中的字节数。

CGI 程序现在必须检查 REQUEST_METHOD 环境变量的值。如果为 HTTP POST 请求方法它的值被赋为 POST。该程序应该首先根据 CONTENT_LENGTH 环境变量的值确定输入数据的容量，并然后从 STDIN 读取数据。注意 Web 服务器不负责在 STDIN 中插入 End-of-File（EOF）标志，这就是为什么 CONTENT_LENGTH 变量被赋为以字节计的数据长度的原因，这使它能较易于为 CGI 程序确定数据的总字节数。

有可能会同时使用 GET 和 POST。下面是使用 POST 方法的简单的 HTML 表单。并且作为 CGI ACTION 的一部分，悄悄地塞进了查询字符串 username=joe。

```
<FORM ACTION="/cgi-bin/edit.cgi?username=joe" METHOD=POST>
<INPUT TYPE=TEXT NAME="PhoneNumber">
</FORM>
```

在这个例子中，username=joe 查询是 URL 的部分。但其他字段（PhoneNumber）是 POST 数据的一部分。效果是：最终用户能将 URL 作为书签，并经常作为 joe 在未设置其他字段值的情况下运行 edit.cgi 脚本。这对联机数据库应用程序和搜索引擎简直是太好了。

不管使用 GET，还是使用 POST，还是两者都使用，都需要对数据进行编码。直到 CGI 程序才对它进行解码。在下一节中，将讨论对编码数据的解码。

8.1.2 输入数据的解码

原始的 HTTP 协议的设计者们计划在任何系统上方便地实现此协议。此外，他们还想使数据编码方案变得简单。

编码方案将某些字符定义为特殊字符。例如，等号（=）用来形成一对“关键字=值”（key=value）；加号（+）用来代替空格符号；逻辑与符号（&）用来将两对“关键字=值”分开。

如果数据本身就带有具有特殊含义的符号，可能会搞不清在传输什么。在这种情况下，使用三字符的编码方案，它能对任何字符编码。百分符（%）表示一个由两位十六进制数组成的编码字符序列的开始。

CGI 程序能确定变量的值，并有条件地利用在 CGI 规范的不同版本中可用的不同特性。例如，如果这个值是 CGI/1.0，那么该程序就不可能使用任何 CGI/1.1 的特性，反过来也一样。

Note

在十进制标点前面的第一个整数被称为 major number（主版本号），而在十进制标点后面的整数被称为 minor number（辅助版本号）。因为这两个整数作为独立的数，所以 CGI/2.2 是比 CGI/2.15 更老的版本。

8.2.1.2 SERVER_ADMIN

如果在 httpd.conf 配置文件中使用 ServerAdmin 命令设置站点管理员的 e-mail 地址，这个变量就会被设置来反映它。还要注意：如果在虚拟主机配置容器中使用 ServerAdmin 命令，而被访问的 CGI 程序是虚拟主机的一部分，那么 SERVER_ADMIN 变量就会被赋值为那个地址。

8.2.1.3 DOCUMENT_ROOT

这个变量被赋值为被访问 Web 站点的 DocumentRoot 命令指定的值。

8.2.2 客户请求变量

Apache 从请求 CGI 程序的客户接收头标，并在此基础上创建一组环境变量。Apache 设置下列环境变量组向 CGI 程序提供信息。

8.2.2.1 SERVER_NAME

该变量告诉 CGI 程序，它访问的是哪一个主机。这个值或者是 IP 地址，或者是完整的主机名：

```
SERVER_NAME = 206.171.50.51
SERVER_NAME = www.nitec.com
```

8.2.2.2 HTTP_HOST

参见 SERVER_NAME 变量。

8.2.2.3 HTTP_ACCEPT

这个变量被赋值为客户能接受的 MIME 类型的列表。例如：

```
HTTP_ACCEPT = image/gif, image/x-xbitmap, image/jpeg,
image/pjpeg, image/png, */*
```

其中，客户宣称可以处理 GIF、JPEG、PNG 及其他影像。这使得 CGI 程序能确定哪一种输出对客户是合适的。例如，CGI 程序能产生 GIF，或者 JPEG。当接收如下的 HTTP_ACCEPT 时：

```
HTTP_ACCEPT = image/gif, */*
```

它将发送 GIF 输出而不是 JPEG，因为客户倾向前一种类型。

8.2.2.4 HTTP_ACCEPT_CHARSET

这个变量指定哪一种字符集是客户所能接受的，例如：

```
HTTP_ACCEPT_CHARSET = iso-8859-1.,*,utf-8
```

8.2.2.5 HTTP_ACCEPT_LANGUAGE

这个变量指定哪一种语言是客户所能接受的，例如：

```
HTTP_ACCEPT_LANGUAGE = en
```

在这种情况下，客户接受 en（英语）语言内容。

8.2.2.6　HTTP_USER_AGENT

这个变量指定发出请求的系统正在运行哪些客户软件和哪一种操作系统，例如：

```
HTTP_USER_AGENT = Mozilla/4.04 [en] (WinNT; I)
```

该信息相当于：

- 客户软件=Netscape Navigator（由于 Mozilla 是 Netscape 使用的关键字）
- 客户软件版本=4.04（英语版）
- 操作系统=Windows NT（Intel）

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

注意，尽管 Mozilla 这个词由 Netscape 浏览器独家使用，但是许多其他销售商开始把它作为 HTTP 头标的内容使用。例如，当在同一台机器上运行 Microsoft IE 4.0 时，它产生以下 HTTP_USER_AGENT 数据。

```
HTTP_USER_AGENT = Mozilla/4.0 (compatible; MSIE 4.0; Windows NT)
```

这一用户代理信息由许多 Web 站点广泛使用。最适合使用 Netscape Navigator 的站点（即它使用 HTML 或 JavaScript，或插件的某种特性，这些特性特别适合于在 Netscape Navigator 中使用）可能使用 HTTP_USER_AGENT 信息来为使用 Microsoft IE 或任何其他不太流行的浏览器返回不同的页。无论如何，建议坚持使用标准的 HTML（对于标准的 HTML 协议，可在 www.w3.org 上找到），并且根本就不实现适合特定浏览器的属性。尽管为单独的浏览器优化的页面可能使它们在其上看起来比较新鲜，但是并不是每个人都在使用那一种特定的浏览器。这同样表示特定浏览器使用的 HTML 标记或插件可能使其他不使用这一浏览器的人访问你的站点变得比较困难。

8.2.2.7　HTTP_REFERER

这个变量被赋值为对 CGI 程序发出请求的 URI。通过使用该变量，能够确定请求是来自你的 Web 页上的一个连接，还是来自远程的 URI。

8.2.2.8　HTTP_CONNECTION

这个变量被赋值为客户和服务器使用的连接类型。例如：

```
HTTP_CONNECTION = Keep-Alive
```

这表明客户能使用 Keep-Alive 处理持续的连接，并且当前正在使用它。

8.2.2.9　SERVER_PORT

这个变量值告诉 CGI 程序哪一个服务器端口是当前被用来访问此程序的。举个例子：

```
SERVER_PORT = 80
```

如果 CGI 程序创建指回服务器的 URL，包含这个端口地址可能是有用的。在 URL 中可以找到作为变量值的端口地址。

8.2.2.10　REMOTE_HOST

这个变量值告诉 CGI 程序有关客户的 IP 地址或 IP 名称的信息，例如：

```
REMOTE_HOST = ppp-007.speedlink.net
```

注意，如果使用 MINIMAL_DNS 选项编译 Apache 服务器，这个变量就不被赋值。

8.2.2.11 REMOTE_PORT

这一端口号由客户用来建立连接以请求这个 CGI 程序。

```
REMOTE_PORT = 1163
```

我尚未看到任何使用这个变量的情况。

8.2.2.12 REMOTE_ADDR

它是客户系统的 IP 地址：

```
REMOTE_ADDR = 206.171.50.51
```

注意，如果客户在防火墙或代理服务器后面，那么存储在此变量中的 IP 地址就可能不是客户系统的 IP 地址。

超星阅览器提醒您：

8.2.2.13 REMOTE_USER

这一变量只当对 CGI 程序的访问需要基本的 HTTP 认证时才被赋值。在基本的认证中用到的用户名被存放在这一变量中。无论如何，CGI 程序都无法辨认用来访问它的口令。如果这个变量被赋值为用户名，CGI 程序能安全地认为用户提供了相应的口令。

8.2.2.14 SERVER_PROTOCOL

这是客户发送 CGI 程序的请求时使用的协议与版本号。例如：

```
SERVER_PROTOCOL = HTTP/1.1
```

8.2.2.15 REQUEST_METHOD

这个变量被赋值成客户用来请求 CGI 程序的 HTTP 请求方法。典型值是：GET、POST、HEAD 等等。例如：

```
REQUEST_METHOD=GET
```

当请求方法是 GET 时，输入被存储在 QUERY_STRING 变量中。当请求方法是 POST 时，输入被存储在 CGI 程序的 STDIN 中。

8.2.2.16 REQUEST_URI

这个变量被赋值为请求的 URI。例如：

```
REQUEST_URI = /cgi-bin/printenv2
```

8.2.2.17 REMOTE_IDENT

这只有当 IdentityCheck 命令被赋值时才被赋值。这个变量存储由远程 identd（识别守护进程）返回的用户识别信息。因为许多系统并不运行这一类守护进程，所以它不应该作识别用户的可靠方法。

建议在企业内联网或外联网环境下使用这个方法，在这种环境下，你或你的组织正在运行 identd 服务器。

8.2.2.18 AUTH_TYPE

如果 CGI 程序被存储在 Web 站点的一个区域里，对此区域的访问需要进行认证。那么这个变量就被赋值为指定的认证方法。

8.2.2.19 CONTENT_TYPE

这个变量指定与请求头标结合在一起的 MIME 类型。例如：

```
CONTENT_TYPE = application/x-www-form-urlencoded
```

当使用 HTML 表单和 POST 请求方法时，可以指定在 HTML 表单里的内容类型，该表单使用<FORM>标记的 TYPE 属性如下：

```
<FORM ACTION="/cgi-bin/search.cgi"  METHOD="POST"  TYPE=
"application/x-www-form-urlencoded">
```

您的浏览器提醒您：
使用本复制品
请尊重相关知识产权！

8.2.2.20　CONTENT_LENGTH

当使用 HTTP POST 请求方法时，Apache 将输入数据（和此请求连接）存储在 CGI 程序的 STDIN 中。然而，服务器并不在 STDIN 中插入 End-of-File（文件结束，EOF）标记。而是将这个变量的值设置为输入数据的总的字节数。例如，如果：

```
CONTENT_LENGTH = 21
```

那么 CGI 程序应该从 STDIN 中读取 21 个字节的数据。

8.2.2.21　SCRIPT_NAME

这是所请求的 CGI 程序的 URI。例如：

```
SCRIPT_NAME = /cgi-bin/search.cgi
```

8.2.2.22　SCRIPT_FILENAME

这是所请求 CGI 程序的实际的、完整的路径名。例如：

```
SCRIPT_FILENAME = /www/kabir/public/cgi-bin/search.cgi
```

8.2.2.23　QUERY_STRING

如果客户使用 HTTP GET 请求方法，并在问号（？）后面提供输入数据，这些数据号被作为变量的值来存储。例如，对下面的 CGI 程序的请求：

```
http://apache.nitec.com/cgi-bin/search.cgi?key1
=value1&key2=value2
```

将使 Apache 对此变量赋值为：

```
QUERY_STRING = key1=value&key2=value2
```

CGI 程序/cgi-bin/search.cgi 在使用之前读取它，并对它进行解码。

8.2.2.24　PATH_INFO

如果将 CGI 程序的输入数据作为 URL 的一部分，路径的多余部分（它实际上是被调用程序的某些数据）将作为这个变量的值来存储。例如：

```
http://apache.nitec.com/cgi-bin/search.cgi/argument1
/argument2
```

将使 Apache 对此变量赋值为：

```
PATH_INFO = /argument1/argument2
```

注意，PATH_INFO 将不会包含查询字符串的任何部分。换句话说，如果 URI 包含查询字符串，应该将这部分数据存储在 QUERY_STRING 变量中。例如：

```
http://apache.nitec.com/cgi-bin/search.cgi/CA/95825?book
=apache&author=kabir
```

将使 Apache 对变量赋值如下：

```
PATH_INFO = /CA/95825
    QUERY_STRING = book=apache&author=kabir
```

8.2.2.25 PATH_TRANSLATED

这是被请求文件的绝对路径。

8.3 为 CGI 配置 Apache

在这一节中，将讨论如何配置 Apache 来处理 CGI 请求。配置过程包括告知 Apache 在哪里存储 CGI 程序，为特定文件扩展名设置 CGI 处理程序，以及指明应该将哪一种文件扩展名作为 CGI 程序。将 CGI 程序集中保存在一个目录里是个好主意。让 CGI 程序分散保存在 Web 的各个位置会使 Web 站点难以管理，而且还会造成难以跟踪的安全漏洞。

使用本复制品请尊重相关知识产权

8.3.1 给 CGI 程序目录创建别名

创建集中的 CGI 程序目录只是建立安全的 CGI 环境的开始。最好将集中的 CGI 程序目录保存在 DocumentRoot 目录外，使得不能直接访问 CGI 程序。为什么呢？是因为当涉及 CGI 程序时，你应该向外界提供尽量少的信息。这将确保你的站点有较好的安全性。人们对你保存 CGI 程序的实际位置知道得越少，对你的站点造成损害的机会就越小。

第一步是在 DocumentRoot 目录外创建一个目录。例如，如果/www/mycompany/public/htdocs 是 Web 站点的 DocumentRoot 目录，那么/www/mycompany/public/cgi-bin 就是 CGI 程序目录的好选择。要为 CGI 程序目录创建别名，可以使用 ScriptAlias 命令。

如果正在为主 Web 服务器设置 CGI 支持，那么编辑 httpd.conf 文件，并按下面的语法插入 ScriptAlias 命令行：

```
ScriptAlias  /alias/  /path/to/the/CGI/program/directory
/ending/with/
```

例如：

```
ScriptAlias  /cgi-bin/  /www/mycompany/pubic/cgi-bin/
```

如果正在为虚拟站点设置 CGI 支持，那么在定义虚拟主机的<VirtualHost...>容器内添加 ScriptAlias 命令行。例如：

```
NameVirtualHost 206.171.50.60
<VirtualHost 206.171.50.60>
ServerName blackhole.nitec.com
ScriptAlias  /apps/  /www/nitec/blackhole/public/cgi-bin/
</VirtualHost>
```

这里“/apps/”被用来创建 CGI 程序目录的别名。如果在/www/nitec/blackhole/public/cgi-bin 目录中有一个名为 feedback.cgi 的 CGI 程序，那么它只能按下面的方式进行访问：

```
http://blackhole.nitec.com/apps/feedback.cgi
```

在设置 ScriptAlias 命令后，要确保该目录只允许 Apache 有读取和执行在此目录中的文件的权限。

由 ScriptAlias 指向的目录应该有非常严格的权限设置。除了 CGI 程序开发员或服务器管理员以外的任何人，都不能拥有对此目录完全的（读、写和执行）权限。注意，可以定义多个 CGI 程序目录别名。而且出于安全考虑，在缺省的情况下，由 ScriptAlias 指定的目录是不可以浏览的。

当被请求时，Apache 试图运行由 ScriptAlias 指定的目录中的任何可执行（文件的权限设置）文件。例如：

```
http://blackhole.nitec.com/apps/foo.cgi
http://blackhole.nitec.com/apps/foo.pl
http://blackhole.nitec.com/apps/foo.bak
http://blackhole.nitec.com/apps/foo.dat
```

浏览器提醒您：
本网页内容受版权保护，请勿复制或传播
请尊重相关知识产权！

上面所有的 URL 请求都将提示 Apache 要试图运行各种不同的 foo 文件。

我并不特别喜欢这种想法，即由 ScriptAlias 指定的目录中的任何文件都可作为 CGI 程序来运行。我喜欢能使我限制 CGI 程序扩展名的解决方案，这样只有那些带有特定扩展名的文件才能作为 CGI 程序。在下一节中，将讨论如何通过在 mod_cgi 模块中找到的 Apache 处理程序来实现这一点。

8.3.2　选择特定的 CGI 文件扩展名

在这一节中，将提供一个配置样本。其中，将使一组选择的文件扩展名作为 CGI 程序来对待。这是通过使用 AddHandler 处理程序来完成的。

假定 Apache 服务器的名称是 www.nitec.com，并且它的 DocumentRoot 目录被设置为 /www/nitec/public/htdocs；CGI 程序目录是/www/nitec/public/cgi-bin。注意，CGI 程序目录被有意地保存在由 DocumentRoot 指定的目录以外。这样能确保该目录不被任何人浏览，因为 Apache 只有使用别名才能看到它。

步骤 1：删除/禁止 ScriptAlias

首先要做的是禁止任何现有的 ScriptAlias 命令。或者将它从配置文件（httpd.conf）中彻底删除，或者在行的第一个字符位置插入#号，使它成为注释行。

步骤 2：为 CGI 程序目录创建别名

不用别名（或符号连接）就无法访问 CGI 程序，因为它保存在文档树以外。可以通过使用 Alias 命令来定义别名，这个命令的语法如下：

```
Alias  /alias/  /path/to/cgi/dir/outside/doc/root/
```

按照此语法，所需要的 alias 命令如下：

```
Alias /cgi-bin/  /www/nitec/public/cgi-bin
```

现在必须告诉 Apache 从这个目录执行 CGI 程序。为了做到这一点，需要为这个特殊目录定义一个<Directory...>容器。

步骤 3：为 CGI 程序目录定义目录容器

要想它发挥作用（即将这个目录转换成 CGI 程序目录），应该按如下方式定义目录容器：

```
<Directory /path/to/cgi/dir/outside/doc/root>
Options ExecCGI
AddHandler cgi-script .extension .extension ...
</Directory>
```

超星阅览器提醒您：
使用非复制品
请尊重相关知识产权！

这个目录容器给出了 Apache 命令。首先，Option ExecCGI 告诉 Apache 允许在此目录中执行 CGI 程序。其次， AddHandler cgi-script .extension .extension ...告诉 Apache 将扩展名列表当作 CGI 程序的扩展名（换句话说，当 Apache 碰到 URL 请求中带有上述扩展名的文件时，它必须把它当作 CGI 程序来执行）。

实际的容器定义如下：

```
<Directory /www/nitec/public/cgi-bin>
Options ExecCGI
AddHandler cgi-script .cgi .pl
</Directory>
```

这里将 .cgi 和 .pl 作为有效的 CGI 程序扩展名。因此，当遇到下列请求时：

```
www.nitec.com/cgi-bin/anything.cgi
www.nitec.com/cgi-bin/anything.cgi.pl
```

Apache 试图将这些文件作为 CGI 程序执行。当然，如果这些文件不是真正的可执行文件，Apache 会显示（并可能记录下）错误信息。

Note 前面提到的有关 CGI 程序目录的权限设置仍然适用于这一配置。同样的配置还适用于虚拟主机站点。

8.3.3 允许用户访问 cgi-bin

许多 Internet 服务提供商（Internet Service Provider，ISP）提供带有用户帐号的 Web 站点。这些 Web 站点通常具有如下的 URL：

```
http://www.isp.net/~username
```

他们常常从用户那儿得到对 cgi-bin 访问的请求。cgi-bin 访问这个词是个通用词，它被许多人用来表示在 Web 服务器上的 CGI 功能。传统上，已经将/cgi-bin/作为 CGI 程序目录的别名，并由此形成了这个词。其他的流行词是主页，它指的是用户的 Web 目录的最上层目录的索引页。

在这一节中，将讨论在 Apache Web 服务器上为用户提供的两种 cgi-bin 访问方法。只需要用一种方法就行了。

8.3.3.1 使用 Directory 或 DirectoryMatch 容器

当 UserDir 命令被赋值为目录名称时，Apache 就把它作为用户 Web 站点的顶层目录。例如：

```
ServerName www.yourcompany.com
UserDir public_html
```

现在，当对 www.yourcompany.com/~username 的请求到来时，Apache 查找指定用户的主目录（在 UNIX 系统上通常通过检查/etc/passwd 文件来得到），然后加上由 UserDir 命令指定的目录来形成此用户的顶级 Web 目录的路径名。例如：

```
www.yourcompany.com/~joe
```

