语　　法：SSLCACertificatePath path/to/CA/cerfiticates

缺 省 值：无

使用范围：服务器配置，虚拟主机

用这个命令指定保存证书颁发机构的证书文件的目录。

15.3.4.5 SSLCACertificateFile

语　　法：SSLCACertificateFile CA_certificate_filename

超星阅览器提醒您：本文复制品 请尊重作者的版权！

缺 省 值：无

使用范围：服务器配置，虚拟主机

这个命令指定包含证书颁发机构的证书的文件。

15.3.4.6 SSLVerifyDepth

语　　法：SSLVerifyDepth number

缺 省 值：SSLVerifyDepth 0

使用范围：服务器配置，虚拟主机

因为一个 CA 证书可以被其他 CA 认证，所以可以形成 CA 认证链。这个命令指定在验证客户证书时服务器应该查看的 CA 证书数。如果不使用客户证书，就不要改变默认值。

15.3.4.7 SSLVerifyClient

语　　法：SSLVerifyClient numeric_option

缺 省 值：SSLVerifyClient 0

使用范围：服务器配置，虚拟主机

这个命令为服务器设置了客户认证方针。如果需要客户证书，把它设置为 2。如果客户证书是可选的，把它设为 1。如果不要求客户证书，把它设为 0。

15.3.4.8 SSLFakeBasicAuth

语　　法：SSLFakeBasicAuth

使用范围：服务器配置，虚拟主机

这个命令把客户的 X509 转换为用于认证的用户名。不推荐使用这一命令。

15.3.4.9 SSLLogFile

语　　法：SSLLogFile log_filename

使用范围：服务器配置，虚拟主机

这个命令指定在哪里写关于 SSL 连接的信息。

15.3.4.10 SSLRequiredCiphers

语　　法：SSLRequiredCiphers cipher1:cipher2:cipher3: ...

使用范围：服务器配置，虚拟主机，针对目录的配置文件(.htaccess)

这个命令指定以冒号分隔的密码算法列表（例如 RC4-MD5、RC4-SHA）。

15.3.4.11 SSLRequireCipher

语　　法：SSLRequireCipher cipher

使用范围：针对目录的配置文件(.htaccess)

这个命令在针对目录的所需加密算法列表中增加加密算法。

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

15.3.4.12 SSLBanCipher

语　　法：SSLBanCipher cipher

使用范围：针对目录的配置文件(.htaccess)

这个命令将禁止在针对目录配置文件中指定的加密算法的使用。使用这个命令，可以禁止客户使用被禁止的加密算法。

15.3.5 为 Apache-SSL 配置 Apache

现在可以创建类似 Apache 的配置文件，例如 httpd.conf、access.conf、srm.conf。Apache-SSL 的作者，Ben Laurie，提供了如列表 15-1 所示的 httpd.conf 文件的例子。Access.conf 和 srm.conf 是空的，因为在作者例子中使用的所有用于 SSL 的配置都出现在 httpd.conf 中。

在这个简单的配置中，Ben 给出了 Apache-SSL 如何用于安全文档服务和非安全文档服务的例子。他使用端口 8887 用于安全连接（https），而不是使用默认 SSL 端口 443；同时他使用端口 8888 用于常规连接（http）。下面的配置例子不能用于测试，因此如果想测试这个例子，需要替换文件目录路径，并取消一些命令的注释。

列表 15-1：httpd.conf

```
# This is an example configuration file for Apache-SSL.
# Copyright (C) 1995,6,7  Ben Laurie

# By popular demand, this file now illustrates the way to
# create two web sites,  one secured (on port 8887), the
# other not (on port 8888).

# You may need one of these
# User web user

# SSL Servers MUST be standlone,  currently.
ServerType  standalone

# The default port for SSL is 443 ...
Port  8887
Listen  8887
Listen  8888

# My test document root
DocumentRoot  /u/ben/www/1/docs

# Note that all SSL options can apply to virtual hosts.

# Disable SSL. Useful in combination with virtual hosts.
# SSLDisable
```

浏览器提醒您：
此为本公司版权，请尊重相关知识产权！

```
# A home for miscellaneous rubbish generated by SSL. Much
# of it is duplicated in the error log file.
SSLLogFile /tmp/ssl.log

# Custom logging
CustomLog logs/ssl_log "%t %{version}c % {cipher}c
%{clientcert}c"

<VirtualHost scuzzy::8888>
SSLDisable
</VirtualHost>

# New and undocumented directives
# SSLRequiredCiphers
# SSLRequireCipher
# SSLBanCipher

# Experiment with authorization...
# <Directory /u/ben/www/l/docs>
# AuthType Basic
# AuthName Experimental
# AuthGroupFile /dev/null
# AuthUserFile /u/ben/www/l/users
# <Limit PUT GET>
# allow from all
# require valid-user
# </Limit>
# </Directory>

ScriptAlias /scripts /u/ben/www/scripts
```

列表 15-2 是用于 blackhole.nitec.com 的安全 Web 服务器的 httpd.conf 文件，它的文档根目录在/www/nitec/secured/htdocs。为了支持端口 80 进行常规的 Web 服务，创建了一个虚拟主机，这个虚拟主机通过 SSLDisable 命令禁止 SSL，并指向不同的文档根目录/www/nitec/public/htdocs。

列表 15-2：httpd.conf

```
# User and group for blackhole.nitec.com
User httpd
Group httpd

# SSL Servers MUST be standalone, currently.
ServerType standalone
```

```
# Use the default HTTPS port
Port 443

# Also listen to the standard HTTP port
Listen 80

# Document root directory for blackhole.nitec.com
DocumentRoot /www/nitec/secured/htdocs

# The directory where you store CA provided certificate file(s)
SSLCACertificatePath /usr/local/etc/httpd/SSLconf/conf

# The full path of the CA provided certificate file
SSLCACertificateFile /usr/local/etc/httpd/SSLconf/conf/httpsd.pem

# The full path to the SSL certificate
SSLCertificateFile /usr/local/etc/httpd/SSLconf/conf/httpsd.pem

# Do not require clients to have certificates
SSLVerifyClient 0
SSLVerifyDepth 0

SSLLogFile /logs/ssl.log

# Since I want to have reqular (HTTP) web services on
# port 80, here is the virtual host that does that.
<VirtualHost 206.171.50.50:80>
DocumentRoot /www/nitec/public/htdocs
ScriptAlias /cgi-bin/ /www/nitec/public/cgi-bin/SSLDisable
ServerName blackhole.nitec.com
</VirtualHost>
```

超星阅览器提醒您：
此为本复制品，
请尊重作者的知识产权！

为新编译的 **httpsd** 文件建立类似的配置。如果只对创建安全虚拟主机有兴趣，并想把主服务器放在端口 80 上，可以在主服务器的配置段（在虚拟主机容器外）使用如下的命令：

```
Port 80
SSLDisable
```

为了创建安全虚拟主机，可以使用类似下面的配置：

```
<VirtualHost host.domain.com:443>
DocumentRoot /path/to/secure/pages
ServerAdmin webmaster@host.domain.com
ServerName host.domain.com
SSLCACertificatePath /usr/local/ssl/certs
SSLCACertificateFile /usr/local/ssl/certs/virtual.host.com.pem
SSLCertificateFile /usr/local/ssl/certs/virtual.host.com.pem
```

```
SSLLogFile /path/to/ssl.log
</VirtualHost>
```

主服务器配置段中的 SSLDisable 命令禁止任何 SSL 服务，除了列出 SSL 命令的虚拟主机以外。现在可以测试你的安全站点了。

15.3.6 测试安全服务器

在开始测试前，要确保具有最新版本的 Netscape Navigator 或 Microsoft IE 浏览器。有些旧的浏览器不支持 SSL。

为了测试安全站点，可以把你的浏览器指向：

`https://your.secured.web.site.domain/`

记住对于所有安全服务器的 URL 都必须用 https://代替 http://。

对前面的配置，安全服务器的访问使用下列 URL：

`https://blackhole.nitec.com/`

普通的 Web 访问使用：

`http://blackhole.nitec.com/`

如果在早些时候通过运行 make certificate 创建了自定义证书，并要访问安全站点，将会从你的浏览器得到一个警告信息。例如，当要访问 https://blackhole.nitec.com 时，Netscape Navigator 将返回如图 15-5 的警告窗口。

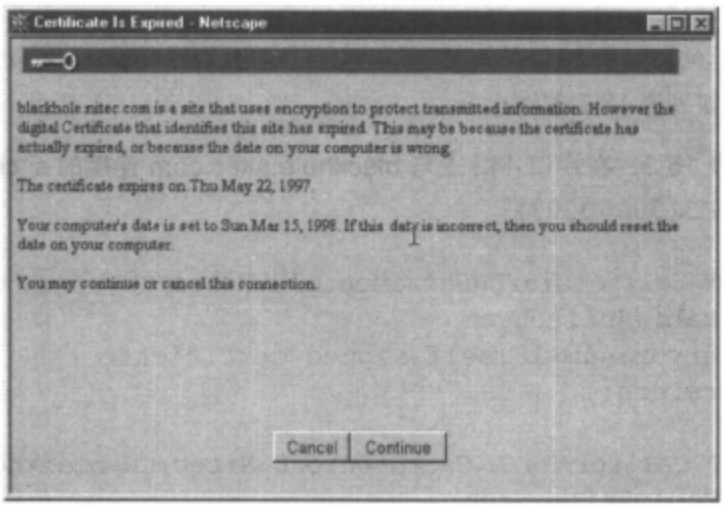

图 15-5 关于自定义证书的警告消息

当每个用户试图访问安全站点时，这种警告信息都将出现，除非得到了浏览器认识的商业 CA 的证书。

如果设法通过了警告对话窗，通常通过单击一个或多个按纽（OK 或 Finish）用手工方式得到证书，浏览器将建立与服务器的安全连接通道。图 15-6 是连接 blackhole.nitec.com 服务器的情况。

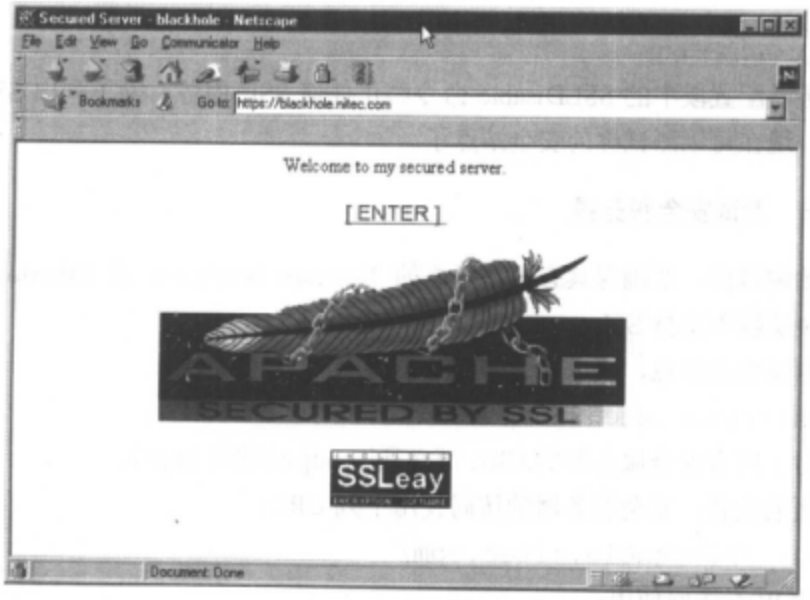

图 15-6 安全的对话过程

注意在左下角的锁图标是关闭的（即锁着的）。这意味着安全通道已经建立。

如果有兴趣看看当客户（例如一个浏览器）连接安全服务器时会发生什么情况，可以运行 SSLeay 软件包中的 s_client 应用程序。对于 blackhole.nitec.com 站点，s_client 可以这样运行：

```
ssleay s_client -host blackhole.nitec.com -port 443
```

它将产生如列表 15-3 的输出。

列表 15-3：在端口 443 上与 blackhole.nitec.com 连接的 s_client 的输出

```
CONNECTED(00000003)
depth=0
/C=US/ST=California/L=Sacramento/O=Nitec/CN=blackhole.nitec.com
/Email=kabir@nitec.com
verify error:num=18:self signed certificate
verify return:1
depth=0
/C=US/ST=California/L=Sacramento/O=Nitec/CN=blackhole.nitec.com
/Email=kabir@nitec.com
verify return:1
--
Certificate chain
 0
s:/C=US/ST=California/L=Sacramento/O=Nitec/CN=blackhole.nitec.com
/Email=kabir@nitec.com

i:/C=US/ST=California/L=Sacramento/O=Nitec/CN=blackhole.nitec.com
/Email=kabir@nitec.com
--
```

```
Server certificate
--BEGIN CERTIFICATE--
MIICezCCAeQCAQAwDQYJKoZIhvcNAQEEBQAwgYUxCzAJBgNVBAYTAlVTMRMwE
QYDVQQIEwpDYWxpZm9ybmlhMRMwEQYDVQQHEwpTYWNyYW1lbnRvMQ4wDAYDVQQKEw
V0AXRlYzEcMBoGA1UEAxMTYmxhY2tob2xlLm5pdGVjLmNvbTEeMBwGCSqGSIb3DQE
JARYPa2FiaXJAbml0ZWMuY29tNB4XDTk3MDQyMjA4NDAwNFoXDTk3MDUyMjA4NDAw
NFowgYUxCzAJBgNVBAYTAlVTMRMwEQYDVQQIEwpDYWxpZm9ybmlhMRMwEQYDVQQHE
wpTYWNyYW1lbnRvMQ4wDAYDVQQKEwV0aXRlYzEcMBoGA1UEAxMTYmxhY2toB2xlLm
5pdGVjLmNvbTEeMBwGCSqGSIb3DQEJARYPa2FiaXJAbml0ZWMuY29tMIGfMAOGCSq
GSIb3DQEBAQUAA4GNADCBiQKBgQDVuXIKqg7p9eSlxVIRyglaLTSZ+Ge+CESRHUBU
QiGuiU+eo97CaKze4B19AM4ZV3xnloJrS8LKTnLmkJ95A++6ymrhHVIVgmXZDgkcz
jQ2LKInWCJ9QzntoEu/1mPUY2KnXbbQo5v/gtf/J6tbhuRJKtLgz2dK4CWM/OPQzz
rYjQIDAQABMAOGCSqGSIb3DQEBBAUAA4GBAIT2FA3Je7Q+tnu3KjBj7NLTVDG5YsE
yGXZKBEwJEzpAzOywnqndiTwwdUgocYl/EADidHsrgiYDsYPD3oVFkWOwevHLwxBE
++6FIcLkRty3C/bfgrdJWWFsrdl4nZq3Vy8y/xFQkZJ9ObOcpo6acOqm5HrrSatn6
DBaWuDXdEI5
--END CERTIFICATE--
subject=/C=US/ST=California/L=Sacramento/O=Nitec/CN=blackhole.
nitec.com/Email=kabir@nitec.com
issuer=/C=US/ST=California/L=Sacramento/O=Nitec/CN=blackhole.
nitec.com/Email=kabir@nitec.com
SSL handshake has read 813 bytes and written 317 bytes
--
New, SSLv3, Cipher is RC4-SHA
SSL-Session:
  Cipher   : RC4-Sha
  Session-ID:
C26196B7FC9C2DBE664FC5C55AD4C89887BC9703B98F9441362853E428C4D
67F
  Master-Key:
303EBE61E04175F81C605E02A2CC264022F6E55F7714FA32DB3A37852A4AD
6C85B2748ED3FAD06FDD733CDD098B443D6
  Key-Arg   : None
  Start Time  : 861699068
  Timeout    : 7200 (sec)
--
```

前面的输出包含了大量的信息。服务器提供了与缩写的可分辨名有关的输出。可分辨名由 X.509 标准定义，它定义了字段、字段名、该字段对应的缩写名。表 15-1 列出了其中的一些字段。

证书颁发机构（CA）定义了哪些字段是必须的，哪些是可选的。它们甚至限制字段的内容。例如，Netscape 浏览器要求证书的公共名字段是服务器的名称或域的规则表达式。

表 15-1 可分辨名称字段

字 段	缩写	描 述	例 子
公共名	CN	被认证的名称。它是系统的完整域名	CN=blackhole.nitec.com
组织或公司	O	这个组织的名称	O = Nitec
组织单位	OU	这个组织单位的名称，例如一个部	OU = I didn't specify one
城市/位置	L	组织所在的城市	L=Sacramento
州/省	ST	组织所在的州/省名	ST=California
国家	C	组织所在的国家名（ISO 代码）	C=US

证书的二进制形式使用抽象语法标记法（Abstract Syntax Notation 1，ASN.1）定义。标记法定义如何指定内容，编码规则定义如何将信息转换为二进制。ASN.1 可以用很多方法编码。但新兴的标准是直接编码规则（DER）——一种很简单的编码方案，它能形成紧凑的二进制证书。如果用于交换电子邮件，二进制证书常常使用基于 64 位的编码，结果得到 ASCII 文本行。正如在上面的列表中在 BEGIN CERTIFICATE 和 END CERTIFICATE 之间看到的那些行。

在列表 15-3 的输出中，也包含认证链。如果一个证书由一个 CA 发行，而这个 CA 的证书又由另一个 CA 发行，那将产生一个链。服务器将显示有关这个链的信息。这样，客户可以调查链中提及的证书。在本例中链是不存在的。

显示列表 15-3 的内容后，s_client 脚本仍然保持与服务器的连接。因此，如果输入如下的 HTTP 请求：

```
GET  /HTTP/1.0
```

那么在一个空行的后面，可以得到列表 15-4 所示的输出。

列表 15-4：安全服务器上的 GET 请求的输出

```
HTTP/1.1 200 OK
Date: Tue, 22 Apr 1997 08:51:56 GMT
Server: Apache/1.2.5 Ben-SSL/1.13
Connection: close
Content-Type: text/html

<HTML><HEAD>
<TITLE>Index of /</TITlE>
</HEAD><BODY>
<H1>Index of /</H1>
<UL><LI> <A HREF="/"> Parent Directory</A>
<LI> <A HREF="development"> development</A>
<LI> <A HREF="future"> future</A>
<LI> <A HREF="public"> public</A>
```

```
</UL></BODY></HTML>
read:error=0
```

安全服务器 blackhole.nitec.com 的顶层目录没有目录索引文件，所以服务器像通常那样生成动态产生的目录列表。

如果在你的 Apache-SSL 服务器上可以执行类似的测试，这就表示已经成功地创建了安全 Web 服务器。

您使用的是浏览器提醒您：
请尊重相关知识产权

15.3.7 得到由 CA 签署的证书

现在，要做的是从某个著名的证书颁发机构（CA）那里得到浏览器能识别的证书。不幸的是，由于它们管理方针的不同，并不是所有的 CA 都为 Apache-SSL 发行服务器证书。

这里是一些可能颁发 Apache-SSL 证书的 CA：

- Thawte Consulting，www.thawte.com/certs/server/request.html
- CertiSign Certificadora Digital Ltda.，www.certisign.com.br
- IKS GmbH，www.iks-jena.de/produkte/ca/
- Uptime Commerce Ltd.，www.uptimecommerce.com
- ID-Pro GmbH，CA-Projekt，www.id-pro.de/security/CA

只有 Thawte 和 VeriSign（它不为 Apache-SSL 提供证书）被所有版本的 Netscape Navigator/Communicator 和 Microsoft Internet Explorer 浏览器支持。这种情况可能会改变，因为越来越多的人开始使用 Apache-SSL 或其他的基于 Apache 的 SSL 服务器。CA 可能会由于商业上的压力而认证这些服务器。最好的方法是确定目标浏览器支持哪些 CA，再看其中那一个 CA 愿意签署你的证书。在当前情况下，Thawte 是唯一现实的选择。

15.4 Stronghold

如果不希望经历使用 SSLeay 的种种麻烦：如配置 SSLeay，用 Apache-SSL 补丁修补 Apache 源文件和其他源文件，以及处理自定义的证书问题等。最好的选择是得到可用于商业的 Apache 衍生服务器——Stronghold。

Stronghold 由 C2NET Technology 公司开发。它基于最新的 Apache 服务器和 SSL。Stronghold 具有很好的基于 Web 的配置管理软件。它可以从 Web 的任何位置管理服务器。如果计划利用 Web 为商业赢利的目的服务，应该坚决使用这个产品。

用户可以很容易地从 C2NET 得到该产品的评估拷贝。访问 Web 站点 www.c2.net，并签约得到 Stronghold 服务器的评估版本。在签约时，要求输入计划用作 Stronghold 服务器平台的 IP 地址。一旦进行了登记，几分钟内将有一份电子邮件发到你的电子邮件地址（你在登记时提供的）。这个电子邮件的信息包括下载 Stronghold 的用户名/口令。它也包括安装时需要的许可证密钥，以及下载站点的 URL。

为了那些想用 Apache-SSL 实现 SSL 的人少一些不眠之夜，Stronghold 的确是一个了

不起的解决方案。公司将提供有用的文档，而且很有帮助。如果认为这种选择太贵，而且也不想使用 Apache-SSL，可以试一试 Apache 的 Raven SSL 模块，它可在下面的 URL 中得到：

```
http://raven.covalent.net/
```

从技术上讲，Covalent Technologies 开发的 Raven SSL 模块可能并不像其他 Apache 模块（例如 mod_cgi 或 mod_rewrite），但它仍然像最终用户的模块那样工作。公司提供包含了 Raven SSL 模块的 Apache 服务器的预编译版本。这个模块还带有预先经过修补的 Apache 源文件。因此，重新编译包括用户自己的模块的新服务器和配置（其中包括 Raven SSL），就像在配置文件中去掉相应行的注释那样简单。注意，由于美国限制加密程序的发行，Covalent Technologies 的发行版不包含 Raven 模块的源代码或 RSA 加密算法。当前，VeriSign 不为基于 Raven 的 Apache 服务器发行证书，但 Thawte Consulting 为它发行证书——而且，它比 Stronghold 便宜（至少在写本书时的情况是这样）。

15.4.1 安装 Stronghold

安装 Stronghold 是轻而易举的。只要解压缩从 Web 站点下载的软件，并运行 INSTALL.sh 脚本即可。它会给出一些提示信息，例如在哪里安装 Stronghold、HTTP 和 HTTPS 服务使用哪些端口、服务器名称、许可证密钥等。大多数的提示都有默认值，它们几乎适用于所有的情况。

Note 不需要为提供 HTTP 服务运行单独的 Apache 服务器。Stronghold 可以同时提供安全和非安全的 HTTP 服务。

如果采用默认值，那么将使用如下的端口：

80 标准 HTTP 服务。例子：http://yourserver/

443 基于 SSL 的 HTTP 服务。例子：https://yourserver/

444 配置管理软件。例子：https://youserer:444/

445 直接（本地主机）接口。例子：https://localhost:445/

如果使用的是 csh/tcsh shell，安装脚本要求在 cshrc 文件中加入如下内容：

```
sentenv SSLTOP /path/to/stronghold/dir/ssl
setenv PATH ${SSLTOP}/bin:$PATH
```

如果使用的是 sh/bash shell，加入如下内容：

```
$ SSLTOP=/path/to/stronghold/dir/ssl
$ PATH=${SSLTOP}/bin:$PATH
$ export SSLTOP PATH
```

安装脚本同时提供下面的选项：或者转换已存在的 Netscape 商业密钥/证书，或者创建新密钥/证书对。如果没有安全的 Netscape 商业服务器，应该创建新密钥/证书对。此时，需要选择密钥的位数。选择范围是 512 位至 4096（4K）位。

这里推荐使用 1024 位的密钥，因为位数少的密钥不够安全，但位数多的密钥使服务器变慢。同时，某些流行的浏览器版本不能处理位数多的密钥。选择了密钥位数后，需要

从键盘上随机地敲击任意键。不用考虑敲的是什么键，因为这样做的实际目的只是衡量击键之间的时间差。一直击键直到听到一声响或计数器回到 0。输入完毕后，将产生一个密钥。

现在需要作出一个重要决定：是否请求 CA 授权的证书。注意，Stronghold 现在和 Thawte 证书颁发机构的不在使用的证书捆绑在一起，但只有对软件付款后才能得到这个证书。如果想先评估一下软件再决定是否使用它，那么现在最好不要产生证书的请求，直到需要购买软件时再说。可以使用 Stronghold 提供的 genreq 实用程序产生证书的请求。如果想先评估一下软件，那么应该回答“不”，系统将给出提示：需要为评估期间创建一个自定义的证书。

在这个过程中创建的专用密钥可以用口令来保护。接着，用户将被询问：是否增加口令来保护专用密钥。如果选择是，那么每次启动安全服务器时都需要输入口令。这意味着可能不能在引导时由 rc.d 或 rc.local 功能自动启动服务器。

最后，安装脚本将启动安全服务器，可以立即用下面的 URL 访问安全服务器：

```
https://yourserver/
```

浏览器将发出警告：它不认识服务器使用的自定义证书。如果在服务器上安装了一个真正的 CA 提供的证书后，就不会显示这个警告了。在安全站点上看到的第一页如图 15-7 所示。

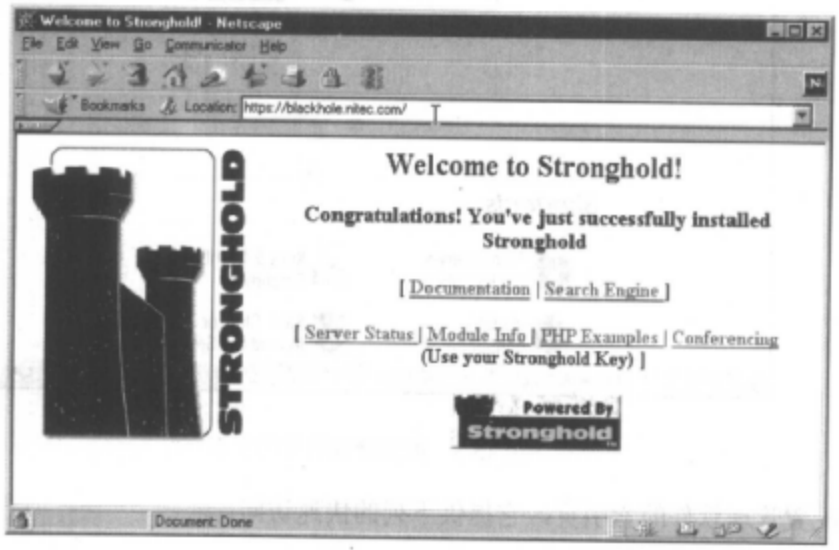

图 15-7 Stronghold 的欢迎页

Note

使用 INSTALL.sh 程序安装 Stronghold 时，它同时也安装 Python（Focus 的引用系统）和 SWISH/WWWWAIS 搜索引擎。

现在已经成功地安装了 Stronghold。可以接着往下做，并在安全 Web 空间创建内容。为了管理安全 Web 服务器，可以使用基于 Web 的配置管理软件。

15.4.2 使用基于 Web 的配置管理软件

基于 Web 的配置管理软件或通常的配置文件，例如 httpd.conf、access.conf、srm.conf，都可以用来配置 Stronghold。为了能从 Web 的任何位置访问配置管理软件，可以将你的浏览器指向安全服务器 URL 的端口 444（如果使用的不是 INSTALL.sh 程序提供的默认值，那么需要使用别的端口地址）。

例如，安全 Stronghold 服务器的配置管理软件可以用如下的 URL 进行访问：

```
https://secured.nitec.com:444/
```

为了避免配置管理软件被滥用，配置管理软件使用基于用户名/口令的认证方案。在安装时提供用户名和口令，需要时就可以使用。如果把自己确定为配置管理人员，将会看到类似图 15-8 所示的页面。