这将使 Apache 寻找/home/joe/public_html（假设/home/joe 是用户 joe 的主目录）。如果存在这个目录，将它的索引页发送给提出请求的客户。

为每个用户添加 CGI 支持的一个方法是在 Apache 的一个配置文件中添加下列配置：

```
<Directory  ~"/home/[a-z]+/public_html/cgi-bin">
Options ExecCGI
AddHandler  cgi-script  .cgi   .pl
</Dirctory>
```

或者，如果正使用最新的 Apache 服务器，可以使用下面的配置：

```
<DirectoryMatch  "/home/[a-z]+/public_html/cgi-bin">
Options ExecCGI
AddHandler cgi-script  .cgi   .pl
</DirectoryMatch>
```

在这两种方法中，Apache 将 www.yourcompany.com/~username/cgi-bin/请求翻译成/home/username/public_html/cgi-bin/，并允许执行任何带有正确扩展名（.cgi 或.pl） 的 CGI 程序。

注意，为了能够正常工作，所有的用户名都必须用小写字符。如果用的是字母数字的用户名，就需要使用不同的规则表达式。

8.3.3.2　使用 ScriptAliasMatch 命令

通过使用 ScriptAliasMatch 命令，也可以为每个用户支持 CGI 程序目录。例如：

```
ScriptAliasMatch ~([a-z]+)/cgi-bin/(.*) /home/$1/public_html/cgi-bin/$2
```

这个命令将用户名与$1 相匹配，其中$1 和~([a-z]+)相等。将/cgi-bin/后面的任何内容与$2 相匹配，其中$2 和(.*)相等。然后，它使用$1 和$2 来建立 CGI 程序的实际的位置。例如：

```
http://www.yourcompany.com/~kabir/
cgi-bin/search.cgi?book=dummies&author=kabir
```

在这里，~([a-z]+)将在波浪符号（~）后面的一个或多个小写字符映射为$1。换句话说，这一对括号（ ）使我们能在~符号和用户名后的反斜杠（/）之间捕获任何内容。所以，$1 被赋值为 kabir。而且（.*）匹配跟在 cgi-bin/后面的任何内容，一对括弧和规则表达式使得可以在$2 中放入任何内容。所以$2 被赋值为 search.cgi?book=dummies&author=kabir。

现在 Apache 可以使用下面的规则表达式建立 CGI 程序目录的实际地址：

```
/home/$1/public_html/cgi-bin/$2
```

为前面的例子使用这一规则表达式将产生下列路径：

```
/home/kabir/public_html/cgi-bin/search.cgi?book
=dummies&author=kabir
```

因为这是保存 CGI 程序 search.cgi 的地方，所以它就执行。

如果你像我一样——不喜欢在 public_html（即由 UserDir 指定的目录）下拥有 CGI 程序目录——可以通过删除表达式的 public_html 部分将它排斥在外。例如：

```
ScriptAliasMatch  ~([a-z]+)/cgi-bin/(.*) /home/$1/cgi-bin/$2
```

这将映射下面的例子：

```
www.yourcompany.com/~ kabir/
cgi-bin/search.cgi?book=dummies&author=kabir
```

到下面的实际文件：

```
/home/kabir/cgi-bin/search.cgi?book=dummies&author=kabir
```

您的浏览器提醒您：
使用本复制品
请尊重相关知识产权！

当然，如果不喜欢让用户子目录（即，public_html）人人可读，那么可以通过为用户创建 Web 分区（或一个目录）来弥补这一点，并提供各自的目录来容纳他们的主页。

8.3.4 使用 AddType 命令创建新的 CGI 扩展名

前面讨论了如何使用 AddHandler 命令来创建 CGI 程序扩展名。然而，如果想在特定目录里创建新的 CGI 程序扩展名，还能使用 .htaccess（或者由 AccessFileName 命令指定的文件）。

在使用针对目录的访问控制文件（.htaccess）添加新的扩展名以前，需要创建一个 <Directory>容器如下：

```
<Directory  /path/to/your/directory>
Options ExecCGI
AllowOverride  FileInfo
</Directory>
```

在上面目录容器里的第一个命令告诉 Apache 你允许在此目录中执行 CGI 程序。第二个命令告诉 Apache 你允许在针对目录的访问控制文件（.htaccess）中使用 FileInfo 特性。这个特性能使你在针对目录的访问控制文件中使用 AddType 命令。

要添加新的 CGI 程序扩展名（.wizard），所要做的是在这个目录中创建 .htaccess（或由 AccessFileName 命令指定的任何名称）文件，其中包含下面的命令：

```
AddType  application/x-httpd-cgi   .wizard
```

现在，重新命名目录中已有的 CGI 程序使它们成为带有.wizard 扩展名的程序，并通过你的浏览器请求它。确保将此目录和 CGI 程序的所有文件权限设置为只能由 Apache 读取与执行。

8.4 运行 CGI 程序

到此为止，已经讨论了 CGI 规范的基础，Apache 如何对它提供支持，以及如何配置 Apache 来运行 CGI 程序。如果你有幸成为 Apache 管理员，就会需要准备 CGI 程序，或者甚至学着写它们。在这一节中，将讨论创建非常简单的 CGI 程序的基础知识。注意，这并不是一本 CGI 编程方面的书，因而，不适合深入讨论 CGI 编程。在这一节中，重点

在于揭示CGI程序的有关内容，它将有助于Apache管理员更好地管理她/他的可以使用CGI的 Web 站点。

在本节中的许多例子都使用 Perl。如果你的系统中没有 Perl，可以从 www.perl.com 处获取它的源代码或二进制文件。要注意，当可能时，好的做法是为系统编译二进制文件，而不是使用由其他人创建的二进制文件。

8.4.1　简单的 CGI 程序

在开发 CGI 程序时，Perl 语言用得最多。因为 Perl 是一种解释语言，用它创建的文件被称为脚本。因此，基于 Perl 的 CGI 程序被称为 CGI 脚本。列表 8-2 列出了一个简单的 Perl 脚本。

列表 8-2：helloworld.pl

```
#!/usr/local/bin/perl

print "Content-type: text/html\n\n";
print <<HTML;

<HTML>
<HEAD>
<TITLE>Listing 8-2: Simple Perl-based CGI Program</TITLE>
</HEAD>
<BODY>
<CENTER>
<H1> Hello World </H1>
</CENTER>
</BODY>
</HTML>

HTML
```

当运行这个程序时，它的输出结果如图 8-4 所示。

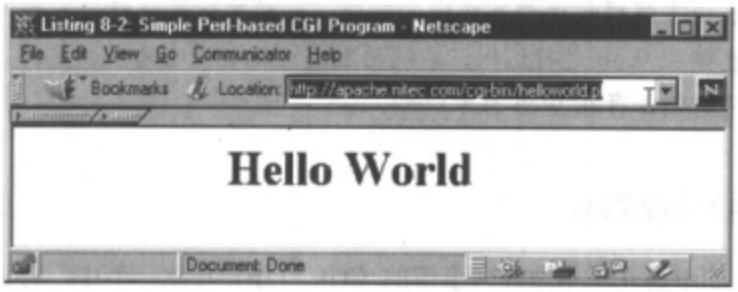

图 8-4　helloworld.pl 的输出

如果仔细检查 helloworld.pl 脚本，就会发现脚本的大部分内容都是普通的 HTML 文本。这个脚本可以进一步简化以便分析：

```
#!/usr/local/bin/perl

print "Content-type: text/html\n\n";

print <<HTML
HTML TEXT GOES HERE
HTML
```

第一行非常重要。当 Apache 运行 Perl 脚本时，第一行告诉系统它是个 Perl 脚本，而 Perl 解释程序是/usr/local/bin/perl。这使得系统能运行 Perl 解释程序并将这个脚本文件传递给它。要注意的是，这种通知系统有关解释程序的方法对所有脚本语言是标准的。下面的例子表示作为 C Shell 脚本的简化的 helloworld.pl 脚本。

```
#!/bin/csh
echo "Conten-type: text/html"
echo ""
echo "<HTML>"
echo "HTML TEXT GOES HERE"
echo "</HTML>
```

正如所看到的，第一行与 Perl 脚本相似。这里，解释程序的路径是/bin/csh。所以当从 Web 上下载自由的脚本时，要确保第一行正确指向你的系统的解释程序的路径。这是检查要做的第一件事情。

现在再回过头看一看 Perl 脚本。会发现第一条打印语句打印 Conten-type: text/html \n\n，它实际上打印一行 Conten-type: text/html 和一空行（在 Perl 中，\n 表示新的一行）。这是绝对必要的。记得前面提到的需要分析的 CGI 脚本必须为 Apache 提供头标信息，所以它能创建相应的 HTTP 头标。这个 Content-type 头标告诉 Apache，它的输出含有 HTML 类型的文本数据。空行将此说明与其余的数据分隔开，这是必要的。C Shell 脚本也这么做，但借助于空的 echo 语句。紧跟在 Conten-type 头标后的是 HTML 文本。

在 Web 上保存解释程序（如 Perl）使它可以从任何目录进行访问是个糟糕的想法。许多人为这种错误吃过不少苦。因为 Perl 和许多其他脚本语言解释程序都有极强的功能，包括执行系统命令。如果解释程序能直接从 Web 进行访问，系统的滥用者就能用它来攻击系统。因此，通常将它们保存在 Web 目录以外。

8.4.2 一个有用的 CGI 脚本

前面讨论了简单而几乎不做任何事情的 CGI 程序。在这一节中，将讨论一个只需稍加修改就能使用的十分有用的 CGI 脚本。

然而，在进一步讨论前，我想说说怎样才能算一个好的 CGI 程序。一个好的 CGI 程

序有下面的一些特点（至少从我个人观点看是这样）：

♦它使用 HTML 模板代替在代码中的硬编码（hard-coded）的 HTML。HTML 模板一般是带有 HTML 标记和一些定制标记的文本文件。定制标记由 CGI 程序用数据取代。使用 HTML 模板是需要的，因为它能使 HTML 作者和图形设计者不需要 CGI 开发人员的帮助，就能对应用程序产生的结果在外观和感受方面作很大的调整。

♦它使用正确的机制以允许应用程序的多例程得以同时运行。例如，如果一个 CGI 程序在某个位置写一个文件，它必须考虑这个 CGI 程序的多个副本也可能运行。因此，它必须谨慎地处理这种情况。在可能应用时，应当使用文件锁定（File-locking）技术。

♦它使用来自外部库和软件包的子程序。这使得这些程序易于维护。

也许你不一定对创建 CGI 程序负责，但作为 Apache 管理员，则需要你来维护它们。这就是说，你将涉及 CGI 程序的修改、它们的配置和安装这一类型的工作。我曾经和许多 Apache 管理员一同工作过，他们专门负责他们组织的 Web 站点的各个方面。我发现他们对 CGI 的深入了解大大帮助了他们在 CGI 程序的外购和内部开发方面做出良好决策。

现在来看一个能检验和修改的 CGI 程序。因为大多数 CGI 程序处理由 HTML 表单返回的用户输入，因此举一个用 Perl 写出的 CGI 程序例子，这个 CGI 程序是个相当好的、通用的 HTML 表单处理程序。列表 8-3 列出了一个这样的程序。

列表 8-3：　wizform.cgi

```
#!/usr/local/bin/perl
#
# Script Name: wizform.cgi
#
# Purpose: this is a general purpose HTML form
# processor script that processes data found it
# one or more HTML forms and stores then in a file
# and/or sends them one or # more persons via email.
#
# Requirements:the HTML form must have a
# HIDDEN variable called conf_file set to
# the name of the of the configuration file
# that MUST be stored in the same file as the script.
# See docs for details
#
# Version Control Information:
# $Id$
# $Author$
# $Revision$
# $Status$
# $Lock$
#
# Copyright (c) 1997 Mohammed J. Kabir (kabir@nitec.com)
#######################################################
```

```
# Tell Perl that we want to use the CGI.pm package
# in this script.
use CGI;

# Create a global CGI object called $cgi
local $cgi = new CGI;

#  Get the name of the HTML form configuration file
#  and store it in a  variable called
#  $conf_file.
my $conf_file = $cgi->param('conf_file');

#  Get the name of the next HTML form page (if any)
#  and store it in a variable called
#  $next_template.
my $next_template = $cgi->param('next_form');

# Other (local) variable declarations
my $key;
my $value;

# we will store configuration data in this
# associated array (hash).
Local %conf;

# Print the necessary Content-Type header.
# Since we only output HTML text, it
# should be he default text/html.
Print $cgi->header();

# Read the configuration file pointed by the $conf_file
variable and store the config. file
# in the %conf associated array.
#
& readConf($conf_file);
# If the $next_tempate variable (which holds the
# file name of the next HTML form)
# is empty then we have no more HTML forms to
# process and therefore we should complete processing
# for this HTML form by calling the processData
# sub routine.
If ($next_template eq ''){
   &processData;
   }

# We do have one or more HTML forms to process
```

超星阅览器提醒您:
使用本复制品
请尊重相关知识产权!

```
# so lets show the next HTML form and
# get more input data by calling the getInput
# sub routine.
else{
   &getInput($next_template);
   }
# We are done; exit the script.
exit 0;

Sub processData{
  # Purpose: this subroutine does all the data processing.

  # an array to hold the saving order of data fields.
  My @saving_order = split (/\,/,$conf{'saving_order'});

  # a variable to hold a data field's value or name
  my $datam;

  # an array to hold all the errors.
  my @errors;

  # a variable to hold a record that gets written to file.
  My $record;

  # a  variable to hold the mail message
  my $mail_msg;

  # keeps count of the number of output methods
  # that are successful.
  My $output_successful = 0;

  # keeps count of the number of output methods
  # that are requested.
  My $output_type = 0;

  # Check the inputs for missing (required) inputs
  # by calling the checkInputs sub routine
  # which returns a list of HTML form variable names
  # that are missing.
  # Store the list in @errors array.
  @errors = &checkInputs;

  # If we have errors (i.e. missing inputs) then
  #  we need to show the error message page
  if ($#errors > -1){
  &showErrors(@errors);
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

```
  return 1;
  }

# Lets create the mail message and the data
# record that we need to send and store
# respectively. So cycle through all the in
# @saving_ouder and add them to the $record
# variable and the $mail_msg variable in that
# order. Use the separator specified in the
# configuration file.
foreach $datam (@saving_order){
$record.= $cgi->param($datam).
$conf{'field_separator'};
  $mail_msg.=ucfirst($datam). "\t=\t".$cgi->param ($datam) ."\n";
}
# Remove the last field separator that got inserted
# in the loop.
$record =~s/$conf{'field_separator'}$//;

# If output by mail is requested in the configuration
# file, then send a mail message to
# specified list of people (found in the configurstion
# file.)
# Store 1 or 0 (depending on success of sendMail
# routine) in the
# $output_successful variable.
# Also increment number of output types we have
# processed so far.
If ($conf{'output_method"} =~ /mail/i){
  $output_successful = &sendMail ($mail_msg);
  $output_type++;
  }

# If output is to be stored in a file (specified
# in the configuration file) then store the
# newly created record in the specified file.
# Increment the $output_successful variable with
# whatever value returned by the
# append File. Also increment number of output
# types we have processed so far.
#
if($conf{'output_method'} =~ /file/i){
    $output_successful +=
    &appendFile($record,$conf{'data_file'});
    $output_type++;
    }
```

超星阅览器提醒您:
使用本复制品
请尊重相关知识产权!

```
# Now in order to figure out if we are successful
# in producing all requested types of output,
# we must compare the $output_successful variable
# with $output_type variable. An equal
# count will indicate that we are completely
# successful. If we are not, then the
# $output_type will have a higher value and there
# we must report a failure by showing the
# failure page (specified in configuration file.)
#
if($output_successful >0 && $output_successful ==
   $output_type){
   &showPage($conf{'success_template'});
}
else{
   $showPage ($conf{'failure_template'});
   }

# Finally, write the log file.
&writeLog;
}

sub writelog{
# Purpose: this sub routine writes a log file which
# records the IP address of the client
# along with time, date, referrer URL etc.

# Local variable delarations
my $log_file;
my $record;
my $status;

# We need to create a string containing current
# date and time using localtime() function.
My (↓tefields) = localtime(time);
My $sec    = $datefields[0];
My $min    = $datefields[1];
My $hr     = $datefields[2];
My $mday   = $datefields[3];
My $mon    = $datefields[4]+1;
My $year   = $datefields[5];
My $today  = sprintf("%02d:%02d:%02d
%s/%s/%s",$hr,$min,$sec,$mon,$mday,$year);

# Create the record we want to store in the log file
```

超星阅览器提醒您:
使用本复制品
请尊重相关知识产权!

```
$record = sprintf("%s, %s, %s, %s, %s",
            $today,
            $ENV{'SERVER_NAME'},
            $ENV{'REMOTE_ADDR'},
            $ENV{'HTTP_USER_AGENT'},
            $ENV{'HTTP_REFERER'}
            ) ;

# Write the log entry to the log file specified
# in the configuration file.
$status = &appendFile($record,$conf{'log_file'});

# Return the success or failure status of the
# log entry attempt.
return ($status);
}

sub sendMail{
# Purpose: this sub routine sends an email message
# to the list of addresses specified in the
# configuration file.

# Get the data (body of the email) from the
# passed parameter list.
My $data = shift;

# Get the FQPN of the mail daemon program from
# the configuration file.
my $sendmail = $conf{'mail_daemon'};

# Get the list of recepients from the configuration file.
my $to = $conf{'mail_to'};

# Get the name of the sender from the configuration file.
my $from = $conf{'mail_from'};

# Get the subject name from the configuratio file.
my $subject = $conf{'mail_subject'};

# Open a PIPE to the mail daemon program.
# if we fail to connect to the daemon, return
# failure status.
Open(MAIL,"!$sendmail -t") || return 0;

# Write the mail message to the pipe
print MAIL << DATA;
```

超星阅览器提醒您:
使用本复制品
请尊重相关知识产权!

```
To: $to
From: $from
Subject: $subject

$data
DATA
# Close the pipe
close(MAIL);

# We have completed the mail transaction so return the
success status.
Return 1;
}

sub showPage{
# Purpose: this sub routine shows an HTML page

# Get the page file name from the parameter list
my $file = shift;

# Local variables
my  @html;
my  %vars;
my  $key;
my  $value;

# Create an associated array of all the HTML
# form inputs that we have value for.
foreach $key ($cgi->param()){
  $value = $cgi->param($key);
  $var{$key} = $value;
  }

# Create the FQPN for the HTML page we want to display.
$file =
sprintf("%s/%s/%s",$conf{'base_dir'},$conf{'template_dir'},
$file);

# Read the page text into a local buffer (an array)
@html = &readFile($file);

# Before printing out each line found in the
# HTML page, replace any custom
# replaceable data that is usually in the
# <VISITOR-STRING> format.
foreach $line (@html){
```

超星阅览器提醒您:
使用本复制品
请尊重相关知识产权!

```
    $line =~ s/<(VISITOR-\w+)>/ucfirst($var{$conf{$1}})/eg;
    print $line, "\n";
    }

}

sub appendFile{
# Purpose: this sub routine appends a line in a file.
# Get the record, from the parameters, we want
# to append in the file.
My $record = shift;

# Get the file name from the parameters.
My $filename = shift;

# Create the FQPN of the file
$datafile =
sprintf("%s/%s/%s",$conf{'base_dir'},$conf{'data_dir'},
$filename);

# Open the file in append mode. If we fail
# send the failure status (0)
open (FP,">$datafile") || return 0;

# *** Critical Zone ***
# Try to get an exclusive lock for the file
flock FP, 2;

# Write the record in the file.
Print FP $record, "\n";

# Release the lock for the file
flock FP, 8;

# ** end of Critical Zone ***

# Close the file.
close(FP);

# we successfully wrote to the file so return
# success status.
return 1;
}
sub checkInputs{
# Purpose: check the inputs for missing fields.
# The required fields are specified in the
```

超星阅览器提醒您:
使用本复制品
请尊重相关知识产权!

```
# configuration file.

# Get the list of the required fields from the
# configuration file.
My @required_data = split(/\,/,$conf{'required_fields'}0;

  # Local variables
  my $datam;
  my $key;
  my $value;
  my @errors;

# Go through each required field and check to see if
# we have received a value of it
# If not, put the variable name in the error list array.
foreach $datam (@required_data){
   $value = $cgi->param($datam);
   if($value eq ''){
     push(@errors, $datam);
     }
  }
# Return the error list.
Return (@errors);
}

sub getInput{
# Purpose: Show an HTML form but before that insert all the
# inputs we have gathered from other forms so far by making
# hidden HTML form variables.

# Get the name of the HTML form file we want to show
my $html_template = shift;

# Local variables
my $hidden_data;
my $key;
my $value;
my @html;

# Create FQPN for the template file
$html_template = sprintf("%s/%s/%s",$conf{'basc_dir'},
$conf{'template_dir'},$html_template);

# Create hidden data list;
  foreach $key ($cgi->param()){
     next if ($key =~ /next_form/i);
```

超星阅览器提醒您:
使用本复制品
请尊重相关知识产权!

```
      $value = $cgi->param($key);
      $hidden_data.= <<HIDDEN;
      <INPUT TYPE="HIDDEN" NAME="$key" VALUE="$value">HIDDEN
      }

# Get the page text by reading the file and store the
# data in a buffer.
@html = &readFile($html_template);

# if the buffer is not empty we have page text.
# So print the page text and replace
# the <HIDDEN-DATA> tag with value of $hidden_data variable.
If($#html > -1){
   foreach  $line (@html){
      $line =~ s/<HIDDEN-DATA>/$hidden_data/g;
      print $line,"\n";
      }
}

# On the other hand, if we don't have any text in
# this page, we have some sort of access problem to the
# page so show a general failure message using the
# failure_template value found in the configuration.
else{
    &showPage($conf{'failure_template'});
    }
}

sub readConf{
# Purpose: this sub routine reads the configuration file
# and stores the configuration data in an associated arry.

# Get the name of the configuration file
my $conf_file = shift;

# Local variables
my @buffer;
my $line;

# Get the data from the file and store it in a local buffer.
@buffer = &readFile($conf_file);

# Cycle through the lines in the buffer and locate valid
# configuration data while
# ignoring empty blandk lines and comment lines that
# starts with a '#' sign.
```

超星阅览器提醒您:
使用本复制品
需尊重相关知识产权!

```
# Get the error list from the parameter list and
# store them in a local list.
My @fields = @_;

# Local variables
my $err_template;
my @html;
my $line;
my $errorlist;

# Get the FQPN of the error template file.
$err_template =
sprintf("%s/%s/%s",$conf{'base_dir'},$conf{'template_dir'},
$conf{'input_error_template'});

# Create an HTML list to display the fields that are
# in error.
$errorlist = "<UL>";
foreach (@fields){
$errorlBt.="<LI> $=";
}
$errorlist.= "<UL>";

# Read the error message template file and store the
# page text in a local buffer.
@html = &readFile($err_template);

# Cycle through the page text and replace
# the <ERROR-LIST> variable with the
# error list created earlier. Print the page.
foreach $line (@html){
   $line =~ s/<ERROR-LIST>/$errorlist/g;
   print $line;
   }
}
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

理解这个脚本的最好方法是查看这个脚本的应用例子。在下一节中，将说明如何通过使用这一通用的表单处理程序来建立简单的名为 On-line Newsletter Signup Tool 的应用程序。

8.4.3 简单的应用程序

这个应用程序的目的是允许站点访问者在由这个站点拥有者定期提供的业务通信上签约。这个签约工具将收集访问者的名称和 e-mail 地址，并将它们储存在一个文件中，这使此站点的拥有者以后能使用这些数据发送业务通信。

在下一节中，将说明如何设置这个应用程序所需要的配置文件和 HTML 模板。对于这个例子，将假设如下内容：

```
DocumentRoot = /www/nitec/public/htdocs
```

首先是为签约工具建立配置文件。通过使用文本编辑程序，建立称为 signup.conf 的配置文件。键入下面这些配置行：

超星阅览器提示：
使用本复制品
请尊重相关知识产权！

```
base_dir=/www/nitec/public
template_dir = htdocs
```

base_dir 行告诉脚本哪一个目录被用作此脚本的基目录。因为这个目录必须是 DocumentRoot（/www/nitec/public/htdocs）以外的一层目录，它被赋值为/www/nitec/public 目录。template_dir 行告诉脚本存储 HTML 模板文件的目录名。此例中，顶级 Web 目录（DocumentRoot）被用于此目的。注意，template_dir 的值必须是 base_dir 的相对路径。

需要在同一个文件中像下面一样指定与输入有关的配置信息：

```
input_fields=firstName,lastName,emailAddress,companyName,
officePhone
required_fields=emailAddress,firstName,lastName
```

其中，firstName、lastName、emailAddress、companyName、officePhone 是这个应用程序的 HTML 表单的输入字段名。注意，列表中的每个字段必须用逗号分开。第二行告诉脚本有关必需字段的信息。在前面的这个例子中，这些必需字段是： e-mailAddress、firstName 和 lastName 字段。换句话说，如果用户不向这些字段键入内容，那么脚本会显示错误信息页。

现在需要告诉脚本将这些从访问者收集的数据写到哪里。需要将下面这些行添加到配置文件中：

```
data_dir=data
data_file=newsletter.cvs
field_separator= ,
saving_order=emailAddress,firstName,lastName,companyName,
officePhone
```

第一行指定写入这些数据文件的目录名。将这个目录名附加到 base_dir 值后面以得到完整的目录名。注意，data_dir 行将此目录设置为/www/nitec/public/data。它在 DocumentRoot 指定的目录以外。确保这个目录可由 Web 服务器读/写。出于安全的考虑，这个目录通常应该被保存在 Web 文档空间以外的地方。第二行指定用来写数据的文件名。第三行为写入 newsletter.cvs 文件中的数据字段指定字段分隔符。这个例子使用逗号（，）作为字段分隔符。第四行指定将这些数据字段写入数据文件时采用的顺序。

现在你需要告诉脚本有关 HTML 模板的信息。这是通过下面这几行做到的：

```
input_error_template = newsletter.err.html
success_template = newsletter.success.html
failure_template = newsletter.failed.html
```

第一行指定错误模板文件的名称，当发生错误时，它被用来显示定制错误信息。这个文件的完整路径是/www/nitec/public/htdocs/newsletter.err.html（即 base_dir/template_dir/

input_error_template)。由第二行指定 newsletter.success.html 文件。当脚本成功地完成它的任务后，显示这一文件作为对用户的确认。

到此文件的完整路径是/www/nitec/public/htdocs/newsletter.success.html（即 base_dir/template_dir/success_template）。

在第三行中指定 newsletter.failed.html 文件，当发生内部错误（如由于权限设置错误而导致发送电子邮件或写文件错）时，它被显示出来。

到此文件的完整路径是/www/nitec/public/htdocs/newsletter.failed.html（即 base_dir/template_dir/failure_template）。

下一步，告诉脚本使用下面几行命令将输出写到一个文件中，并且用电子邮件发送数据：

```
output_method=file,email
mail_to=newsletter@nitec.com
mail_from=signup@nitec.com
mail_subject=New-registration
mail_daemon=/usr/sbin/sendmail
```

第一行告诉脚本去写数据文件，并将数据用电子邮件发出。第二行指定将数据发送到目的 e-mail 地址。第三行指定程序用来发送电子邮件的发送端地址。第四行指定电子邮件的主题。第五行指定邮件守护进程的完整路径。注意，程序的这一版本只支持这一广泛使用的 Sendmail 邮件守护进程。如果使用不同的邮件服务器程序，就需要对这个 CGI 程序作某些修改。这个脚本允许创建记录访问者每次成功的数据录入操作的日志。可以指定日志文件的名称如下：

```
Log_file=newsletter.log
```

日志文件被写入 data_dir 目录中。因此，可以使用 base_dir/data_dir/log_file 来建立它的完整路径名。

这个脚本还允许定制成功模板（success_template）。它使用定制标记定义来找到并替代数据值。这些定制标记都以 VISITOR 前缀开头。例如：

```
VISITOR-LAST=lastName
```

在配置文件中设置这一项表示该脚本会在成功消息模板（由 success_template 设置）中找寻称为<VISITOR-LAST>的定制标记，并用输入变量 lastName 的值替代它。注意，在配置文件和实际的 HTML 消息文件中，VISITOR-变量名必须用大写字母写出。

带有注释行（以#符号开头标的行）的最终配置文件如列表 8-4 所示。

列表 8-4：signup.conf

```
#
# signup.conf
#
# Purpose: this file is used by the general purpose
#        form processor script.
#
# Application: Newsletter sign-up tool example.
```

```
# $Author$
# $Id$
# $Version$
# $Status$
##################################################################

# base and template diretory
base_dir=/www/nitec/public
template_dir = htdocs

# input requirements
input_fields=firstName,lastName,emailAddress,companyName,
officePhone
required_fields=emailAddress,firstName,lastName

# data
data_dir=data
data_file=newsletter.cvs
field_separator,
saving_order=emailAddress,firstName,lastName,companyName,
office Phone

# response template
input_error_template = newsletter.err.html
success_template = newsletter.success.html
failure_template = newsletter.failed.html

# output
output_method=file,email
mail_to=newsletter@nitec.com
mail_from=signup@nitec.com
mail_subject=New-registration
mail_daemon=/usr/sbin/sendmail

# Logging
log_file=newsletter.log

# common variables
VISITOR-LAST=lastName
VISITOR-FIRST=firstName
VISITOR-EMAIL=emailAddress
VISITOR-HOMEADDR1=homeAddress1
VISITOR-HOMEADDR2=homeAddress2
VISITOR-HOMEPHONE=homePhone
VISITOR-OFFICEPHONE=officePhone
```

将此文件和脚本保存在同一目录中（即 Web 服务器的 cgi-bin 目录中）。然后创建如列表 8-5 所示的 HTML 表单。

列表 8-5：newsletter.html

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

```
<HTML>
<HEAD><TITLE> On-line Newsletter Sign-up Form </TITLE>
</HEAD>

<BODY BGCOLOR="white">
<TABLE BORDER=1>
<TR BGCOLOR="#abcdef">
<TD>

<FORM ACTION="/cgi-bin/wizform.cgi" METHOD="POST">

<TABLE BORDER=0 CELLPADDING=6 CELLSPACING=0>

<TR> <TD BGCOLOR="#abcdef" ALIGN="LEFT" COLSPAN="2">
    <FONT FACE="Arial Bold"> Newsletter Sign-up Form
</FONT><BR>
    </TD>
</TR>

<TR> <TD BGCOLOR="#abcdef" ALIGN="RIGHT"> First Name * </TD>
    <TD BGCOLOR="#aabbcc"> <INPUT TYPE="TEXT" NAME
="firstName" SIZE="20" MAXSIZE="30" > </TD>
</TR>

<TR> <TD BGCOLOR="#abcdef" ALIGN="RIGHT"> Last Name * </TD>
     <TD BGCOLOR="#aabbcc"> <INPUT TYPE="TEXT" NAME
="lastName" SIZE="20" MAXSIZE="30" > </TD>
</TR>

<TR> <TD BGCOLOR="#abcdef" ALIGN="RIGHT"> Email * </TD>
     <TD BGCOLOR="#aabbcc"> <INPUT TYPE="TEXT"
NAME="emailAddress" SIZE="20" MAXSIZE="30" > </TD>
</TR>

<TR> <TD BGCOLOR="#abcdef" ALIGN="RIGHT"> Company </TD>
     <TD BGCOLOR="#aabbcc"> <INPUT TYPE="TEXT"
NAME="companyName" SIZE="20" MAXSIZE="20" > </TD>
</TR>

<TR> <TD BGCOLOR="#abcdef" ALIGN="RIGHT"> Office Phone </TD>
     <TD BGCOLOR="#aabbcc"> <INPUT TYPE="TEXT"
```

```
NAME="officePhone" SIZE="20" MAXSIZE="15" > </TD>
</TR>

<TR> <TD BGCOLOR="#abcdef" ALIGN="RIGHT">   </TD>
     <TD BGCOLOR="#aabbcc"> <INPUT TYPE="RESET" VALUE="Clear
All">    <INPUT TYPE="SUBMIT" VALUE="Submit"> </TD>
</TR>

<TR> <TD BGCOLOR="#abcdef" ALIGN="LEFT" COLSPAN="2">
    <FONT SIZE="-1"> * Required field. </FONT>
    </TD>
</TR>

</TABLE>

<INPUT TYPE=HIDDEN NAME="conf_file" VALUE="newsletter.conf">

</FORM>

</TD>
<TD BGCOLOR="abcdef" VALIGN="TOP">
<FONT SIZE="+2">w</FONT>e send out a monthly newsletter to all
on-line subscribers. If you would like to subscribe, please
sign up here.
</TD>
</TR>

</TABLE>
</BODY>
</HTML>
```

在这一 HTML 表单中有些内容值得注意。首先要注意的是下面这一行：

```
<FORM ACTION="/cgi-bin/wizform.cgi" METHOD="POST">
```

在这里，表单的 ACTION 被赋值为存储在 cgi-bin 目录中的 wizform.cgi 脚本。方法是 POST。注意，所有数据录入字段都使用在配置文件中指定的同样的名称（如 lastName 或 firstName）。这是该脚本的要求。如果字段名称不一样，脚本就不会发挥预期的作用。

另外需要注意如下的隐藏输入字段：

```
<INPUT TYPE=HIDDEN  NAME="conf_file"  VALUE="newsletter.conf">
```

这样就指定了配置文件名。

在创建了表单后，将它存储在 htdocs 目录下。下面的这些列表，列表 8-6、列表 8-7 和列表 8-8，显示可以创建简单的 HTML 页来定制错误消息页、确认（成功消息）页和内

部错误消息页。这都是些简单的 HTML 页。newsletter.success.html 页用到一些 VISITOR 标记，使此页有定制的感觉。

列表 8-6：newsletter.success.html

```
<HTML>
<HEAD>
<TITLE> On-line Newsletter Sign-up Form (Success) </TITLE>
</HEAD>

<BODY BGCOLOR="#abcdef">
<BLOCKQUOTE>

<HR>
<FONT FACE="Arial Bold">
Hello <VISITOR-FIRST><VISITOR-LAST><BR><BR>
</FONT>

Thank you for subscribing our newsletter. We will send you the
latest copy of our newsletter
via the email address (<STRONG><VISITOR-EMAIL></STRONG>) you
provided here. Have a nice day.
<BR>

<FORM ACTION="/" METHOD="GET"> <INPUT TYPE=SUBMIT  VALUE
="Return to Index"> </FORM>
<HR>
</BLOCKQUOTE>
</BODY>
</HTML>
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

当访问者忘了提交所有必需的数据时，就会显示 newsletter.err.html 文件。定制标记 **<ERROR-LIST>**用访问者忘了提交的所有字段名代替。

列表 8-7：newsletter.err.html

```
<HTML>
<HEAD>
<TITLE> On-line Newsletter Sign-up Form (Error) </TITLE>

</HEAD>

<BODY BGCOLOR="#abcdef">
<BLOCKQUOTE>
<FONT FACE="Arial Bold" SIZE="+1">Incomplete Subscription
Request</FONT><BR>
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

```
<HR>

Your newsletter subscription request could not be processed
since one or  more of the listed
information are missing.
<BR>
<ERROR-LIST>
<BR>
Please try again.
<BR>

<FORM ACTION="/newsletter.html" METHOD="GET"> <INPUT
TYPE=SUBMIT  VALUE="Try Again"> </FORM>
</BLOCKQUOTE>
</BODY>
</HTML>
```

当脚本发现文件访问出问题时，如由于文件权限设置问题导致写数据文件失败，会显示 newsletter.failure.html。

列表 8-8：newsletter.failure.html

```
<HTML>
<HEAD>
<TITLE> On-line Newsletter Sign-up Form (Internal Error)
</TITLE>
 .
</HEAD>

<BODY BGCOLOR="#abcdef">
<BLOCKQUOTE>
<FONT FACE="Arial Bold" SIZE="+1">Internal Subscription
Error</FONT><BR>
<HR>
Sorry, we are unable to process your subscription request at
this time.
<BR>
<BR>

<FORM ACTION="/ newsletter.html" METHOD="GET"> <INPUT
TYPE=SUBMIT  VALUE="Try Again"> </FORM>
</BLOCKQUOTE>
</BODY>
</HTML>
```

将这些页（模板）保存在配置文件中由 template_dir 变量指定的目录中。现在，确保

将脚本和 CGI.pm 软件包文件（包含在 CD-ROM 中）保存在 cgi-bin 目录中，并且正确设置权限使 Apache 能读取和执行这些文件。

8.4.3.1 测试 newsletter 签约工具

现在来测试 newsletter 签约工具。图 8-5 显示了 newsletter.html 在 Web 浏览器中看到的样子。一旦填好这一页并由访问者提交，这个脚本就会运行。首先，它检查在配置文件所要求的任何输入字段是否丢失。

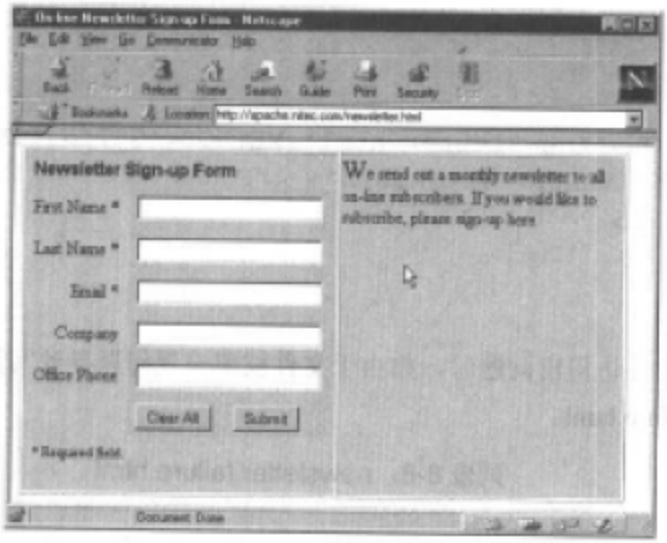

图 8-5 newsletter.html 页

如果丢失了任何必要的字段，它就显示如图 8-6 所示的错误页。

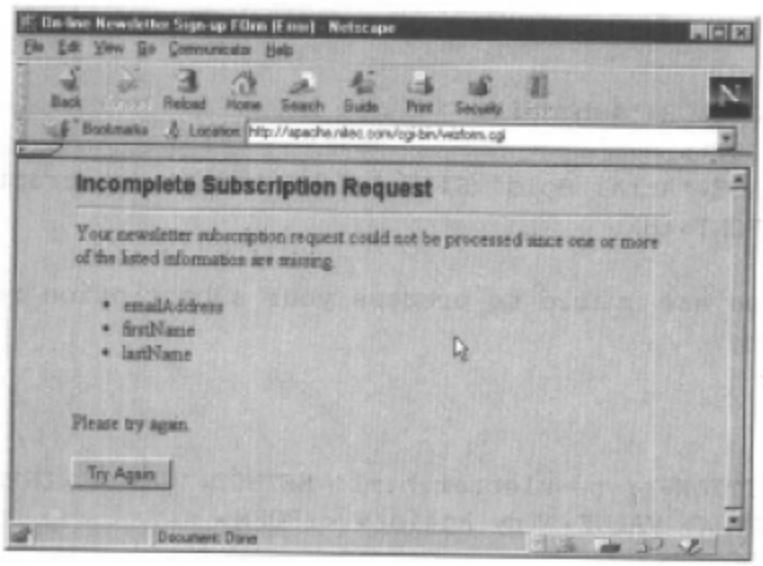

图 8-6 输入错误

否则，就显示如图 8-7 所示的确认页。

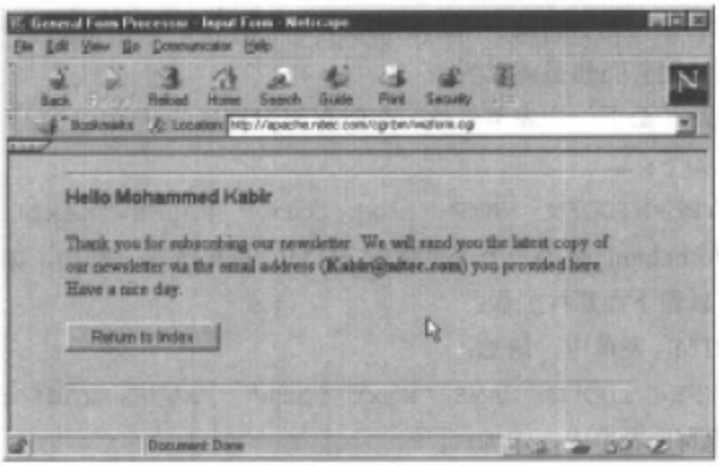

图 8-7　确认页

在提交若干业务通信的请求后，数据文件 newsletter.cvs 看起来像下面这样：

```
kabir@nitec.com,Mohammed,Kabir,Nitec,555-1111
joe@gunchy.org,Joe,Gunchy,Gunchy Inc,555-2222
someone@somewhere.net,First,Last,Company Name,555-3333
```

newsletter.log 文件看起来像下面这样：

```
11:26:00 4/1/98, apache.nitec.com, 206.171.50.51, Mozilla/4.04
[en]
(WinNT; I)
11:27:00 4/1/98, apache.nitec.com, 206.171.50.51, Mozilla/4.04
[en]
(WinNT; I)
11:30:35 4/13/98, apache.nitec.com, 206.171.50.51, Mozilla/4.0
(compatible; MSIE 4.01; Windows NT)
```