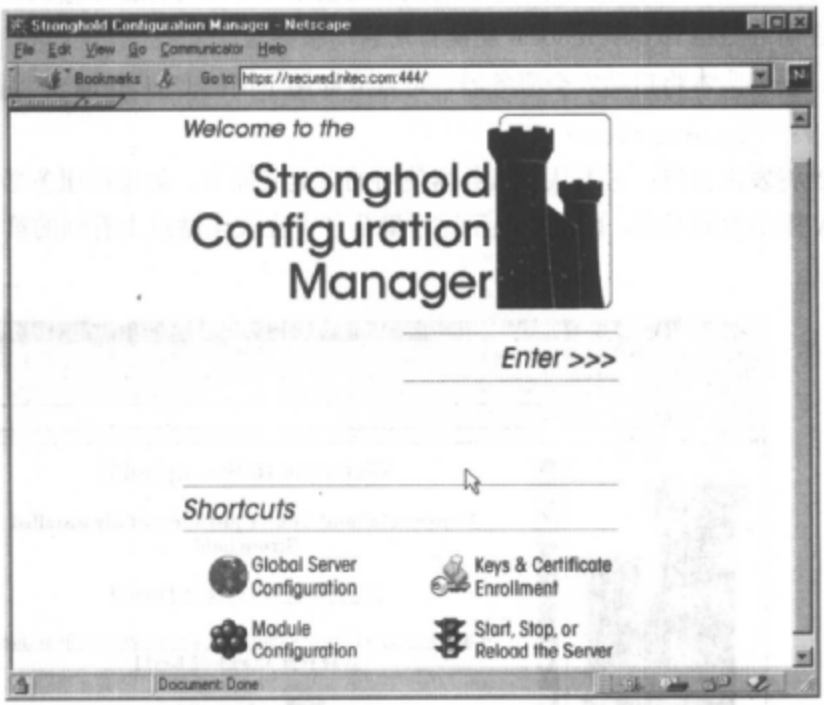

图 15-8 配置管理软件的主页

这是配置管理软件的索引页，它提供下列的快速访问：

- 总的服务器配置
- 密钥和证书登记
- 模块配置
- 启动、停止或重新装入服务器

单击 Enter 图标，可以看到类似图 15-9 所示的页面。

在这儿，用户可以访问：

- 总的服务器配置

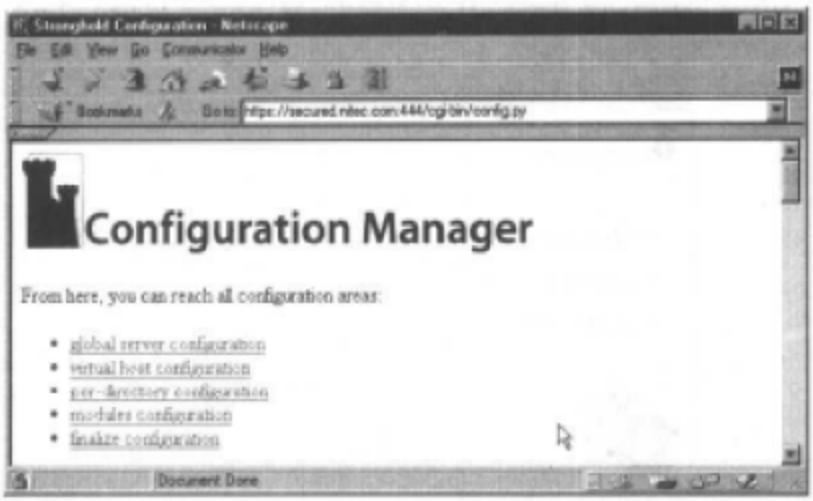

图 15-9 配置管理软件的总的服务器配置页

- 虚拟主机配置
- 针对目录的配置
- 模块配置
- 最终配置

最后的选项允许提交对服务器的任何修改或增加。单击这一项可以看到如图 15-10 所示的页面。

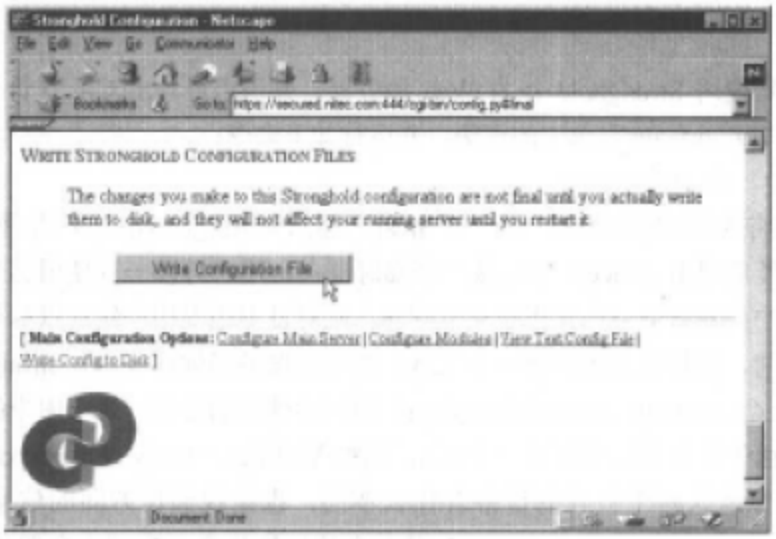

图 15-10 保存修改

如果准备提交新配置，单击“Write Configuration File（写配置文件）”按钮就可以把修改过的配置写到硬盘上。保存修改后，必须重新启动服务器，这样新的修改才有效。可以使用如图 15-11 所示的重新启动、停止、重新安装页来重新启动服务器。

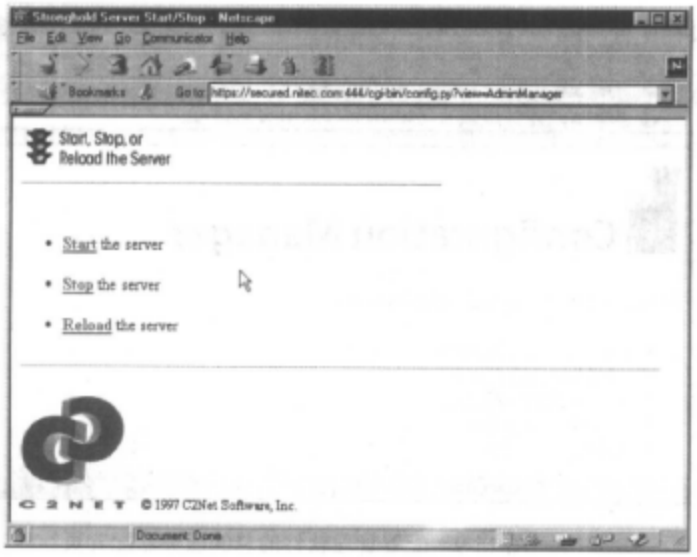

图 15-11 从配置管理软件启动、停止和重新安装

Note

如果在安全服务器系统上有一个支持 SSL 的浏览器，那么可以使用下面的 URL https://localhost:445/通过直接（本地主机）接口访问配置管理软件。如果安装时采用的不是 445 端口，则需要改变端口号码。如果在 Web 服务器上运行 Telnet，并希望能快速改变某些设置时，就可以用到这个特点。可实现 SSL 的 Lynx 浏览器就是这种浏览器。

15.4.3 取得由 CA 签署的证书

如果决定使用 Stronghold 作为 Web 服务器，并请求得到由 CA 签署的证书，可以备份存在$SSLTOP/private 目录中的密钥，并运行如下命令：

```
genkey <hostname>
```

该命令将为指定的主机产生一对密钥。然后调用 genreq 来产生签署证书的请求（CSR）。最后调用 gencert 来生成一个临时证书，用户可以一直使用这个临时证书，直到 CA 对用户的签署证书的请求做出了回答。临时证书用专用密钥加以标记。但在运行这个实用程序前，应该决定使用哪一个 CA。在购买服务器软件包时，Stronghold 就提供了来自 Thawte Consulting（www.thawte.com）的自由使用的证书。这个附带的证书也可以用得上，除非由于某种原因需要另一个 CA，例如 VeriSign（www.verisign.com/）。

如果想知道这两个公司的价格有什么不同，也能找到这方面的信息。VeriSign 在 www.verisign.com/enroll.s/payment.html 发布它的价格信息，它的安全服务器服务协议在 www.verisign.com/enroll.s/legal.html。Thawte 在 htttps://www.thawte.com/pricing.html 提供价格表。如果要使用另一个 CA，需要确认所选的 CA 是访问你的站点的客户所能接受的 CA。

在向 CA 提交 CSR 后，还要向 CA 提供一些 CA 要求的内容，这些有助于 CA 认证你的组织。现在只需要等待 CA 返回你的站点的证书。通常这个证书是一个电子邮件的附件。

在接到它后，必须用 Stronghold 的 getca 实用程序安装它。

安装站点证书的步骤如下：

1. 把电子邮件的附带的证书文件保存在一个临时位置，例如/tmp/01.pem。

2. 运行 getca，指定拥有这个证书的主机名并把临时文件作为输入：

```
getca  hostname  < /tmp/01.pem
```

3. 使用重装入脚本重新启动服务器，或用下面的命令重新启动服务器。

```
kill  -HUP  'cat /path/to/httpd.pid'
```

要求对每个使用 SSL 的主机都有一个密钥/证书对。其中也包括你运行的虚拟主机。因为需要为每个请求的证书付款，如果你有许多站点支持 SSL，这将是一种昂贵的做法。在这种情况下，重新组织虚拟主机并使它们指向一个支持 SSL 的主机是明智的。

例如，可以创建一个叫 secured.yourdomain.com 的主机，并为这个主机得到一个密钥/证书对。如果一个叫 virtualone.yourdomain.com 的虚拟站点需要实现安全传输，那么只需在 secured.yourdomain.com 的文档根目录下创建一个子目录，并把这个虚拟站点的文件放在那里即可。在虚拟站点上需要链接到安全区域的 HTML 文档可以使用如下的链接：

```
<A HREF=https://secured.yourdomain.com
/virtualhost-directory/filename>A secured Page</A>
```

需要确保访问者在完成对安全文档或脚本的浏览后有办法回到非安全虚拟站点。可以通过增加返回虚拟站点的链接来保证这一点。为减少虚拟站点访问者的混淆，也可能需要在你的证书请求中使可选字段（例如组织单位）为空。

如果将服务器用于企业内联网或校园网，可以考虑建立你的专用证书颁发机构。

15.4.4 建立专用的证书颁发机构

专用CA 允许签署自己的证书。创建专用的 CA 可能出于以下两个原因：

- 你只想满足自己的好奇心。
- 你的组织有大量的安全需求，需要在部门中广泛使用 SSL。

第二种情况下，需要确保要建立 CA 的系统实际是安全的，并且更可取地它根本不和其他网络连接。这样做可保证很难强行闯入你的 CA。如果不采取措施保护 CA 系统，而同时入侵者获得了对 CA 的访问权限，那么入侵者可以建立假的证书，并使用它访问需要保护的文档，甚至创建一个软件来假冒你的服务器。

Stronghold Web 服务器为创建基本证书授权机构提供了工具。注意，可以使用为 Apache-SSL 安装的 SSLeay 软件包创建专用的 CA。在下面的章节中，讲的是用 Stronghold 带来的 SSLeay 实用程序来创建专用的 CA。

15.4.4.1 建立专用的 CA

建立专用 CA 的第一步是为 CA 创建密钥/证书对。可以使用$SSLTOP/bin 目录下的 makeca 脚本来创建它们。可使用为 CA 创建的专用密钥签署你的 CA 提供的服务器证书。客户系统将使用你为 CA 创建的证书来验证它发行的服务器认书上的 CA 的签名。

Makeca 脚本把它创建的 CA 证书存在$SSLTOP/CA/cacert.pem 文件中。然而，Netscape

Navigator 要求 CA 证书是 DER 格式。为了把 PEM 证书格式转换为 DER 格式，可以使用 x509 程序：

```
X509 -outform DER < $SSLTOP /CA/cacert.pem>
/path/to/document/root/your_organization.cacert
```

结果文件，your_organization.cacert，将放在服务器的文档根目录下。如果希望客户浏览器也可以访问这个结果文件，应当把它放在别的位置。文件的后缀.cacert 很重要，因为它与 MIME 类型的 application/x-x509-ca-cert 联系在一起。确保检查在 Stronghold 的顶层目录的 mime 类型文件中有下面这样一行：

```
application/x-x509-ca-cert  cacert
```

它把后缀 cacert 和证书授权机构的证书联系起来，并通知浏览器下载和安装证书。例如，当一个 Nitec 用户把它的 Netscape Navigator 浏览器指向 https://blackhole.nitec.com/nitec.cacert 时，浏览器将显示如图 15-12 的对话窗口。

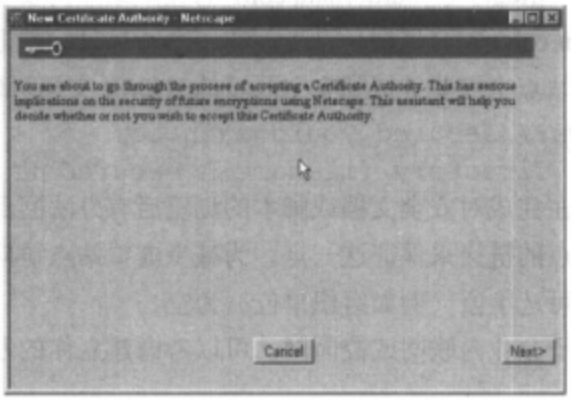

图 15-12　在浏览器上安装专用 CA 的证书

还有一些对话窗口向用户提供关于证书的信息。用户可以浏览证书的细节，如图 15-13 所示。

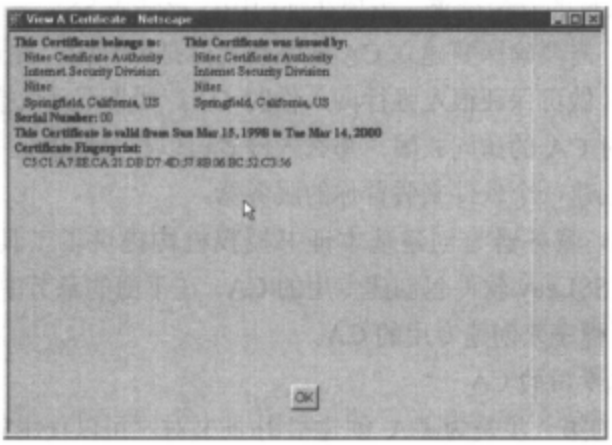

图 15-13　Nitec 证书的细节

在这个过程中，用户有机会决定这个专用 CA 可以认证什么。例如，他可以决定允许

专用 CA 认证其他（网络）站点、电子邮件用户或软件开发者。图 15-14 表示用户决定相信这个 CA 用它来认证任何事。

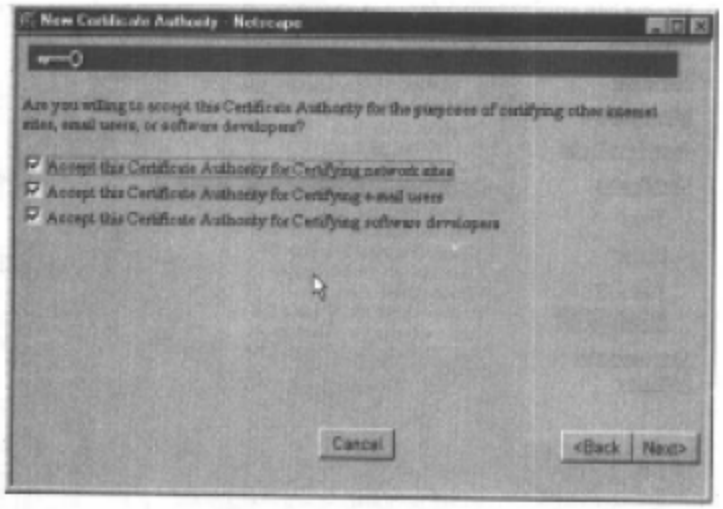

图 15-14 用户选择专用 CA 认证的项目

最后，当用户接受专用 CA 时，他必须为 CA 取个昵称。在图 15-15 中，昵称是 Nitec CA。

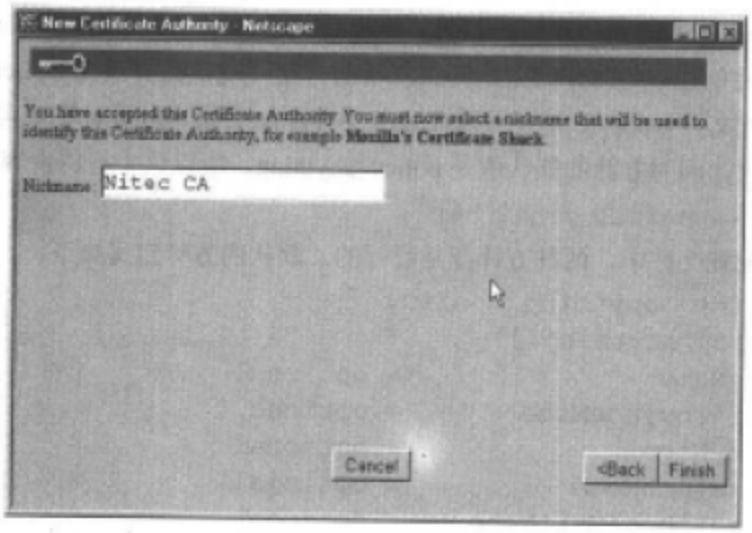

图 15-15 为专用 CA 选择昵称

用户完成这些步骤后，这个 CA 的证书就和其他正式的 CA 证书列在一起了。在图 15-16 中可以看到 Netic CA 和许多其他商业 CA 列在一起。

用户可以在任何时候编辑、删除或改变 CA 的证书和它的使用。

如果将 CA 的认证安装在用户的浏览器上，当 CA 是未知时，浏览器不会警告用户（除非在证书安装过程中用户选择了警告选项）。

创建专用 CA 的下一步是配置 SSLeay 库配置文件。

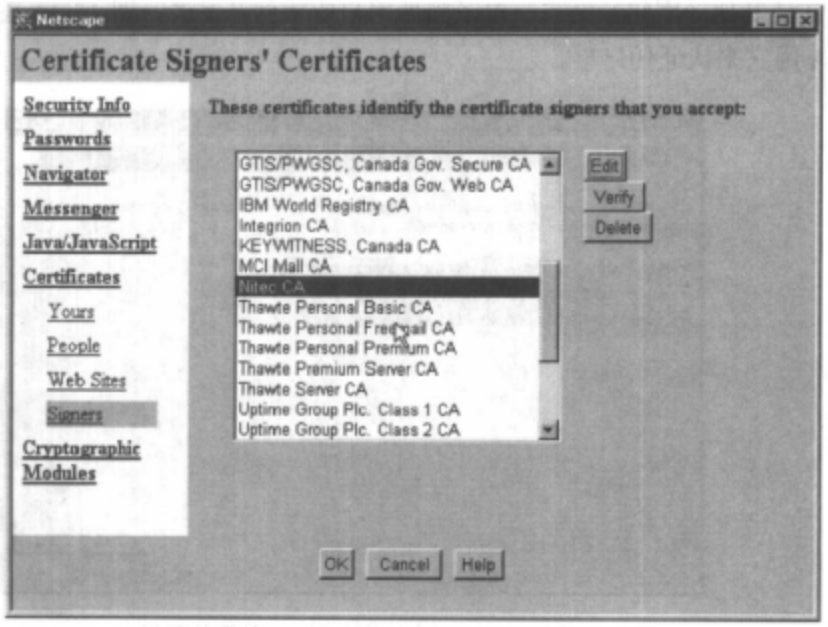

图 15-16 专用 CA 证书与商业 CA 证书一起显示

15.4.4.2 配置 SSLeay

在默认情况下，SSLeay 安全方针可能太松散，不足以满足专用 CA 的需要。例如，默认 SSLeay 配置允许服务器证书请求只提供公共名信息，其他的都是可选的。

如果使用专用 CA 为你的组织提供严格的安全环境，需要改变 SSLeay 配置文件的方针段。在配置文件 ssleay.conf（可以在$SSLTOP/CA 目录中找到）中，可以找到设置所用方针的那一行。因为方针的默认值是 policy_anything，所以这一行的内容是这样的：

```
policy = policy_anything
```

在默认配置文件中，两种方针都是可行的。默认的方针定义如下：

```
# For the 'anything' policy
 [ policy_anything ]
countryName                = optional
stateOrProvinceName        = optional
localityName               = optional
organizationName           = optional
organizationalUnitName     = optional
commonName                 = supplied
emailAddress               = optional
```

可以看到，除了 commonName 之外其他的项都是可选择的。另一个方针叫 policy_match，它是这样定义的：

```
#  For the CA policy
 [ policy_match ]
countryName                = match
stateOrProvinceName        = match
organizationName           = match
```

```
organizationalUnitName    = optional
commonName                = supplied
emailAddress              = optional
```

这种方针要求服务器认证请求与 CA 的国家名、州名或省名、组织名匹配。同时它也要求在请求中提供公共名。现在可以改变这两种方针（policy_match 和 policy_anything）或创建一种新方针。

创建新方针的最简单的方法是拷贝前述的方针，并把它的值改为所需要的值。值的选项有三种：

- match——字段内容必须与 CA 证书中的相同字段的值相匹配。
- optional——在证书请求中不需要这个字段。
- supplied——在证书请求中需要提供这个字段。

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

每个方针的参数字段都要与证书请求字段相符合，并且必须有这三种值中的一种。如果希望用新方针替代默认值，应该确保在括号中给新方针填入一个新名称；并确保把这个方针设置为当前方针。

如果新方针叫 my_policy，可以用下面的语句定义方针段（my_policy）：

```
policy = my_policy
```

通常情况下，只需要在 ssleay.conf 文件中编辑方针参数字段。如果想编辑其他字段，要仔细阅读文件内容。

现在可以准备接收证书请求（CSR）并签署它了！

15.4.4.3 签署证书

实际上，在开始签署服务器证书前，应该建立另一项方针。它指出要求使用哪一种证明方式以什么格式对证书进行处理。也就是验证原始的 CSR 请求。

一旦通过电子邮件或 Web 站点的 CGI 脚本得到一个 CSR 请求，可以把 CSR 的内容存在一个临时文件中，并把它作为参数传给 sign_csr 脚本。例如：

```
sign_csr  /tmp/a_csr_request_file
```

此时需要输入你的 CA 口令，CA 口令是在通过 makeca 脚本生成 CA 时建立的。sign_csr 要求检查请求中的签名、打印所请求的字段、提示决定是否签署这个请求。如果你认为提供的所有信息都是正确的，并与得到的证书申请的其他认证表单（通常是书面的）符合，就可以签署这个请求。脚本接着提醒你提交签名。

如果输入 Y，程序将打印新证书的内容，并把它存在$SSLTOP/CA/new_certs 目录中。

在 Stronghold 中没有创建 new_certs，因此用手工创建它。

每个新证书的文件名是这个 CA 签署的所有的证书的序列号。例如，签署的第一个证书是 01.pem，下一个是 02.pem 等等。一旦创建了证书，就应该把证书传给申请者，然后，申请者就可以安装它。

为 blackhole.nitec.com 提出一个 CSR 请求，并用前面讨论的 Nitec CA 工具对它进行

签署。然后将得到的证书安装在 Stronghold 服务器。如果一个也安装了 Nitec CA 证书（在早些的例子中）的浏览器访问这一服务器，服务器将不显示那些令人厌烦的警告对话窗口，发出“不认识服务器的证书”之类的消息。这正是所期待的，因此可从 Netscape Navigator 菜单选择安全信息选项，单击显示证书按钮，将看到如图 15-17 所示的窗口。

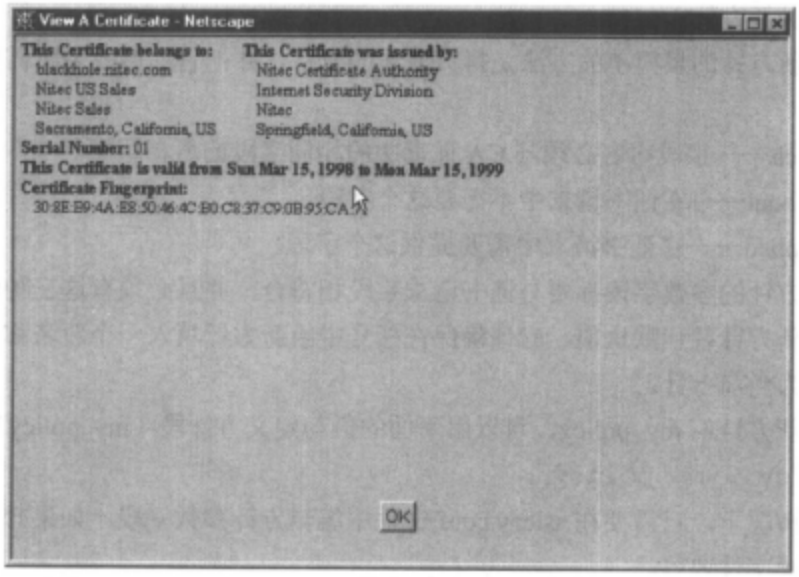

图 15-17 Nitec CA 认证的 blackhole.nitec.com 服务器

所以，为了建这个精彩的自签署的 CA，只需按如下步骤去做：

1. 为 blackhole.nitec.com 产生 CSR。
2. 建立称为 Nitec CA 的 CA。
3. 客户浏览器下载并安装 Nitec CA 证书。
4. 使 Nitec CA 签署 CSR。
5. 把 Nitec CA 签署的服务器证书安装在 blackhole.nitec.com 上。

因为所有这些步骤都在同一个组织中完成，这个过程几乎是不费劲的，在很短时间就完成。在现实情况下，在这个过程中还有其他事情要做，例如，建立接收 CSR 请求的电子邮件的特殊帐号，和发送已签署的服务器证书的帐号等。

15.5 小　　结

在本章中，主要讨论了什么是 SSL 和为什么在默认情况下 Apache 不支持 SSL。还讨论了怎样在 Apache 中使用非商业的 Apache-SSL 实现 SSL、怎样在从 Apache 衍生的商业服务器——Stronghold 中实现 SSL。最后讨论了证书、证书颁发机构和怎样创建自己的专有证书颁发机构。

下一章讨论 URL 重写工具。

第16章 重 写 URL

本章内容:

- Apache 的 URL 重写引擎
- URL 格式
- 内容处理
- 访问限制

超星阅览器提醒您:
使用本复制品
请尊重相关知识产权!