正如所看到的，所有都工作得十分正常。如果在写数据文件时遇到权限问题，那么失败消息就如图 8-8 这样显示。

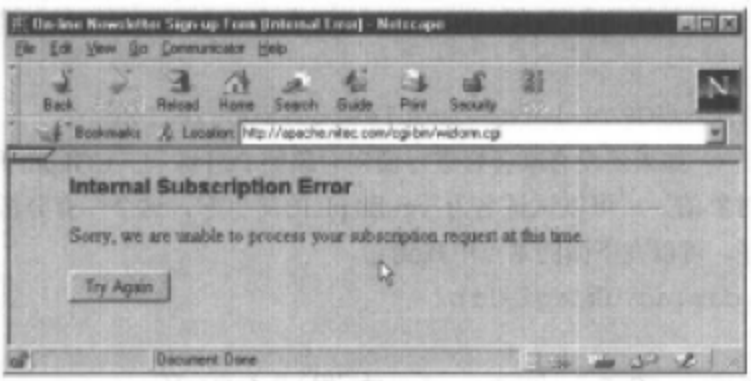

图 8-8　失败消息

8.4.3.2　修改业务通信签约工具

现在，可能想知道能否将此脚本用在多表单的应用中，其中有多于一个 HTML 表单需要访问者填写。回答当然是可以的。

所要做的是除了最后一个表单以外，在所有的 HTML 表单中添加一个新的隐藏变量。隐藏变量的形式如下：

```
<INPUT TYPE=HIDDEN  NAME="next_form"  VALUE="nextform.html">
```

这里，nextform.html 是第一个表单后的 HTML 表单等等。例如，如果一个应用程序有三个表单，可以像下面那样去做：

在第一个 HTML 表单中，增加：

```
<INPUT TYPE=HIDDEN  NAME="next_form"  VALUE="2ndform.html">
```

在第二个 HTML 表单中，增加：

```
<INPUT TYPE=HIDDEN  NAME="next_form"  VALUE="lastform.html">
```

在最后的表单中不需要隐藏变量。注意，需要将所有表单都放在 template_dir 指定的目录中。通过使用通用表单处理程序，可以为 Web 站点创建各种常用的应用程序。此类应用程序的例子有：FeedBack Script、GuestBook 和 Mailing List Sign-up Script 等等。

8.4.4 用于 Perl 的广泛使用的 CGI 模块

在前面的表单生成器（form generator）脚本中，使用了由 Lincoln D. Stein 创建的被广泛使用的 CGI.pm 软件包。其他软件包也能用来做快速的基于 Perl 的 CGI 脚本开发。现在看一些用于 Perl 语言的流行的的开发软件包：

♦CGI.pm——这是很好的 Perl 软件包。它可以用来快速和干净利落地开发 CGI 脚本。它能创建可以访问 CGI 查询字符串的 CGI 对象，创建相应的头标、读/写 cookie，以及在此脚本中实际上不用任何 HTML 写出 HTML 表单。可以在下面的站点找到该软件包：

www-genome.wi.mit.edu/ftp/pub/software/WWW/cgi_docs.html

♦Libwww——如果你对开发功能强大的用户代理、机器人、迷你 Web 服务器感兴趣，那么别为代码需要从头写起而感到烦恼。可以使用 Libwww 的 8 个模块 HTML、LWP、MIME、WWW、HTTP、URL、File 和 Font 来做所有的事。可以在下面的站点找到该软件包：

www.perl.com/CPAN-local/modules/by-module/LWP/

♦Cgi-lib.pl——如果还没有准备好进行面向对象的 Perl 编程（CGI.pm 和 libwww 两个模块都是面向对象的）。可以试试名为 cgi-lib.pl 的简单库。这个库有分析表单输入和创建头标的子程序。可以在下面的站点中找到它。

www.bio.cam.ac.uk/cgi-lib/

8.5 在 Apache 中调试 CGI 程序

当写了或安装了其他人的 CGI 程序时，它有可能不正常运行。调试 CGI 程序通常是困难的，因为在很多情况下不能从命令行运行该程序并给它提供输入。

Tip 如果在基于 Perl 的 CGI 脚本中使用 CGI.pm 软件包，就可以从命令行进行大量的调试工作。这个软件包允许以交互模式工作，在这里可以键入 key=value 输入对，就像这些数据来自 HTML 表单。

为帮助 CGI 开发人员，Apache 有用于 CGI 输出的日志。对于每个 CGI 程序错误，日志文件包含相应的由若干行组成的日志条目。前面两行包含请求的时间、请求的 URI、HTTP 状态、CGI 程序名等等。如果 CGI 程序不能运行，另外两个附加行包含有关错误的信息。如果错误是由于脚本返回不正确头标信息引起的，那么这些信息就被记录如下：所有的 HTTP 请求头标，所有头标由 CGI 程序及其 STDIN 和 STDOUT 输出。注意，如果这个脚本不能输出任何内容，将不包括 STDOUT 在内。

要在 Apache 中记录 CGI 程序输出，可以使用下面这些命令。它们可以在 mod_cgi 模块（这是标准发行版的一部分）中找到。

8.5.1 ScripLog

语　　法：ScriptLog filename
缺 省 值：none
使用范围：资源配置

这个命令为 CGI 程序错误设置日志文件名。如果日志文件名用的是相对路径（即不以斜杠/开头），那它就相对于由 ServerRoot 命令设置的服务器的根目录。

当使用这个命令时，要确保由 UserDir 命令指定的用户可以写日志目录。注意，就效率或性能而言，在日常基础上使用这个命令并不是好主意。建议当需要时才使用它，而且当测试完成后就关闭它。

8.5.2 ScripLogLength

语　　法：ScriptLogLength size
缺 省 值：10385760
使用范围：资源配置

这个命令限制由 ScriptLog 命令指定的日志文件的容量。该脚本日志文件对每个 CGI 错误记录大量的信息，从而能迅速增长。通过使用这个命令，就能限制日志文件。一旦这个文件达到最大长度，就不让它记录更多的信息。

8.5.3 ScriptLogBuffer

语　　法：ScriptLogBuffer size
缺 省 值：1024
使用范围：资源配置

这个命令限制被记录的 POST 或 PUT 数据的容量。

通过使用这些命令，可以设置 CGI 程序的日志。

8.6 小　　结

在本章中讨论了 CGI。学习了 CGI 如何利用不同的 HTTP 请求方法，如 GET 和 POST；如何配置 Apache 来为 Web 站点建立安全的 CGI 环境；在多用户系统上怎样为用户提供被频繁请求的 cgi-bin 访问。

本章提供了能灵活配置的 CGI 程序的例子，并说明如何用这个 CGI 程序生成有用的 CGI 应用程序。还讨论了如何配置 Apache 来支持 CGI 程序的调试。这是 CGI 开发人员极其需要的。下面，将讨论如何使用 FastCGI 来提高 CGI 脚本的运行速度。

第 9 章　FastCGI

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

本章内容：

- 什么是 FastCGI
- 为 FastCGI 设置 Apache
- 为 FastCGI 配置 Perl

若干年来，通用网关接口（CGI）已成为服务器端应用程序开发的事实标准。随着时间的推移，发现在沉重的 CGI 负担下许多 Web 服务器系统的表现不尽人意。研究表明 CGI 规范存在瓶颈问题：每当客户系统请求 CGI 应用程序时，Web 服务器就必须启动新的 CGI 进程，然后进程完成它的任务并结束。当负载低时，它会工作得很好。然而，在高负载时，进程设置和初始化所需要的时间就成为性能的瓶颈。

新的标准现在脱颖而出以弥补这种不足。尽管仍处在接受的早期阶段，但 FastCGI 是很有前途的开放标准。在本章中，将讨论标准的细节以及如何在 Apache 中实现 FastCGI。

9.1　作为新选择的 FastCGI

简单说，FastCGI 应用程序就像服务器应用程序一样进行操作。不像 CGI 应用程序，它始终处于活动状态并为来自 Web 服务器的请求提供服务。这正是 FastCGI 的优点所在。不同于 CGI，在每次处理请求时，它没有启动新进程和做应用程序初始化（例如与数据库连接）方面的开销。可以使用各种不同的语言，包括 Perl、C、C++、Java 和 Python，编写 Web 服务器应用程序。这是 CGI 的高性能的代用方案。

应该注意：FastCGI 是有发展前途的开放标准，并且有可能在不久的将来作为 Web 标准被世人所接受。然而，一些主要服务器销售商并不接受 FastCGI 的概念。他们有自己的标准，这些标准允许使用某些类似的功能和大量专用的服务器属性。它们可以归属于服务器应用程序编程接口（API）。例如，主要的服务器销售商（如 Netscape 和 Microsoft）都有他们自己的服务器 API：分别称为 NSAPI 和 ISAPI。甚至 Apache 也有它自己的 API。

Note

如果对有关 FastCGI 的最新消息感兴趣，可以订阅邮递表 fastcgi-developers@openmarket.com。要注意的是，这种邮递表由 FastCGI 开发人员使用，可能不适合询问初级的问题。要加入此邮递表，用包含单词 subscribe 的消息体向 fastcgi-developers-request@openmarket.com 发送一则消息就行了。发送给这个邮递表的邮件会被归档，而且可以在 WWW 上的 www.fastcgi.com/mail/使用它。

CGI、FastCGI 和 ServerAPI 的并存给开发人员和服务器管理员带来了许多麻烦。为能清楚说明这个隐晦的主题，表 9-1 提供了这些技术之间的关键特征比较。

表 9-1 比较 CGI、ServerAPI 和 FastCGI

特征	CGI	ServerAPI	FastCGI
编程语言独立性	独立于语言。CGI 应用程序几乎可以用任何编程语言编写	应用程序必须使用销售商 API 支持的语言（通常是 C/C++）编写	独立于语言。像 CGI 一样，FastCGI 应用程序能用任何编程语言编写
进程隔离	支持。应用程序作为独立的进程运行；带隐错的应用程序不会使 Web 服务器崩溃或访问服务器专用的内部状态	没有进程隔离。因为这些应用程序在此服务器的地址空间内运行，带隐错的应用程序会毁坏服务器内核和危及安全，而且服务器内核的隐错也能破坏应用程序	支持。带隐错的 FastCGI 应用程序不能使服务器内核或者其他应用程序毁坏或崩溃
标准类型	开放标准。一些 CGI 表单已经在每个 Web 服务器上实现	专有。在特定的 API 上开发应用程序，将使你的应用程序和特定经销商的服务器拴在一起	非专有。推荐的开放标准。在其他 Web 服务器上尚未得到广泛支持，包括 Microsoft 和 Netscape 的商业服务器。Apache 当前用第三方模块支持 FastCGI
平台独立性	独立于平台。CGI 没和任何特定的服务器结构（单线程的，多线程的等等）联系在一起	与服务器结构紧密联系。API 应用程序必须使用与此服务器相同的结构。如果 Web 服务器是多线程的，那么应用程序就必须是线程安全的。如果是单线程的，那么多线程应用程序就得不到任何性能优势	独立于平台。FastCGI 没有和任何特定的服务器结构联系在一起。任何 Web 服务器都能实现 FastCGI 接口
性能	为每个请求建立一个新的进程，并且当请求完成后丢弃这个进程；效率低	在服务器进程中运行应用程序，并且常驻于内存持续地服务于请求。不存在 CGI 启动或初始化问题	FastCGI 进程是常驻的；它们被重复用来服务于多个请求。不存在 CGI 启动或初始化问题
复杂性	易于理解	非常复杂。销售商的 API 的学习过程缓慢，并增加实施和维护费用	简单，能容易地从 CGI 转换到 FastCGI
分布式结构	不支持。要在远程系统上运行 CGI 应用程序，在那个系统上需要有 Web 服务器，因为 CGI 应用程序是由 Web 服务器运行的	取决于销售商	支持。FastCGI 应用程序能在任何支持 TCP/IP 的主机上使用

现在，你知道自己应该选择什么，但仍想试一试 FastCGI，这就对了。让我们看使用 FastCGI 的一些好处。

9.2　FastCGI 的效益

首先让我们考虑与 CGI 或服务器 API 相比使用 FastCGI 取得的效益。

9.2.1　通过缓存取得高性能

FastCGI 有多快？答案取决于应用程序。如果应用程序从文件读取数据，而且将这些数据缓存在内存中，那么这个应用程序的 FastCGI 版就会比 CGI 或基于 API 的 Web 服务器应用程序提供更好的性能。根据规范的要求，CGI 应用程序不能利用内存作缓存。因为在 CGI 中将为每个请求运行一次应用程序，并在处理请求任务后退出。与此类似，许多被广泛使用的基于 API 的 Web 服务器应用程序以不共享内存的子进程方式运行，从而也不能使用缓存。即使该模块的每个子进程都使用了缓存，它仍然效率很低。因为每个子进程必须在内存中有缓存的副本，因而浪费了大量的内存。

FastCGI 被设计成可以有效利用内存作缓存。来自任何子进程的请求被路由到 FastCGI 应用程序服务器。FastCGI 应用程序进程维护在内存中的缓存。注意，在某些情况下，单独的 FastCGI 应用程序服务器不会提供足够好的性能。通过使用多线程，可以运行被设计成同时处理若干请求的应用程序进程。这些处理并发请求的线程共享进程内存，所以它们都访问同一个缓存。

FastCGI 的开发人员通过与数据库服务器交换数据的应用程序的三种版本（基于 CGI、FastCGI 和基于流行的服务器 API 规范）做了一次测试。他们发现，当 FastCGI 版本的应用程序使用内存缓存，并与数据库服务器进行持续连接时，它的性能就大大超过其他两个版本的性能。

当基于 FastCGI 的应用程序不用内存缓存，并对基于 API 的应用程序使用持续连接时，基于 API 的应用程序的性能会比 FastCGI 版本稍微好些。这表示只有改变了游戏规则（当 FastCGI 的优点诸如内存缓存被禁止使用时），这时 API 版本才得以胜出。但是为什么要禁止使用缓存呢？换句话说，只要编写充分利用 FastCGI 特征的应用程序，那它就比 CGI 和 API 版的应用程序好。

测试结果表明：基于 FastCGI 的应用程序结构的优势会产生比 API 版本好三倍的性能。如果应用程序必须与远程资源相连接时，如远程数据库服务器，这个因素可能会更加突出。但是，测试结果同样表明：如果多线程 Web 服务器能为它的 API 应用程序线程维护缓存和保持持续连接，它可能比 FastCGI 应用程序取得更好的性能。这是由于在线程的环境下，没有进程间通信的开销。然而，开发多线程应用程序需要非常仔细的设计和编程，因为每个有故障的线程都可能导致整个 Web 服务器系统关闭。

另一方面，FastCGI 作为外部进程运行，利用了进程隔离模式的优点。这就为 Web 服务器系统提供了安全。万一出现有故障的 FastCGI 应用程序，Web 服务器仍能有效发挥作用。如果你十分喜欢多线程，没有它简直就不行的话，可以经常在多线程模式下编写 FastCGI 应用程序，同时仍利用进程隔离模式的优点。

9.2.2 通过分布式应用程序调节系统规模

不像 CGI 应用程序，FastCGI 应用程序不从进程环境表中获取 CGI 环境变量。而是在应用程序与 Web 服务器之间使用全双工连接，来传送环境信息、标准输入、输出和错误信息。这使 FastCGI 应用程序能在使用 TCP/IP 与 Web 服务器连接的远程机器上运行，如图 9-1 所示。

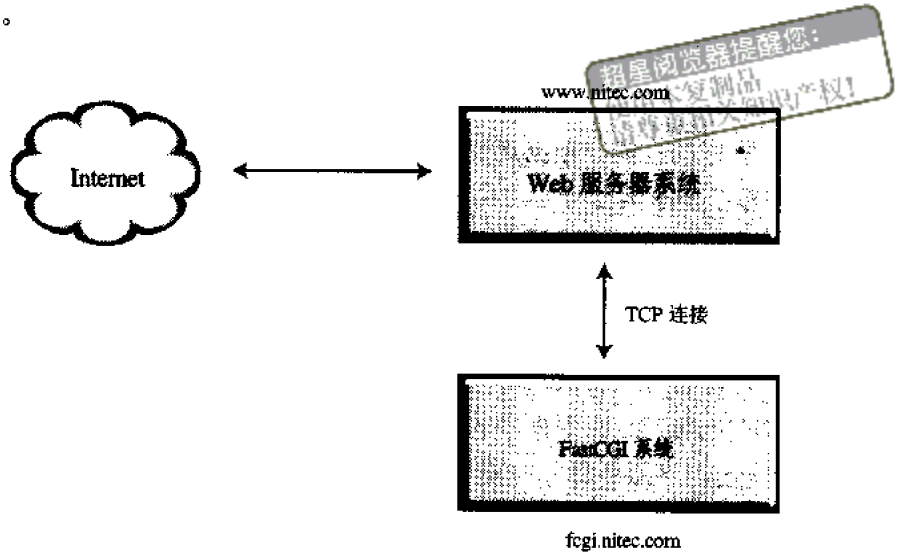

图 9-1 在远程机上运行的 FastCGI

当基于 CGI 和 API 的应用程序由于负载过重成为性能瓶颈时，典型的解决方案是：或者使用更强有力的 Web 服务器，或者使用多个 Web 服务器来运行它们。使用 FastCGI，这个问题可以以不同方式解决。FastCGI 应用程序能在网上专用的应用程序服务器上运行，因而减轻了 Web 服务器的负荷。使它能充分发挥它的长处——服务于 Web 请求。同时，FastCGI 应用程序服务器可以更有效地运行应用程序。Web 管理员也不必再为权衡同一台机器上的 Web 服务器和应用程序资源而操心。这将使 Web 服务器端以及应用程序端的配置更加灵活。

许多机构需要在他们的 Web 站点上提供数据库访问。由于受到 CGI 和销售商的 API 的限制，他们必须在 Web 服务器上的复制数据库的受限版本以提供该项服务。这给管理员带来大量的工作。通过使用远程 FastCGI，应用程序能在内部网络上运行，就简化了管理员的工作。当使用适当的防火墙配置和审查功能时，这种方法能提供安全的、高性能的、可调节规模的途径，将内部应用程序和数据送到 Internet 上。

Note

远程 FastCGI 连接有两个安全问题：认证和专用的问题。FastCGI 应用程序只接受来自它们可信任的 Web 服务器的连接（应用程序库包括对 IP 地址验证的支持）。这个规范的未来版本会包括对应用程序认证 Web 服务器的支持，以及对通过诸如 Secured Socket Layer（SSL）的安全协议进行远程连接的支持。

9.3　理解 FastCGI

FastCGI 应用程序使用单独的连接来与 Web 服务器通信。该连接用来将环境变量和标准输入（STDIN）数据传递给应用程序，而将标准输出和错误（STDOUT 和 STDERR）数据传递给 Web 服务器。