URL 把访问者带到你的 Web 站点。作为 Apache 的管理人员，需要确保到你的 Web 站点的所有可能 URL 都能发挥它们应有的作用。怎么办呢？应该保持监视服务器错误日志中的被断开 URL 请求。如果发现返回 404（即没找到）状态码的请求，则需要检查这些 URL。通常，HTML 文档设计人员更新 Web 站点后，忘了重新命名一个已存在的目录的情况时有发生，这将导致用户用书签保存的大量 URL 不能进行访问。

作为管理人员，怎样解决这个问题呢？值得高兴得是，有一个称为 mod_rewrite 的模块不仅可以帮管理人员解决这个问题，而且让管理人员根据 URL 重写规则创建其他有趣的解决办法。在本章中将讨论这个模块，并提供一些重写 URL 的实际例子。

在当前所有的可用 Apache 模块中，mod_rewrite 显得很突出。由于下面两个原因，它是一个很成熟的模块：

- 它为 Apache 管理人员提供很好和很实用的重写 URL 的灵活性。
- 它具有组织良好的文档。

现在来看看 mod_rewrite 是怎样工作的。

16.1 Apache 的 URL 重写引擎

当 Apache 接收 URL 请求时，它处理请求的方法是把文件传给客户（Web 浏览器）。如果需要干预这个处理过程，将 URL 指向不同的文件，或甚至一个不同的 URL，那该怎么办呢？

这正是 mod_rewrite 要解决的问题。mod_rewrite 提供了灵活的机制来建立 URL 规则。URL 规则具有如下的格式：

```
regex-pattern-to-be-matched      regex-substitution-pattern
```

当然，还可以将条件加到规则中（例如更多的 regex-pattern-to-be-matched），使得只在满足条件时，才进行替代。被替代的 URL 由 Apache 作为内部的子请求处理，或把它作为外部的重新定向传回 Web 浏览器。为了更好理解这一点，看看图 16-1 的例子，它包括请求和使用 mod_rewrite 规则得到的结果。

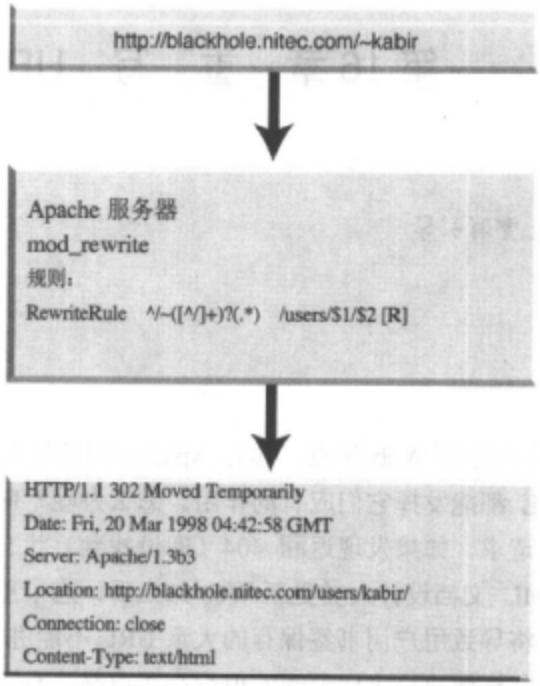

图 16-1 基于规则的重写 URL 操作的例子

其中，向 Apache 服务器提出 http://blackhole.nitec.com/~kabir 的请求。服务器接受请求，然后在处理请求的 URL 转换阶段将它传给 mod_rewrite 模块。mod_rewrite 模块应用 RewriteRule 命令定义的重写规则。在这个例子中，规则表示：如果匹配/~([^/]+)/?(.*)这样的模式，应该用/users/$1/$2 来代替它。因为规则中有一个[R]标志，所以外部的 URL 重新定向将被送回 Web 浏览器。输出结果表示，重新定向的位置是 http://blackhole.nitec.com/user/kabir。

正如上所见，在许多情况下都很容易出现这种重新定向。下面来看一看重写 URL 使用的命令。同时也应该熟悉这些可用于许多重写规则和条件中的服务器变量：

SERVER_NAME	Web 服务器的主机名
SERVER_ADMIN	Web 服务器管理人员的电子邮件地址
SERVER_PORT	Web 服务器的端口地址
SERVER_PROTOCOL	Web 服务器使用的 HTTP 协议版本
SERVER_SOFTWARE	Web 服务器软件的发行机构
SERVER_VERSION	Web 服务器软件的版本号
DOCUMENT_ROOT	Web 站点的顶层文档目录
HTTP_ACCEPT	Web 客户所接受的 MIME 类型
HTTP_COOKIE	从 Web 客户接收的 cookie
HTTP_FORWARD	转寄 URL
HTTP_HOST	Web 服务器的主机名

HTTP_PROXY_CONNECTION	HTTP 代理连接信息
HTTP_REFERER	引用当前 URL 的 URL
HTTP_USER_AGENT	Web 客户的信息
REMOTE_ADDR	Web 客户的 IP 地址
REMOTE_HOST	Web 客户的主机名
REMOTE_USER	已认证用户的用户名
REMOTE_IDENT	远程用户的识别信息
REQUEST_METHOD	用于请求当前 URL 的 HTTP 请求方法
SCRIPT_FILENAME	被请求脚本文件的实际路径
PATH_INFO	被请求 URL 的路径
QUERY_STRING	与被请求 URL 一起发送的查询数据
AUTH_TYPE	所使用的认证的类型
REQUEST_URI	被请求的 URL
REQUEST_FILENAME	同 SCRIPT_FILENAME 一样
THE_REQUEST	被请求的 URL
TIME_YEAR	当前年份
TIME_MON	当前月份
TIME_DAY	当前日期
TIME_HOUR	当前小时
TIME_MIN	当前分钟
TIME_SEC	当前秒
TIME_WDAY	当前星期
TIME	当前时间
API_VERSION	所使用的 API 版本
IS_SUBREQ	如果请求是子请求就设置

16.1.1　RewriteEngine

语　　法：`RewriteEngine on | off`

缺 省 值：`RewriteEngine off`

使用范围：服务器配置，虚拟主机，针对目录的配置文件(.htaccess)

这个命令为 mod_rewrite 模块中的 URL 重写引擎提供开关。默认情况下，所有的重写都是关闭的。为了使用重写引擎，必须把这个命令置为 on。

当允许使用 URL 重写针对目录的配置文件（.htaccess）时，必须把这个命令放在针对目录的配置文件中（设为 on），并保证在合适的位置允许使用下面的命令：

```
Options FollowSymLinks
```

也就是说，如果该目录属于虚拟主机站点，要保证在相应的虚拟主机容器中允许使用这个选项。类似的，如果目录是主服务器的 Web 文档的一部分，要保证在主服务器配置

中允许使用这个选项。

注意，在针对目录的配置文件中允许使用重写规则会降低 Apache 服务器的性能。这是因为 mod_write 使用了一种技巧来支持针对目录的重写规则，而这种技巧会增加服务器的处理负载。因此，只要可能，应该避免在针对目录的配置文件中使用重写规则。

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

16.1.2 RewriteOptions

语　　法：RewriteOptions option1 option2 ...

缺 省 值：无

使用范围：服务器配置，虚拟主机，针对目录的配置文件(.htaccess)

这个命令指定用于改变重写引擎的行为的选项。当前唯一的可选项是继承。把这个命令设为继承选项，可以强制低层的配置继承高层的配置。例如，如果在主服务配置中设置了这个命令，在配置文件中定义的虚拟主机就继承所有的重写配置，例如重写规则、条件、映射等等。

类似的，如果在针对目录的配置文件（.htaccess）中设置这个命令，它的子目录将继承父目录的重写规则、条件和映射等。在默认情况下，重写引擎不允许重写配置的继承，但这个命令允许改变默认值。

16.1.3 RewriteRule

语　　法：RewriteRule search-pattern substitution-string
[flag list]

缺 省 值：无

使用范围：服务器配置，虚拟主机，针对目录的配置文件(.htaccess)

这个命令允许定义重写规则。规则必须有两个参数。第一个参数是搜索模式，必须符合搜索模式才使用代替字符串。搜索模式是用规则表达式写的（参见附录 B：规则表达式基础）。代替字符串可以使用纯文本、反向引用搜索模式中的子字符串、服务器变量的值，或者甚至是映射函数。标志列表可以容纳一个或更多的由逗号分隔标志串，它用于告诉重写引擎替换后下一步干什么。看一看前面的例子：

```
RewriteRule  /~([^/]+)/?(.*)  /user/$1/$2  [R]
```

这里，搜索模式是 /~([^/]+)/?(.*)，替换字符串是 /users/$1/$2。注意在替换字符串中使用了反向引用。第一个反向引用串是$1，它替换第一个括号中的内容。$1 匹配的是([^/]+)，$2 匹配的是(.*)。如果有一个 URL 如下：

```
http://blackhole.nitec.com/~kabir/welcome.html
```

那么$1 的值是 kabir，$2 的值是 welcome.html，因此替换串是：

```
/users/kabir/welcome.html
```

如果指定的重写规则多于一个时，第一个重写规则将对原始 URL 进行操作。如果匹配，第二条规则操作的就不是原始 URL，而是第一条规则替换后的 URL。假如每一步都匹配，那么三条重写规则的作用如下所示：

```
RewriteRule  search-pattern-for-original-URL  substitution-
string-1  [flags]
RewriteRule  search-pattern-for-substitution-string-1
substitution-string-2 [flags]
RewriteRule  search-pattern-for-substitution-string-2
substitution-string-3  [flags]
```

是否可能将一条以上的规则用于原始 URL？是的。可以使用 C 标志来指示重写引擎链接了多条规则。在这种情况下，可能希望所有的规则都匹配后再替换，此时可以用一个特殊的替换字符串使规则中的替换不起作用。

可能的标志的详细情况如下所示：

C | chain	这一标志指定当前的规则链接着下一规则。如果使用 C 标志进行链接，只有链中的前一规则匹配才继续比较下一条规则。链中的每个规则都必须包含这一标志，如果第一个规则不匹配，整个规则链将被忽视。
E=var:value | env=var:value	你可以使用这个命令设置环境变量。该变量可以由重写条件、SSI 和 CGI 脚本等访问。
F | forbidden	如果使用这个标志的规则匹配成功，将一个称为 FORBIDDEN 的 HTTP 响应头标（状态代码为 403）返回给浏览器。它将有效地禁止被请求的 URL。
G | gone	如果使用这个标志的规则匹配成功，将一个称为 GONE 的 HTTP 响应头标（状态代码为 410）返回给浏览器。它通知浏览器被请求的 URL 在这个服务器上不再使用。
L | last	它告诉重写引擎立即结束规则处理，因此就没有其他的规则用于最后被替换的 URL。
N | next	它告诉重写引擎从第一条规则重新开始。但是，第一条规则不再去试图匹配原始 URL，因为它现在操作的是最新被替换的 URL。它等效于创建了一个循环，在循环中必须有终止条件，这样才能避免一个死循环。
NS | nosubreq	使用这个标志可以避免把规则用在内部产生的请求上。
P | proxy	使用这个标志可以把一个 URL 请求转换为内部的代理请求。如果把 Apache 和 mod_proxy 模块一起编译，并把它配置为使用代理模块，它才能工作。
PT | passthrough	这是老而无用的标志，在新的 Apache 版本中可能会消失。这个标志强制重写引擎改变内部请求的记录结构，例如把结构的 URL 元变量设为文件名元变量。只当使用 mod_rewrite 模块和其他包含 URL-to-filemane 转换程序的模块时，才使用这个标志。
QSA | qsappend	这一标志允许向被替换的 URL 的查询字符串部分添加数据，例如 key=value 对。

S=n | skip=n　　跳过 n 条规则。

T=mime-type | type=mime-type　　强制将请求的目标文件的 MIME 类型改变为特殊的 MIME-类型。

超星阅览器提醒您：使用本复制品需尊重原作者的版权！

通过在规则前面使用一个或更多的 RewriteCond 命令，即可为规则增加条件。下面将要讲 RewriteCond 命令。

16.1.4 RewriteCond

语　　法：RewriteCond test-string condition-pattern [flag list]

缺 省 值：无

使用范围：服务器配置，虚拟主机，针对目录的配置文件(.htaccess)

如果需要为 RewriteRule 命令指定的规则增加附加条件，这个命令很有用。每个 RewriteRule 命令可以有几个 RewriteCond。必须在规则前指定所有的重写条件。

测试字符串可以由下面的一些内容组成：纯文本、服务器变量，或者对当前的重写规则和最后的重写条件的反向引用。为了从当前的 RewriteRule 命令访问第一个反向引用，使用$1，为了从最后的 RewriteCond 命令访问第一个反向引用，使用%1。

为了访问服务器变量，可以使用%{变量名称}的格式。例如，为了访问 REMOTE_USER 变量，可以在测试字符串中指定%{REMOTE_USER}。

下面是一些特殊的数据访问格式：

%{ENV: variable}　　使用它可以访问适用于 Apache 进程的任何环境变量。

%{HTTP: header}　　使用它可以访问用于请求中的 HTTP 头标。

%{LA-U: variable}　　使用它可以访问在当前处理阶段不可用的变量。例如，如果需要用到存在服务器配置文件（httpd.conf）的重写条件中的 REMOVE_USER 服务器变量，你不能使用%{REMOVE_USER}，因为这个变量只在服务器执行了认证阶段之后才被定义，认证阶段跟在 mod_rewrite 的 URL 处理阶段的后面。为了预先看到认证成功的用户名，可以使用%{LA-U:REMOTE_USER}。然而，如果从针对目录的配置文件中访问 REMOTE_USER 数据，可以使用%{REMOVE _USER}，因为此时认证阶段已经完成、服务器变量已经象通常那样变为可用。查找是通过生成基于 URL 的内部子请求来实现的。

%{LA-F:variable}　　在大多数情况下，它与%{LA-U:variable}相同。但是，查找是使用基于文件名的内部子请求来实现。

为了形成规则表达式，条件模式也可以使用一些特殊符号。例如，如果希望在测试字符串和条件模式之间进行词汇比较，只要在条件模式加上前缀<、>、=字符即可。这样，条件模式作为纯文本字符串与测试字符串进行比较。

同时注意有时需要检查测试字符串是否是一个文件、目录或符号链接。在这种情况下，可以用下面的特殊字符串代替条件模式：

-d　检查由测试字符串指定的目录是否存在。

-f　检查由测试字符串指定的文件是否存在。

-s　检查由测试字符串指定的非空文件是否存在。

-l　检查由测试字符串指定的符号链接是否存在。

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

-F　检查由测试字符串指定的文件是否存在和可以访问。

-U　检查由测试字符串指定的 URL 是否有效和可以访问。

选项标志列表可以包含如下所示的一个或多个用逗号分隔的字符串：

NC | nocase　执行不区分大小写字母的条件测试。

OR | ornext　当一个 RewriteRule 命令中有多个 RewriteCond 时，通常这些条件之间的关系是 AND。如果希望在两个条件间建立 OR 关系，可以使用这个标志。

16.1.5 RewriteMap

语　法：`RewriteMap name-of-map type-of-map:source-of-map`

缺 省 值：无

使用范围：服务器配置，虚拟主机

这个命令通过使用映射（map）使关键字到值（key-to-value）的查找变得容易。可以把映射看作一个数据表，表的每一行有一个关键字和一个值。通常将 map 存在文件中。但 map 可以是文本文件、DBM 文件、内部 Apache 函数，也可以是一个外部程序。映射的类型与映射的源相同。可以使用的映射类型有：

txt　具有关键字和值的行的普通文本文件。每个关键字/值对就是一行，行中至少用一个空白字符分隔。该文件可以包含以#开始的注释，可以有空白行。注释和空白行都将被忽略。例如：

```
key1    value1
key2    value2
```

定义了两个关键字/值对。注意，基于文本文件的映射在 Apache 启动时被读入，并且只在服务器已启动运行后文件被修改时才被重读。在服务器重新启动时，这个文件也将被重读。

rnd　一个特殊的普通文本文件，它具有 txt 类型的所有限制，但在定义值时允许有灵活性。可以使用 |（竖线）字符作为“逻辑或”将每个关键字的值定义为一系列的值。例如：

```
key1  first_value_for_key1  |  second_value_for_key1
key2  first_value_for_key2  |  second_value_for_key2
```

它定义了两个关键字/值对，这里每个关键字有多个值。可以任意选择其中的一个值。

int　内部的 Apache 函数 toupper（关键字）和 tolower（关键字）可以用作 map 源。第一个函数把关键字转换为大写，第二个函数把关键字转换为小

缺 省 值：RewriteLogLevel　0

使用范围：服务器配置，虚拟主机

这个命令允许指定在日志文件中记录什么。默认值 0 表示什么也不记录。实际上，0 级记录表示在模块中不进行记录处理。因此，如果希望禁止记录，将它置为 0。

注意，如果把 RewriteLog 命令设为/dev/null，并把 RewriteLogLevel 设为非 0 值，那么内部的与记录有关的处理将发挥作用，但没有产生记录。这在浪费系统的计算资源。记录级别为 0~9。记录级别越高，所写的记录数据越多。

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

16.1.9　RewriteLock

语　　法：RewriteLock　filename

缺 省 值：无

使用范围：服务器配置，虚拟主机

如果使用外部映射程序来创建重写映射，那么需要用这个命令指定一个文件名。这个文件用作锁文件，用于与外部映射程序的同步传输。

16.2　URL　格　式

在本节中将看到一些 URL 重写的例子，它们处理的是 URL 的格式。经常需要把一个 URL 请求重新定向或扩展为另一个 URL。在下面的例子中，可以看到 mod_rewrite 是怎样做的。

16.2.1　把请求的 URL 扩展为规范的 URL

提供用户主页的 Web 站点通常支持如下的 URL 方案：

```
http://hostname/~username
```

这是简捷的 URL 表示方式，它需要映射为规范的 URL。可能还有其他简捷的 URL 表示方或内部的 URL 需要被扩展为规范的 URL。在这个例子中，~username 被转换为/u/username。图 16-2 表明这一转换的过程。

当收到对 http://hostname/~username 的请求时，重写规则将把它转化为/u/username，并把新 URL 重新定向给浏览器。接着，浏览器重新请求 http://hostname/u/username URL，并由通常的 Apache 请求处理实现这个请求。

外部的 HTTP 重新定向是必要的，因为任何后继的请求也都必须用转换后的规范 URL 代替~username。转换的规则是这样的：

```
Rewriterule  ^/~([^/]+)/?(.*)   /u/$1/$2  [R]
```

许多有成千上万用户的 ISP 站点使用结构化的 home 目录方案，即每个用户的主目录在以用户名的第一个字符命名的子目录下。因此，/~foo/anypath 就变为/home/f/foo/www/anypath，而 /~bar/anypath 就变为/home/b/bar/www/anypath。在这种情况下，为了把简捷的 URL 表示方式转换为规范的 URL，可以使用下面的规则：

```
RewriteRule ^/~(([a-z])[a-z0-9]+)(.*)  /home/$2/$1/www$3
```

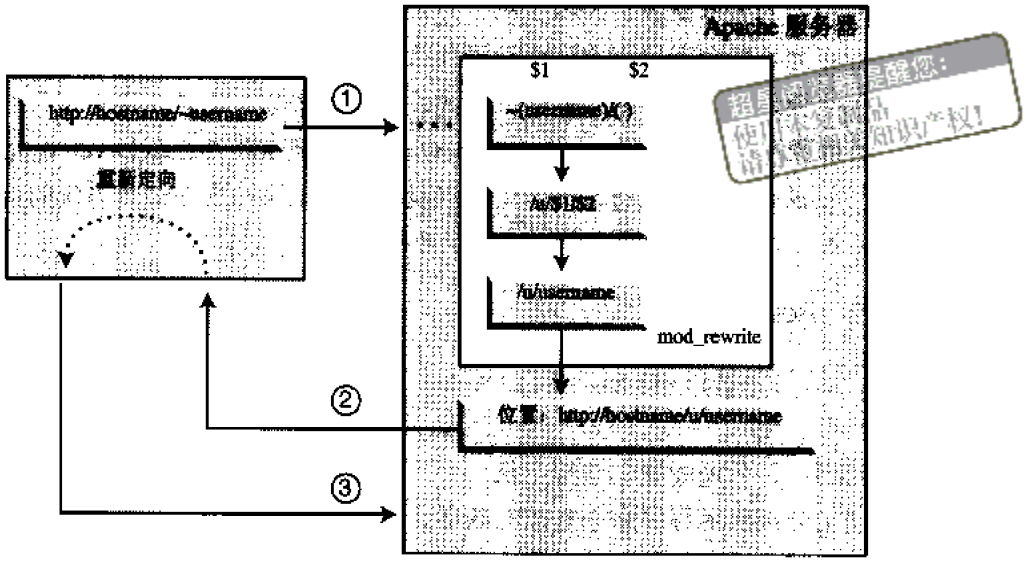

图 16-2 把请求的 URL 扩展为规范的 URL

16.2.2 把用户主目录重新定向给新的 Web 服务器

如果在 Web 服务器有许多用户主目录，并且出于某种原因需要把它们移到一个新机器上，需要有如图 16-3 所示的重新定向规则。

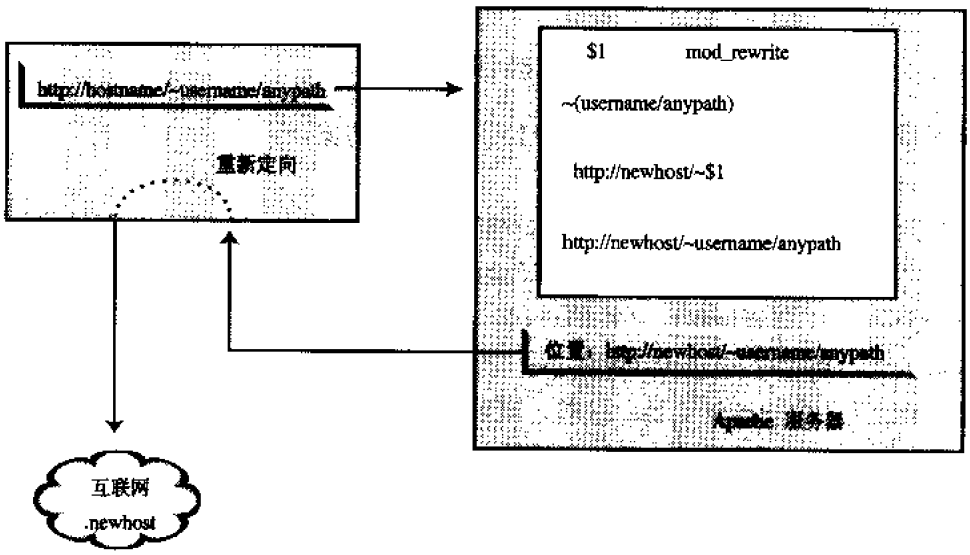

图 16-3 把用户主目录重新定向给新的 Web 服务器

有了 mod_rewrite 解决方法变得很简单。在旧的 Web 服务器上使用如下的规则就可以把所有的/~user/anypath URL 重新定向为 http://newserver/~user/anypath。

```
RewriteRule  ^/~(.+)  http://newserver/~$1  [R,L]
```

注意 L 标记表示没有其他的重写规则可以用于被替换的 URL。

16.2.3　在多个目录中查找一页

有时需要 Web 服务器在多于一个目录中查找一页。这里 MultiViews 或其他技术都没有帮助。例如，假设需要处理对 http://hostname/filename.html 的请求，如果 filename.html 不在 Web 服务器的 dir1 目录中，那么需要服务器去 dir2 子目录试一试。图 16-4 说明了这个过程。

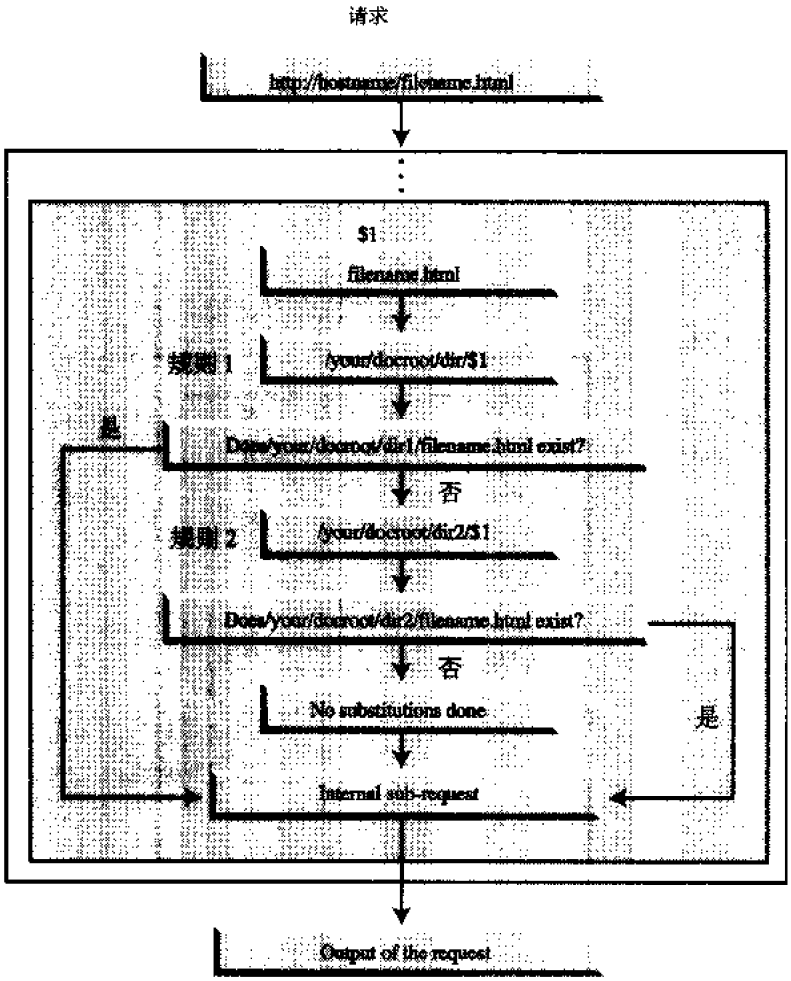

图 16-4　在多个目录中查找一页

需要执行的规则如下所示：

```
RewriteCond/your/docroot/dir1/%{REQUEST_FILENAME} -f
RewriteRule  ^(.+)     /your/docroot/dir1/$1  [L]

RewriteCond/your/docroot/dir2/%{REQUEST_FILENAME} -f
RewriteRule  ^(.+)     /your/docroot/dir2/$1  [L]
```

```
RewriteRule  ^(.+)  -  [PT]
```

只有当被请求文件在 your/docroot/dir1/子目录下，第一条规则才用/your/docroot/dir1/$1（这里$1 是请求中的目标文件）替换被请求的 URL。如果条件匹配，那么它是用于这个 URL 的最后规则。如果不匹配，则适用下一条规则。这条规则与第一条规则做同样的事，但这一次路径中是 dir2 子目录。如果这条规则的条件匹配，那么这条规则是最后规则。如果没有规则匹配，请求不被替换并像平常一样传给 Apache。

16.2.4 设置基于 URL 的环境变量

可能需要保持请求之间的状态信息并使用 URL 对它编码。但并不想只是为了得到状态信息就对所有页使用 CGI wrapper 脚本。此时可以使用重写规则得到状态信息，并通过环境变量将它保存起来，这个环境变量可以从 XSSI 或 CGI 废弃。这样，URL /foo/S=java/bar/ 将被转换为 /foo/bar/，环境变量 STATUS 将被设置为 java 值。图 16-5 说明了这个过程。

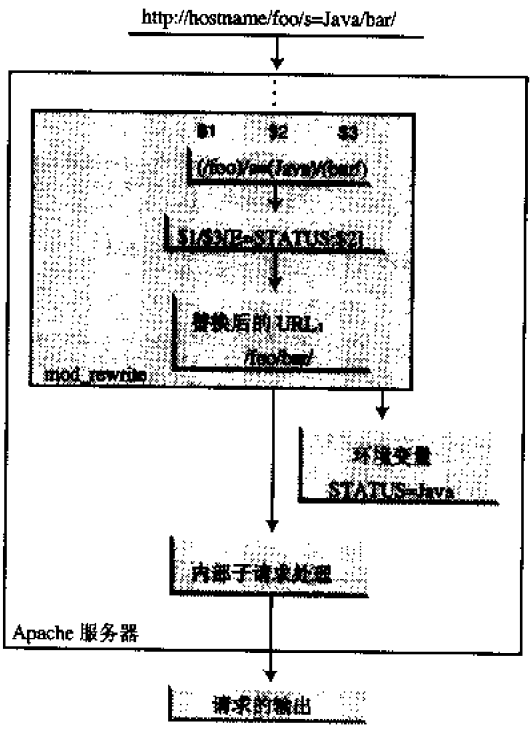

图 16-5 设置基于 URL 的环境变量

图 16-5 中显示的 URL 重写步骤可以通过下面规则实现：

```
RewriteRule  ^(.*)/S=([^/]+)/(.*)  $1/$3  [E=STATUS:$2]
```

这里$3 的值使用 E 标记存在环境变量 STATUS 中。

16.2.5　创建 www.username.host.com 站点

假设需要给 www.username.host.com 提供用户名的主页，只有通过同一台机器的 DNS 地址（A）记录，该机器上没有任何虚拟主机。对于 HTTP/1.0 请求没有解决方法，对于 HTTP/1.1 请求，它含有 Host:HTTP 头标，可以使用下面的重写规则把 http://www.username.host.com/anypath 内部转换为 /home/username/anypath：

```
RewriteCond  %{HTTP_HOST}   ^www\.[^.]+\.host\.come$
RewriteRule  ^(.+)    %{HTTP_HOST}$1   [C]
RewriteRule  ^www\.([^.]+)\.host\.com(.*)  /home/$1/$2
```