使用简单的通信协议允许将 FastCGI 应用程序驻留在与 Web 服务器不同的机器（或者多台不同机器）上。这使得系统规模的调节不局限于一台机器上，并且易于实现与现有系统的集成。对于本地的应用程序，服务器使用全双工管道来与 FastCGI 应用程序进程连接。对于远程应用程序，服务器则使用 TCP/IP 连接。

用在 Web 服务器和应用程序之间进行传输的 FastCGI 协议使用简单的数据包记录格式。许多应用程序开发员能使用 FastCGI 应用程序库，并不用为协议细节而费心。无论怎样，特定的应用程序能直接实现 FastCGI 协议。

由于 CGI 与 FastCGI 非常相似，让我们回顾 CGI 的请求过程。图 9-2 显示简化的 CGI 请求处理模式。

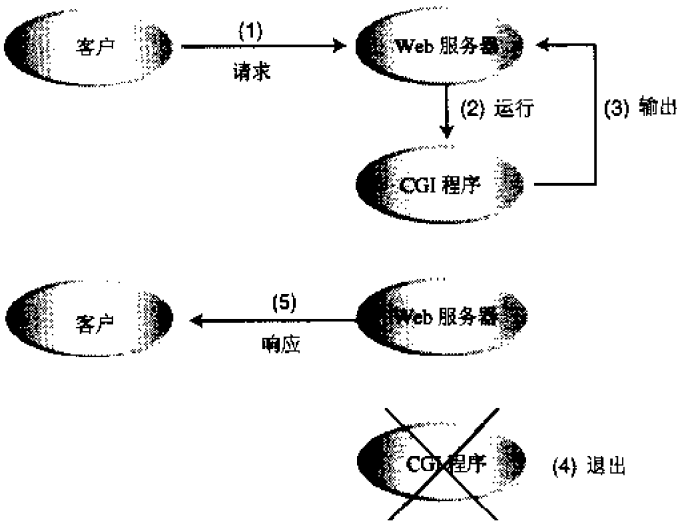

图 9-2　CGI 请求处理模式

对于每个 CGI 请求，Web 服务器都创建新的 CGI 进程，并且该进程对它自己进行初始化。Web 服务器使用环境变量将不同的与请求有关的信息传给该程序。根据请求方法（GET 或 POST）不同，用户数据或者被存储在名为 QUERY_STRING 的环境变量中，或者被放在进程的标准输入中。

CGI 应用程序完成它的任务，并将它的所有输出送到标准输出上，Web 服务器读取并分析它（不需要分析头标的应用程序除外）。然后 CGI 程序退出，并且服务器向客户返回 CGI 的输出。

FastCGI 进程是常驻的：它们完成请求后，等待另一个新的请求以取代已有的请求。如图 9-3 中所示。

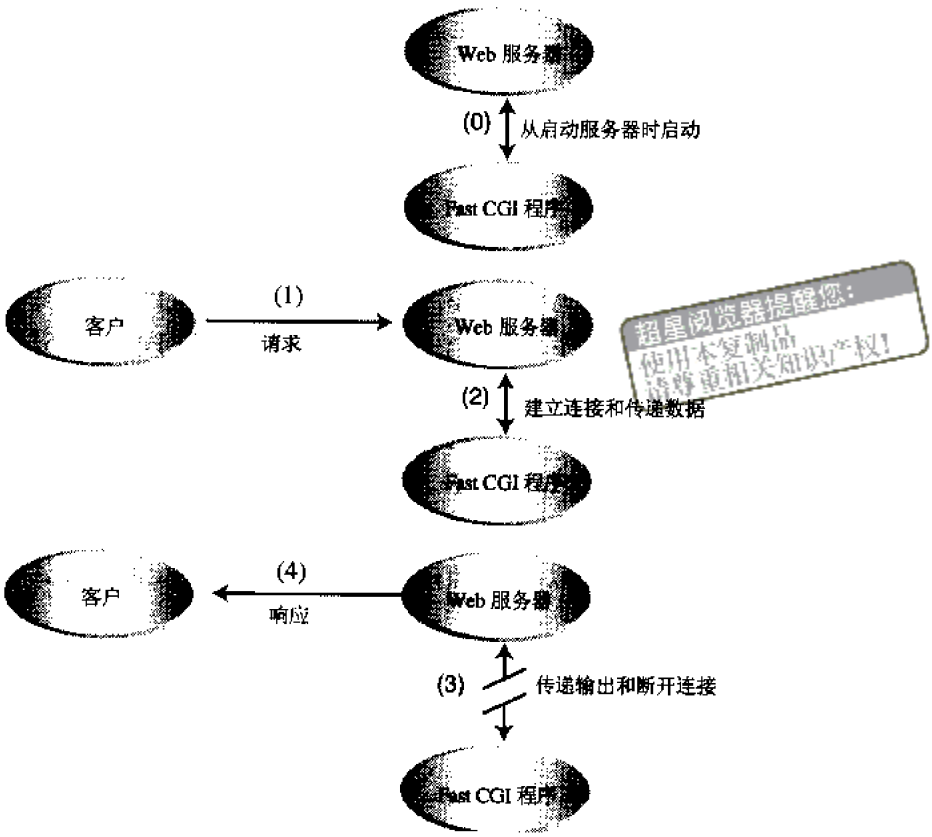

图 9-3 FastCGI 请求处理模式

Note

对不需要分析头标的应用程序，CGI 应用程序负责产生相应的 HTTP 头标。而在其他情况下，Web 服务器根据它在 STDOUT 中找到的输出内容类型，产生相应的 HTTP 头标。任何被写入 CGI 程序的标准错误的错误信息都会由 Web 服务器记录在案。

在单线程 FastCGI 应用程序中客户请求的处理过程如下：

1. Web 服务器创建 FastCGI 应用程序进程来处理请求。这些进程可以在启动服务器时或根据需要创建。

2. FastCGI 应用程序对它自己进行初始化，并等待来自 Web 服务器的新的连接。

3. 当客户请求到来时，Web 服务器打开与 FastCGI 进程的连接。这台服务器通过连接发送 CGI 环境变量信息和标准输入。

4. FastCGI 进程通过同样的连接，将标准输出和错误信息发送回服务器。

5. 当 FastCGI 进程关闭这个连接时，请求完成。然后 FastCGI 进程等待来自 Web 服务器的另外一次连接。

9.5 FastCGI 应用程序的不同类型

FastCGI 的另一个重要方面是它对应用程序的支持作用。不像 CGI 应用程序，FastCGI 应用程序是常驻的，因而它能实现基于 CGI 的应用程序不能实现的那些用途。让我们看一看 FastCGI 支持的应用程序的不同类型。

显然，FastCGI 应用程序能做 CGI 应用程序所能做的所有事情。因此典型的应用程序和 CGI 应用程序一样。现在具有 FastCGI 支持的新的应用程序类型是过滤程序和外部认证应用程序。

超星网络版权所有 使用本复制品 请尊重相关知识产权!

在被请求的文件返回给客户以前，可以创建 FastCGI 过滤程序来处理它。例如，假设需要为每个由 Web 服务器返回的 HTML（.html）页应用一组标准的头标和脚注。通过使用 FastCGI 过滤器程序，这是可能做到的。当对.html 文件的请求来到服务器时，它将此请求发送给负责添加标准的头标和脚注的 FastCGI 过滤程序。在给 HTML 页添加头标和脚注后，FastCGI 应用程序将结果返回给服务器，然后再由服务器传输给客户。

FastCGI 过滤程序可以通过缓存过滤的结果而大大提高性能（服务器在请求信息中提供修改时间，当服务器文件被修改完毕时，应用程序能刷新此缓存）。过滤程序在为具有嵌入 SQL 语句的 HTML 页开发分析程序（parser）、在数据传输过程中进行格式转换等方面非常有用。

利用 FastCGI 支持能开发另一种新类型的应用程序。这些应用程序包括外部认证程序和到第三方认证应用程序的网关。例如，如果使用外部数据库服务器来存储认证信息，如用户名、口令或其他特定权限数据，则可以创建 FastCGI 应用程序来保持与数据库服务器的持续连接并完成对认证访问请求的查询。使用 CGI 应用程序能做到这点吗？可以，但是 CGI 应用程序每次运行时都必须打开与数据库服务器的连接。从资源（CPU、网络）利用角度来说，它的代价太高。

从另一方面看，同一应用程序的 FastCGI 版会保持与数据库服务器的持续连接，完成查询，并返回基于查询结果的相应的 HTTP 状态代码。例如，当访问请求带有有效的用户名/口令对时，FastCGI 应用程序查询数据库服务器，核对是否允许使用这一用户名/口令对来访问被请求资源。如果数据库服务器返回一个指定值表明允许进行访问，FastCGI 应用程序将返回 “200 OK” HTTP 状态代码；当认证失败时，它将发送不同的 HTTP 状态代码，如“401 Unauthorized”。

你已经知道了 FastCGI 应用程序的类型，让我们看看如何将现有的 CGI 程序转换为 FastCGI 程序。

9.6 从 CGI 转换到 FastCGI

FastCGI 的另一个优点是从 CGI 转换到 FastCGI 的过程相当简单。在这一节中，将说明如何转换简单的（用 Perl 写的）CGI 应用程序 fontsize.cgi。见列表 9-1。

列表 9-1：fontsize.cgi

```
#!/usr/local/bin/perl

# Variables
my $MAX = 10;
my $i;

# Content Header
print "Content-type text/html\n\n";


# Main loop
for ($I=0; $I < $MAX; $i++){
print "<FONT SIZE=$i>FONT SIZE = $i</FONT><BR>";
}
exit 0;
```

这个 CGI 应用程序产生如图 9-5 所示的输出。

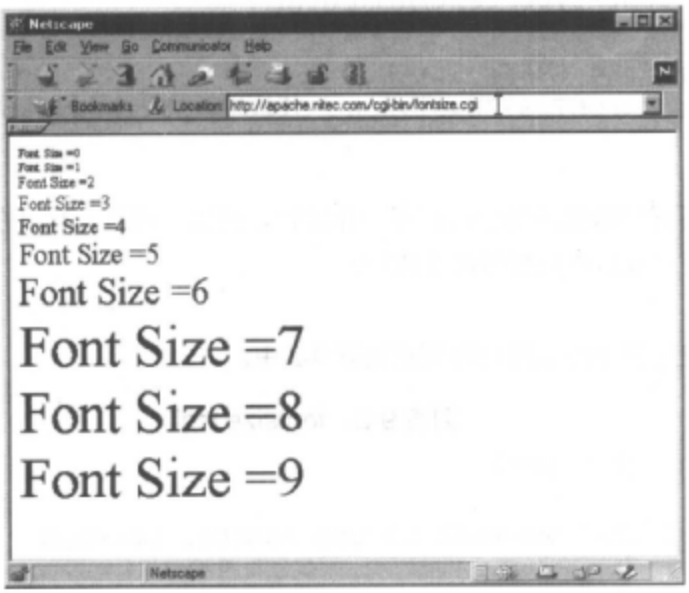

图 9-5　fontsize.cgi CGI 程序的输出

现在将它转换为 FastCGI 应用程序。为简单起见，假定在系统上已经安装了 Perl 的 FastCGI 版。阅读 CD-ROM 上的 FastCGI 开发人员工具集（FastCGI Developer Kit）文档，了解如何在系统上编译和安装 Perl 的 FastCGI 版。下一行告诉 shell 装入 Perl 解释程序：

```
#!/usr/local/bin/perl
```

如果对 Perl 解释程序的 FastCGI 版使用不同的名称或存放在不同位置，需要对这一行作相应的更改以指向 FastCGI-savvy Perl 解释程序。照原样保留它，这是因为在系统上用的是 Perl 解释程序的 FastCGI-savvy 版。

为了在 Perl 中使用 FastCGI 软件包，需要添加下面这一行：

```
Use  FCGI;
```

超星阅览器提醒您：
使用本复制品
须尊重知识产权！

下一步，确定 CGI 应用程序的初始化段。对于这个脚本，初始化段含有下列两个变量的说明：

```
my $MAX = 10;
my $i;
```

现在需要确定 CGI 应用程序的响应循环和主体代码段。因为每个请求都必须用相应的 Content-type 头标予以响应，所以主体代码段从下一行开始：

```
print "Content-type text/html\n\n";
```

用于实际处理的 for 循环是主体代码段的其余部分。现在主体代码段已经确定，需要将它放入响应循环中。这是使用下列代码做到的：

```
while(FCGIaccept() >=0) {
# body code goes here
     }
```

现在主体代码段看起来就像这样：

```
while(FCGI: :accept() >=0) {
print "Content-type text/html\n\n";

for ($i=0; $i < $MAX; $i++){
     print "<FONT SIZE=$i>Font Size = $i</FONT><BR>";
}
     }
```

FastCGI 应用程序像服务器应用程序一样运行，所以下面这一行只在服务器或管理员向 FastCGI 应用程序发送结束信号时才执行：

```
exit 0;
```

FastCGI 版应用程序的全部内容列在列表 9-2 中。

列表 9-2：fontsize.fcgi

```
#!/usr/local/bin/perl

# Tell Perl that we want to use FastCGI package
use FCGI;

# Variables
my $MAX = 10;
my $i;

# FastCGI Loop
while(FCGI::accept() >=0) {

# Content Header
print "Content-type text/html\n\n";
```

```
# Main Loop
for ($i=0; $i < $MAX; $i++){
     print "<FONT SIZE=$i>Font Size = $i</FONT><BR>";
}
     }

exit 0;
```

超星阅览器提醒您：
您此本复制品
请尊重作者版权！

为了使转换到 FastCGI 应用程序的工作变得容易，用 FastCGI 应用程序库（在 FastCGI Development Kit 中提供）建立的可执行程序可以作为 CGI 或者 FastCGI 程序运行，主要取决于如何调用它们。库程序检测运行环境，并根据检测结果自动选取 FastCGI 或常规 I/O 子程序。

当将 CGI 应用程序转换为 FastCGI 应用程序时，要记住：许多 CGI 应用程序并不试图完成任何内存管理操作。这是由于 CGI 应用程序执行后退出，而且在多数情况下，由操作系统恢复内存供其他用途使用。最重要的是，许多 CGI 应用程序甚至不关闭文件，在退出时将这一任务移交给操作系统去做。

在将这种类型的应用程序转换为 FastCGI 应用程序时，克服这一问题显得非常重要。记住：只要 Web 服务器或管理员不撤消这些 FastCGI 应用程序，它们就一直驻留在内存中。如果将可能造成内存漏损的 CGI 应用程序转换成 FastCGI ，而又没有任何人处理内存问题的话，那么 FastCGI 版的应用程序就可能随着时间的推移不断漏损内存，并最终导致内存资源匮乏。为避免周末办公室内长时间无人值守，提前检查有没有这类问题是个好主意。如果 CGI 应用程序非常复杂，而且从时间与精力的角度看，克服它（内存使用问题）代价又那么大，可以考虑使用另外一种可行的解决方案。

可以保持对有内存漏损的应用程序服务的请求次数进行计数，并编程撤消它。列表 9-3 表示简单的例子：它如何将处理请求的次数计数保存在简单的基于 C 的 FastCGI 应用程序中。

列表 9-3：memory_hog.c

```
#include <fcgi_stdio.h>
void main(void){
int maxRequests = 100;   // maximum num of request before
exiting.
int requestCount = 0;          // used to keep count of
requests Processed so far.

While(FCGI_Accept() > 0){       // start of response loop
/* body code */

printf("Content-type:  text/html\r\n");
printf("\r\n");
printf("Number of requests processed so far is %d.",
requestCount++);
```

```
/* some memory leaking code could go here */

/* end of application's reall processing code */

if(requestCount >= maxRequests) {
   /* clean-up code goes here */
   exit 0;
   }

/* end of body code */

}
/* termination or clean-up code (if any) */
exit(0);
}
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

正如从前面的源代码中所看到的，当处理了最大数量的请求时，FastCGI 应用程序就退出。

另外要注意的问题是：如果被转换的 CGI 应用程序含有可能干扰主体代码第二次运行的代码时，就必须对它进行改正。对此问题的解决方案比较简单，只要添加重新初始化某些变量、数组等等的代码就可以了。应用程序必须保证在处理一个请求的过程中，建立的任何状态都不会对以后的请求产生影响。

最后必须意识到，在 CGI 开发人员中将大的应用程序拆成一个个小程序是常有的事。这样做弥补了与 CGI 应用程序初始化有关的缺陷。使用 FastCGI，较好的方法是在单个可执行程序中完成有关的功能，以减少需要管理的进程数，而应用程序的各项功能之间又能充分利用共享缓存信息的优点。

FastCGI 协议的开发员提供了可自由使用的软件开发工具集，使 FastCGI 应用程序开发过程变得容易。这个工具集可以从 FastCGI 的 Web 站点上获得，它的 URL 是：

www.fastcgi.com/applibs

在 CD-ROM 中也可以找到它。以压缩的 tar 文件提供的工具集有助于用 C、C++、Perl、Tcl 和 Java 编写 FastCGI 应用程序。当解压缩和还原(untar)这个文件时，它创建 fcgi-devel-kit 目录。可以找到 index.html 文件，它提供有关工具集的详细情况。

9.7 用于 Apache 的 FastCGI

像许多其他功能一样，在 Apache 中通过添加模块来支持 FastCGI 功能。如果不打算在 Apache Web 服务器中添加 mod_fastcgi.c 模块，仍然可以使用称为 cgi-fcgi 的传统 CGI 应用程序运行 FastCGI 应用程序。在开发人员工具集中含有这一应用程序。

cgi-fcgi 程序使你能用 Apache 或任何支持 CGI 的 Web 服务器运行 FastCGI 应用程序。

它是一个用 UNIX 域（domain）或 TCP/IP socket 来与 FastCGI 应用程序通信的 CGI 应用程序。cgi-fcgi 程序采用侦听 socket 的路径名或主机/端口地址作为参数，并且与在这一 socket 上侦听的 FastCGI 应用程序连接。然后它将 CGI 环境变量和 STDIN 数据传递给 FastCGI 应用程序，并将来自 FastCGI 应用程序的 STDOUT 和 STDERR 数据传递给 Web 服务器。当 FastCGI 应用程序发出它的响应结束的信号时，cgi-fcgi 就刷新缓冲区并退出。

注意，使用 cgi-fcgi 并不具有支持 FastCGI 的 Apache 服务器一样的性能。因为需要为每个请求启动 CGI 应用程序（cgi-fcgi），每个请求都有 fork/exec 调用开销。它不能完成任何应用程序的管理工作，所以需要自己来完成它。然而，cgi-fcgi 的确允许开发在各次连接之间保持内存状态的应用程序，这通常提供超过普通 CGI 的性能。所有使用 cgi-fcgi 开发的应用程序都能在支持 FastCGI 的 Web 服务器上工作。文件 exmaples/tiny-fcgi.cgi 演示了使用 cgi-fcgi 来运行一般应用程序的方法，下面是运行 exmaples/tiny-fcgi 应用程序的情况：

```
#!/path/to/cgi-fcgi/cgi-fcgi -f -connect sockets /tiny-fcgi
tiny-fcgi
```

在大多数 UNIX 平台上，执行这一命令解释程序文件来运行 cgi-fcgi 需要要带有变元 -f 和 exmaples/tiny-fcgi.cgi。

在某些 UNIX 平台上，包括 HP-UX，命令解释程序文件的第一行的字符不得多于 32 个，包括换行字符。必须在标准位置上（如/usr/local/bin）安装 cgi-fcgi 应用程序，或者在含有应用程序的目录中建立到 cgi-fcgi 应用程序的符号连接。

如果想发挥 FastCGI 的性能，极力推荐使用 mod_fastcgi.c 模块取代 cgi-fcgi 应用程序。这个模块并不是标准 Apache 发行版的一部分。mod_fastcgi.c 模块提供下列命令。

9.7.1 AppClass 命令

语　法：AppClass <path to FastCGI application> [-prcesses N] [-listen-queue-depth N] [-restart-delay N] [-priority N] [-port N] [-socket sock-name] [-initial-env key=value]

使用范围：服务器配置

这个命令能直接启动 FastCGI 应用程序，例如：