图 16-6 说明了这个过程。

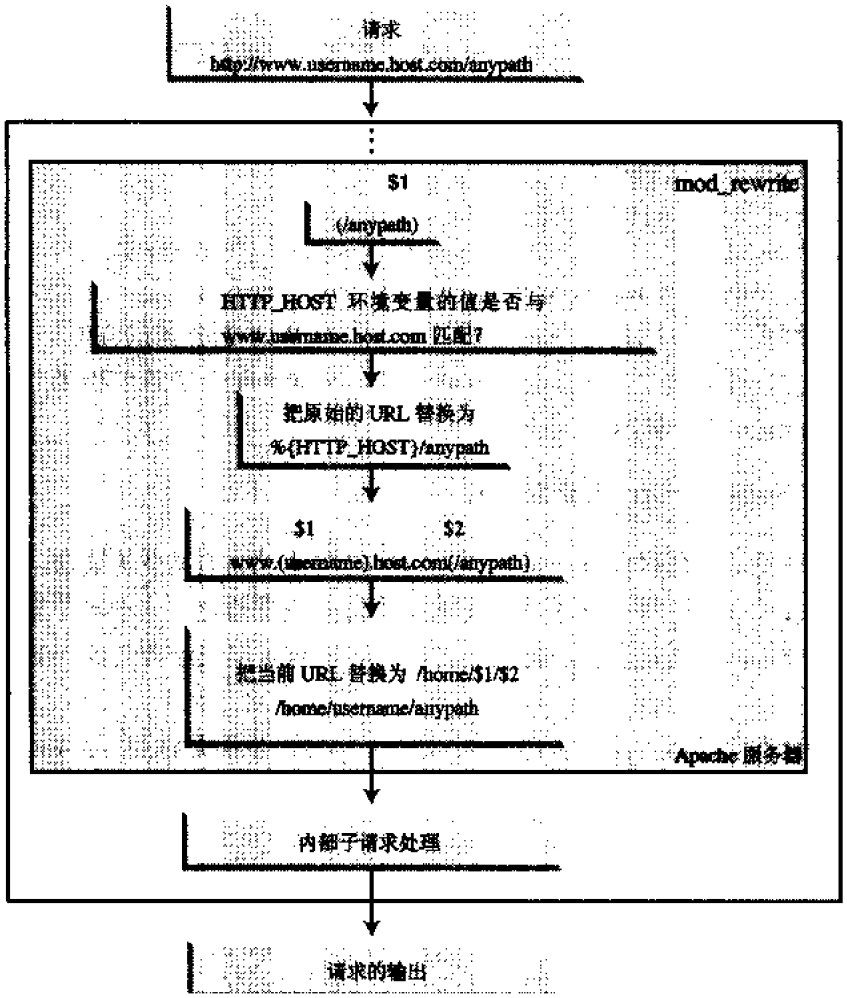

图 16-6　对每个用户名的虚拟主机

这是规则链的例子。第一条规则的条件是检查环境变量 HTTP_HOST 是否与类似 www.username.host.com. 的模式匹配。如果匹配，规则就起作用。也就是说，一个 www.username.host.com/anypath 请求将被替换为 www.username.host.com/anypath。可能会觉得混淆，因为变化很不明显。但它是必要的，这样用户名才可用第二条规则提取。第二

条规则处理被替换的请求的用户名部分，并为内部子请求产生一个新 URL /home /username/anypath。

16.2.6 把访问失败的 URL 重新定向给另一个 Web 服务器

超星阅览器提醒您：
使用本复制品
须遵守相关知识产权！

如果拥有多个服务器的 Web 网络，经常需要将内容从一个服务器移至另一个服务器，可能会面对这样的问题：需要把访问失败的 URL 从 Web 服务器 A 重新定向给 Web 服务器 B。有许多方法可以做到这一点：可以使用 ErrrorDocument 命令（参见第 4 章）；写一个 CGI 脚本来处理它；或者使用 mod_rewrite 将失败的 URL 重新定向给其他服务器。注意，使用 ErroeDocument CGI 脚本的方法比基于 mod_rewrite 的方法更好。第一种方法具有最好的性能，但灵活性和安全性较差。

```
RewriteCond  /your/docroot/%{REQUEST_FILENAME} ! -f
RewriteRule  ^(.+)          http://web ServerB.dom/$1
```

这里的问题是它只为 DocumentRoot 目录下的页工作。可以增加条件（例如，也处理主目录），这样的效果就更好。

```
RewriteCond  %{REQUEST_URI}  ! -U
RewriteRule  ^(.+)   http://Web serverB.dom/$1
```

它利用了 mod_rewrite 的预先使用的特征，可以为所有类型的 URL 工作。然而，由于每个请求都多了一个内部子请求，所以它将影响 Web 服务器的性能。如果 Web 服务器运行在强有力的计算机上，可以使用这种方法。如果不是，应该使用第一种方法，或者使用 ErrorDocument CGI 脚本。

16.2.7 创建一个访问多路复用器

这个例子将显示怎样创建一条规则来重新定向基于域的类型的请求，例如：.com、.net、.edu、.org、.uk、.de 等等。它的基本想法是将访问者重新定向到地理位置最近的 Web 站点。许多大公司都使用这种技术把它们的国际客户重新定向给合适的 Web 站点或 FTP 服务器。

实现这种方案的第一步是创建 map 文件。例如，下面是称为 site-redirect.map 的基于文本的 map 文件。

```
com     http://www.mydomain.com/download/
net     http://www.mydomain.com/download/
edu     http://www.mydomain.com/download/
org     http://www.mydomain.com/download/
uk      http://www.mydomain.uk/download/
de      http://www.mydomain.de/download/
ch      http://www.mydomain.ch/download/
```

如果收到来自 dialup001.demon.uk 主机的对 http://www.mycompany.com/download /anypath 的请求，它需要被重新定向到 www.mydomain.uk/download/ Web 站点；类似的，任何来自属于顶层域名（TLD）.com、.net、.edu 和.org 的主机的请求都将传给 www.mycompany.com/download/ 站点。

下面的规则是上面的设置所需要的：

```
RewriteMap  sitemap    txt:/path/to/site-redirect.map
RewriteRule  ^/download/(.*)   %{REMOTE_HOST}::$1  [C]
RewriteRule  ^.+\.([a-zA-Z]+)::(.*)$
%{sitemap:$1|www.mydomain.com/download/}$2  [R,L]
```

超星阅览器提醒您：
被本作制品
请尊重相关知识产权！

图 16-7 说明这个规则的使用。

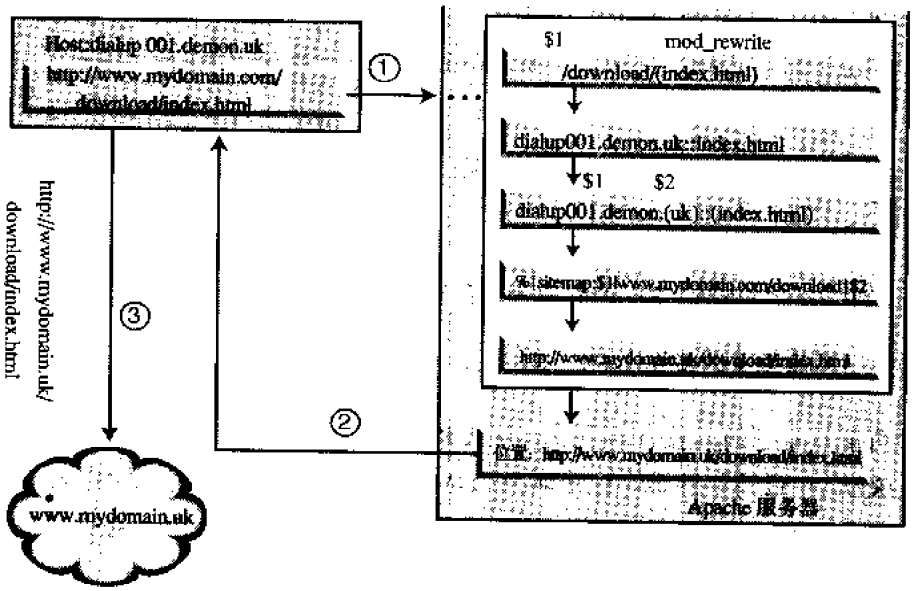

图 16-7　基于 URL 的访问多路复用器

如图 16-7 所示，当一个主机，例如 dialup001.demon.uk，请求 www.mydomain.com/download/index.html 页时，第一条规则用请求主机的主机名重写请求：

```
dialup001.demon.uk::index.html
```

接着应用第二条规则。当搜索模式匹配时，第二条规则适用。查询 map 文件找到对应于 TLD 的替换 URL。当搜索模式不匹配时，则使用默认的 www.mydomain.com。它由替换 URL 中的 |（or）操作符完成。使用如列表 16-1 的算法更容易理解第二条规则。

列表 16-1：第二条重写规则的算法

```
if(current URL matches a fully-qualified-hostname::anything) then

#  substitute the current URL using the domain type information
stored in $1 perform a
# lookup in the map file.

If (map file has a key that matches the domain type) then
#  use the key's value as follows:
```

```
    Substituted URL = value-of-the-key$2  # where $2 is
anything after

#  fully-qualified-hostname::  pattern

Else
    #  Use default value www.mydomain.com/download/$2
    Substituted URL = www.mydomain.com/download/$2
endif

Endif
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

R 标志表示这个外部 URL 重新定向，L 标志表示这是替换 URL 的最后一个规则。

16.2.8 创建基于时间的 URL

是否可能有基于时间指向不同文件的 URL 呢？是的，mod_rewrite 可以很容易地创建这样的 URL。在重写条件中有许多叫作 TIME_xxx 变量。使用特殊的词典式的比较模式 <STRING、>STRING、=STRING 就可以实现根据时间的重新定向，例如：

```
RewriteCond      %{TIME_HOUR}%{TIME_MIN} >0700
RewriteCond      %{TIME_HOUR}%{TIME_MIN} <1900
RewriteRule      ^foo\.html$             foo.day.html
RewriteRule      ^foo\.html$             foo.night.html
```

它表示从 07：00 到 19：00 提供的是 URL foo.day.html 的内容，其他时间提供的是 URL foo.night.html 的内容。

16.3 内 容 处 理

在本节中的例子涉及与内容有关的重写规则。将了解如何创建向后兼容的 URL、基于浏览器的内容重写、最终用户透明的 HTML 到 CGI 重新定向等等。

16.3.1 使 URL 向后兼容

假设在最近将 bar.html 页重新命名为 foo.html，而且需要使它与旧的 URL 向后兼容。此外，还希望让使用旧的 URL 的用户觉察不到该页已经重新命名。怎么办呢？下面就是答案：

```
RewriteRule  ^foo\.html$  bar.html
```

如果希望浏览器知道这个改变，可以写一个外部重写规则，这样浏览器将显示新 URL。这只需要增加 R 标记即可：

```
RewriteRule ^foo\.html$ bar.html [R]
```

16.3.2 创建传送的内容与浏览器类型有关的 URL

可以使用重写规则把不同的内容（使用内部子请求）送给不同的浏览器。但不能通过内容，因为浏览器不提供它们的类型。因此，需要使用 HTTP 头标 User-Agent。例如，如果浏览器的 HTTP 头标 User-Agent 与 Mozilla/3 匹配，那么可以发送 Netscape Navigator 3（或更高）的友好特征页，如果浏览器是旧的 Navigator 版本或是其他类型的，可以发送另一个不同的页。

如果 HTTP 头标 User-Agent 从 Mozilla/3 开始，foo.html 页被重写为 foo.NS.html 后，重写就停止。如果浏览器是 Lynx 或 Mozilla 版本 1 或 2，URL 将变成 foo.20.html。所有其他浏览器都接收 foo.32.html 页。它用如下规则完成：

```
RewriteCond %{HTTP_USER_AGENT} ^Mozilla/3.*
RewriteRule ^foo\.html$ foo.3x.html [L]

RewriteCond %{HTTP_USER_AGENT} ^Lynx/.* [OR]
RewriteCond %{HTTP_USER_AGENT} ^Mozilla/[12].*
RewriteRule ^foo\.html$ foo.2x.html [L]
```

如果接到一个请求，例如对 http://hostname/foo.html，第一个条件检查环境变量的值是否与字符串 Mozilla/3. 匹配。如果匹配，第一条规则就适用。这条规则把原始的 URL 替换为 foo.3x.html，并且所有的重写就完成了。然而，如果第一条规则不适用，则调用第二条规则。第二条规则有两个条件，它们之间是或的关系。也就是说，在规则适用前，必须有一个条件满足。

第一个条件检查同一个环境变量的值是否与子串 Lynx/ 匹配，第二个条件检查同一个环境变量是否与子串 Mozilla/1 或 Mozilla/2 匹配。如果条件中的任一个满足，规则就适用。该规则把原始的 URL 替换为 foo.2x.html。替换后的 URL 转变为一个子请求，由 Apache 做常规处理。

16.3.3 为 CGI 网关创建 HTML

可能需要将静态页 foo.html 以无痕方式转换为动态的变体 foo.cgi，也就是说不通知浏览器或用户。可以这样实现：

```
RewriteRule ^foo\ .html$ foo.cgi [T=application/x-httpd-cgi]
```

该规则将对 foo.html 的请求重写为对 foo.cgi 的请求。它同时强制纠正 MIME 类型。这样，它可以像 CGI 脚本一样运行。一个请求，例如 http://hostname/foo.html，被内部转换为对 CGI 脚本的请求。浏览器不知道它的请求被重新定向。

16.4 访 问 限 制

这些例子涉及访问控制方面的问题。能从中了解如何利用 URL 重写模块控制对 Web

站点的特定区域的访问。

16.4.1 阻塞机器人

阻塞机器人令人讨厌地检索特定站点的 Web 页是很容易做到的。可以试一试/robots.txt 文件，它包含 Robot Exclusion 协议条目。但通常 Robot Exclusion 协议条目还不足以完全摆脱机器人。下面是一个例子：

```
RewriteCond  %{HTTP_USER_AGENT}  ^NameOfBadRobot.*
RewriteCond  %{REMOTE_ADDR}  ^123\.45\.67\.[8-9]$
RewriteRule  ^/not/to/be/indexed/by/robots/.+  -  [F]
```

这条规则有两个条件，如下所示：

```
If (HTTP_USER_AGENT of the robot matches a pattern "NameOfBadRobot*" )
    and(REMOTE_ADDR of the requesting host is 123.45.67.8 to
    123.45.67.9 )  then
No substitution but send a HTTP "Forbidden" header (status code 403)
endif
```

就如看到的，机器人的用户代理头标与它使用的主机的 IP 地址相匹配。后一个条件允许检查多个 IP 地址（123.45.67.8 到 123.45.67.9）。

16.4.2 创建基于 HTTP 引用的 URL 转向功能

可以编写基于 HTTP 引用头标的灵活的 URL 转向功能，并且只要愿意，就可以为它配置许多引用页。它可以这样实现：

```
RewriteMap  deflector  txt: /path/to/deflector.map
RewriteRule  ^/(.*)  ${deflector:%{HTTP_REFERER}|/$1}
RewriteRule  ^/DEFLECTED  %{HTTP_REFERER}  [R,L]
RewriteRule  .*   [PT]
```

它常常与相应的重写 map 一起使用，例如：

```
http://www.badguys.com/bad/index.html  DEFLECTED
http://www.badguys.com/bad/index2.html  DEFLECTED
http://www.badguys.com/bad/index3.html   http://somewhere.com/
```

如果 URL 与 map 文件中的 DEFLECTED 值匹配，它将自动把请求重新定向回到引用页。在所有其他情况下，请求将被重新定向给特定的 URL。

16.5 小　　结

在本章中，讨论的是强有力的 URL 重写引擎——mod_rewrite。还讨论了如何使用 URL 重写规则解决许多实际问题。然而，如果管理人员不熟悉规则表达式，那么写 URL 重写规则将是一项考验。建议如果不熟悉规则表达式，在使用重写规则时一定要小心。

下一章将要讨论如何处理好影响 Web 服务器的每一件事。

第五部分　使用 Apache 的现在和将来

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

第 17 章　性能提示
第 18 章　运行完善的 Web 站点
第 19 章　使用 Apache 组建 Web 网络
第 20 章　用于 Windows 95/NT 的 Apache

第 17 章　性 能 提 示

本章内容:

- 精确调整计算机
- 提高 Apache 及其配置的效率
- 从网络获得更好的性能
- 开发高性能内容

为了最大限度发挥 Apache 服务器的性能，需要不断监视四个方面：计算机、网络、内容和所涉及的人员。在本章中，将对引起性能瓶颈有关的问题展开讨论，并说明如何解决它们。

需要说明的是：如果你是经常修改操作系统内核以获取最大性能的人，那么你大概不会从本章中学到任何新的知识。本章是为那些开始关注性能问题的人们而写的。还有，也许会感到主要讨论的是基于 PC 的 Apache 安装。这是出于这样的假设：大多数人是在基于 PC 的系统上运行 Apache。然而，这里所讨论的大部分内容都能用在所有的 Apache 平台上。

17.1　Apache 计算机

Apache 可运行在各种计算机上。尽管这些计算机在体系结构上有很大的差别，但是导致性能瓶颈的硬件组成部分实质上是一样的。

Note

顺便说一句，从有关性能的章节开始，假定所使用的 Apache 计算机是较流行的机器。如果用一台 PC 兼容机作为 Web 服务器，并且关注其性能，将假定它是奔腾级的系统而不是基于 i386 的系统。

在分析性能需求时，首先应该问自己一个问题："我所使用的 Apache 计算机需要快速的 CPU 吗？"问题的答案取决于如何使用 Apache 服务器。如果服务器提供的大部分内容是静态的 HTML，则 CPU 不会是重要的性能因素。另一方面，如果使用服务器端包含命令（SSI）、CGL 脚本等等产生了大量的动态内容，Apache 服务器就能从快速的 CPU 中获得好处。在这种情况，使用快速的 CPU 是明智的。

那么，多快才算足够快？没有什么是足够快的，越快越好。但必须清楚的认识到，如动态内容的容量很大，只有快速的 CPU 并不会得到多大好处。在这种情况，解决性能瓶颈的最好选择是加大 RAM。

但这也不能得到足够内存，而且内存也不便宜。那么多大的内存才能获得较好软件性能呢？要想回答这个问题，就必须经常监视系统的负载。例如，在大部分 UNIX 系统上，可以使用实用程序监视系统性能。图 17-1 显示了可容易获得的 UNIX 实用程序 top 的输出结果。

```
/bin/csh (ttyp2)
 9:35pm  up 18 days,  6:48,  1 user,  load average: 0.06, 0.05, 0.01
48 processes: 47 sleeping, 1 running, 0 zombie, 0 stopped
CPU states:  0.3% user,  0.5% system,  0.0% nice, 99.2% idle
Mem:  257068K av, 251904K used,   5004K free,  26556K shrd, 129228K buff
Swap: 128484K av,      0K used, 128484K free                96828K cached

  PID USER     PRI  NI  SIZE  RSS SHARE STAT  LIB %CPU %MEM   TIME COMMAND
22679 kabir     13   0   588  588   456 R       0  0.7  0.2   0:00 top
22548 httpd      1   0  1824 1824   876 S       0  0.1  0.3   0:00 httpd
    1 root       0   0   484  484   340 S       0  0.0  0.1   0:03 init
    2 root       0   0     0    0     0 SW      0  0.0  0.0   0:00 kflushd
    3 root     -12 -12     0    0     0 SW<     0  0.0  0.0   0:00 kswapd
22546 httpd      5   0  1048 1048   880 S       0  0.0  0.4   0:00 httpd
  304 root       0   0   316  316   260 S       0  0.0  0.1   0:00 mingetty
  305 root       0   0   316  316   260 S       0  0.0  0.1   0:00 mingetty
   23 root       0   0   380  380   328 S       0  0.0  0.1   0:00 kerneld
  209 root       0   0   448  448   372 S       0  0.0  0.1   5:56 syslogd
  218 root       0   0   556  556   340 S       0  0.0  0.2   0:00 klogd
  229 daemon     0   0   488  488   332 S       0  0.0  0.1   0:00 atd
  240 root       0   0   488  488   400 S       0  0.0  0.1   0:00 crond
  251 bin        0   0   488  488   328 S       0  0.0  0.1   0:14 portmap
  263 root       0   0   484  484   336 S       0  0.0  0.1   0:04 inetd
  286 root       9   0   844  844   772 S       0  0.0  0.3   0:01 httpd
```

图 17-1　top 程序的输出结果

像 top 之类的程序能够显示系统运行时的许多信息。从上图的输出结果中可看到，此计算机几乎使用了它的全部的实际内存（256MB），但却还未使用到任何虚拟内存。

Note

如果在 Windows 平台上运行 Apache，可使用任务管理（Task Manager）程序监视内存、虚拟内存和 CPU 的使用。

在大多数 UNIX 系统上可用来检查系统虚拟内存使用情况的另一个程序称为 vmstat。作为最后一个例子，图 17-2 显示了在同一台机器上 vmstat 会话过程的输出样本。当观察到你的 Web 服务器在使用虚拟内存（即交换区），则系统内存短缺！这就是该在 RAM 上投资的时候了。

如果不选择购买更多的内存，或认为内存已经够用了，那么就要找到一种能减少内存使用的方法。要减少内存使用，可以使用本章的软件部分所提到的一些软件技巧。

```
[kabir@www1 ~]$ vmstat 1
 procs                      memory    swap       io    system        cpu
 r b w   swpd   free   buff cache  si  so   bi   bo   in   cs  us  sy  id
 0 0 0      0  5232 129220 96912   0   0    1    5   35   15   1   4  47
 0 0 0      0  5232 129220 96912   0   0    0    0  107   18   0   2  98
 0 0 0      0  5232 129220 96912   0   0    0   18  115   18   0   2  98
 0 0 0      0  5156 129220 96916   0   0    1    0  133   87   1   4  95
 0 0 0      0  5204 129220 96916   0   0    0    0  137   24   0   2  98
 0 0 0      0  5224 129220 96916   0   0    0    0  112   24   0   2  98
 0 0 0      0  5220 129220 96916   0   0    0    0  107   10   0   2  98
 0 0 0      0  5220 129220 96916   0   0    0    6  112   12   1   1  98
 0 0 0      0  5232 129220 96916   0   0    0    0  113    8   0   2  98
 0 0 0      0  5232 129220 96916   0   0    0    0  110    8   0   2  98

[kabir@www1 ~]$
```

图 17-2　vmstat 程序的输出

作为影响 Web 服务器性能的主要因素，下一个应考虑到的硬件组成部份是硬盘。Web 服务器要花费大量的时间访问硬盘，并且因为硬盘速度仍很慢，所以它经常是导致糟糕性能的主要因素。

确保 Web 服务器使用高档的 SCSI 磁盘控制器和一组高档的 SCSI 硬盘驱动器。最近的超大容量 SCSI 硬盘是最好的选择。在 Web 服务器上使用多个硬盘是个好主意，例如，不应把操作系统和 Web 数据放在同一个硬盘上。至少要有两个硬盘：一个用作操作系统，另一个用作数据。使操作系统软件和 Web 服务器软件分离也是不错的安全措施。

硬盘会出错。如果担心硬盘出错，可以考虑经常备份数据。若有冗余的低价磁盘阵列（RAID）子系统也是很值得的。从本质上讲，RAID 是用一组较小的磁盘模拟较大的磁盘。如果一个磁盘坏了，其他的磁盘将替补这个空缺直到坏盘被替换。对 Web 服务器来说，使用 RAID 子系统就意味着具有更高的 I/O 性能和合理的数据安全。

许多人从销售商那里购买整机，而这些厂商主要使用大的 IDE 或 EIDE 硬盘。如果不想把服务器作为玩具，强烈建议在 Web 服务器上不要使用 IDE/EIDE 硬盘。

在计算机里影响服务器性能的最后一个硬件部件是网络适配卡，假设你的 Web 服务器在某处连到以太网上，所使用的网络适配卡应该是高质量的，并且尽可能快。例如，若连接 Web 服务器的以太网既可以处理 10Mbps 的节点也可以处理 100Mbps 的节点，那么从名牌厂商那里买一个 100Mbps 的网络适配卡。

还有其他硬件也会影响 Web 服务器性能，它们将在本章的网络部分中予以讨论。

在确保使用的 Apache 服务器硬件不存在这里所讨论的问题后，就要准备在软件上去驯服 Apache 服务器这个“野兽”了。

17.2　软　　件

在不同类型的操作系统上运行 Apache 也会造成服务器性能有很大差异。如果在使用

早期的 Apache 的 Winodows 版本，并且想知道它为什么这么慢，那么全部所能做的就是等待在这个平台上的 Apache 的成熟版本。

当前，Apache 在大部分 UNIX 平台上闪耀着光辉。Apache 的爱好者普遍相信：Apache 作为自由和开放源代码的服务器，大部分是在 UNIX 的变体如 FreeBSD 和 Linux 系统上运行。若正在使用这些操作系统之一，就有好的基础。若正在使用不同的 UNIX 操作系统流行的版本，确保你的操作系统的网络是为当今的 TCP/IP 需要而建立的。查阅位于 www.apache.com 的服务器文档。可以找到更多有关特定的操作系统的信息。

获取正在使用的操作系统的最新的补丁和升级总是很明智的。许多操作系统的升级去掉了安全漏洞，增强了对网络支持。因此，竖起耳朵，睁大眼睛，关注操作系统的升级。

确保卓越的操作系统在强有力的硬件上运行后，就可以查找其他可能成为性能瓶颈的软件“部位”。那么 Apache 服务器本身怎么样呢？它能成为瓶颈吗？是的，它确实能成为瓶颈。

正如本书在早些时候提到的那样，可以通过两种方式获得 Apache。即可以下载 Apache 的可执行文件发行版，并在计算机上安装它，也可编译自己的 Apache 可执行文件，然后安装它。强烈建议使用后一种方法，因为它一开始就可以调整 Apache。检查当前拥有的可执行文件，决定是否需要它提供的全部内容。运行如下命令：

```
httpd -l
```

这将显示当前 Apache 可执行文件中包含的所有模块。如果不需要其中一些模块，可以编辑配置文件后重新编译，并重新进行安装。这样就可删去不需要的模块。如果你的操作系统允许从可执行文件去掉不必要的符号信息，那么就去掉它们。在 Linux 上使用下面的命令：

```
strip  httpd
```

将从 httpd 中去掉多余的符号。这样可以使可执行文件变得更小，将为每个 httpd 节省内存。

Note

由于许多人可能正在使用 Linux，所以应当知道：若有许多 RAM，应使用诸如 top 之类的程序向系统报告可用的实际内存。由于某些 BIOS 设置的原因，许多（并非全部）Linux 发行版所带的内核不支持多于 64MB 的 RAM。当然，这很容易得到纠正。办法是在 etc/lilo.conf 文件中增加如下的一行：

```
append="mem=256M"
```

这儿，将限制设置为 256M。不要忘记在修改 lilo 配置文件后，重新运行 lilo。

如果认为 Apache 可执行文件（httpd）像它应该有的那样好，但却仍怀凝 Apache 内部存在瓶颈，那么仔细检查 Apache 的配置文件。一些 Apache 命令对性能影响很大，这些通常是那些需要域名解析、系统调用、磁盘 I/O 和进程处理的命令。在建立 Apache 配置文件时可以使用下面的主要规则：

- 少使用 DNS

- 减少磁盘 I/O
- 限制子进程数

17.2.1 少使用 DNS

如果正在使用既可用主机名又可使用 IP 地址的命令，那么使用 IP 地址以避免域名解析带来的性能损失。例如，若使用如下命令：

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

```
allow from .nitec.com
```

或

```
deny from .nitel.com
```

从性能的角度看，这些命令的代价是很大的。服务器将执行多次域名解析调用以确定每个主机请求是否落入命令所指定的范围。这些就是上面的命令设置造成的。在繁忙的服务器上，这些是很不受欢迎的。因此使用 IP 地址列表去代替域名。使用 IP 地址意味着不需进行域名解析。当然，这使得配置服务器变的不那么容易，因为必须添加许多 IP 地址。但谁又能说性能就不付出代价呢？

Tip

许多新的 Apache 系统管理员打开 HostnameLookups 在日志文件中记录主机名称，这是一个坏主意，因为每个请求都需要解析主机名，这是一个很费时的过程。

17.2.2 减少磁盘 I/O

如果不使用针对目录的配置文件，那么就应该告诉 Apache 不要查找这个文件。否则 Apache 将在 URL 请求范围内的每个目录中查找此文件。这就意味着要完成许多不必要的磁盘 I/O。可以通过把 AllowOverride 设为 none 来禁止此功能。

Cross-Reference

关于如何告诉 Apache 不要查找针对目录的配置文件，详见第 4 章。

若在 Apache 上使用诸如快速 CGI、嵌入式 Perl 模块等新概念，可以编写出使用内存缓冲的 Web 应用程序，它们可以减少磁盘 I/O。对于高通信流量的站点，这种缓冲技术可以有效地减少 Web 服务器上的负载。

Cross-Reference

关于如何使用快速 GGI 和嵌入式 Perl 模块的详细内容，参见第 9 章和第 13 章。

17.2.3 限制子进程数

在 UNIX 上运行 Apache，一个主进程产生许多子进程来处理各种请求。有些命令可以用来控制使用多少个子进程，以及在请求完成后子进程可以挂起多长时间。如果系统负载很重，就可以通过改变像 MinSpareServers、MaxSpareServers、MaxRequestsPerChild 等命令而获得一个较好的结果。通过改变 RLimitCPU、RLimitMEM，RLimitNPROC 等命令，可以限制 Apache 和其子进程在处理请求时所占用的系统资源。

所有上面的命令在第 4 章中有详细介绍。

已经精确调整了 Apache 配置后，确保已去掉了所有不必要的软件。在 UNIX 和 NT 服务器系统上，许多应用软件（在 UNIX 上称为服务器软件/守护进程，在 NT 上称为服务）在后台运行。因此，应决定哪些应该保留和哪些应该去掉。举例来说，若发现服务器正在运行打印机服务软件（print server，lpd），可能需要禁止它。除了杀死这些正在运行的进程外，还应确保它们不会在下次重新启动机器时被运行。

超星阅览器提醒您：
使用本复制品
请尊重知识产权！

在改变软件后重新启动机器是个好主意。这可以检查出系统是否如期望的那样在运行。如果已经删除或禁止了一个或多个已加载的守护进程，重新启动机器是很有帮助的。检查经常使用的服务，确保服务器像想象那样工作。例如，若系统上有一个邮件服务器，也许应在重新启动后再试试能否发送或接收 e-mail，或至少检查一下邮件服务器是否在运行。

17.3 网　　络

Web 服务器运行在网络上，因此网络配置对于 Web 服务器性能有巨大的影响。这取决于如何使用 Web 服务器。可以调整 Web 服务器网络。典型的 Web 服务器网络安装如下：

- 内部 Internet Web 服务器
- ISP 集合 Web 服务器
- 内联网 Web 服务器
- 分布式 Web 网络

17.3.1 内部 Internet Web 服务器

许多组织正在连接到 Internet，并把 Web 服务器安装在自己的网络上。典型的，这些接入是从专用 ISDN 连接到部分 T-1（DS-1）的低带宽连接。若对 Web 服务器上提供的内容没有高的要求，这种接入也许不是一个问题。你想做的全部工作只是给已存在和潜在的用户获得公司有关信息的途径。

但是，若提供的内容很丰富，要求也很高，显然需要更多的带宽。以我的经验，许多负责取得带宽人不对所需带宽进行任何计算来决定他们的需要，而靠猜测和道听途说。可以实际进行简单的计算，对所需带宽获得清醒的认识。

计算出平均 Web 页面大小（字节数），确定每天的高峰的时间段和高峰时间段内的访问次数。使用下面的公式计算出平均字节传输率：（每天峰值时间段总的访问次数×平均页面字节数）÷每天总的峰值时间。一旦知道了高峰时间的平均字节传输要求，可以很容易地计算出平均带宽需求。使用的公式如下：（平均每一高峰小时的字节传输率÷3600 秒）。这就给出了高峰期的字节/秒的平均速率。如果当前的带宽很好地满足了这个以字节/秒计的峰值，那就成了。

Note

大多带宽以千位/秒（Kbps）或兆位/秒（Mbps）来表示，也可以把高峰字节/秒的速率转换成这种形式。要变换字节/秒到 Kbps 或 Mbps，使用下面算法：

Kbps=（字节/秒）*（8 位/字节）*（1 千位/1024 位）

Mbps=（字节/秒）*（8 位/字节）*（1 千位/1024 位）*91 兆位/1024 千位）

使带宽为当前所需带宽的 2 倍是一个好想法。一旦确保你的 Internet 连接有足够的带宽，就可以把注意力放在网络上如何配置 Web 服务器。图 17-3 显示了在组织的局域网上的 Web 服务器通过路由器接入 Internet。

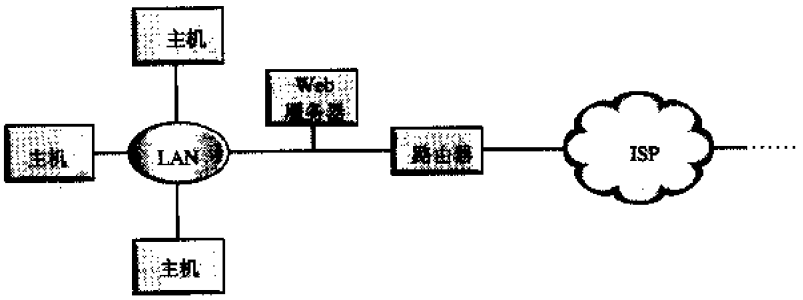

图 17-3 将 LAN 上的 Web 服务器接入 Internet

这种设置有什么错误吗？它是有问题的。正如在图中所看到的，所有的 LAN（以太网）传输都能被 Web 服务器见到。这意味着，在繁忙的 LAN 上 Web 服务器同其他 PC 主机有同样的性能，因为它和其他主机共享以太网。这也是一种高度危险的网络设计，因为整个 LAN 通过路由器都暴露在 Internet 上。希望所有用户不要使用这种设计。这种设计在性能上是差的，从网络安全考虑上也很糟。为了增加性能和安全，LAN 应像图 17-4 所示的那样进行分段。

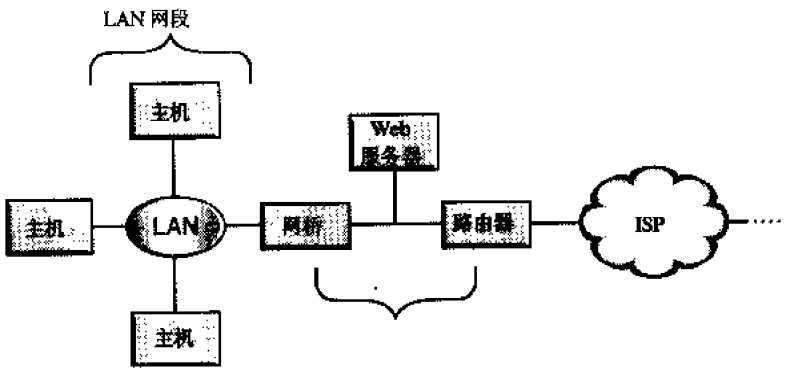

图 17-4 将分段 LAN 上的 Web 服务器接入 Internet

经过这样改变后，LAN 传输隔离在第一网段内，Web 服务器和以太网上的流量分开。事实上，Web 服务器所看到的流量只是那些从路由器来或到路由器去的进出 Internet 的传输，这使 LAN 传输不再占用 Web 服务器的带宽，有效地优化了服务器接受和发送数据包

的性能。网桥既可以通过网络硬件实现，也可以通过连到多个网络上的 PC 机实现，该机器运行的操作系统可以将 IP 通信流从一个网络传输到另一个网络。

因此，要记住的关键是：当你的专用网络上有 Web 服务器，并且将它接入 Internet，你必须有合适的带宽去处理高峰负载，并且内部网络（LAN）通信流量不应当影响 Web 服务器。

内部的 Web 服务器在经济上并不总是可行的，也并非必需的。如果没有管理服务器和处理网络问题的知识丰富的员工，强烈建议把 Web 服务器安装在 ISP 网络上。这样的服务器称之为集合服务器。

17.3.2 ISP 集合 Web 服务器

集合服务器可以提供许多好处。如果选择了合适的 ISP，就无需担心网络设计和带宽阻塞。ISP 将处理所有这些问题。当前许多 ISP 容纳集合服务器，当然，需要选择一个好的。在和 ISP 签定协议之前，最好到那里看一看，问一些问题。

询问潜在的 ISP 怎样把你的服务器同其余的集合服务器隔离开来。还要问一下，这个 ISP 是否把你放到专有网段上。这对你应当是需要的，因为你决不希望同一网段上的其他人刺探你的数据包。若你的 ISP 告诉说：你的服务器系统同其他集合服务器共享一个以太网段。这就意味着那些服务器中的任何一个都可以使用像 tcpdump 或其他网络数据包刺探工具来监视你的服务器数据包。不管从性能或是从安全角度来讲，这怎么都是不可接受的。询问 ISP 它是如何接入 Internet 的，有什么网络冗余措施。它上面有那些提供商（也就是说，你询问的 ISP 的 ISP），可能有不止一条路径路由数据包到 Internet。这些是附加的。

当做完这些后，可以用常用的 TCP/IP 工具做一个简单的调查分析。例如，可以用 ping 程序检查潜在的 ISP 的 Web 站点。ping 将告诉将数据包传送到网络和从网络返回用了多长时间。其花费的时间越少，从使用 ping 的主机到被检查的主机之间建立的连接性能越好。

如果使用 traceroute 程序（Windows NT 下称为 tracert）追踪到 ISP Web 服务器的路由，数一下从你的机器到 ISP Web 服务器经过的段数（输出结果最左边那一列显示的数）。同样的，从 Internet 上不同的节点多作几次这样的试验。可以请在不同城市的朋友使用 ping 或 traceroute，尽可能多地收集这些数据。

如果对当前的 ISP 做了这样试验，就可以获得 ISP 服务情况的比较。这虽然是很原始的方法，但它可以提供 ISP 怎样连到 Internet 上的有价值的信息。建议同时对多个 ISP 做这些试验，这样就可以更好地理解每个 ISP 的性能。另外，试着在不同的时间做这些试验，而不仅只在大早上或深夜。收集一天内不同时间的数据，这样得到的数据就更全面。

通过做上面的“作业”，就可增加选择到能提供好的网络性能的 ISP 的机会。

17.3.3 内联网 Web 服务器

如果在内联网上使用 Web 服务器，应该不会为网络性能而担心，除非网络有很严重的拥塞。如果出现这种情况，把 Web 服务器放到不太繁忙的网段上，或增加一个新网段，

隔离 Web 服务器，使 Web 服务器只能接触到和它有关的传输。

若 Web 已成为你经商很重要方式，那么很可能你的 Web 服务器将因负载多而变得十分繁忙。一个单个的服务器也许不能完成赋予它的任务。在这种情况下，应考虑把负载在两个或更多的 Web 服务器上分担。

17.3.4 分布式 Web 网络

当 Web 站点变成高通信量的热门站点时，就必须面对负载平衡问题。这个问题许多 Web 系统管理员每天都会遇到。如果已经实际运用了本章前面所讲的全部或大部分性能提示，你的单个 Web 服务器大概已在优化的水平上运行。但是，单个高性能的 Web 服务器对于高负载的 Web 站点来说也许并不够用。在这种情况下，就需要考虑分配负载到另外的系统上。可以使用两种方法把 Web 负载分配到多个服务器上。现在让我们仔细看一下这两种负载平衡方法，然后指出这两种配置方法的潜在问题。

17.3.4.1 分配任务到专用的服务器

当 Web 服务器承担的任务可以分配给其他服务器时，可以采用这种方法。例如，运行许多 CGI 应用程序，并且发现 Web 服务器不能很好地为这么多的 CGI 请求提供服务，那就可以新增一个或多个 CGI 服务器，把 CGI 应用程序分配到这些服务器上。图 17-5 显示了这种网络的例子。

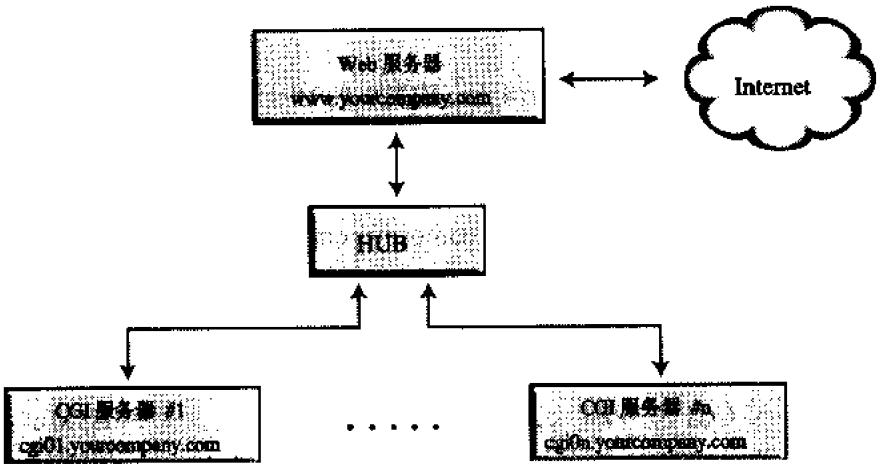

图 17-5 分配 CGI 负载到外部 CGI 服务器

在本图中，Web 服务器主机 www.yourcompany.com 连接了一个或多个 CGI 服务器，如 cgi01.yourcompany.com。要使用这些 CGI 服务器就必须修改那些和 CGI 脚本有链接的 HTML 页及调用 CGI 脚本的 HTML 表单。例如，列表 17-1 显示了一个位于 www.yourcompany.com 的 HTML 订购表单。

列表 17-1：order.html

```
<HTML>
<HEAD>
<TITLE> Order Form </TITLE>
```

```
</HEAD>
<BODY BGCOLOR ="White">
<H1> Order Form </H1>
<HR>
<PRE>

<FORM ACTION="/cgi-bin/takeorder.pl" METHOD="POST">
Last Name <INPUT TYPE=TEXT NAME="LastName" SIZE="30">
First Name <INPUT TYPE=TEXT NAME="FistName" SIZE="30">
Email Addr <INPUT TYPE=TEXT NAME="EmailAddr" SIZE="30">
Product <INPUT TYPE=TEXT NAME="Product" SIZE="30">
Quantity <INPUT TYPE=TEXT NAME="Qty" SIZE="3">

<INPDT TYPE=SUBMIT VALUE="Order Now">

<INPUT TYPE=RESET VALUE="Clear All">

</BLOCKQUTE>
</PRE>
</FORM>
</BODY>
</HTML>
```

请尊重相关知识产权!

在单一 Web 服务器情况下，访问此订购表单的 URL 如下：

www.yourcompany.com/order.html

访问者可以在此表单中输入订购信息，然后点击 Order Now 按钮提交此订购表单，并由给服务器处理。在这点上，www.yourcompany.com 服务器收到一个关于 takeorder.pl CGI 脚本的请求，如下：

www.yourcompany.com/cgi-bin/takeorder.pl

访问者输入的数据通过 HTTP 的 POST 方法提交给同一个的服务器。

为了利用外部 CGI 服务器（cgi01.yourcompany.com），上面的 HTML 表单需要加以改变。订购表单的 ACTION 值应从/cgi-bin/takeorder.pl 改为 http://cgi01.yourcompany.com/cgi-bin/takeorder.pl，CGI 服务器主机 cgi0l.yourcompany.com 就会收到订购表单的处理请求。这样，就将 CGI 请求分配给 CGI 服务器。同时，主 Web 服务器就可以留出时间为其他页提供服务。

在这种设置中，需要提出一个问题。大多数 CGI 脚本执行任务并产生输出页。输出页需要这样建立，即输出页中的所有链接必须指向 www.yourcompany.com 服务器。在输出页上的每一个链接都需用绝对引用，不能用相对引用。例如，如果 takeorder.pl 脚本在单一服务器配置下输出一个 HTML 页，该页的源代码如列表 17-2 所示，但在分离负载的情况下，它需要被修改。00

列表 17-2：takeorder.pl 的 HTML 输出

```
<HTML>
<HEAD>
<TITLE> Successful Order </TITLE>
</HEAD>
<BODY BGCOLOR ="white">

<IMG SRC="/imags/header.gif" HEIGHT=50 WIDTH=50 BORDER=0>

<P> Thank you for your order Moahmmed Kabir </P>
<HR>
<BLOCRQUTE>
We have already sent an order confirmation number at your email
address (kabir@nitec.com) for your future reference.
<P>
Click <A HREF="/index.html"> here </A> to return to home page.
</P>
    </BLOCKQUTE>
    <CENTER>
    <IMG src="/images/copyright.gif" HEIGHT=20 WIDTH=400 BOR-
DER=0>
    </CENTER>
    </FORM>
    </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！

URL 链接必须由：

```
<A HREF="/index.html"> here </A>
```

改为：

```
<A HREF="http://www.yourcompany.com/index.html">here </A>
```

相似的，图像标记中的图像源值也应该使用 www.yourcompany.com/images/替换/images/。也许想知道如果对 Web 服务器和 CGI Web 服务器上的这些文件不做改变会发生什么事。在这种情况下，在第一个 CGI 脚本执行后，访问者将会停留在 CGI 服务器上，当访问者看到 cgi01.yourcompany.com 在其浏览器的 URL 地址框中时会感到困惑。另外，经常访问你的站点的人也许会把你的 Web 站点的区域在 CGI 服务器上做书签，并直接访问这些站点而不通过 Web 服务器。这样如果只有一个 CGI 服务器，它很有可能被 CGI 请求和正常的文件请求弄得疲倦不堪。这就有违使用两个服务器的初衷。

另外，使用外部 CGI 服务器的真正原因可能是你有耗费 CPU 时间和大量 I/O 的 CGI 程序。因此，必须要有很强的意识，确保 CGI 脚本的输出页里有合适的链接能够返回到原 Web 服务器。注意，也可以使用 HTTP 重定向头标 Location:在 CGI 脚本里把访问者访问的文件的 URL 自动重定向到主 Web 服务器上。例如，下面的 CGI 脚本 goback.pl，就说明了在基于 Perl 的 CGI 脚本中如何使用重定向头标：

```
#!/usr/local/bin/perl
#
print "Location:
www.yourcompany.com/thanks_for_order.html\\n\\nn\";
   exit 0;
```

超星阅览器提醒您：
此为试读，需要完整PDF请访问：www.ertongbook.com
请尊重相关知识产权！

无论何时，使用 URL http://cgi01.yourcompany.com/cgi-bin/goback.pl 运行此脚本，它都使访问者访问 www.yourcompany.com 服务器上的 thanks_for_order.html 页。这种技术也能用于前面订购表单的处理脚本里，把访问自动重定向到主 Web 服务器，使用这种方法，访问者就不会混淆他或她到底在哪里。如果 takeorder.pl 使用 goback.pl 所示的重定向，大多访问者不会知道 CGI 脚本是在另外的机器中处理的（除非他们查看订购表单的源代码）。

正如看到的那样，这种分布式 Web 网络对于繁忙的 Web 站点来说在性能上有相当的优势。注意，这种分布式网络设计不但有助于提高 Web 性能，也有助于形成更安全的环境。例如，假设需要将一个数据库服务器连到 Web 服务器上，以便访问者可以使用数据库应用程序，但是关心数据库服务器的安全，就可以像图 17-6 所示的那样实现网络。

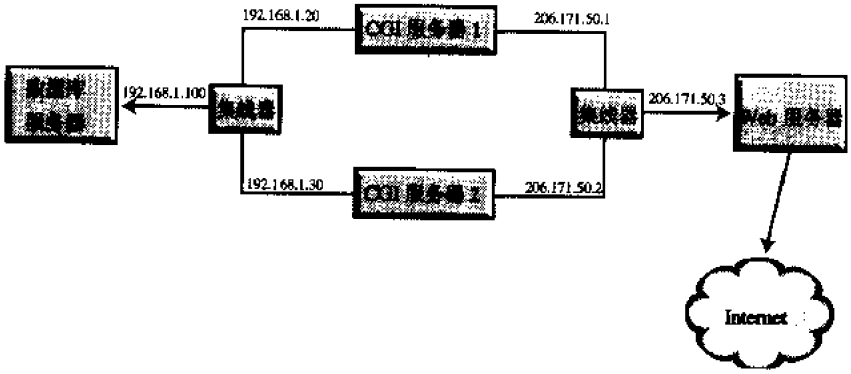

图 17-6 安全和高性能的 Web 网络

这里，数据库服务器位于不能路由的 IP 网络（192.168.1.0）上，它只连到两个多地址的 CGI 服务器上。CGI 服务器再连到 Internet 可路由的网络 206.171.50.0，这样它们就可以实现本章前面所提到的功能：隔离数据库服务器。而所需要做的就是在 CGI 服务器上安装数据库客户端软件，以便 CGI 服务器能够在不可路由的网络内访问数据库服务器。这可以把负载分配到三个外部系统上，因此提供了既有高性能又安全的解决方案。

注意，安全程度依赖于如何保护 CGI 服务器，因为任何人，只要能突破这两个服务器中的一个就有可能接触到甚至是进入数据库服务器。但是，由于数据库服务器总的来说位于整个系统的后面，要想揭露这种网路配置是不太容易的。

17.3.4.2 为平衡负载重复配置 Web 服务器

如果 Web 服务站点包含大量的静态文档和适量的由 CGI 生成的文档或其他动态文档，并且负载很重。也许应考虑重复配置 Web 服务器以便把负载分摊到多个完全相同的 Web 服务器上。

举例来说，假设使用三个 Web 服务器，它们有相同的主机名（www.your

company.com），但IP地址不同。这怎么可能？答案在DNS的设置上。

使用称为循环（round-robin）解析的DNS技巧，可以把多个IP地址赋予相同的主机名。本例中的网络的DNS配置如下：

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

```
www        IN A    206.171.50.1
www        IN A    206.171.50.2
www        IN A    206.171.50.3
```

地址（A）记录告诉DNS主机以列表的顺序循环给出称为www主机（例如，www.yourcompany.com）的全部IP地址。DNS服务器第一次解析www名字时给出第一个IP地址（206.171.50.1），然后是第二个，第三个，不断地重复循环。

只要保持所有Web服务器内容一致，访问者就不会注意到有何不同。负载以优雅的方式分担。注意，当访问者指到www.yourcompany.com/时，DNS服务器就会给出列表上上次给出的IP地址记录的下一个记录的IP地址。浏览器在整个会话期间用同一个Web服务器。因此这种负载平衡是通过把不同的客户重定向到不同的服务器上而获得的。

这种设置很快会出现两个问题。首先是如何保持服务器一致。如果试图通过手工做的话（可能通过FTP来做），就要更新三个服务器上的文件。但这很不方便，解决办法是用一个服务器作为预备区，利用软件如rdist来定期自动更新其他服务器。

在第19章中可找到循环（round-robin）DNS设置的更详细信息。

当有一些CGI脚本，它们写数据文件，而随后其他CGI脚本要在这些数据文件中查找记录，那么第二个问题就出现了。例如，你的Web站点有一个抽彩的应用程序，它让访问者签名并给访问者一个cookie作为标识。你已想出了一种方法，让访问可以通过访问特殊的页面而增加点数。这些点数存在磁盘上，这样以后就可以累加出访问者所得的点数，找出访问这一站点次数最多的访问者。访问者也可以通过CGI脚本查看他的总点数。此CGI脚本要读点数数据文件，但因为使用重复设置的Web网络，此CGI脚本存在于每个服务器上，它们在各自服务器上创建或读取数据文件。

因此，当一个访问者访问其中一个Web站点并点击签名参加抽彩活动，一切都正常，但在另一天，他回到此站点，并且被定向到另外一个没有存放他数据的服务器上，他会感到震惊——他根本没收集到任何点数。你是如何考虑这个问题的。当然，可以通过引入网络文件系统（Network File System，NFS）解决这个问题。把服务器上的一片磁盘空间安装到其他服务器上，然后对那些写文件的CGI脚本作一些必要的更改，访问定位到安装的NFS文件上，就可以使所有服务器共享同一组数据。

要更详细的了解这种设置的NFS配置，见第19章。

使用循环的DNS配置，能够生成重复配置的Web服务器网络，提供负载平衡，因此比单一的服务器提供了更好的性能。这种配置的一大好处是便宜。但是，循环设置也有一些负面影响应予以关注。任何时候要改变DNS设置，它要花费时间在Internet上传播。若一个服务器停止了，想要请求从停止的服务器重定向的到其他工作着的服务器上，这时

就可能产生问题。另外有相同主机名的 DNS 记录数也有一个限制，当前的限制是不大于 32，这意味着当前最多只能有 32 个重复配置服务器。若要找到了一个更好的解决方案，其价格也将会达到几千甚至几万美元。使用硬件的成套设备是一个可能的解决方案。

17.3.4.3 负载平衡所带来的问题

无论是分配负载到多个服务器，还是使用循环 DNS 配置的分布式 Web 网络都存在的问题是：它们仍不能提供真正的负载平衡。在这些设置中，并不能说出哪个服务器是繁忙的，哪个不是。为了获得基于服务器能力的真正的负载平衡，必须额外买一些硬件。现在市场上已经出现了能够实现负载平衡的硬件部件。其中的一些称为 Web 多路复用器；另一些是具有负载平衡算法的以太网交换器，它可以从外部检测服务器负载。这些硬件部件通常还有许多其他的特征，使得它们很昂贵。包括失败探测和重新路由 HTTP 请求到一个工作着的 Web 服务器。然而，若有适当的预算，可以考虑使用这些特征，若觉得有能力负担这些选择的费用，可以到如 CISCO（www.cisco.com/）和 HydraWeb（www.hydraweb.com/）之类的厂商那里寻找有关负载平衡产品的信息。

17.4 内　　容

一旦使 Apache 计算机、软件和网络都已能充分发挥它们的作用，就可以把注意力集中到调整内容上来。

静态内容如 HTML 页和图像不会像动态内容那样产生很重的负载，因为动态内容需要由诸如 CGI 脚本、Java Servlet 和数据库应用程序生成。为了减少 Web 服务器的负载，应当寻找能够加快动态内容速度的产品。例如，若要使用 Perl 编写 CGI 脚本，应当考虑 mod_perl 或 FastCGI。

关于 mod_perl 更多的信息见第 13 章，要了解 FastCGI 见第 9 章。

可以通过生成使用客户端脚本或客户端 applet 的应用程序以减轻 Web 服务器端动态内容的负载。这可使浏览器在同 Web 服务器通信之前已做了大量工作。例如，使用 Java 脚本或 Java applet，可以把用户输入合法性检查任务分担到浏览器端运行。还有，使用 Java applet，可以减少服务器的大部分工作，Java applet 把这些工作移到了客户端去。

同样的思想，应尽可能地减少 SSI 命令的使用。更要很小心的选择 SSI 文件的扩展名，因为服务器对具有同样扩展名的文件同那些可能有 SSI 调用的文件进行同样的分析处理，例如，使用.html 作为 SSI 文件扩展名，如果有大量有.html 扩展名的非 SSL HTML 页面，将会大大降低服务器的性能。

虽然服务器上的大部分负载由动态内容产生，但在许多方面，静态内容也会成为获得服务器最大性能的障碍。下面是对于开发静态内容的一些指导原则。

- 使用合适 HTML 元标记，鼓励在客户端和代理服务器缓冲存储不经常变化的页面。
- 使用 IMG 标记中 HEIGHT 和 WIDTH 属性，给访问者较快页面下载的印象。
- 使用小图像。

♦使用客户端图像映射以减少由于处理服务器端图像映射而给服务器造成的负担。

17.5 人 员

Web 站点的性能在很大程度上取决于管理和开发站点的人。因此，应当注意，简单地调整一下硬件、软件和网络并不能使 Web 服务器变成一个高性能的机器，必须要考虑人的因素。下面是一些建议：

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

♦强制执行第 18 章所提到的 CGI 开发指导原则。

♦不鼓励人们注册到 Web 服务器上运行用户进程，或在 Web 服务器上做开发工作。

♦给 Web 开发小组提供接触最新工具和技术的权力。提供培训机会。

♦告诉每个人重视更好的性能，但不能以降低标准为代价。例如，并不希望开发人员为减少浏览器端的 DNS 请求而在链接中直接插入 IP 地址。当然，在这种情况下，访问者也许会感到页面下载较快，但当改变服务器的 IP 地址时，就要在有链接的地方留下大量要改变的 IP 地址。

对服务器的性能目标应当和 Apache 组织对 Apache 的目标一样：正确第一，然后才是速度。

17.6 小 结

在本章中，讨论了如何从计算机、Apache 及其配置、网络、内容和所涉及的人员等方面着手使服务器获得更好的性能。本章使你能够真正从改善性能的正确方向去考虑问题。

下一章讨论如何建立完善的 Web 服务器。

第 18 章　运行完善的 Web 站点

本章内容:

- 创建 Web 周期
- 标准化指导原则
- 使 Web 对用户友好
- 在 Web 上宣传 Web 站点

超星阅览器提醒您:
使用本复制品
请尊重相关知识产权!