```
AppClass /www/development/fcgi-devel-kit/echo.fcgi -port 9000
```

这个命令使 mod_fastcgi 装入/www/development/fcgi-devel-kit/echo.fcg FastCGI 应用程序。它将侦听端口 9000。当由这条命令装入的 FastCGI 应用程序停止工作时，mod_fastcgi 将重新启动这个应用程序，并在错误日志文件中写入一个记录条目。AppClass 命令的可选参数如下：

♦processes——此选项指定要建立多少个 FastCGI 进程。在高负载的情况下，装入同

一 FastCGI 应用程序的多个进程可以提供较好的性能。它的缺省值是 1。

♦listen-queue-depth——此选项指定用此命令装入的 FastCGI 应用程序的侦听队列的长度。在大多数情况下，它的缺省值就足够了。但是在高负载环境下，可以加长队列的长度。这样做能减少由于应用程序繁忙而拒绝请求的可能。然而，如果预计有高的负载，并且服务器能运行额外的 FastCGI 进程，可以用增加进程的数量来代替加长侦听队列的长度。它的缺省值是 5。

♦restart-delay——此选项指定用于控制延迟重新启动停止工作的 FastCGI 进程的秒数。这只在使用同一 FastCGI 应用程序的多个进程时才有用。在只有单个 FastCGI 应用程序情况下，它会被立即重启，而且此选项不起任何作用。其缺省值为 5。

也许对为什么需要延迟感到迷惑。FastCGI 应用程序一般不应该停止工作；如果它停止工作，那么它可能有什么毛病。在这种情况下，这个延迟至少允许服务器去完成其他有用的任务，不仅仅一次又一次地重新启动有错误的程序。

♦Priority——此选项设定 FastCGI 应用程序的进程优先级。缺省值使 FastCGI 应用程序拥有和 Apache 服务器本身一样的优先级。其他有关值是由操作系统的 setpriority 系统调用设定。在 RedHat Linux 系统中，这个 setpriority 系统调用允许使用从-20 到 20 之间的任意值。值越小，进程被调用的机会就越多。然而，mod_fastcgi 不允许使用负值，这表示不能设置一个比 Apache 服务器本身具有更高优先级的 FastCGI 进程。所以应该使用正整数，以降低应用程序的优先级。这个选项的缺省值是 0。

♦Port——此选项指定 FastCGI 应用程序侦听的 TCP 端口。因为低于 1024 的端口被用于标准服务，所以需要使用更高的端口号。使用此选项能从另一个系统访问这个应用程序。在缺省情况下，不提供任何环境变量。

♦Socket——此选项指定将由应用程序侦听的 UNIX 域 socket 的路径名。这个模块在由 FastCgiIpcDir 命令指定的目录中创建 socket。其缺省值是 Default Socket。

如果不提供端口和 socket 选项，这个模块将为应用程序本身创建 Unix 域 socket。

♦Initial-env——此选项可以被用于在 FastCGI 应用程序的环境表中插入（带有值的）环境变量。可以多次使用这个选项在应用程序环境表中插入多个 key=value 对。在缺省情况下，不需要提供任何环境变量。

注：-socket 和-port 选项是相互排斥的。Path-name 必须不同于提供给前面 AppClass 或 ExternalAppClass 命令的路径名。

9.7.2 ExternalAppClass 命令

语　　法：ExternalAppClass <FastCGI-appliaction-name>
[-host host:port] [-socket sock-name]

使用范围：服务器配置

当在另外一个系统上运行 FastCGI 应用程序时，使用这个命令。例如：

```
ExternalAppClass  echo.fcg  -host fcgi.nitec.com:9090
```

这将在名为 fcgi.nitec.com 的主机上运行 echo.fcg 应用程序，并侦听端口 9090。<FastCGI-application-name>是个标识符，它用来说明运行在远程主机上的应用程序。所以它可以是想要的任意名称。确保所选择的名称没有被用在另一条 AppClass 或 ExternalAppClass 命令中。ExternalAppClass 的可选参数如下：

♦host——这个选项能指定在另一系统上运行 FastCGI 应用程序的主机和 TCP 端口号。使用 host:port 格式来指定主机和端口。可以使用主机名，或者 IP 地址。

♦socket——这个选项能指定由 FastCGI 应用程序使用的 UNIX 域 socket 的路径名。

超星阅览器提醒：
使用本复制品
请尊重相关知识产权！

9.7.3 FastCgiIpcDir 命令

语　　法：FastCgiIpcDir path

缺 省 值：FastCgiIpcDir /tmp

使用范围：服务器配置

这个命令指定由此模块创建的 UNIX 域 socket 的缺省路径。只要没有建立 cron 作业定时清除/tmp 目录，那么缺省的/tmp 位置就是好的选择。Socket 名称有具有下列格式：

```
OM_WS_n.pid
```

其中 n 是个数，而 pid 是主 Apache 进程的进程 ID。如果用这一命令设置一个路径，需要确保只有 Apache 对这个目录有读和写的访问权限。

9.8 将 mod_fastcgi 编译到 Apache 中

在 Apache 中添加 FastCGI 支持是相当容易的。第一步是取得 Apache 最新的 FastCGI 模块。可以在下面的站点中找到 Apache 最新的 FastCGI 模块：

```
www.fastcgi.com
```

Note　在随本书一起提供的 CD-ROM 中也包括这个模块的最新版。

下一步需要配置 Apache 配置文件。将模块文件（mod_fastcgi.c）拷贝到 Apache 源目录树（source tree）的 src/modules 目录下的子目录中。我喜欢在 src/modules/standard 中保存所有模块。一旦将模块文件放在 Apache 源目录树下，将下面这一行添加到在 src 目录下的配置文件中：

```
AddModule modules/standard/mod_fastcgi.o
```

Note　如果在不同目录中保存这个模块，要确保用实际路径取代 modules/standard/。

现在已准备好编译带有 FastCGI 模块的 Apache。运行 Configure 脚本和 make 实用程序。应该在源目录中有一个新的 Apache 可执行程序（httpd）。为确保这个模块已用在新

的 Apache 可执行程序中，可以在源目录下运行 ./httpd -l 命令。如果看到 mod_fastcgi.c 列在命令的输出中，那么在 Apache 中已成功地添加了这个模块。

Tip 如果你的系统支持删除二进制文件的多余符号（stripping），可以使 Apache 的可执行程序容量变得小些。例如，在 RedHat Linux 系统中，可以用 strip httpd 命令来从 Apache 二进制中删除不必要的符号。

用新的执行程序代替现有的 httpd。为了配置 FastCGI 应用程序，需要组合使用 mod_fastcgi 的命令和由 Apache 服务器提供的其他命令。

使用 AppClass 命令来启动需要由 Web 服务器控制的 FastCGI 应用程序。这种情形下，服务器进行两方面的管理：①当受控进程停止工作时，记录错误消息；②试图重新启动那些停止工作的受控进程。

使用 AppClass 和 ExternalAppClass 命令中的一个或者它们两者来定义路径名与 FastCGI 应用程序的连接信息之间的联系（association）。连接信息或者是 UNIX 域 socket 的路径名，或者是 IP 地址和 TCP 端口的端口号。这两个命令之间不同之处在于：单独一个 AppClass 命令启动应用程序，而且建立与它通信的联系；而 ExternalAppClass 命令只定义联系。在 AppClass 情况下，用在联系中的路径名经常是这个应用程序的可执行程序的路径名；使用 ExternalAppClass，这个路径名是任意的。

对于由 mod_fastcgi 处理的 HTTP 请求，它的处理程序必须是 fastcgi-script 或者请求的 MIME 类型必须是 application/x-httpd-fcgi。Apache 提供几种设置处理程序和请求的 MIME 类型的方法：

♦SetHandler（在 Location 或 Directory 段或 per-directory 访问控制文件的使用范围内）将处理程序 fastcgi-script 和特定文件或目录中的所有文件联系在一起。

♦AddHandler 将处理程序 fastcgi-script 和基于文件扩展名的文件联系在一起。

♦ForceType（在 Location 或 Directory 段或.htaccess 文件的使用范围内）将 MIME 类型 application/x-httpd-fcgi 和特定文件或目录中的所有文件联系在一起。

♦AddType 将 MIME 类型 application/x-httpd-fcgi 和基于文件扩展名的文件联系在一起。

一旦配置了 mod_fastcgi 模块，它已准备好处理如下请求：

1. 提取与被请求的路径名有关的连接信息。如果没有任何与此路径名有关的连接信息，服务器就返回错误代码“404 Not Found.”。

2. mod_fastcgi 模块连接 FastCGI 应用程序进程。如果这个连接尝试失败，服务器就返回“500 Server Error.”。

3. 传输请求给 FastCGI 应用程序进程，这个进程产生相应的响应。

4. 接受应用程序的响应，并将它转换成 HTTP 响应。服务器将响应送给提出请求的客户。

9.9 配置文件例子

建议备份已有的 httpd.conf 文件，但不要立即使用这个例子配置文件。现为测试目的提供最小配置的 httpd.conf 文件，该文件如列表 9-4 所示。可使用这一配置测试 FastCGI 是否正常工作。如果证明配置正常，将这一配置文件中专门用于 FastCGI 的内容合并到你自己的配置文件中。

列表 9-4：httpd.conf 配置例子

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

```
# httpd.conf-minimal for mod_fastcgi
#
# One config file
#
ResourceConfig /dev/null
AccessConfig /dev/null

# The port number below is above 1024 because we don't want to
# confuse this server with some other standard service
#
Port 9999

# You should replace the User/Group directive with appropriate
# user/group name
User $HTTP_USER
Group $HTTP_GROUP

# Configure just one idle httpd child process, to simplify
# debugging
StartServers 1
MinSpareServers 1
MaxSpareServers 1
# Tell httpd where it should live, turn on access and error
# logging
#ServerRoot $APACHE
ErrorLog      logs/error.log
TransferLog   logs/access.log
ScoreBoardFile   logs/httpd.scoreboard

# Tell httpd where to get documents
#
#DocumentRoot $FASTCGI
# This is how you'd place the Unix-domain socket files in the
# logs directory (you'd probably want to create a subdirectory
# for them.)
# Don't do this until you've verified that everything works
```

```
# with the socket files stored locally, in /tmp!
# FastCgiIpcDir $APACHE/logs
# Start the echo app
#
AppClass $FASTCGI/examples/echo -initial -env SOMETHING=NOTHING

# Have mod_fastcgi handle requests for the echo app #
# (otherwise the server will return the app's binary as a file!)
#
<Location /examples/echo>
SetHandler fastcgi-script
</Location>

# Start a FastCGI application that's accessible from other
# machines
AppClass  $FastCGI/examples/echo.fcg -port 8978
<Location /example/echo.fcg>
SetHandler fastcgi-script
</Location>

# Connect to "remote" app started above. Since the app is
# actually local, communication will take place using TCP
# loopback.
# To test true remote operation, start one copy of this # Web
# server on one machine, then start another copy with
# "localhost" in the line below changed to the host name of the
# first machine.
#
# ExternalAppClass remote-echo -host localhost: 8978
<Location /examples/remote-echo>
SetHandler fastcgi-script
</Location>

# This is how you'd have mod_fastcgi handle any request for a file
# whose name ends in .fcg:
# AddHandler fastcgi-script fcg
# End of httpd.conf
```

超星阅览器提醒您：使用本复制品请尊重相关知识产权！

确保已建立了包含 mod_fastcgi 模块的新的 httpd，并正确地建立了 FastCGI 开发人员工具集（FastCGI Developer's Kit）。其中可利用随 FastCGI 开发人员工具集一起提供的、在 example 目录中的 echo 程序。在把列表 9-4 中的新 httpd.conf 文件放入$APACHE/conf 目录后，不要忘记重新启动 Apache。可以使用浏览器访问下面的 URL：

http: //$YOUR_HOST:9999/examples/echo

其中，$YOUR_HOST 是运行 httpd 的主机的 IP 地址。在 echo 显示的初始环境中寻找 SOMETHING=NOTHING。请求计数器在每次重载此页时加 1。在使用这个配置前，需

要做一些如表 9-2 中所示的替换。

表 9-2　例子配置的替代项

关键字	替代为
$APACHE	含有 Apache 的目录的路径名
$FASTCGI	含有 FastCGI 开发工具集的目录路径名
$HTTP_USER	用于 User 命令的用户名称
$HTTP_GROUP	用于 Group 命令的组名称

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

9.10　小　　结

在本章中，讨论了在没有引入专用 API 的开销与复杂性的情况下，如何使用 FastCGI 解决 CGI 所固有的性能问题。FastCGI 是快速的、开放的和可维护的。它提供诸如内存缓存、持续连接和分布式结构等特性。从 CGI 到 FastCGI 的转换相当简单。然而，它仍然只是建议中的开放标准，至今还未被广泛使用。

在下一章中，将讨论基本的 HTTP 认证。

第三部分　安全机制

第 10 章　基本认证
第 11 章　服务器状态和日志
第 12 章　Web 服务器的安全

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

第 10 章　基本认证

本章内容：

- 什么是基本的 HTTP 认证
- 基本 HTTP 认证的安全程度
- 如何限制特定主机访问你的网页
- 如何使访问需要用户名和/或口令
- 如何使用外部认证程序
- 什么是基于主机的和摘要认证

对一般客户/服务器系统讲，当客户登录到网络上后，在客户和服务器之间的对话过程将一直维持，直到用户注销登录为止。对 Web 来说，典型情况是：客户端是浏览器，而在服务器端则运行着 Web 服务器软件。然而，由于超文本传输协议（HTTP）是无状态协议，因此在 Web 服务器（如 Apache）和 Web 浏览器之间并不始终维持对话过程。只要 Web 服务器完成对一个 URL 请求的服务，连接就断开了。虽然 Apache 能够使用 Keep-Alive 功能来维持连接的打开状态，继续为以后的请求提供服务，但并不是所有的浏览器都支持 Keep-Alive 功能。

这一功能也不能用来维持认证的对话过程。换言之，在 Web 上保证可以使用的唯一认证机制是由 HTTP 本身提供的。所有其他的认证机制都是针对特定的服务器的。也就是说，其他的认证机制或者是基于主机的，或者要求特殊的编程才能实现。在标准的 Apache 服务器中实现了这样的认证模式，它能控制哪些主机可能访问特定的站点或特定的站点的一部分。

然而，基于主机的认证机制在许多方面有限制。当用户通过代理服务器或防火墙访问 Web 时，此时企图进行用户认证根本就是不可能的。因为 Apache 服务器根本就得不到用户主机的 IP 地址，它当然不能区分用户。互联网上的大多数用户从它们的服务提供商（ISP）那里动态地获得 IP 地址，因此基于主机的认证方式并不总是适用。在大多数情况

下，传统的基于用户名/口令的认证方式更为可取。基本的 HTTP 认证就提供了这种功能。

Web 对登录认证的支持很弱，因为 HTTP 只为认证访问提供极少的支持。如果对高度安全的认证机制感兴趣，可以考虑基于安全 Socket 层（Secured Socket Layer，SSL）的认证模式。除了基于 SSL 的认证外，目前几乎所有其他基于 Web 的认证模式均可认为是基本认证。

本章将讨论两种基本的认证模式：基于主机的认证和基本 HTTP 认证。

使用本复制品请尊重相关知识产权！

10.1 基于主机的认证过程

在这种认证模式中，访问是用主机名或主机 IP 地址来控制的。当 Web 服务器收到对某种资源的请求，它就检查提出请求的主机是否有权访问请求的资源，并根据检查结果采取相应的行动。

标准的 Apache 发行版本包括称为 mod_access 的模块，它能够完成基于 Web 客户主机名的访问控制。主机名可以是全称域名（FQDN），如 blackhole.nitec.com，也可以是 IP 地址，如 206.171.50.50。该模块用以下几种 Apache 命令来提供访问控制功能。这些命令是：allow，deny，order，allow from env=variable 和 deny from env=variable。

10.1.1 allow 命令

语　　法：`allow from host1 host2 host3...`

使用范围：目录，位置，针对目录的访问控制文件

覆　　盖：`Limit`

该命令允许定义可以访问某个目录的主机清单（包括一个或几个主机名或 IP 地址）。当指定一个或几个主机名或 IP 地址时，主机名或 IP 地址之间应该用空格分开。表 10-1 表示该命令可能取的值。

表 10-1　allow 命令可能取的值

值	例　子	说　　明
All	allow from all	该保留字允许所有的主机访问，本例中演示了如何使用该选项
主机的全称域名（FQDN）	allow from wormhole.nilec.com	仅允许拥有指定的 FQDN 的主机访问。本例中仅允许名为 wormhole.nitec.com 的主机访问。注意这里要匹配整个主机名；很明显 toys.com 并不和 etoys.com 匹配
主机的部分域名	allow from .mainoffice.nitec.com	仅允许与部分域名匹配的所有主机进行访问。本例中允许在.mainoffice.nitec.com 网络上的所有主机进行访问；而 developer3.baoffice.nitec.com 就不能访问
主机的完整 IP 地址	allow from 206.171.50.50	仅允许拥有指定 IP 地址的主机访问，本例中允许拥有完整 IP 地址（具有 4 个 8 位字节）206.171.50.50 的主机访问

值	例　子	说　　明
部分 IP 地址	例 1： allow from 206.171.50 例 2： allow from 130.86	当 allow 命令中并未指定所有 4 个 8 位字节时，部分 IP 地址从左向右匹配，并允许与地址模式匹配的主机（即属于同一子网的所有主机）进行访问。在第一个例子中，所有从 206.171.50.1 到 206.171.50.255 范围内的主机均可访问，在第二个例子中，属于 130.86 子网的所有主机均可访问
网络地址/掩码对	allow from 206.171.50.0/ 255.255.255.0	使用网络地址和子网掩码允许指定一个地址范围，允许该范围内的主机访问。本例中仅允许 IP 地址范围在 206.171.50.1 和 206.171.50.255 之间的主机访问。这一特征仅在 Apache 1.3 及更高版本中使用
网络地址/nnn CIDR 定义	allow 206.171.50.0/24	除了网络掩码由从高位开始的 1 的位数来指定外，类似于上一项。本例的效果和 allow from 206.171.50.0/255.255.255.0 相同。本特征在 Apache 1.3 以及更高版本中得到支持

10.1.2 deny 命令

语　　法：deny from host1 host2 host3...

使用范围：目录，位置，针对目录的访问控制文件

覆　　盖：Limit

这一命令和 allow 命令恰好相反，它允许定义不能访问指定目录的主机清单。和 allow 令相似，它能接受表 10-1 中的所有值。

10.1.3 order 命令

语　　法：order deny, allow | allow, deny | mutual-failure

缺 省 值：order deny, allow

使用范围：目录，位置，针对目录的访问控制文件

覆　　盖：Limit

该命令控制评价 allow 和 deny 命令的先后顺序，例如：

```
<Directory  /mysite/myboss/rants>
order deny, allow
deny from myboss.mycompany.com
allow from all
</Directory>
```

本例不允许 myboss.mycompany.com 主机访问这一目录，但允许所有其他主机访问。order 命令的值是由逗号分开的列表，它指出了哪个命令优先。典型情况是：对影响所有主机的命令给以最低的优先级，在本例中因为 allow 命令影响所有主机，所以它的优先级较低。

虽然 allow，deny 和 deny，allow 是 order 命令最常用的值，但也可以使用该命令的另一个值 mutual-failure。它用来说明仅允许出现在 allow 列表中，但不出现在 deny 列表中

的所有主机进行访问。

注意，在所情况下，总是对每一条 allow 和 deny 命令进行评价。

10.1.4 allow from env=variable 命令

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

语　　法：allow from env=varible

使用范围：目录，位置，针对目录的访问控制文件。

覆　　盖：Linit

该命令是 allow 命令的变体，当设置了指名的环境变量后才可访问。如果用其他命令（如 BrowserMatch）设置了环境变量这条命令才有用。例如，想允许微软最新版本的 IE4.01 浏览器访问存有嵌入 VB 脚本的 HTML 文件的目录。因为其他流行的 Web 浏览器如 Netscape Navigator 并不直接支持 VB 脚本，因此并不想让使用 Navigator 的用户进入该目录。在这种情况下，当检测到 IE4.01 在访问时，可以用 BrowerMatch 命令设置一个环境变量。命令如下：

```
BrowerMatch "MSIE" ms-brower
```

现在可以使用<Directory>来指定 allow 命令，如下：

```
<Directory  /path/to/Vbscript_directory>
order deny, allow
deny from all
allow from env=ms_browser
</Directory>
```

其中，Apache 服务器将为所有用户代理标识符中包含“MSIE 4.01”字符串的浏览器设置 ms-browser 环境变量。而 allow 命令只允许设置了 ms-brower 变量的浏览器访问这一目录。

10.1.5 deny from env=variable 命令

语　　法：deny from env=variable

使用范围：目录，位置，针对目录的访问控制文件

覆　　盖：Limit

该命令是 deny 命令的变体，它拒绝所有设置了指定环境变量的主机访问。假设不想让 IE 访问某一目录，可以对任何以“MSIE”字符串作为标识的浏览器使用 BroswerMatch 命令设置 ms-browser 环境变量。

```
BrowserMatch "MSIE" ms-browser
```

现在可以使用<Directory>来指定 deny 命令，如下：

```
<Directory  /path/to/Vbscript_directory>
order deny, allow
allow from all
deny from env=ms-browser
</Directory>
```

如果想阻塞特定的 HTTP 请求方法的使用，如 GET、POST 和 PUT，可使用<Limit>

容器来完成这一任务，如：

```
<Location /cgi-bin>
<Limit POST>
order deny, allow
deny from all
allow from yourdomain.com
</Location>
```

这个例子仅允许 yourdomain.com 域内的主机用 POST 方法向 cgi-bin 目录发请求。换言之，如果这一站点有一些表单，它们用 HTTP POST 方法传送用户输入的数据。那么，在现在的情况下，仅在 yourdomain.com 域中的用户才可以有效地使用这些表单。一般情况下，CGI 程序存放在 cgi-bin 目录下，许多站点允许使用 POST 方法的 HTML 表单向 CGI 程序发数据。使用前述基于主机的访问控制配置，一个网站可以允许任何人运行 CGI 脚本，但是只允许某些主机（本例中是 yourdomain.com 域中的主机）才可以向一个或多个 CGI 脚本发送数据。这赋予该站点的 CGI 访问具有只读的色彩，任何人均能运行无需用户输入任何数据即可产生输出的程序，但只有特定域内的用户才能提供输入。

10.2 基本 HTTP 认证过程

基本的 HTTP 认证其实相当简单。当 Web 浏览器请求由 HTTP 认证模式保护的 URL 时，Web 服务器返回 WWW 认证应答头标（WWW-Authenticate response header）和 401 状态头标，WWW 认证应答头标表示访问这一 URL 需要进行认证（见图 10-1）。该头标包含所使用的认证模式（目前仅支持基本 HTTP 认证）和区域（realm）的名称。

此时，在 Web 浏览器中出现一个对话框，要求用户键入用户名和口令。一旦用户键入了用户名和口令，浏览器就将它们和前面对服务器的 URL 请求一起送给服务器。服务器检查用户名和口令是否有效，如果有效，服务器返回所请求的网页。如果无效，服务器用 401 状态和同样的 WWW 认证应答头标予以回答。在每一个后续的、对同一个 Web 服务器的同一区域的请求中，浏览器就继续向服务器发送用户名/口令对，这样服务器就不必为对该站点同样的请求失败而产生 401 状态头标。例如，如果 URL http://apache.nitec.com/protected/ 需要基本 HTTP 认证，后续的请求 http://apache.nitec.com/protected/a-page.html 和 http://apache.nitec.com/protected/b-page.html 同样需要用户名/口令对。这就是为什么浏览器需要在服务器发回另一次验证口令的信息（也就是 401 状态头标和 WWW 认证应答头标）之前发送用户名/口令对的原因。这样做比服务器对每次请求都发回验证口令信息，再由用户一次又一次输入用户名/口令对更快，也更实用。

当口令从客户机（运行 Web 浏览器的主机）发出，它既不是以不显示正文的形式也不是以加密的形式发出。它以 UU 编码（Uuencoded）的形式通过互联网发出，这是这种认证方法的一个缺陷。因为任何配备网络监听硬件或软件的人都可以截取含有 UU 编码的口令的 IP 数据包。因为 UU 编码是一种广泛使用的编码方式，因此它的解码程序（Uudecode）也很容易获得，实际使任何人都能对 UU 编码的口令解码，因而可能造成口

令被滥用。虽然说监听器必须先从网上所有 IP 数据包中截取所需的 IP 数据包，但这一点至少在技术上是可以做到的。

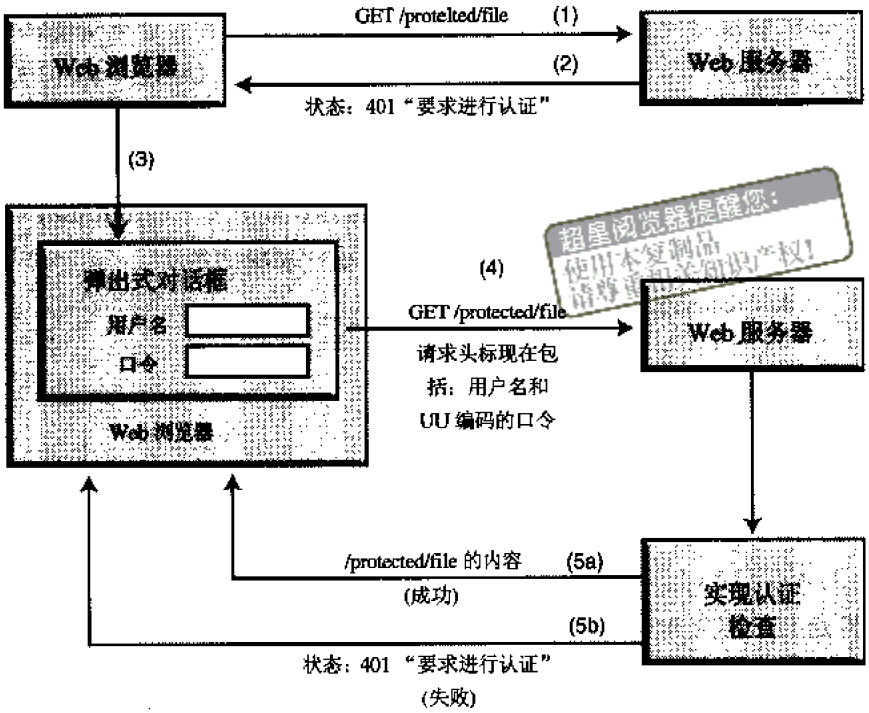

图 10-1　基本的 HTTP 认证过程

这就是为什么强调对于任何关键性应用决不应使用基本 HTTP 认证的原因。使用这种认证模式并不能保护国家的机密。然而，如果已经允许用 Telnet 或 ftp 访问你的系统，就已经为这些服务使用了非常类似于基本 HTTP 认证的认证方法。如果你放心将你的机器连到互联网上，试图对任何想通过 Telnet 登录到你的系统的人开放，那么就没有理由不信任这种方法。

在 Apache 中对基本 HTTP 认证的支持已经很久了，其中许多模块可用来提供基本 HTTP 认证。

在 Web 站点上用 Apache 提供的一个或多个认证模块来建立一个受限制的区域相当容易，应该和认证模块的特殊命令一起使用以下的内核命令：

- AccessFileName
- AllowOverride
- AuthType
- AuthName
- Satisfy
- Require

上述命令在第 4 章中已经讨论过了，在以下几节中，将讨论在当前的 Apache 版本中的几种通用认证模块。

10.2.1 使用标准的 mod_auth 模块

在缺省条件下，在 Apache 标准发行版本中编译了本模块。标准的基于基本 HTTP 认证的 mod_auth 模块将用存储在文本文件中的用户名、组名和口令来实现认证。当处理少量用户时，这种方法工作得很好。然而，如果有大量的用户（数以千计的用户或更多），使用 mod_auth 模块就会使性能急剧下降。在这种情况下，应该使用更先进的方法，如 DBM 文件，Berkeley DB 文件，或者甚至使用 SQL 数据库（在本章后面将讨论这些内容）。

标准的 mod_auth 模块提供 AuthUserFile、AuthGroupFile 和 AuthAuthoritative 等几个 Apache 命令。让我们进一步讨论这些命令和使用这一模块的例子。

超星网点提醒您：
使用本复制品
请尊重相关知识产权！

10.2.1.1 AuthUserFile 命令

语　　法：`AuthUserFile filename`

使用范围：目录，针对目录的访问控制文件

覆　　盖：`AuthConfig`

该命令用来设置文本文件的名称，在这一文件中包含用于基本 HTTP 认证的用户名和口令。必须为该文件提供完整的路径名，例如：

```
AuthUserFile /www/nitec/secrets/.htpasswd
```

该文件通常用称为 htpasswd 的实用程序来建立，这个程序包括在标准的 Apache 发行版中。该文件格式极简单，每行包含一个用户名和一个加密了的口令，口令是用标准的 crypt( )函数加密的。

AuthUserfile 所指定的文件一定要放在 Web 站点的文档树之外，这一点很重要，不然任何用户可能将这些文档下载。

10.2.1.2 AuthGroupFile 命令

语　　法：`AuthGroupFile filename`

使用范围：目录，针对目录的访问控制文件

覆　　盖：`AuthConfig`

该命令指定一个用于存放基本 HTTP 认证的用户组名的文本文件，文件名必须是到这一文件的绝对路径。可以使用任何文本编辑程序来建立这个文件，文件格式如下：

```
groupname:username username username
```

例如：

```
Startrek: kirk spock picard data
```

这就建立了称为 Startrek 的组，组成员有 kirk、spock、picard 和 data 四个用户。

对 AuthUserFile 命令的警告同样适用于本命令。

10.2.1.3 AuthAuthoritative 命令

语　　法：`AuthAuthoritative on|off`

缺 省 值：on

使用范围：目录，针对目录的访问控制文件。

覆　　盖：AuthConfig

如果对同一目录用了不止一个认证模式，可以将这个命令设置为 on。当第一种验证模式对用户名/口令的验证失败后，可以将它传送到下一层（更低层）。

例如，如果用 mod_auth_db（在下一节中讨论）和标准的 mod_auth 模块来提供认证服务，其中一个模块对用户提供的用户名/口令的认证失败后，则另一个模块用来认证这一用户。到底哪一个模块被认为是较低或较高层的认证模块呢？这由这些模块在配置文件中出现的顺序来决定。如果配置文件如下：

超星阅览器提醒
使用本复制品
请尊重相关知识产权！

```
AddModule modules/standard/mod_auth.o
AddModule modules/standard/mod_auth_db.o
```

那么 mod_Auth_db 模块的优先级就较高，因为配置文件中的模块是以从低到高的优先级顺序列出的。

在任何情况下，当所有模块对用户名/口令的认证失败，那么服务器将重新发回 401 状态头标和 WWW 认证应答头标要求重新进行认证。然而，如果用户名/口令成功地通过了某一模块的认证，那么较低层的模块就收不到用户名/口令对。

10.2.1.4 *例子 1：要求用户名和口令的访问控制*

本例演示了如何建立需要用户名/口令才能进行访问的目录。为了简化示例，假设名为 apache.nitec.com 的站点具有如下的设置：

```
DocumentRoot   /data/web/apache/public/htdocs
AccessFileName  .htaccess
AllowOverride   All
```

现在假设想限制对下列目录的访问，并且只有用户名“reader”和口令“bought-it”才可以访问这一目录：

```
/data/web/apache/public/htdocs/chapter_10/
```

按照以下步骤来完成访问控制：

步骤 1：使用 htpasswd 建立用户文件

标准的 Apache 发行版中有一个 htpasswd 实用程序，该程序能建立用于 AuthUserFile 命令的用户文件。使用方法如下：

```
htpasswd -c /data/web/apache/secrets/.htpasswd reader
```

该实用程序会询问用户“reader”的口令，键入“bought-it”，然后重新键入口令来确认以防键入错误。在重新键入口令后，该实用程序就会在/data/web/apache/secrets 目录下建立名为 .htpasswd 的文件。注意以下各点：

♦使用-c 选项用来通知 htpasswd 程序你想建立新的用户文件。如果已经有了口令文件，并且只想在该文件中添加一个新用户，就无需使用此选项。

♦将用户文件放到 apache.nitec.com 站点的文档根目录以外，以防其他人通过 Web 下载这一文件。

♦在文件名前加一个点（.），这样当在 Unix 系统上用 ls 命令列出文件名时，这个文

件就不会出现在输出结果中。这样做并不会提供任何实际的好处，但这确实是 Unix 的传统。许多配置文件（如 .login 和 .profile 文件）前面都有“.”号。

为了减少后面的麻烦，可执行下列命令：

```
cat /data/web/apche/secrets/.htpasswd
```

这将显示类似下列的信息（口令不一定和下面的相同）：

```
reader:hulR6FFh1sxK6
```

这就确保在.htpasswd 文件中确实有一个名为“reader”的用户，口令“bought-it”由 htpasswd 程序使用标准的 crypt( )函数加密。

步骤 2：建立.htaccess 文件

超星阅览器提醒您：
请尊重版权！

使用文本编辑程序在 /data/web/apache/public/htdocs/chapter_10/.htaccess 文件中添加以下各行：

```
AuthName Apache Server Bible Readers Only
AuthType Basic
AuthUserFile /data/web/apache/secrets/.htpasswd
require user reader
```

第一个命令 AuthName 设置认证区域，实际上这只是一个标号而已，将它发到浏览器使用户得到关于她或他将访问什么的启示。在这个例子中，字符串“Apache Server Bible Reader Only”表示只有《Apache 服务器实用大全》的读者才可访问此目录。

第二个命令 AuthType 指定使用的认证类型，因为仅支持基本认证类型，AuthType 总是被设为“Basic”。下一个命令是“AuthUserFile”，它指定了用户文件名。在这里提供到用户文件的路径。最后一个 require 命令指定：允许名为“reader”的用户访问这一目录。

步骤 3：设置文件权限

建立 .htaccess 和 .htpasswd 文件后，要确保只有 Apache 才能读这些文件，这一点很重要。除了文件所有者和 Apache 外任何用户不得访问该文件。

步骤 4：测试

接下来用 Web 浏览器来访问下列 URL：

```
http://apache.nitec.com/chapter_10
```

然后 Apache 服务器就将认证区域（在 AuthName 中设置）和认证类型（在 AuthType 中设置）信息连同 401 状态头标和 WWW 认证应答头标一起送回给浏览器。然后，浏览器弹出对话框，用户应在对话框中输入用户名和口令，如图 10-2 所示。

试一试能不能不用用户名和口令来访问该 URL，在用户名和口令录入框中什么都不输入，然后按“OK”按钮。这将导致认证失败。浏览器又收到同样的验证口令的要求，

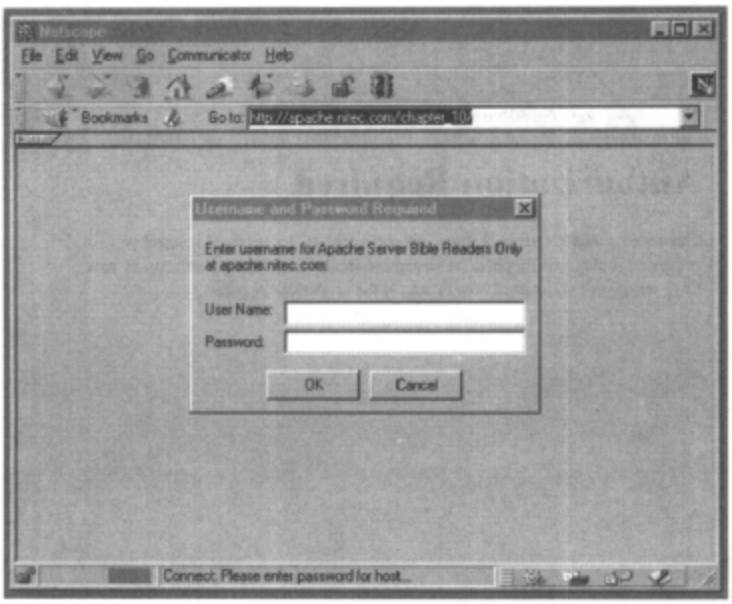

图 10-2 要求输入用户名和口令

显示另一个如图 10-3 所示的对话框，问是否想重试。

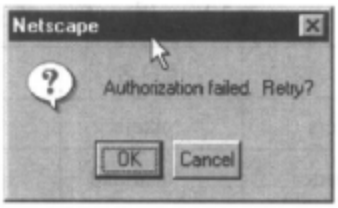

图 10-3 浏览器显示重试对话框

如果选择了“Cancel”按钮，浏览器就会从 Apache 服务器获得并显示标准的“Authentication Required”错误消息，如图 10-4 所示。

按浏览器的“Reload”按钮再次请求同一个 URL 资源，浏览器就会再次收到同样的验证口令的要求。现在键入“reader”用户名和“bought-it”口令，并按“OK”按钮，这时 Apache 服务器就允许访问这个目录了。

如果愿意可以使用 ErrorDocument 命令来改变“Authentication Required”消息：

```
ErrorDocument 401 /nice_401message.html
```

在你的 srm.conf 文件中插入上面这行正文，并在 nice_401message.html 文件中添加容易看懂的消息，使你的用户更容易明白错误的性质。

10.2.1.5 例子 2：允许一组用户访问一个目录

不像前例那样只允许名为“reader”的单个用户访问受限制的区域，现在将允许属于

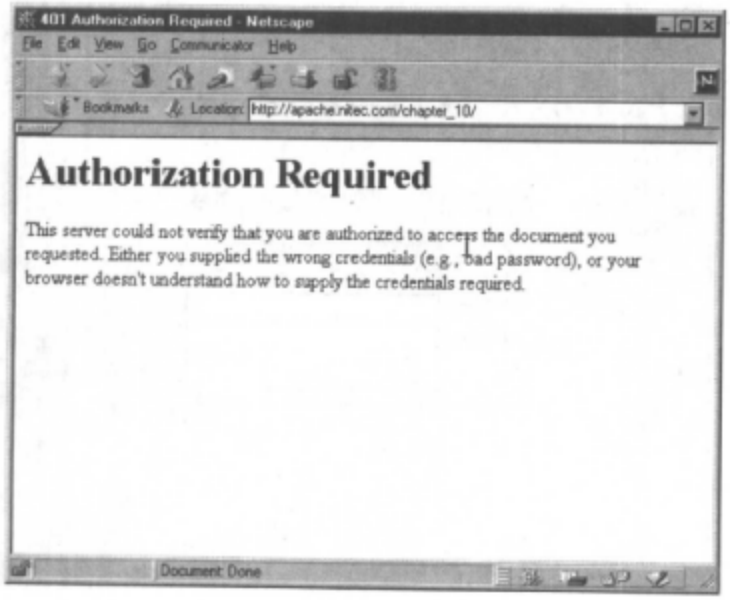

图 10-4 浏览器显示认证失败的消息

名为 asb_readers 的用户组的所有用户访问同一目录。假设想建立如表 10-2 所示的用户表。

表 10-2 用 户 表

组	用户名	口令
asb_readers	pikejb	Red#hat
asb_readers	bcaridad	Net#rat

按照以下步骤指定表 10-2 中的用户访问权限。

步骤 1：使用 htpasswd 建立用户文件

使用 htpasswd 实用程序建立用户 pikejb 和 bcaridad。

步骤 2：建立组文件

使用诸如 vi（大多数 Unix 系统中都有它）的文本编辑程序建立名为/data/web/apache/secrets/.htgroup 的文件，该文件只有一行，如下：

```
asb_readers: pikejb bcaridad
```

步骤 3：在 /data/Web/apache/public/htdocs/chapter_10 目录中建立一个.htaccess 文件

使用文本编辑程序在 /data/web/apache/public/htdocs/chapter-10/.htaccess 文件中添加以下各行：

```
AuthName Apache Server Bible Reader Only
AuthType Basic
AuthUserFile /data/web/apache/secrets/.htpasswd
AuthGroupFile /data/web/apache/secrets/.htgroup
reguire group asb_reader
```

这个配置几乎和前面的例子相同，但有两处变化。一处是加了一个新命令

AuthGroupFile，它使用前面建立的 .htgroup 组文件。另一处改变是在 require 命令行，现在需要名为“asb_readers”的组。实际上完全可以使用下面一行命令：