到现在为止，你大概已建立一个或多个 Web 站点，它们运行在新 Apache Web 服务器上。组织里的每个人都因为你做了这样漂亮的工作而赞许你，你进入 Web 天堂了，是吗？ 错了！很不幸的，许多同事会问你如何更新他（她)）们在 Web 站点上的页面。例如，市场部会问你如何更新价格信息，法律部门也许会问你他们如何在 Web 站点上增加更多的法律内容。

这就是在中型或大型组织里 Web 管理员所遇到的种种事情。很快，你就发现自己陷于一大堆内容更新请求和愿望之中。那么，如何管理你的 Web 呢？ 在本章中，将讨论如何创建专业化的 Web 管理环境。它将使你和你的 Web 开发人员保持清醒的头脑，并和 Web 发展保持协调一致。

18.1　创建 Web 周期

不幸的是，典型的 Web 开发项目并不是从设计可维护的 Web 站点开始的。在大多数项目中，大部分时间花费在使服务器运行起来和开发 Web 内容上，很少花时间去考虑 Web 站点长期管理方面的问题。具有讽刺意义的是：由于缺乏明确、可维护的完整过程，当系统开始运转时，各种问题就来了。在本部分，将讨论称为 Web 周期的内容，它能使你产生高度可管理的 Web 解决方案。

一个 Web 周期由三个阶段组成： 开发，试用和生产阶段。系统地实现所有的阶段，就能产生可维持和可管理的 Web。图 18-1 显示了 Web 周期的框图。

正如从图中所能看到的，Web 周期开始于开发阶段，持续到试用阶段，终止于生产阶段。当重新开始时，又从生产阶段开始，重复以前的周期。你也许想知道这是什么意思，好，下面让我们看看每一个阶段：

开发阶段——在这个阶段，开始开发 Web 内容。内容既可以是 HTML 文档也可以是 CGI 脚本。Web 内容完全在本阶段开发和测试完成。一旦开发人员确认，他们的工作已为和 Web 站点集成作好了准备，新开发的内容就进入下一阶段。

试用阶段——本阶段的目的是把新开发的内容和已有的内容集成，并执行测试周期。

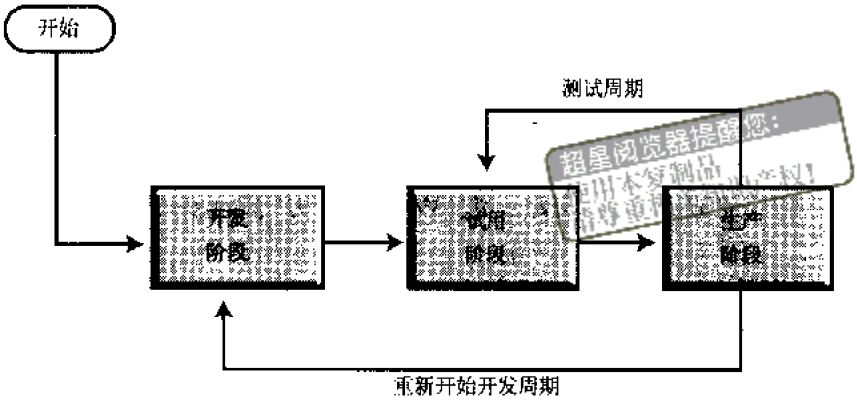

图 18-1　Web 周期的框图

在试用阶段，开发人员不再参与试用过程。在这个过程里，引入测试人员，他们不是开发人员，以便排除程序员们的成见。换句话说，由于开发人员过分自信，也许并未全面测试开发的内容。在此阶段，或者出现到问题，或者在一组成功测试的基础上结束此阶段。若是后一种情况，就可以把新开发的、已试用和测试的内容移交到生产阶段。若新开发的内容出现了问题，则需要开发人员在开发区把问题改正后把内容再放入开发阶段，不允许开发人员在试用区修正问题。

生产阶段——本阶段由内容备份和内容部署两方面的任务组成，首先，备份已有的（可正常工作的）内容，然后把试用阶段的内容移到生产的 Web 空间，移动时应尽可能少花费时间，以减少断开与可能的访问者的连接时间，并防止通过 Web 收集的数据的丢失。

当准备开始新的开发周期（重新开始整个过程）时，需要将生产阶段的内容拷贝到开发区，以便开发人员能在其上进行开发工作。只要需要，Web 周期就应该以同样方式继续。

所有这些带来了什么？ 它带来了可靠和管理的选择。例如，如果把当前正在开发的内容不经过全面的测试就直接搬到生产系统上，那就很危险。在大多数情况下，内容开发员都声称他们已在开发环境中测试了新开发的内容。然而，很快就会面对这些内容与已有内容之间的不协调的问题。因为开发人员在开发环境中一般都缺少新内容与当前内容的集成，其测试也是不现实的。只有把已存在内容与新内容集成到一起，才会发现可能存在的不兼容。例如，若没有试用阶段，把新内容直接从开发阶段转储到生产系统上，可能就会造成下面的问题：

♦生产系统内的文件可能被新内容覆盖。这种情况一般发生在图像文件上，主要是由于缺乏标准的文件命名规则，或者使用了图像文件的公共目录的原因。

♦生产系统上的数据文件被覆盖，这是由于 CGI 开发员在开发时使用了旧的数据文件。

♦当涉及多个开发人员时，由于每个开发人员可能使用不同时间的拷贝开始工作，一些旧文件也许会在生产服务器上重新出现。一旦一个开发人员复制了拷贝，其他开发人员应复制这个开发人员的拷贝，以确保结果一致。

如果涉及到多个开发人员，并且他们的项目之间又相互交叉的话，会出现许多另外的问题。如果不想让生产服务器处于这种危险之中，就需要有试用阶段。现在让我们看看 Apache 如何帮助实现这些周期。

超星阅览器提醒您：
本书为数字版权产品
请遵守相关知识产权！

18.2 把 Web 周期付诸实施

现在已准备好把 Web 周期付诸实施。

首先，需要为 Web 周期建立服务器。这有许多方法，但在这儿只讨论三种。在这三种方法中，有两种是针对那些组织中只有单个 Apache 服务器机器的情况，另一种是面向组织中有多个 Apache 服务器的情况。

在理想情况下，不应该在生产服务器系统上执行任何开发工作。但是，如果预算不允许为 Web 使用多个机器，就必须使用单个服务器去实现这个周期。

一旦已建立了服务器，就应准备实施 Web 周期了。

18.2.1 为实施 Web 周期进行设置

可以用两种方法为实施 Web 周期进行设置：既可在生产服务器上用两个新的虚拟主机来实现开发和试用站点，也可形成三个单独的 Apache 配置，分别为生产服务器、开发服务器和试用服务器。

如果开发工作包括调试 Apache 配置文件或测试最近发行的 Apache 服务器，就应当使用为生产服务器和其他两个服务器各生成一个配置文件的方法。如果正常 Web 开发不包括与 Apache 有关的改变，可以使用虚拟机的方法。

在进入详细地准备 Web 周期之前，我愿意指出：好的 Web 周期需要有很好计划的文件目录结构。图 18-2 显示了这样的用于 Web 周期的目录结构。

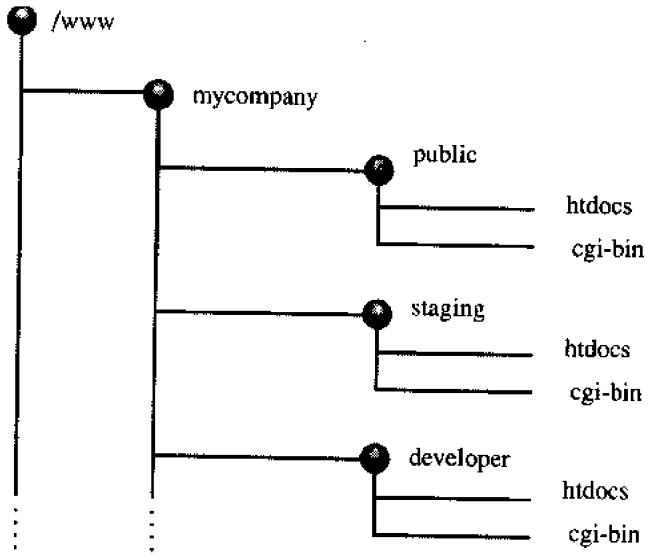

图 18-2 公司使用的 public、staging 和 developer 站点的目录结构

```
<VirtualHost 206.171.50.50:8080>
ServerName
DocumentRoot /www/mycompany/developer/htdocs
ScriptAlias /cgi-bin/ /www/mycompany/developer/cgi-bin/
TransferLog logs/developer-server.access.log
ErrorLog logs/developer-server.error.log
</VirtualHost>
```

超星阅览器提醒您：
请勿复制品
使用本相关知识产权！

注意，在上面的例子中，两个虚拟主机使用了相同的 IP 地址，但在<VirtualHost...>容器里标明了不同的端口。IP 地址同主服务器（www.mycompany.com）是一样的。ServerName 命令也被设为同主服务器一样。主服务器的配置未变。

下面的 URL 用于访问试用阶段的站点：

```
www.mycompany.com:1080/
```

要访问开发阶段的服务器，使用下面的 URL：

```
www.mycompany.com:8080/
```

如果需要在开发过程中改变 Apache 的配置文件，不要使用上面的方法。在这种设置中只有一套 Apache 配置文件，任何为试验目的改变这些文件都会影响到生产服务器。此时，如果仍用一台机器，就需要运行多个 Apache（主）服务器，下面介绍这种方法。

18.2.1.2 使用多个 Apache（主）服务器进程

如果计划试验 Apache 服务器本身作为开发阶段工作的一部分，应使用多个（主）服务器进程。要现实 Web 周期，按照下面的说明去做。

在 Apache 配置目录里生成两个子目录（分别为 staging 和 developer）如下：

```
mkdir /path/to/Apache/server/root/conf/staging
mkdir /Path/to/Apache/server/root/conf/developer
```

不要忘记用你的实际服务器配置目录路径替换“/path/to/Apache/server/root/conf”。拷贝所有的*.conf 文件到 staging 和 developer 子目录，命令如下：

```
cp /path/to/Apache/server/root/conf/*.conf
/path/to/Apache/server/root/conf/staging/*
cp /path/to/Apache/server/root/conf/*.conf
/path/to/Apache/server/root/conf/developer/*
```

现在修改 staging 目录下的 httpd.conf 文件。使其监听 1080 端口，替代缺省的 80 端口。要使其监听 1080 端口，可以使用 Port 命令，也可使用 Listen 命令，相似地修改 developer 目录下的 httpd.conf 文件，将 Port 或 Listen 命令设为 8080。

现在，应该去修改 staging 和 developer 子目录下的 srm.conf（或 httpd.conf），使其 DocumentRoot 和 ScriptAlias 命令指向合适的路径。例如，在前面图 18-2 中的所示的目录结构例子中，对于试用站点的配置 DocumentRoot 和 ScriptAlias 设置如下：

```
DocumentRoot    /www/mycompany/staging/htdocs
ScriptAlias  /cgi-bin/  /www/mycompany/staging/cgi-bin/
```

对于开发站点配置，设置变为：

```
DocumentRoot    /www/mycompany/developer/htdocs
```

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

注意，如果对于生产服务器使用特殊的配置，其中使用绝对路径信息，那么也许就要为试用和开发者服务器进一步编辑新的配置文件。

其思想是生成三套配置文件，每个指向不同的 DocumentRoot 和 ScriptAlias。做了这些工作后，可以启动三个 Apache（主）服务器，命令如下：

```
httpd -f /path/to/Apache/server/root/conf/httpd.conf
httpd -f/path/to/Apache/server/root/conf/staging/httpd.conf
httpd -f/path/to/Apache/server/root/conf/developer/httpd.conf
```

当决定编译 Apache 新版本，并要将它作为开发者服务器时，只需简单地把开发者服务器的配置文件作为参数传递给它。例如，如果决定要增加一个新模块，检查此新模块在内容上的效果，只需用这个可执行文件去运行试用和开发者服务器，而不是用生产服务器可执行文件（httpd）去运行即可。在编译新的可执行文件后，重新将它命名为类似 httpd-xx80 是一个好的主意，这样可确保不会偶然地用它覆盖生产服务器的可执行文件。

现在，让我们讨论为 Web 周期环境创建多个主机需要做什么。

18.2.1.3　使用多个 Apache 服务器计算机实现 Web 周期

如果能为建立 Web 周期的环境提供多个 Apache 服务器计算机（即一个用于开发，一个用于试用，另一个用于生产），则不需要特殊的 Apache 配置。只需简单地在每个主机上安装 Apache，选其中之一作为开发者站点主机，另一个作为试用站点主机，第三个作为生产站点。因为现在有了三个 Apache 服务器分别运行在三台主机上，这样每个 Apache 都可运行在 80 端口。这就是创建多主机的 Web 周期环境所需做的全部。

18.2.2　实现 Web 周期

到此为止，在上面三种 Web 环境中，应该有其中之一：

♦使用单个计算机，具有两个虚拟主机，分别用于开发和试用。生产服务器是主 Apache 服务器。在这种设置里，修改任何一个 Apache 设置都需小心，因为改变将会影响到生产服务器如何运作。

♦使用单一计算机。运行三个主 Apache 服务器分别用于开发、试用和生产，每个（主）Apache 服务器生成各自的配置文件。创建这种配置，就可以在开发站点试验 Apache 配置，而不干扰生产服务器配置。

♦创建至少三个不同的计算机分别作为开发、试用和生产 Apache 服务器。所有三个计算机上的 Apache 服务器都运行在 80 端口。

为了开始新的 Web 周期，把生产内容从生产服务器的文档根目录里拷贝到开发站点。例如，如果配置是列表中的前两个之中的一个，可以很容易地把整个 Web 内容拷贝在开发站点，使用的命令如下：

```
cd /path/to/production/docroot/dir
tar cvf - . |(cd /path/to/development/site/docroot/dir;
tar xvf - )
```

此命令拷贝生产服务文档根目录下所有的文件和目录到开发站点的文档根目录下。确

保把路径信息改变成自己系统上相应的目录。

对于多主机 Web 环境，可以生成一个生产服务器的 tar 文档，并通过 FTP 把它拷贝到开发站点。

设置文件的访问权限，以便 Apache 能读所有的文件，执行 CGI 脚本。如果有些目录 Apache 应该有写权限（用于 CGI 脚本写数据），也应设置相应的权限，做完这些后，启动或重新启动开发站点的 Apache 服务器。

现在，确保开发站点看起来完全同生产站点一样（可通过浏览器）。执行一些手工比较和热点检查，确保脚本也能工作。

如果任何 CGI 脚本产生了自定义的生产服务器的 URL，则它们应该为开发站点做同样的事情。使用 SERVER_NAME 环境变量和 SERVER_PORT 端口地址来改正它们，或者忽略这些 URL。

当各个服务器都正常工作后，则已经成功地创建了实施 Web 周期的环境。现在，开发人员可以把新内容和脚本放到开发站点并测试它们。无论何时完成的新内容开发，都应先在开发区内测试它。这种测试应集中关注下面的问题：

- 它为其目的服务吗？ 换种说法，新内容提供的功能符合要求吗？
- 新内容会有任何负面影响吗？例如，如果新内容真是一个新的 CGI 脚本，就应使用 Apache 提供的调试支持工具（见第 8 章）去监视这个脚本如何工作。

一旦对测试结果满意，就停止对新内容作任何进一步的开发，以避免必须对它进行另一次功能测试。当到了更新生产服务器的时候，拷贝生产站点上的内容，把它放到试用站点。确保试用站点的表现精确地与生产服务器相同。做一些手工检查确信一切看起来都同生产站点一样，然后就可以把新内容移到试用站点并同已有内容集成到一起。

一次移动一个项目是一个好主意，这样就能够在试用阶段发现问题。例如，如果在开发系统里增加了三个新脚本，那么每次只移动一个到试用区。要执行功能和站点集成两方面的测试。如果这个脚本通过了测试，就可以移动下一个项目。一旦把所有新内容都移到了试用区，就可以执行站点级集成测试。小心监视试用站点的日志，注意错误日志里是否显示出任何不正常事情。如果一切正常，就可以准备更新生产结点。然而，更新生产站点必须要十分小心仔细。例如，如果生产服务器有 CGI 脚本在生产区创建了数据文件，任何数据文件都不应该被试用区的文件覆盖。

因此，更新生产站点最好的方法是在生产服务器最不繁忙时更新它。这段时间，可以从生产服务器取得这些数据文件并把它们放到试用站点相应的目录下。使试用站点同生产站点一致。此时，必须快速把试用站点内容转储到生产区域。这很有技巧，因为生产站点是活动的，并不知道什么时候访问者正在访问页面或 CGI 脚本，它们最终需要读、写数据文件。

为了使切换时间达到最短（至少在单个服务器设置下），可以生成 Shell 脚本去做下面的工作：

1. 把所有活动的文件拷贝到试用站点相应的区域。

2. 重新命名顶级生产目录（如图 18-2 中的 public 目录），诸如 public.old。

3. 把顶级试用目录（如图 18-2 中的 staging 目录）重命名为以前使用的顶级生产目录，例如 public。

4. 把旧的生产目录(如 public.old) 重命名为以前使用的试用阶段顶级目录——如 staging。

在这种方法里，试用站点只在几步内就变成了生产站点，不需要大量的文件拷贝操作。列表 18-1 提供了这样一个脚本例子，它对应于图 18-2 中所示的环境。

请尊重相关知识产权！

列表 18-1：Stage2production.sh 脚本

```
#!/bin/sh
# Purpose:a simple shell script to copy live data files
# to staging area and rename the staging area into live
# production site. It also renames the old production area into
# staging area
#
# $Author$
# $Revision$
# $Id$
# $Status
# Yon will need to change there variables to use this script.
DATA_FILES="/www/mycompany/public/htdocs/cgi-data/*.dat";
TEMP_DIR="/www/mycompany/public.old";
PRODUCTION_DIR="/www/mycompany/public";
STAGE_DIR="/www.mycompany/staging";
# Copy the live data to the staging directory.
/bin/cp $DATA_FILES $STAGE_DIR

# Temporarily rename current production directory to TEMP_DIR
/bin/mv PRODUCTION_DIR TEMP_DIR

# Now rename the current staging site to production directory
/bin/mv TEMP_DIR PRODUCTION_DIR

# Now rename the temporary (old) production directory
# to staging directory
/bin/mv TEMP_DIR STAGE_SITE

# To be safe lets change the current production directory's
# permission setting such that the Apache user (httpd)
# and Apache group (httpd) can read all files. Note that
# if you use some other user and group for Apache, you
# with have to modify this command according to
# your setup.

/bin/chown -R httpd.httpd $PRODUCTION_DIR
```

```
# Now lets change the file permission such that the
# owner (httpd in this case) has read, write, execute
# permission, the group (httpd in this case) has read,
# execute permission and everyone else has no
# permission to see the production directory files

/bin /chmod -R 750 $PRODUCTION_DIR
```

运行此脚本后，应当完成快速检查以保一切正常。若出现问题，可以把当前生产目录重命名为别的名字，并把试用目录变回生产目录名以便恢复原来的生产站点。

一旦习惯使用这种 Web 周期后，就会发现它使得跟踪开发和集成问题变得很容易，它也保证生产站点始终工作正常，除了严格保持实施 Web 周期外，还需多做一些工作，使 Web 处于完善的状态。这些任务属于维护类。

18.3 维 护 Web

一旦实施了 Web 周期后，维护 Web 是很重要的。典型的 Web 维护任务包括服务器监视、记录日志和数据备份。在第 11 章中包含了 Web 站点维护的服务器监视和记录日志方面的内容。在这里，将讨论数据备份，如果可能，应有两种类型的备份。

18.3.1 联机备份

联机备份在应急情况下是很有用的。大部分情况下可以在几分钟内迅速地访问备份数据，完成必要的恢复工作。为了实施联机备份解决方案，可以找商业联机备份厂商，也可以和你的 ISP 谈。然而，若把 Web 服务器放在自己的网络上，可以将数据备份到网络的另一台主机上。在大部分 UNIX 系统上，可以使用称为 rdist 的程序在其他 UNIX 主机上生成你的 Web 站点的镜像目录（第 19 章讨论基于 rdist 的站点镜像应用的例子）。

在 Web 服务器本身保存 Web 数据的压缩版本也是好办法。在 UNIX 系统上，可以建立一个 cron 作业按需要的周期生成 Web 数据的压缩的 tar 文件。下面就是一个例子：

```
# for system V-ish Unix, Weekday range is 0-6 where 0=sunday
# For BSD-ish system use weakday range 1-7 where 1=Monday
# This example is for a Linux system (System V-ish cornd)
30 2 * * 0,1,3,5 root /bin/tar czf /backup/M-W-F-Sun.tgz /www/*
30 2 * * 2,4,6   root /bin/tar czf /backup/T-TH-Sat.tgz/www/*
```

如果将这两个 cron 项保存在/etc/crontab 文件中，就会建立两个文件，每星期一、星期三、星期五和星期天，第一个 cron 作业将在早上 2：30 运行，建立 /www 目录下所有内容的备份，并把这个压缩备份文件存储到 /backup/M-W-F-Sun.tgz 文件里。同样的，在星期二、星期四和星期六的早上 2：30，第二个 cron 作业建立 T-TH-Sat.tgz 文件，对同样的数据进行备份，并将保存相同的目录下。确保在两个压缩文件里至少有最近两天的备份。

18.3.2　脱机备份

也应该在可移动的存储介质上做备份，把它们放到安全的地方。从这种备份恢复通常是耗时的工作。可以使用磁带驱动器、可移动磁盘（如 Jaz 盘）实现这种备份。更赞同基于 8mm 的磁带备份，因为它提供 8GB 的数据存储容量。另外 8mm 的磁带驱动器比新的大容量可移动介质出现在市场上的时间更长。

随着 Web 站点内容越来越丰富，应能注意 Web 服务器上的磁盘空间被迅速填满。这是由那些无用的文件造成的。通常因为担心移去这些文件可能会在某处断开（如链接），所以一直将它们保留下来。如果认为真的出现了这种情况，并且是在 UNIX 平台上，可以考虑运行 find 实用程序去发现那些长时期未被访问的文件，例如：

```
find  /www  -name "*.bak" -type f  -atime +10  -exec ls -l {}\;
```

这将列出/www 目录下所有以.bak 为扩展名且最近十天内未被访问的文件。若想删去这些文件，可以替换 ls -l 命令，并按如下方式运行一次 find：

```
find  /www  -name "*.bak" -type f  -atime +10 -exec rm -f {}\;
```

如果这样做有帮助，可以建立 cron 作业按所希望的周期运行这个命令。

18.4　标准化指导原则

随着实施 Web 周期的环境的建立，就能容纳许多开发人员。但是，仅仅建立实施 Web 周期的环境并不保证有高质量的 Web 生产。高质量的 Web 需要有高质量的内容，这里有一些在开发内容时应该遵循的指导原则，这里的主题是指导原则的标准化。

每个 Web 站点都应当提供独特的内容以吸引潜在的访问者。所有的 Web 内容类型可被分为两类：或者静态，或者动态。静态内容典型地由 HTML 文件生成，动态内容通常是 CGI 或其他的服务器端及客户端应用的输出结果。大部分站点使用静态和动态内容相混合的方式出版它们的信息，因此，静态内容和动态内容开发都需要标准。

18.4.1　HTML 文档开发策略

虽然可以有很多方式提供静态内容，如普通的文本文件或 PDF 文件，例如，大多 Web 站点用 HTML 文档作为主要的信息仓库，但为了指导 HTML 编辑人员，应形成一套 HTML 开发策略。下面是一些指导原则，可以做出必要的增删以适应自己组织需要。

18.4.1.1　坚持使用标准的 HTML 标记

HTML 开发人员应当坚持使用标准的 HTML。使用依赖于浏览器的 HTML 也许会使页面在一种浏览器上很好看，但在其他浏览器上却会变得很糟：

例如，下面显示了一个 HTML 文档，它符合最小的 HTML 文档标准。

```
<HTML>
<HEAD>
<TITLE> Document title goes here </TITLE>
</HEAD>
```

```
<BODY>
Document body goes here
</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！

每一个文档都应当至少包含这些标记。

18.4.1.2 保持内嵌图像与文档在一起

文档内嵌的图像应存放于文档目录的子目录内。然而，这个规则有一个例外：如果一些图像是可重用的，应该把它们放到一个集中的图像目录下。这种情况的例子是用图像文件配置的标准导航条。导航条能在多个文档里重用，因此要把它存在集中的目录下而不是把它同每个文档保存在一起。这提供了更好的控制并节省了空间。除了提及的特例外，其他所有的图像文件应该同使用它们的文档在一起。对于这些图像文件的源引用应当采用相对地址，这样如果文档同图像目录从一处移到另一处时，这些图像能够同以前一样提供相同的显示方式。在下面的例子里，将展示如何生成一个使用多个图像连接的的可移植的 HTML 文档。

假设想出版两个 HTML 文档（mydoc.html 和 mydoc2.html），它们包含三个图像（image1.gif，image2.gif 和 image3.gif）。首先在文档根目录下或其他相应的子目录下生成一个有意义的子目录，假设在服务器的文档根目录（/www/mycompany/htdocs）下生成这个目录并称它为 mydir。

现在，在 mydir 子目录下生成一个称为 images 的子目录，并把三个图像文件存放在其下。编辑 HTML 文档，使所有指向这些图像的链接使用的 SRC 属性值如下：

```
SRC="images/image1.gif"
SRC="images/image2.gif"
SRC="images/image3.gif"
```

一个使用 image3 的内嵌图像链接看起来像下面那样：

```
<IMG SRC="images/image3.gif" HEIGHT="20" WIDTH="30"
ALT="Image 3 Description">
```

注意，在上面行里的 SRC 属性不包含任何绝对的路径信息。如果这些文档同图像目录一起从 mydir 移到 otherdir 目录下，没有图像被中断。然而，如果链接包含的路径信息如下：

```
<IMGSRC="mydir/images/image3.gif" HEIGHT="20" WIDTH="30" ALT="Image3
Description">
```

或：

```
<IMGSRC="/mydir/imges/image3.gif" HEIGH="20" WIDTH="30" ALT="Image 3
Description">
```

那么移动后将需要修改这些文档。许多站点把图像文件保持在文档根目录下的一个集中的图像目录（如 images）里，链接文档使用的 IMG 标注如下：

```
<IMG SRC="/images/image3.gif" HEIGHT="20" WIDTH="30"
ALT="Image 3 Description">
```

这虽很好，但有下面两个问题：

♦当要想删除 HTML 文档时，必须要确保删除了存放在集中的图像目录下的相应图像文件。

♦若不能解决第一个问题，最终许多磁盘空间就会消耗在图像文件里。

因此，把图像文件保存在中的目录里并不是好主意，应当把文件保存在它们链接的子目录里。

18.4.1.3　在每个文档上显示明确的版权信息

每个文档都应该包含内嵌的（评注的）版权消息，它明确地标明文档和所用图像的所有者。相似的版本信息也应出现在每一页上。为了容易更新版权信息，也许应考虑使用 SSI 命令如下：

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

现在，所需做的是生成一个称为 copyright.html 的 HTML 页面，并将它放到服务器文档的根目录下。注意，因为这个 HTML 页的内容是通过 SSI 调用被插入到嵌入 SSI 命令的文档里的，故在此文档里不应用<HTML>、<HEAD>、<TITLE>或<BODY>标记。如果有许多文档使用这一 SSI 调用，那么当需要改变版权信息里的年月或做其他改变时，这就会变得很容易。

18.4.2　动态应用的开发策略

Web 内容的另一种类型是动态内容，它通常是由 CGI 脚本或其他实现 CGI 功能的应用程序或一些服务器端的应用程序产生的。绝大多数动态文档是由使用 Perl 编写的 CGI 脚本产生。因为 CGI 脚本和应用通常只有很短的生命周期，所以许多 CGI 开发人员并不致力于产生高质量的应用。

如果计划使用 FastCGI 或 mod_perl 语言编写脚本和应用，采取合适的开发方式编写它们是很重要的。当用脚本和应用程序实现动态内容时，应考虑下面的策略。

18.4.2.1　坚持使用版本控制

CGI 开发员必须使用版本控制，这样，当新开发的或扩展的版本包含缺陷时，就能够返回到的较早的版本。在大多数 UNIX 系统上，可以使用协调版本系统（Concurrent Versions System，CVS）软件实现版本控制环境。可以在下面站点找到 CVS 的最近版本：