```
require user pikejb bcaridad
```

来代替：

```
require group asb_readers
```

超星阅览器提醒您：

然而，在 require 命令后边列出一大堆用户显得很麻烦，并且也容易出错，而使用组却可以轻松地增加或删除多个用户。

步骤 5：测试

测试新的设置和前面的例子基本相同，就不再重复了。

10.2.1.6 例子 3：混合使用基于主机的访问控制和基本 HTTP 认证

在本例中，将看到如何将 Apache 中基于主机的访问控制方法和基本的 HTTP 认证方法混合起来使用。为了简化，假设允许“asb_reader”组中的用户访问他们在例子 2 中访问的同一个目录，并且想让在 apache-training.nitec.com 域中的所有用户无需用户名和口令就能访问同一个目录。

换言之，如果对 URL http://apache.nitec.com/chapter_10 的请求来自名为 apache-training.nitec.com 的域，通过以下各步，这个请求无需进行 HTTP 认证就能够得到结果。

步骤 1：修改.htaccess 文件

修改例子 2 中的 .htaccess 文件如下：

```
AuthName Apache Server Bible Readers Only
AuthType Basic
AuthUserFile /data/web/apache/secrets/.htpasswd
AuthGroupFile /data/Web/apache/secrets/.htgroup
require group asb_readers
order deny, allow
deny from all
allow from apache-training.nitec.com
```

该文件中添加了在前面几节讨论过的基于主机的访问控制命令，第一个命令是“order”，它告诉 Apache 服务器先评价 deny 命令，再评价 allow 命令。deny 命令告诉 Apache 拒绝所有主机的访问，而 allow 命令允许 apache-training.nitec.com 域中的主机访问，这就告诉 Apache 服务器允许在 apache-training.nitec.com 域中的所有主机访问这一目录。

步骤 2：测试

如果在名为 user01.apache-training.nitec.com 的主机上使用浏览器访问 URL http://apache.nitec.com/chapter_10，在浏览器中会显示要求用户名/口令的认证对话框，也就是说需要进行认证。这并不是所希望发生的，那是什么原因呢？ 这是因为 Apache 假定：对这一目录而言，基于主机的访问控制和基本的 HTTP 认证都是必须的，因此两种方法有一种通不过就不能访问。解决这个问题的令人满意的办法是使用 Satisfy 命令：

```
AuthName Apache Server Bible Readers Only
AuthType Basic
AuthUserFile /data/web/apache/secrets/.htpasswd
```

该表是从 Perl 5 文档得到的信息。在 Apache 中使用 DBM 之前应保证你的系统已经安装了所使用的 DBM。可以通过确认 DBM 文件确实位于你的系统缺省库目录中来做到这一点。可能还需要对该 DBM 支持的 Perl，应该确保拥有与支持所选的 DBM 选项一起编译的最新版的 Perl 目标代码。

Tip

可从 http://www.perl.com 获得 Perl。配置 Perl 使其支持 DBM 相当简单，只需运行配置脚本，然后它就会提示是否需要 DBM 支持。例如，如果选择了 GDBM 或 NDBM 作为需要的 DBM，并将它们安装在系统中，Perl 配置脚本就会提示是否需要用-lndbm、-lgdbm 和库标志编译 Perl。

一旦已经在你的系统中安装了 DBM 库，就应该配置 Apache 使其支持 DBM 文件，因为标准的 Apache 发行版并不自动实现 DBM 支持。在重新编译之前应检查你的 Apache 可执行文件（httpd），确保已将这个模块预编译过了。下面的命令将会显示在 httpd 可执行文件中的所有已编译过的模块：

```
/user/sbin/httpd -l
```

如果在上面的命令所列出的清单中没有 mod_auth_dbm.c 模块，应该重新配置和编译 Apache。为了添加 DBM 认证模块，应该编辑 Apache 源目录中的配置文件，将下列行的注释去掉：

```
# AddModule modules/standard/mod_auth_dbm.o
```

删除“#”号即去掉注释。保存该文件，并运行配置脚本建立一个新的 Makefile。一旦有了新 Makefile 文件后，即可运行 make 命令建立一个新的 Apache 可执行文件（httpd）。

Tip

如果编译有问题，请在配置文件中为 EXTRA_LIBS 添加-l<your dbmname>。例如，如果使用 GDBM，就可以加 -lgdbm，即：EXTRA_LIBS=-lgdbm。然后重新运行配置脚本和 make 命令。如果还有问题，最好试一下 GNU GDBM，因为许多不同的系统都在使用它，并且还可以从 USENET 上获得帮助。

建立了新 httpd 可执行文件后，运行 httpd -l 命令来检验 mod_auth_dbm.c 模块是否出现在列出的清单中。可用 strip 命令去除该可执行文件中的多余符号，如下：

```
strip httpd
```

然后用新的 httpd 文件替换旧文件。

Tip

替换正在运行的 httpd 文件前，为防止新文件有问题，需要将旧文件改名。这样做一旦新文件出现问题，可以迅速回到旧的 httpd，无须停止 Web 服务。

一旦 Apache 为 DBM 文件进行了正确的编译，就可以用 dbmmanage 建立一个新的用户 DBM 文件。为了建立 DBM 用户文件，应先用标准 Apache 发行版的 support 目录中的 dbmmanage 脚本。dbmmanage Perl 脚本能建立许多流行的 DBM 文件，如 NDBM、GDBM

和 Berkeley DB 等。脚本可用来建立新的 DBM 文件、添加用户名和口令、改变口令或者显示用户信息。使用脚本前应先修改脚本中的下列行，使所选的 DBM 列在 ISA 数组的第一项上：

```
BEGIN{ @AnyDBM_File::ISA=qw(DB_File, NDBM_File, GDBM-File)}
```

如使用 GDBM 文件，像下面那样进行修改：

```
BEGIN{ @AnyDBM_File::ISA=qw(GDBM_File, DB-File, NDBM-File)}
```

为了找出这一脚本提供哪些选项，像下面那样运行它：

```
./dbmmanage
```

这将列出语法行及其所有可能选项。为了建立新的名为 /www/secrets/myuserdbm 的 DBM 文件，并增加名为“reader”的用户，输入如下命令：

```
dbmmanage /www/secrets/myuserdbm adduser reader
```

然后脚本提示为“reader”用户键入（并重键入）口令，这样就将一个用户名和一个加密的口令添加到了名为 myuserdbm 的 DBM 文件中。

Tip 不要用“add”选项来添加用户，因为它不给口令加密。

为了看 DBM 文件中的所有用户名列表，使用下列脚本命令：

```
dbmmanage /path/to/your/dmbfile view
```

重新编译了具有 DBM 支持的 Apache 后，就可使用 mod_auth_dbm 模块来提供基于 DBM 的基本 HTTP 认证。但对于 Berkeley DB，应该使用 mod_Auth_db 模块而不是 mod_auth_dbm。

mod_auth_dbm 提供 AuthDBMUserFile、AuthDBMGroupFile 和 AuthDBMAuthoritative 三个命令。现在来看一下每个命令和使用该模块的例子。

10.2.2.1 AuthDBMUserFile 命令

语　　法：AuthDBMUserFile filename

使用范围：目录，针对目录的访问控制文件（.htaccess）

覆　　盖：AuthConfig

该命令设置 DBM 文件的完整路径名，DBM 文件是用作 DBM 认证的用户文件。这个文件的每个记录均包括 key=value 对，其中 key 是用户名，而 value 则是用 crypt（）加密后的口令。注意，记录中的每个字段都用冒号分开，在开头的用户名和口令字段之后可以加任意数据。

Tip 千万别把用户数据库文件存在 Web 文档树中。

10.2.2.2 AuthDBMGroupFile 命令

语　　法：AuthDBMGroupFile filename

使用范围：目录，针对目录的访问控制文件（.htaccess）

覆　盖：AuthConfig

该命令设置包含用户组列表的组文件的完整路径。文件中的每一个记录是 key= value 对，其中 key 是用户名，而 value 则是以逗号分开的用户所属的组名列表。

如果不喜欢将组文件分开，可以使用单个 DBM 文件提供口令和组信息，文件的格式如下：

```
username: encrypted password: 用逗号分开的组合列表
```

其中用户名是 key，而口令和组列表是 value 的两个字段。如果需要，可在后面另外再加一冒号，并将其他数据写在后面。认证模块自动忽略这些数据。如果用单个 DBM 文件提供口令和组信息，必须使 AuthDBMGroup 命令和 AuthDBMUserFile 命令都指向这个 DBM 文件。

10.2.2.3　AuthDBMAuthoritative 命令

语　法：AuthDBMAuthoritative on | off

缺 省 值：on

使用范围：目录，针对目录的访问控制文件（.htaccess）

覆　盖：AuthConfig

当对同一目录使用多种认证模式时，如 mod_dbm 和标准的 mod_auth，可使用这一命令来决定是否使 mod_auth_dbm 模块成为唯一有效的认证模式。

该命令的缺省值是使 mod_auth_dbm 成为唯一有效的认证模式，这表示当基于 DBM 的用户验证失败，低层的认证模式不会接着进行认证。如果设置为“off”，基于 DBM 的认证失败后，就会将认证任务交给低一层的认证模式。

该命令通常的用法是将它与基本认证模块一起使用，如 mod_auth.c。对大批量的用户就可以由 DBM 模块进行认证，而将少量认证失败的访问（像管理员的访问）交给拥有受到很好保护的.htpasswd 文件的下一层模块来进行认证。

10.2.2.4　例子：要求 DBM 用户名/口令

既然建立了 DBM 用户文件，现在就可以随意限制对任何 Web 目录的访问。在下面的例子中，假定 DBM 用户文件是/www/secrets/myuserdbm。无论用<Directory>容器还是用 .htaccess 文件你都能对你的主或虚拟服务器添加认证模式，二者并无差别。示例配置如下：

```
AuthName Apache Server Bible Readers Only
AuthType Basic
AuthUserDBMFile /www/secrets/myuserdbm
require valid-user
```

现在 Apache 就会用 mod_auth_dbm 模块对使用上面配置的目录进行认证。

确保只有 Apache 和文件所有者才能读这个 dbm 文件，只有所有者才能写 dbm 文件。

10.2.3 使用 mod_auth_db

如果你的系统不能用 DBM，但支持 Berkeley DB 文件，则可以用 mod_auth_db 模块和 DB 文件来代替。和 DBM 模块一样，该模块也没有编译在标准的 Apache 发行版中。要使用该模块，必须配置 Apache 使它支持这个模块，然后对 Apache 重新进行编译。在编译前可能想看一看是否已经预编译了这个模块，可以用下面的命令显示在 httpd 可执行文件中的已经编译过的模块：

```
/usr/sbin/httpd -l
```

如果在该命令产生的列表中没有 mod_auth_db.c，则必须重新配置和编译 Apache。

首先必须知道 DB 库文件存在系统的哪个目录中。例如，对于 Linux 系统，该文件存在标准的/usr/lib 目录下。如果你的系统没有 DB 库，你得先搞到源码，然后编译 DB 支持软件。在下列站点中可找到 DB 库的信息：

```
www.sleepycat.com/
```

一旦确认系统中已安装了 DB 库，就可以重新配置和编译 Apache 了。首先应在配置文件中将下列行的注释去掉：

```
#Addmodale modules/standard/mod_auth_db.o
```

删去#号后，运行配置脚本建立新的 Makefile 文件，然后运行 make 实用程序建立新的 httpd 文件。用-l 命令参数检查是否已将 mod_auth_db 模块加到新 httpd 文件中。现在可以用新的 httpd 文件覆盖旧的 httpd 文件了。

现在可以使用 mod_auth_db 模块了，该模块提供 AuthDBUserFile、AuthDBGroupFile 和 AuthDBAuthoritative 三个命令。

10.2.3.1 AuthDBUserFile

语　　法：AuthDBUserFile filename

使用范围：目录，针对目录的访问控制文件（.htaccess）

覆　　盖：AuthConfig

该命令设置含有用户名和加密口令的 DB 用户文件的完整路径名。

类似于 DBM 文件，DB 用户文件将用户名作为 key 并以使用 crypt( )加密后的口令作为 value。

确保你的 DB 用户文件存在 Web 文档树之外，并且只有 Apache 具有读权限，只有文件所有者才具有写权限。

10.2.3.2 AuthDBGroupFile

语　　法：AuthDBGroupFile filename

使用范围：目录，针对目录的访问控制文件（.htaccess）

覆　　盖：AuthConfig

该命令设置用户组 DB 文件的完整路径名，该文件中包含用于用户认证的用户组名清单。和 DBM 用户组文件类似，这里的组文件也用用户名作 key，以逗号分隔开的组名列

表作为 value。value 中一定不能有空格和冒号。

如果不想用分开的组文件，可以使用单个 DB 文件提供口令和组信息，文件的格式如下：

```
username: encrypted password: 用逗号分开的组名列表
```

其中用户名是 key，而口令和组列表是 value 的两个字段。可以在 value 后面加一冒号，这样就可将其他数据写在后面。认证模块自动忽略这些数据。如果用单个 DB 文件提供组名和口令信息，必须使 AuthDBUserFile 和 AuthDBGroupFile 命令指向这一 DB 文件。

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

10.2.3.3 AuthDBAuthoritative

语　　法：AuthDBAuthoritative on|off

缺 省 值：on

使用范围：目录，针对目录的访问控制文件（.htaccess）

覆盖：AuthConfig

当对同一目录使用多种认证模式，如 mod_auth_db、mod_auth_dbm 和标准的 mod_auth 时，可以使用本命令来指定是否 mod_auth_db 是唯一有效的认证模式。缺省情况下 mod_auth_db 是目录的唯一有效认证模式。这表示当基于 DB 的认证对特定用户失败后，将不继续进行低一层的认证。当设为“off”时，如果 mod_auth_db 认证失败，会由低一层的认证模块继续处理。

虽然用 DB 和 DBM 文件使管理员管理大量用户变得更加简单，但这并不适用于已经将用户信息存在 SQL 关系数据库中的组织。SQL 数据库能提供许多 DB 和 DBM 文件所不能提供的功能。Apache 同样对名为 mSQL 的流行数据库提供支持。

10.2.4 使用 mod_auth_msql 模块

用 mod_auth_msql 模块可以使用存在 mSQL 数据库中的用户名/口令数据来实现基本 HTTP 验证。应先建立包含有用户名/口令信息的 mSQL 数据库。如何建 mSQL 数据库并不是本书讨论的范围，详细情况可以查阅 mSQL 文档。虽然确实建立了 mod_auth_msql 模块使用的用户数据库，还应考虑以下这些问题：

- 用户名字段不应长于 32 个字符，许多浏览器不允许在认证对话框的用户名字段中键入超过 32 个字符。
- 用户名中间不应有空格。

口令字段和用户名字段有相同的限制，但可能不希望让口令少于 5 个字符，因为太短的口令太容易被破译。

mSQL 简介：

mSQL 一词表示迷你 SQL（mini-SQL），它是由澳大利亚 Bond 大学的 David J. Hughes 开发的轻量级结构化查询语言（SQL）数据库引擎。虽然 mSQL 只提供 ANSI SQL 的一个子集，但它以访问速度快和内存消耗少作为设计目标。它支持各种流行的平台如 Sun OS 4.1.1、Solaris 2.x、Ultrix 4.3、Linux、OSF/1、大多数 BSD 衍生系统、基于 SVR4 的系统

或 POSIX O/S、HP-UX、NeXT、SCO、Sequent、Cray、Tandem 和 *BSD 等。数据库源码可从下列 URL 下载：

```
ftp://ftp.bond.edu.au/pub/Minerva/msql
```

和 DB 和 DBM 认证模块一样，该模块也不包括在 Apache 标准版中。必须像下行所示那样将这个模块添加到配置文件中：

```
AddModule modules/standard/mod_auth_msql.o
```

上例中假设将 mod_auth_msql 文件存在 Apache 源发行版的 modules/standard 目录中。如果已将该行加到配置文件中就可运行配置脚本和 make 实用程序。然后可以用 httpd -l 命令检查 httpd 中是否包含了 mod_auth_msql.c 代码。

现在来看一个示例的访问控制配置（如列表 10-1 所示）。该例中所作的假设如下：

- mSQL 服务器和 Apache 服务器装在同一台机器上
- 包含用户信息的 mSQL 数据库名为 asb_readers_db
- 包含用户信息的 mSQL 表名为 asb_users_table
- 包含用户名的 mSQL 字段名为 Uname
- 包含口令的 mSQL 字段名为 Passwd
- 包含组信息的 mSQL 表名为 asb_group_table
- 包含组名的 mSQL 字段名为 asb_readers

列表 10-1： 访问控制配置示例

```
Auty_MSQLhost localhost
Auth_MSQLdatabase asb_readers_db
Auth_MSQLpwd_table asb_users_table
Auth_MSQLuid_field Uname
Auth_MSQLpwd_field Passwd
Auth_MSQLgrp_table asb_group_table
Auth_MSQLgrp_field asb_readers
Auth_MSQL_nopasswd off
Auth_MSQL_Authoritative on
Auth_MSQL_EncryptedPasswords on
AuthName Apache Server Bible Readers Only
AuthType Basic
Require valid-user
```

示例中对 Auth_MSQL_nopasswd、Auth_MSQL_authoritative 和 Auth_MSQL_EncryptedPasswords 三个命令使用了缺省值。

Note 必须保证 mSQL 访问控制文件（msql.acl）允许有效的 Apache 用户 ID 对数据库有读访问权限。

在做了这些以后，即可使用 mod_auth_msql 模块了。该模块有以下命令。

10.2.4.1 Auth_MSQLhost 命令

语　　法：Auth_MSQLhost hostname|IP address

使用范围：目录，针对目录的访问控制文件（.htaccess）

覆　　盖：AuthConfig

该命令设置运行 mSQL 服务器（守护进程）的主机名。如果未设置此项或设置为“localhost”，将强制该模块使用 /dev/msql 而不是用 socket 连接到 mSQL 服务器。如果 mSQL 服务器运行在本地主机上，推荐使用/dev/msql。注意，必须允许 Apache 服务器（UID）访问 mSQL 服务器。

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

10.2.4.2 Auth_MSQLdatabase mSQL 命令

语　　法：Auth_MSQLdatabase mSQL 数据库名

使用范围：目录，针对目录的访问控制文件（.htaccess）

覆　　盖：Authconfig

该命令设置由 mSQL 服务器服务的数据库名，该库中包含用户信息。

10.2.4.3 Auth_MSQLpwd_table mSQL 命令

语　　法：Auth_MSQLpwd_table mSQL 包含用户信息的表名

使用范围：目录，针对目录的访问控制文件（.htaccess）

覆　　盖：Authconfig

本命令设置至少包含用户名和加密口令的表的名称，用户名应设置为表的主键。

10.2.4.4 Auth_MSQLgrp_table mSQL 命令

语　　法：Auth_MSQLgrp_table mSQL 包含组信息的表名

使用范围：目录，针对目录的访问控制文件（.htaccess）

覆　　盖：AuthConfig

该命令设置含有组信息的表名，该表至少包含一个用户名和组名字段。注意，如果一个用户属于多个组，那么在该表中应该有这一用户的多个条目。

10.2.4.5 AuthMSQLuid_field mSQL 命令

语　　法：Auth_MSQLuid_field mSQL 含有用户名的字段名

使用范围：目录，针对目录的访问控制文件（.htaccess）

覆　　盖：Authconfig

本命令为 Auth_MSQLpwd_table 命令指定的表设置字段名，该字段包含用户名信息。如果用 Auth_MSQLgrp_table 命令建立了组信息表，本命令同样为组信息表指定字段名。换言之，必须保证 Auth_MSQLpwd_table 命令和 Auth_MSQLgrp_table 命令所指定的两个表中有同一字段名。

10.2.4.6 Auth_MSQLpwd_field mSQL 命令

语　　法：Auth_MSQLpwd_field mSQL 含有口令的字段名

使用范围：目录，针对目录的访问控制文件（.htaccess）

覆　　盖：AuthConfig

本命令为 Auth_MSQLpwd_table 命令指定的表设置口令信息字段的名称。

10.2.4.7 Auth_MSQLgrp_field mSQL 命令