```
ftp://prep.ai.mit.edu
```

18.4.2.2　不在 CGI 脚本和应用中使用绝对路径名

为确保脚本从一个 Web 站点移到另一个后无需改变就能够使用，不要在 CGI 脚本使用绝对路径名。如果由于一些特殊的用途需要使用绝对路径名，应为此脚本提供一个配置文件。使用这种方式，路径可以通过修改文本配置文件而得到更新。

在第 8 章中，讨论了如何使用基于配置文件的 CGI 脚本，它能用于一般的表单处理。

18.4.2.3　提供用户级和源代码级两种文档

需要写好源代码的文档，以便将来的开发人员更新脚本时不需要为弄懂它如何工作

而花费太多的时间。

18.4.2.4 避免在脚本和应用中嵌入 HTML 标记

CGI 脚本的输出应当是模板驱动的。换一种说法，CGI 脚本读一个输出文件的模板，并替换动态数据字段（可以使用定制标记表示）。对于 HTML 开发人员，这可使输出页很容易更新，因为 HTML 不在 CGI 脚本内部。实际上，CGI 脚本应该尽可能少地包含 HTML。

18.4.2.5 不要信任用户输入的数据

为了减少安全风险，确保在使用用户输入的数据之前对它进行检查。

在第 12 章中，讨论了检查用户输入的问题。还详细讨论与输入相关的安全风险和解决方案。

18.4.2.6 在基于 Perl 的 CGI 脚本里避免使用全局变量

当用 Perl 开发 CGI 脚本时，应避免使用全局变量。限制变量的范围是消除脚本不可预料行为的方法。Perl 程序员应按如下方式说明变量：

```
my $variable;
```

而不是：

```
local $variable;
```

因为后者生成全局变量，而前者生成局部变量。

18.5 使用用户友好的界面

使用标准的 HTML 和高质量的 CGI 脚本/应用，某种程度上使你的 Web 站点比外部许多已存在的站点更好。然而，还应考虑 Web 站点设计的另外一个方面：用户界面。

考虑 Web 站点是可以在浏览器中看到的具有图形用户界面（Graphical User Interface，GUI）的交互应用，为了人们在访问 Web 站点时有一种愉快的感受，需要使用用户友好的 GUI。具有对访问者友好的用户界面（UI）会使访问者感到愉快。

本节将讨论开发用户友好的 GUI 的关键问题。除了使用用户友好的 GUI 外，需要注意断开的链接和对已删除文件的请求。使用服务器记录错误信息的日志去发现这一类问题。还应当给访问者提供向你反馈信息的方法，大多数站点使用简单的基于 HTML 表单的 CGI 反馈脚本。也可以开发这种适合自身需要的脚本。收集反馈信息是发现访问者想从你的 Web 站点得到什么的好方法。

18.5.1 使站点易于导航

用户必须不用费大劲就能够从一页跳到另一页。应能让访问者找到使他们返回、前进或跳转到相关信息的按钮或菜单条。

许多 Web 页面设计者争论说：流行的 Web 浏览器已经包含有 Back 和 Next 按钮，因此在页面上有 Back 或 Next 按钮是多余的。错了！想象一个用户从搜索引擎的输出结果登

上你的 Web 页面中的一页（而不是主页）。他简单地搜索一个或多个关键词，搜索引擎给提供到你的 Web 站上的一页的 URL。他很有兴趣了解你的站点上关于这个主题的更多信息，这样他想移到这个文档的开始——但这里没任何方式使他能做这些，因为浏览器的 Back 按钮只会把他带回到搜索引擎的输出页！

只要在这个页面上有一个到前面页面的链接（或按钮），此用户就能很容易地翻到前面。也许不喜欢额外按钮的 Web 页面设计者会坚持说：用户应当能够简单地操作 URL 回到主页面，并从那里开始。好，这已假定了主页上有到那个页面（匹配搜索关键字的那个）的明确的链接。这种情况总是对的吧？ 我希望你同意我的观点。

我想，实现一个菜单条使用户可以向前向后翻页，也能使用户跳转到相关的位置或者甚至是主页，这是个好主意。图 18-3 显示了这种页面的一个例子。

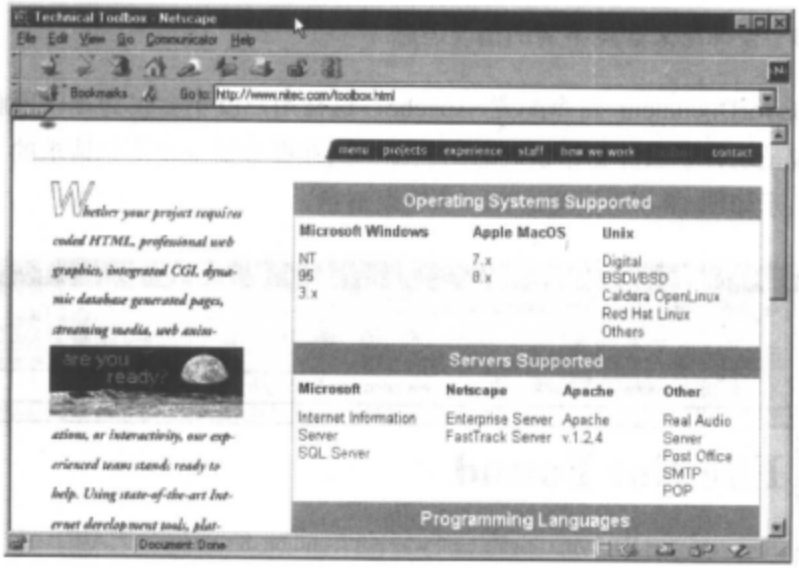

图 18-3　具有导航菜单条的页面的例子

18.5.2　创建有吸引力的设计

设想 Web 站点是一天 24 小时的开放的、进行丰富多彩的和交互式的展示。如果这种展示并不吸引人，访问者将会离开你的 Web 站点。为开发设计有吸引力的站点，应该考虑下面的指导原则。

18.5.2.1　合适的前景和背景颜色

确保不要着迷于选择颜色。极端色彩的使用会使 Web 站点看起来不专业并且乏味。应赋予色彩意义并使用合适的配色方案。例如，若 Web 站点是关于儿童玩具的，它应当是一个色彩鲜艳的站点。但如果站点是关于数字信号处理器评测的，大概不需要许多颜色和闪烁的背景。

18.5.2.2　适当的文本大小

尽量使主要内容以普通的字体显示。使用特殊的字体（通过 FONT FACE=

“myVerySpecialFont”>）也许会使页面在你的浏览器上很好看（因为你碰巧有这种字体），但是在别的浏览器上，此页面也许看起来完全不同，并且也许会很难读。另外，正文的长短也要小心选择：既不要太长，也不要太短。记住，如果访问者看不懂你的 Web 页面说什么，他们不会喜欢你所要说的东西。

18.5.2.3 少用图像

不使用不必要的图像，图像会使 Web 页面下载很慢。记住，不是每个人都连到 ISDN 线路上。大部分人到 Internet 的连接仍使用 28.8K 的调制解调器。缓慢的页面下载也许会使潜在的客户离开你的页面。

同样，谨慎使用动画。即使最有趣的动画，看看几回后也令人厌烦。因此，确保不因动画而使页面过分拥挤。

18.5.3 去掉意义含糊不清的错误消息

使用 ErrorDocument 命令配置 Apache，这样用户就不会收到难以理解的（至少对一般用户来说是这样）服务器错误消息。例如，当在服务器上找不到请求的 URL 时，服务器也许会显示如图 18-4 所示的这样一条错误消息。

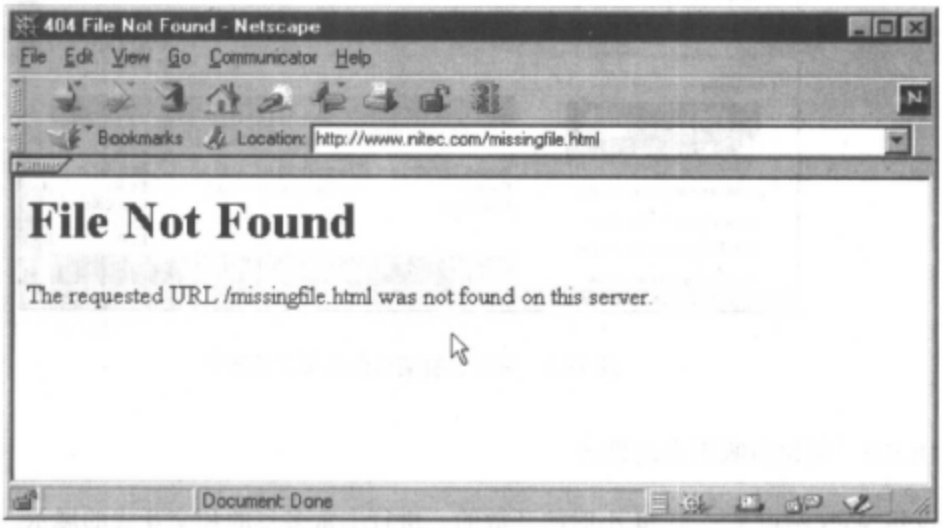

图 18-4 没有发现文档

为了使这条错误消息变得更友好一些，可以在 Apache 的配置文件里（如 srm.conf）加一条 ErrorDoument 命令，形式如下：

```
ErrorDocument 404 /Sorry.html
```

这样这条错误消息就看起来如图 18-5 所示的那样。

Cross-Reference 对于其他错误消息如何使用这条命令，详见第 4 章。

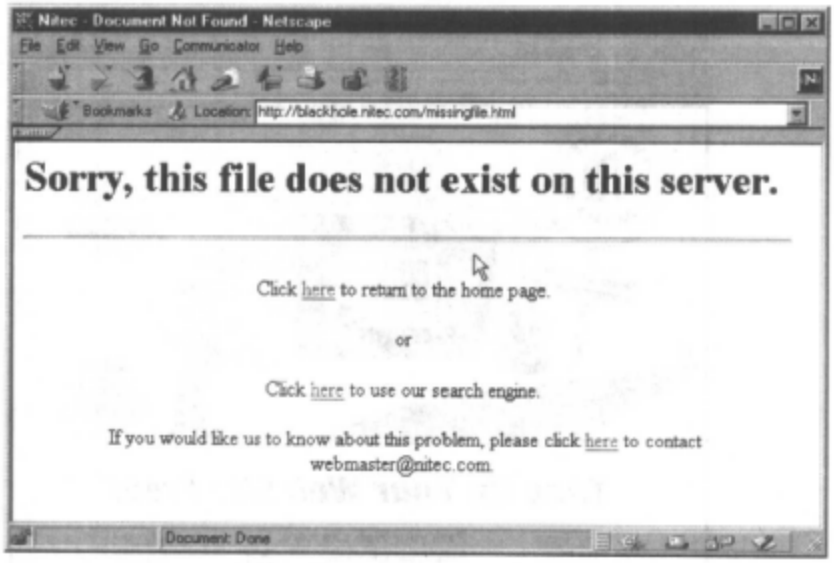

图 18-5　友好的错误消息

18.5.4　测试 Web GUI

测试图形用户界面的最好方法之一是使用与一般 Internet 用户或潜在用户的计算机相似的系统。若认为客户都有高性能的计算机和快速的连接，那么无须因为使用较少的图形或客户端应用程序而担心，诸如 Java apples 和激波动画。

在大多数情况下，并不知道潜在客户的计算机和网络的规格。这样，应该使用一般用户的配置。使用一个低端的奔腾计算机，有 16M RAM 和 28.8K 的调制解调器从一个 ISP 帐号连接到你的测试 Web 站点。试试诸如 640×480 或 800×600 像素的低监视分辨率，如果访问者使用 Web-TV 系统，要试试更低的 550×400 像素的分辨率。

如果你对从你的 Web 站点看到的东西很高兴，那么其他人大概也会喜欢它。另一方面，如果你不喜欢你所看到的，其他人大概也不会喜欢。

如果愿意，也可以使用第三方测试你的 Web 站点。例如，一个称作 AtWeb 的公司提供基于 Web 的自由调整服务，这个公司的 URL 如下：

www.websitegarage.com/

他们的后端应用程序能够检测任何 Web 站点，包括页面下载时间、HTML 质量、死链接、拼写错误、HTML 设计质量和链接流行程度。要试试它，只需到上面的 Web 站点，在文本输入窗体内输入自己的 Web 站点地址即可。

例如，如果决定从他们的站点获得 www.nitec.com 的免费测试报告，输入站点的 URL 如图 18-6 所示。

图 18-6 请求测试报告

点击 GO 图标后，必须等几秒，然后这个应用程序产生一个报告，如图 18-7 所示。

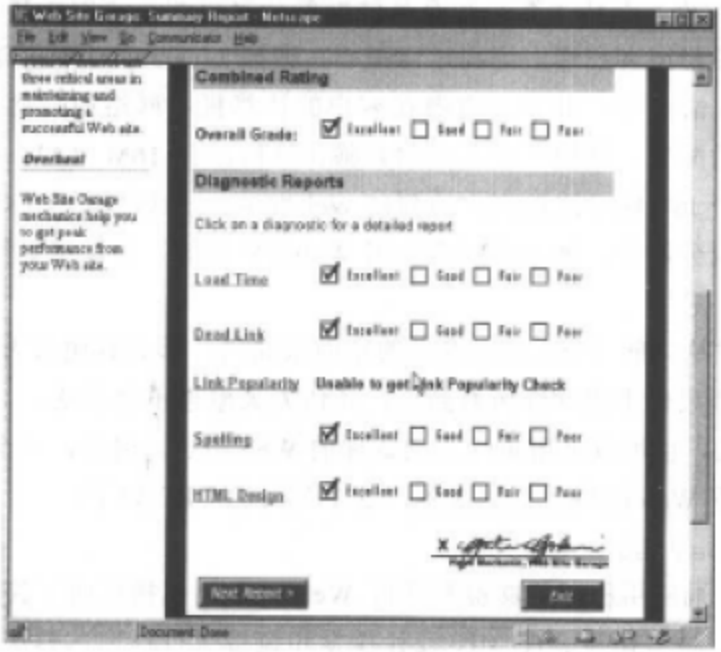

图 18-7 来自 www.websitegarage.com 的报告

这个应用程序报告按照 AtWeb 的标准测试站点的情况。此免费版本的功能是有限的，但此公司也提供用于站点测试的其他的商业软件包。

18.6　宣传 Web 站点的方法

一个完善的 Web 站点，如果没有人知道它，有什么好处？应该考虑在 Web 上宣传你的 Web 站点。可以雇用广告代理商帮你做这些。然而 Web 上的广告可能很昂贵。如果你有预算，可以给自己站点做一些宣传。

在宣传站点之前，问问自己，“我在 Web 上如何发现信息？”答案是：通过搜索引擎。你的公司被列在搜索引擎上了吗？ 如果没有，这是宣传你的站点的第一步。

几乎所有的搜索引擎都可以让你把你的 URL 提交到它们的搜索机器人的数据库里，以便以后它能导游你的 Web 站点。你应该把你所能考虑到的重要搜索引擎列一个表，把 Web 站点的 URL 提交给它们。虽然这个过程也许要几天甚至几周的时间，然而能够在内容内加上 META 信息，以帮助你的 URL 在一个潜在用户搜索时出现在恰当的位置。例如，加如下的 META 信息：

```
<META NAME="KEYWORD" CONTENT="keyword1 keyword2 keyword3...">
<META NAME="DESCRIPTION" CONTENT="Description of yourCompany">
```

为了增加访问率，也可以参加链接交换，如站点 www.linkexchange.com。链接交换需要在 Web 页面里放置一组专门的 HTML 标记，这些标记把广告图形（通栏标题广告等）文件加入你的页面里。同样，你的通栏标题广告也显示在那些同意在他们的页面上显示别人的通栏标题广告的人操作的 Web 站点上。这种广告分享在个人和小型商业站点中间非常流行。

不论在高知名度的站点上买了广告空间，如 Yahoo、AltaVista 或 Netscape，或者使用了链接交换方法，你应通过查询自己的方法在搜索引擎输出结果里定期检查的自己站点的地位。

18.7　小　　结

本章讨论与开发完善的 Web 站点有关的各种的问题。介绍了 Web 周期，它使用三个阶段（开发，试用，生产）管理 Web 站点。本章还讨论了如何遵循标准化指导原则提高静态和动态 Web 内容的质量以及改进用户界面。本章还提供了如何在 Web 上宣传你的 Web 站点的有用的信息。希望本章所给出的信息将有助你创建完善的 Web 站点。

在下一章，将学习如何建立 Web 网络。

第 19 章　使用 Apache 组建 Web 网络

本章内容:

- 什么是 Web 网络
- 如何实现基于 Apache 的多系统 Web 网络
- 如何在 Web 网络中使用循环 DNS
- 如何通过 NFS 共享公共数据

超星阅览器提醒您:
使用本复制品
请尊重相关知识产权!

到现在为止，本书已讨论了许多可以在实际中运用的 Apache 特征和功能。如果你已经有一台机器在运行 Apache，你才真正开始使用到目前为止在本书中所讨论的所有内容。

如果你的需求是小规模的系统，可以只在一台机器上安装 Apache 建立几个 Web 站点。但是，如果你的需求是大规模的系统——一个 Web 网络又该怎么办呢？如果你和你的组织需要一个 Web 网络来创建可靠的和可扩展的环境，在网络上使用多个系统来容纳所有的 Web 站点，本章将帮助你创建这样一个网络。

本章中的例子来自于利用 Apache 建立的 Web 网络的实际体验。

19.1　什么是 Web 网络

首先，让我们了解 Web 网络的需求。有一个称为 XC News 的虚构公司，它需要建立一个可扩展的 Web 解决方案。方案中使用的负载平衡法是把请求分配给两个 Web 服务器系统。

Note

在本章中使用的名称 XC News 和 IP 地址，都是虚构的，如与某公司及其 IP 地址有任何类同，纯属巧合。

让我们看看这个项目详细的需求:

- 目的ó ó XC News 有成千上万的客户，他们希望 XC News 为他们建立 Web 站点，并提供对 XC News 开发的 Web 应用和其他服务的访问。在集成环境上使用了单个基于 Windows NT 的服务器后，XC News 现在需要更好的解决方案，因为他们未来的项目表明他们所要提供的基于 Internet 的服务可能迅猛地增长。这样，公司需要开发一个由多个 Web 服务器系统组成的 Web 网络，它能为客户站点提供一个可扩展的、可靠的环境。这个网络最初有两个 Web 服务器和一个监视服务器系统，但网络的设计和实施必须有足够的灵活性以便在需要时可以增加多个服务器。
- 服务器平台ó ó XC News 认识到分布式网络及如何利用分布式网络的体系结构提供可扩展的解决方案的重要性。为了满足 Web 网络上通信量的突然增加，必须具有迅速增

需求。XC News 需要建一个 Web 服务器网络，在网络上的每个服务器将容纳所有客户的 Web 站点和 XC News 的站点。这个网络应该这样工作，即 URL 请求被自动地路由到运行同样的 Web 站点的不同机器上。图 19-1 显示了这种需求如何实现。

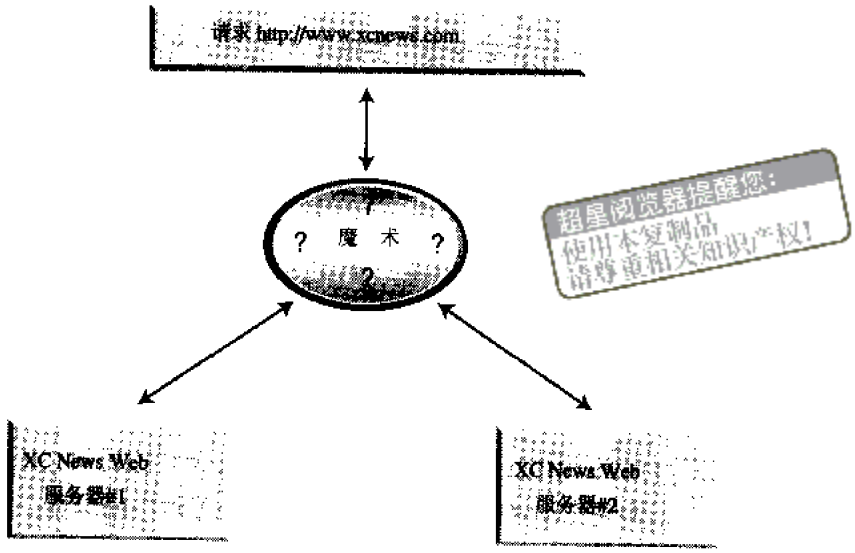

图 19-1 URL 请求如何进行分配

正如你所看到的，一个请求如 www.xcnews.com 必须能够魔术般地既可以由 Web 服务器#1 服务，也可由 Web 服务器#2 服务。这种“魔术”将必须使用循环域名服务来实现。在单个 Web 服务器失败的情况下，必须容易地把传输重新定向到另一个工作着的服务器。

需求同样说明必须有一个单独的试用区用于客户 FTP 文件。这个试用区还必须分发到每个 Web 服务器上，以便每个服务器有为所有 URL 请求服务所需要的文件。那么试用区能放到哪儿呢？在回答这个问题之前，需要区分服务器系统上两种不同的需求。如果仔细地阅读了这些需求，应知道 XC News 公司想把 Web 服务器作为生产系统。减少用户同生产系统的交换数据总的来说是个好主意，因此试用区不能放在生产系统上。它必须放在第三个系统——DNS 服务器和监视服务器上。现在应该做的是在 DNS 服务器上创建一个试用区，并把试用文件分发到生产 Web 服务器。

如何使这些文件分发到 Web 服务器上，这里有两种方案：

- 使用网络文件系统（Network File System，NFS），与 Web 服务器共享试用区。
- 使用名叫 rdist 的流行的远程分发软件把文件分发到 Web 服务器。

如果所有 Web 站点使用 NFS，并且有大量的文件，那么 NFS 将会成为性能瓶颈，因为 NFS 并不像本地文件系统一样好。这时，rdist 成为好的选择。与 UNIX 系统上的 cron 功能一起使用 rdist，就可在定时的基础上实现文件分发。然而这里仍有一个突出问题（同步问题）需要关注，即 CGI 脚本写数据的同步问题。如果使用 rdist 同步 CGI 数据文件，有一个时间间隔，在间隔期间在服务器上的数据文件将不同步。另外，还会存在同一文件的多个拷贝，这可能造成各种问题。因此，很明显 CGI 数据文件应该以其他方式保持同步。

一个好主意是：在同一个目录下写所有的 CGI 数据文件。这样要做的只是把这个目录作为 NFS 输出目录安装到所有的 Web 服务器上。作为结果，所有 Web 服务器将读、写同一个文件。

NFS 不提供任何文件锁定机制。因此，需要 CGI 脚本自己实现某种文件锁定机制是很重要的。

超星阅览器提醒您：
请勿本复制品
请尊重知识产权！

从这些需求可以明显看出安全服务器必须运行在 DNS 服务器上，表 19-1 显示了网络内每个系统所运行服务器。

表 19-1　每个系统提供的服务

系　统	服　务
DNS 服务器	域名服务 监视服务 用于 CGI 数据文件目录的 NFS 服务器 试用区的 FTP 服务器 从试用区分发文件的 rdist 服务器 安全服务器
Web 服务器#1	Apache 服务器 使用 CGI 数据文件目录的 NFS 客户 rdist 客户
Web 服务器#2	Apache 服务器 使用 CGI 数据文件目录的 NFS 客户 rdist 客户

注意，不必开发内部的监视方案，一个称为 Spong 的简单的基于 Perl 的监视软件包（包含在 CD-ROM 中）能用来满足这些需求。由于其安装和设置过程非常简单，对它不作进一步的讨论。

现在，每台机器上的服务已确定，让我们看一下这个网络可能的传输类型。对于 XC News，有三种类型的网络传输：

♦普通的 Internet 传输——每个服务器要同 Internet 客户通信。

♦rdist 文件同步传输——这种特定的传输类型将只涉及包含在文件分发过程中的三台机器。这种传输不能也不应传到任何外部主机，或来自任何外部主机。

♦NFS 传输——这种特定的传输类型将只涉及三台机器。DNS 服务器作为 NFS 服务器，它向其他两台 Web 服务器输出某些目录。NFS 传输不能也不应该传到任何外部主机，或从任何外部主机传入。

现在已经确定了服务项目和传输需求，该是设计网络的时候了。

19.3 设计 Web 网络

设计 Web 网络的第一步是画出总体方案。图 19-2 显示了一个总的网络图。如图中看到的，Web 服务器和 DNS 服务器通过以太网集线器连到 ISP 的网络。所有来自和通向 Internet 的传输通过这个集线器到达这个系统。这种设计的网络能够工作，但不是优化的。为什么？ 如果仔细想一想，这个网络用单个以太网集线器把三台机器连接到 Internet，并实现它们之间相互的连接。这意味着所有 NFS 和 rdist 传输将和 Internet 传输通过同一个以太网集线器。这使得以太网很繁忙，所以它不是好的解决方案。

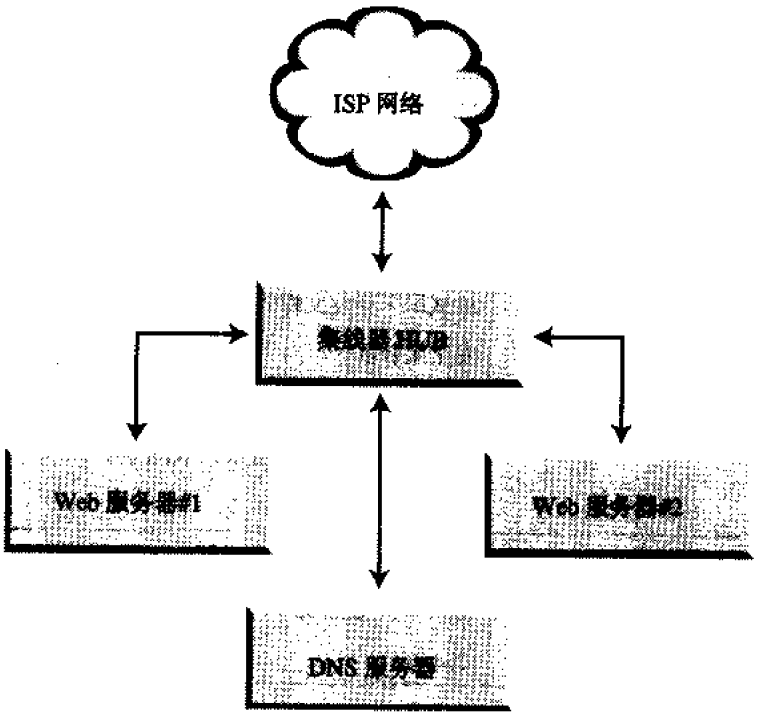

图 19-2 网络框图（单以太网）

要解决这个问题，可以如图 19-3 所示重新设计这个网络。在这种设计里，使用了两个以太网集线器（hub）。一个把服务器连接到 ISP 网络，另一个把服务器相互连接起来。这种设计有什么好处？ 这样能把 NFS 和 rdist 传输保持在服务器之间的以太网上，而连接到 ISP 网络的以太网只用于 Internet 服务。换句话说，一个以太网用作局域网（LAN），另一个用于 Internet 访问。

图 19-4 表示网络传输框图。这是一种更好的方案，因为 NFS 和 rdist 传输只通过 LAN，因此使另一个的以太网只用于 Internet 连接。它也为网络的安全增加了保护。在系统管理员中，普遍相信 NFS 并没有其应该达到的那样安全。因此如果使用单个以太网，并且向 Web 服务器输出 NFS 目录（的内容），那么以太网将很可能成为攻击点。

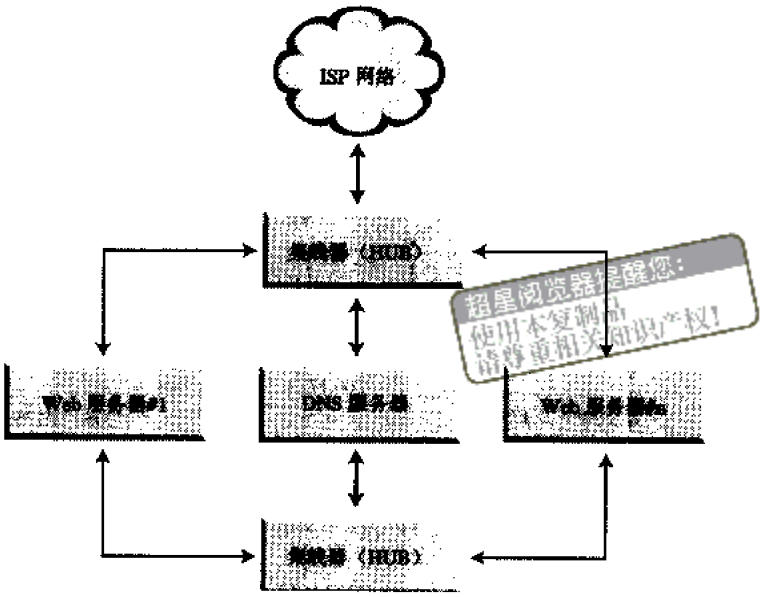

图 19-3　网络框图（多以太网）

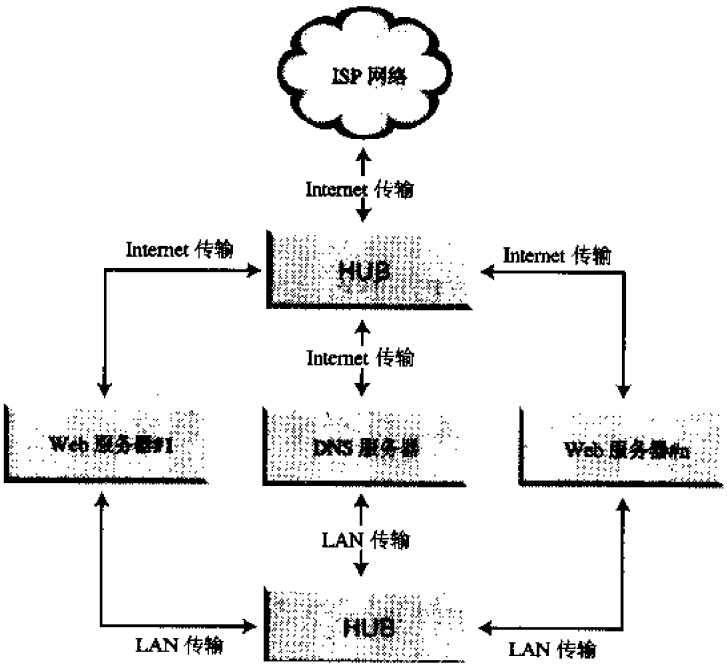

图 19-4　网络传输框图

许多系统已经因为与 NFS 有关的安全漏洞而受到攻击，因此避免通过 Internet 使用 NFS 目录是个好主意。为了确保除了三个系统外 LAN 不被任何其他系统访问，不要在两个以太网之间存在直接的路由。这能阻止人们从 Internet 上访问 LAN。可以在 LAN 上使用不可路由的 IP 地址，这进一步增强了方案的安全。图 19-5 显示在两个网络上节点的 IP 地址分配，如图所示，LAN 使用网络地址 192.108.1.60，它在 Internet 上是不可路由的。

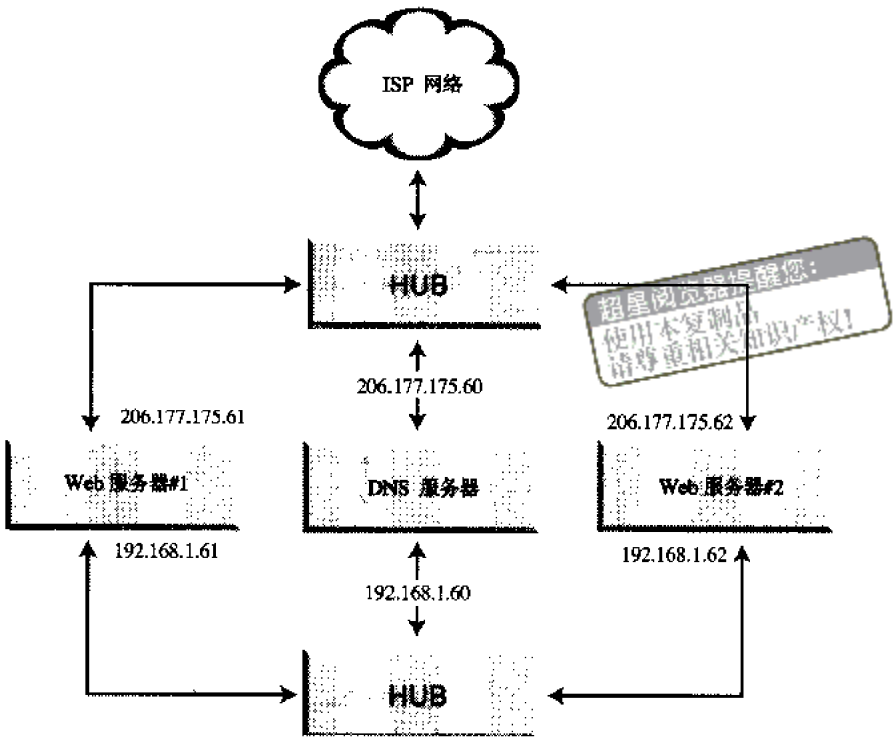

图 19-5 网络的 IP 地址分配

Note 也可使用 A 类网的其他不可路由的 IP，如 10.0.0.0。

在 Internet 一端使用的 IP 地址依赖于 ISP，这些必须是在 Internet 上可路由 IP 地址。

现在关于网络的知识已经够用了，可以讨论硬件和软件的列表了。

19.4 选择硬件和软件

正如前面提到的，XC News 公司需要使用 PC 服务器而不是昂贵的服务器平台，这样当需求增长时可以很容易地增加服务器。从这点出发，可以列出用于网络服务器的技术要求清单。以个人的经验看，使用名牌的 PC 比非名牌的 PC 更值得，因此更喜欢使用下面厂商的机器作为服务器系统：

- Micron (www.micronpc.com)
- ASA Computers (www.asacomputers.com)

现在让我们看一下服务器系统需要什么硬件。

19.4.1 建立 PC 服务器

如果有一个容易升级的系统，总是件好事。因为 Intel 已经决定在它的新 CPU 上使用 Slot 系统，所以使用类似 Socket7 的 Socket CPU 并不是好主意。我想这种 CPU 很快将会

难以找到。另外，新的 Pentium II 处理器非常快，应成为下一代 PC 服务器必要组成部分。推荐使用 233MHz Intel 处理器作为服务器。系统的母板是非常重要的，因此推荐使用来自 Intel 的高端母板。

下一项应考虑的是 RAM 的容量。一般规则是 RAM 越大越好，然而 RAM 是昂贵的，特别是 10ns 的 SDRAM。建议每台服务器使用 256MB 的 SDRAM。

每个系统应该有两个连接到名牌的 SCSI UW 控制器的大容量 SCSI 硬盘。每个系统至少有两个磁盘是个好主意，因为它可以使操作系统在一个硬盘上，而数据在另一个磁盘上。数据硬盘的总的数据容量可由公司当前和将来的需要决定。对于 XC News，可计算出用于所有 XC News 站点的总的磁盘空间容量接近 200MB。这样，现在有一个 4GB 的硬盘用于存放数据就够了。

使用 SCSI 的磁盘系统有很多好处。当需求增长时，可以很容易在 SCSI 控制器上增加磁盘。几乎所有流行品牌的 SCSI 控制器都可以连接最小 7 设备，最大到 15 个设备。

每个系统也应当有两块著名品牌的 100Mbps PCI 以太网络适配卡，一块正规的 SVGA 视频卡，和一个具有双电源支持的服务器级的机箱。视频卡很少使用，因此它不需要太多的 RAM。

类似的，每个系统应有一个 IDE CD-ROM 驱动器和一个软盘驱动器。服务器机箱必须很结实，有足够的空间和高质量的电源供应系统。双电源支持系统提供安全网络支持。如果一个电源出故障，另一个启动以提供不间断电源服务。

对于 XC News 网络，还需要额外的一些设备：两个 100base-TX 以太网集线器（如图 19-5 中所示），一些做好接头的 5 类 100base-TX 电缆，一个 9 英寸 SVGA 监视器和一个 101 键键盘，把它们存放到 ISP 以备在紧急情况下使用。

到此为止， XC News 使用的 PC 服务器配置和附加的硬件也已经选好，该是考虑软件的时候了。

19.4.2　选择操作系统

可以获得许多基于 x86 PC 的 UNIX 操作系统。例如，Linux、FreeBSD、BSDI 和 Solaris。对于 XC News，让我们使用 Linux。我曾经在许多实际的网络上使用它，并且总是很成功。特别的，让我们使用 Linux 的 RedHat 发行版，因为它易于安装，并且 RedHat 保持它们的 Linux 版本紧跟最新的发展。RedHat Linux 的商业版本可以从 www.redhat.com 订购。

19.4.3　选择 Web 服务器

Web 服务器很容易选择，让我们使用 Apache 稳定的发行版 1.2.6 替代最新的 Bata 版。本人不喜欢在生产系统上运行 beta 版软件。

正如第 15 章中讨论的，软件 Stronghold 对于基于 Apache 的安全服务器是好的选择，因此，让我们把这个程序加到硬件和软件清单上。

到现在为止，硬件和软件都已经选好，可以建立系统了。

19.5 建立系统

为了建立系统，需在每台系统上安装 RedHat Linux，但在这之前，必须要决定如何进行磁盘分区。

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

19.5.1 磁盘分区

几乎所有 UNIX 系统都有许多标准的磁盘分区，如 root（根）分区、/usr 和/home。磁盘必须进行分区，以便操作系统和其他软件的管理能够实现某种程度的标准化。即使新入门的 UNIX 系统管理员也能告诉你操作系统软件应放在 root 分区，大部分的其他软件应驻留在/usr 分区。这已是 UNIX 的传统，主要使系统管理变的更容易。表 19-2 表示对每个 XC News 系统推荐的操作系统磁盘分区。

表 19-2 操作系统盘的磁盘分区

分区	容量（MB）	解释
/	512	根分区
/usr	1024	安装各种软件的标准分区
/home	384	用户主目录
swap	128	交换区

注意，用户主目录位于 /home 分区。但它并不需要太大。这是因为注册的用户大部分在其主目录里没有很多东西。事实上，大部分 XC News 客户的主目录被指向一个 Web 目录，这将在本章后面更详细进行讨论。

表 19-3 表示为每个 XC News 系统推荐的数据盘分区。注意，虽然用户并不注册到 Web 服务器系统上，但还是保持在所有三个系统上操作系统磁盘分区是一样的。

表 19-3 数据盘的磁盘分区

分区	容量（MB）	解释
/www	3072	Web 数据区
/log	1024	日志文件区

日志分区需要 1GB（1024MB），这是因为许多 Web 站点对于每个请求都产生一个日志条目。

19.5.2 安装 Linux

现在是在每个 XC News 系统上安装 Linux 的时候了。RedHat 软件安装起来很轻松，所需做的就是用一个启动盘启动系统并分区磁盘，正如所期望的那样。安装 RedHat 有许

多方式，但我更愿意从本地 CD-ROM 上安装它，因为这种方法似乎花费的时间最少。

下一步，使用简单的菜单安装程序安装操作系统。当安装服务器的操作系统时，不要安装任何不必要的程序，限制只安装必要的程序以减少磁盘空间的使用，同时也增强了系统的安全性。例如，XC News 的服务器不需要任何 X Windows 软件，因此 X Windows 和与 X Windows 相关的软件包应排除在安装之外。XC News 的每个系统确实需要基本的操作系统和网络组件，如 FTP、NFS、SMTP 和 Apache 服务器。其他软件，诸如：新闻服务器，新闻阅读器，其他编辑器，用于网管服务的支持软件，打印机守护进程等等，按需要应不被安装到服务器系统上，因此在每个系统上都不安装它们。

超星扫描仪：请尊重相关知识产权！

19.5.3　配置网络

现在为每个系统配置两个网络。

从 Internet 网络开始，它是连接到 ISP 网的网络。用于 Internet 网络的系统应该使用表 19-4 所示的 IP 地址进行配置。

表 19-4　Internet 网络使用的主机名和 IP 地址

主机名	IP 地址	机器
ns.xcnews.com	206.177.175.60	DNS 服务器
www1.xcnews.com	206.117.175.61	Web 服务器#1
www2.xcnews.com	206.177.175.62	Web 服务器#2

这些 IP 地址不是任意选取的，它们由 ISP 分配。ISP 提供 IP 地址的反向域名解析服务，同时，ns.xcnens.com 为它自己和所有客户的域名提供主域名解析服务。ISP 也为 XC News 的所有域名提供二级 DNS 服务。

ISP 为这个网络提供网络掩码（255.255.285.0）和缺省的网关 IP，以使所有的传输通过集线器到达他们的网关。

现在，来为局域网（LAN）选择虚构的域名，如 xcnews-lan.com，并分配不可路由的 IP 地址，如表 19-5 所示。

表 19-5　xcnews-lan.com 的主机名和 IP 地址分配

主机名	IP 地址	机器
ns.xcnews-lan.com	192.168.1.60	DNS 服务器
www1.xcnews-lan.com	192.168.1.61	Web 服务器#1
www2.xcnews-lan.com	192.168.1.62	Web 服务器#2

使 IP 地址的主机部分同另外一个网络一样，这样便于系统管理员记忆。此网络的掩码也是 255.255.255.0，因为用了一个 C 类网址（192.168.1.0），缺省的网关是域名服务器

系统——ns.xcnews-lan.com。

一旦已安装操作系统且所有的网络也已配置好，就可以使用 ping 程序测试网络。

要测试 Internet 网络，从一台主机发一个 ping 请求到另一个主机，每次使用合适的 IP 地址。例如，为了从 ns.xcnews.com ping www1.xcnems.com 系统，使用下面的命令：

```
ping 206.177.175.61
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

因为名字服务器还未建立，故用主机名 ping 是不可能的。当从一个主机 ping 另一个主机时，注意观察 Internet 端的集线器的 LED 显示急速的闪烁而另一个却仍很稳定。这是证明，ping 的数据包通过了正确的集线器。ping xcnews-lan.com 网络上的 IP 地址会使另一个集线器闪烁。当然，也能从路由表确定数据包的路由，命令如下：

```
/sbin/route  -n
```

它显示的路由表如列表 19-1 中所示。

列表 19-1：名字服务器的路由表

```
Kernel IP routing table
Destination   Gateway        Genmask        Flags Metric Ref  Use
Iface
206.177.175.0 0.0.0.0        255.255.255.0   U     0      0    0
179 eth0
192.168.1.0   0.0.0.0        255.255.255.0   U     0      0    0
23 eth1
127.0.0.0     0.0.0.0        255.0.0.0       U     0      0    0
2 lo
0.0.0.0       206.177.175.1  0.0.0.0         UG    0      0    53116
eth0
```

下一步，应该设置叫做 BIND 的 DNS 服务器软件。

19.6 设置 DNS 服务器

XC News 的 Web 网络的 DNS 使用称为循环（round-robin）DNS 的 DNS 技术。在这种设置里，DNS 服务器的配置为，一个主机名有多个 IP 地址，因为 XC News 需要使它的所有 Web 服务器以分布式方式为同一组域名服务，每个主机名必须指向两个 IP 地址。图 19-6 说明了这一点。

DNS 服务器使用的文件/etc/named.boot 显示在列表 19-2 中。注意：将 DNS 服务器配置为 xcnews.com 和 xnews-lan.com 的主 DNS 服务器。因为 ISP 提供主反向 DNS 解析服务，它被设置为 175.177.206.in-addr.arpa 域的第二 DNS，而对于 1.168.192.ma-addr.arpa 域，它被设置为主反向 DNS。

```
ns          IN  A          206.177.175.60
www1        IN  A          206.177.175.61
www2        IN  A          206.177.175.62

; CNAME records
mail            IN  CNAME      ns
ftp             IN  CNAME      ns
www             IN  CNAME      www1
www             IN  CNAME      www2
secured         IN  CNAME      ns
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

特殊的 CNAME 记录如下：

```
www IN  CNAME      www1
www IN  CNAME      www2
```

这些记录表明 www.xcnews.com 是 www1.xcnews.com 和 www2.xcnews.com 的 CNAME。那么，当客户请求 www.xcnews.com 的地址时，域名服务器会做些什么呢？ 它会给出 www1.xcnews.com 或者 www2.xcnews.com。在循环方式下， DNS 服务器对第一个指向 www.xcnews.com 的请求返回 206.177.175.61（www1.xcnews.com），对于下一个指向 www.xcnews.com 的请求返回 206.177.175.62（www2.xcnews.com）。xcnews.com 的反向数据库记录显示在列表 19-4 中。

列表 19-4：/var/mamed/xcnews.rev

```
@  IN   SOA    xcnews.com.       hostmaster.xcnews.com. (
               19980205000       ; serial YYYYMMDDXXX
               7200              ; refresh
               3600              ; (1 hour) retry
               1728000           ; (20 days) expire
               3600)             ; (1 hour) minimal TTL

; Name server and Mail exchange records
   IN      NS      ns.xcnews.com.
   IN      MX      5 mail.xcnews.com.
; PTR  Records
60      IN      PTR     ns.xcnews.com.
61      IN      PTR     www1.xcnews.com.
62      IN      PTR     www2.xcnews.com.
```

注意，这里并不需要为 CNAME 记录生成 PTR 记录，因为它们只是那些必须有 PTR 记录的记录的别名。因此，当客户想执行反向域名查找，它将得到规范名 www1.xcnens.com 或 www2.xcnews.com，这两个每个都有一条 PTR 记录。

xcnews-lan.com 的 DNS 记录显示在列表 19-5，它们很简单明了。

列表 19-5：/var/named/xcnews-lan.db

```
@  IN SOA  xcnews-lan.com.      hostmaster.xcnews-lan.com. (
        19980205000             ; serial YYYYMMDDXXX
            7200                    ; refresh
            3600                    ; (1 hour) retry
            1728000                 ; (20 days) expire
            3600)                   ; (1 hour)minimal TTL

; Name server and Mail exchange records
        IN   NS        ns.xcnews.com.

; A records
ns      IN  A      192.168.1.60
www1    IN  A      192.168.1.61
www2    IN  A      192.168.1.62
```

注意，没有提供邮件交换（MX）记录，因为这里不需要。xcnews-lan.com 的反向 DNS 记录在列表 19-6 列出，也同样的简单。

列表 19-6：/var/named/xcnews-lan.rev

```
@ IN    SOA     xcnews-lan.com.      hostmaster.xcnews-lan.com. (
                19980205000          ; serial YYYYMMDDXXX
                7200                 ; refresh
                3600                 ; (1 hour) retry
                1728000              ; (20 days) expire
                3600)                ; (1 hour) minimal TTL

; Name server and Mail exchange records
        IN NS          ns.xcnews-lan.com.

; PTR Records
60      IN  PTR        ns.xcnews-lan.com.
61      IN  PTR        www1.xcnews-lan.com.
62      IN  PTR        www2.xcnews-lan.com.
```

这里关键的事情是如何使 XC News 增加新 Web 服务器和提供循环域名解析服务变得很容易。如果公司的系统管理员决定要增加一个新的 Web 服务器系统，称为 www3.xcnews.com（206.177.175.63）。他们所有需要做的是在/var/named/xcnews.db 文件添加如下记录：

```
www3    IN      A          206.177.175.63
www     IN      CNAME      www3
```

然后在/var/named/xcnews.rev 文件里增加一个新的 PTR 记录，如下：

```
63      IN      PTR         www3. xcnews.com.
```

当然也要更新/var/named/xcnews-lan.db 文件，如下：

```
www3    IN      A           192.168.1.63
```

最后要更新/var/named/xcnews-lan.rev 文件，更新如下：

```
63      IN      PRT         www3.xcnews-lan.com
```

一旦这些改变完成后，就可重新启动域名服务器。现在 www.xcnews.com 将指向所有三个 Web 服务器。

所有这些都很好，但是客户站点如何从循环 DNS 方案中获取得好处？让我们看一个例子，假设 XC News 的客户需要在这个网络上建立 client-one.com 域名。在这种情况，应在/etc/named.boot 文件里增加如下一行：

```
Primary   client-one.com   client-one.db
```

然后，需要在/var/named 创建一个名为 client-one.db 的文件，内容如列表 19-7 所示。

列表 19-7：client-one.db

```
@  IN   SOA    client-one.com.      hostmaster.client-one.com. (
               19980205000          ; serial YYYYMMDDXXX
               7200                 ; refresh
               3600                 ; (1 hour) retry
               1728000              ; (20 days) expire
               3600)                ; (1 hour) minimal TTL

; Name server and Mail exchange records
       IN      NS          ns.xcnews.com.

; CNAME records for        www.client-a.com
www    IN      CNAME       www.xcnews.com.
```

这里设置 ns.xcnews.com 作为 client-one.com 的 DNS 服务器，并使 www.client-one.com 主机指向 www.xcnews.com。因为 www.xcnews.com 是 www1.xcnews.com 和 www2.xcnews.com 的别名，对于 www.client-one.com，DNS 服务器也将给出 www1.xcnews.com 和 www2.xcnews.com 的主机 IP 地址。这基本上如所期望的那样实现了循环方案。现在，如果 XC News 增加一个新 Web 服务器，client-one.com 的 DNS 记录不需改变，只要将 www.xcnews.com 成为新 Web 服务器的别名就行了。

当一个 Web 服务器不可使用时将发什么情况？ 例如，假设 www1.xcnews.com 需要修理。作为结果，DNS 里需要改变什么？ /var/named/xcnews.db 文件需要进行如下的修改。将下面的记录：

```
www     IN      CNAME      www1
```

加注释如下：

```
; www     IN      CNAME      www1
```

改变后，需重新启动 DNS 服务器以使改变生效。然而，由于 DNS 的本身性质，改变仍需花费一些时间才能传播开来。

虽然与 XC News 客户相关的 DNS 设置变得很简单，但是在设置过程中为人为的错误留下了空间。例如，如下一行所示，如果在行末不加句号，这一设置将不发挥作用：

```
www     IN      CNAME      www.xcnews.com
```

超星阅览器产品
使用本复制品
需重视知识产权！

为了减少出现人为错误的机会，很有必要生成一个 perl 的脚本，它用一个模板（如列表 19-8 所示）生成新的客户数据库文件。另外，模板也要修改/etc/named.boot 文件以便把 ns.xcnews.com 配置为客户域的主 DNS 服务器。

列表 19-8：named.template

```
@    IN SOA    <DOMAIN>.          hostmaster.<DOMAIN>. (
               19980205000        ; serial YYYYMMDDXXX
               7200               ; refresh
               3600               ; (1 hour) retry
               1728000            ; (20 days) expire
               3600)              ; (1 hour) minimal TTL

; Name Servers
       IN      NS         ns.xcnews.com.
       IN      MX         10mail.xcnews.com.

; CNAME records
mail       IN      CNAME      ns.xcnews.com.
ftp        IN      CNAME      ns.xcnews.com.
www        IN      CNAME      www.xcnews.com.
```

注意，在模板文件里，域名由<DOMAIN>标记指定，脚本用客户的域名替换此标记，XC News 还要为每个用户进行附加的 Apache 配置。因为在此过程中也可能出现人为的错误，所以也使用 Perl 脚本用另外一个模板文件生成必要的<VirtualHost ...>容器配置，所用模板文件如列表 19-9 所示。

列表 19-9：httpd.template

```
# Domain Configuration for <WWW-SITE>
#
<VitualHost REPLACE-THIS-WITH-WEB-SERVER-IP-ADDR>
ServerAdmin webmaster@<DOMAIN>
DocumentRoot <HTDOCS-DIR>
#ScriptAlias  /cgi-bin/ <CGI-BIN-DIR>/
ScriptAlias       /cgi-bin/  /www/xcnews/cgi-bin/
Alias         /cgi-data/ /www/cgi-data/
```

```
ServerName <WWW-SITE>
ErrorLog          logs/<WWW-SITE>.error.log
TransferLog       logs/<WWW-SITE>.accees.log
</VirtualHost>
#
# End of domain Configuration for <WWW-SITE>
```

这个 Perl 脚本称为 makesite，它用 www.client-domain-name 替换<WWW-SITE>标记。脚本列在列表 19-10 中。

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

列表 19-10：makesite 脚本

```
#!/usr/local/bin/perl
#
# Purpose: makesite creates victual sites. It uses set
# of templates to create DNS, HTTPD, and SMTP
# configurations and appends these configuration data
# to appropriate configuration files.
#
# $Author$ (kabir@nitec.com)
# $Version$
# $Id$
# $Status
#
##########################################

# You might need to change some of these values before
# you can use this script on your system
#

# Directory where the script is stored
my $MAKESITE_DIR ='/usr/local/build/makesite';

# The username which owns the newly created client
# directory
my $USER = 'httpd';

# The group which owns the newly created client
# directory
my $GROUP = 'httpd';

# Default file permission for $USER.$GROUP
my $PERMISSION = '2770';

# The Web partition directory
my $BASE_DIR = '/www';
```

```
# Document Root directory (relative to
# /$BASE_DIR/<CLIENT-DOMAIN-DIR>)
my $HTDOCS = 'htdocs';

# CGI program directory (relative to
# /$BASE_DIR/<CLIENT-DOMAIN-DIR>)
my $CGIBIN = 'cgi-bin';

# FQPN of BIND DNS record directory
my $NAMED_PATH = '/var/named';

# File extension for each domain record file
my $NAMED_FILE_EXT = '.db';

# FQPN of the BIND DNS record template file
my $NAMED_TEMPLATE_FILE = "$MAKESITE_DIR/named.template";

# FQPN of BIND named.boot file
my $NAMED_ETC_FILE = '/etc/named.boot';

# FQPN of the Apache httpd.conf file
my $HTTPD_CONF_FILE = "/www/apache/conf/httpd.conf";

# FQPN of the virtual host template file
my $VIRTUAL_HOST_TEMPLATE = "$MAKESITE_DIR/httpd.template";

# FQPN of the log file
my $LOG_FILE = "$BASE_DIR/makesite.log";

# FQPN of the 'date' program in back-ticks.
my $date = '/bin/date';

# Remove the newline character from date.
chomp ($date);

# Get the first argument passed to the script.
my $site = $ARGV[0];

# The argument is the site name (such as nitec.com)
# So assign it to a variable called $dir.
my $dir = $site;

# Store the allowed domain types in an array
my @domain_types = ( com , net , org , edu ) ;

my $domain_ext;  # used to store domain extension
```

超星阅览器提醒您:
使用本复制品
请尊重相关知识产权!

```
my $thesite_dir; # used to store site's dir. name
my $htdocs_dir:  # used to store DocumentRoot dir. name
my $cgibin_dir;  # used to store cgi-bin dir. name
my $named_file;  # used to store site's DNS file name
my $dir_len;     # length of dir. name
my $temp_len;    # temp. variable

# Lowercase the site name supplied by user.
$site =~ y/[A-Z]/[a-z]/;

# If no arguments to the script was suppplied
# then $site will be empty show syntax in such a case
&syntax if($site eq ' ' );

# Get the lenght with the EXT
$dir_len = length($dir);

# Remove the domain extension from the supplied name
foreach $domain_ext (@domain-types){
$dir =~ S/\.$domain_ext//g;
}

# Get the new length without the EXT
$temp_len = length($dir);

# If the user has not entered an extension then show syntax.
&syntaX if($temp_len == $dir_len);

# Create FQFN names
$named_file = $NAMED_PATH . '/' . $dir . $NAMED_FILE_EXT ;
$thesite_dir = $BASE_DIR . '/' . $dir;
$htdocs_dir = $BASE_DIR . '/' . $dir . '/' . $HTDOCS;
$cgibin_dir = $BASE_DIR . '/' . $dir . ' /' . $CGIBIN;

# If this site's dir. Already exists show error msg and exit
if(-e $thesite_dir) {
print "$thesite_dir already exist! No action taken for
$thesite_dir\n";
exit 0;
}

# OK. It is time to create the necessary directories.
system( "mkdir $thesite_dir");
system( "mkdir $htdocs_dir");
system( "mkdir $cgibin_dir");
```

超星阅览器提醒您:
使用本复制品
请尊重相关知识产权!

```
# Create the DNS record file for this site
&createNamedFile ($named_file , $site,"$dir$NAMED_FILE_EXT");

# Create the dummy index file in the DocumentRoot dir.
&createIndexFile($htdocs_dir, $site);

# Create the virtual host configuration in httpd.conf
&createVirtualHostConf(domain=>$site, website=>"www.$site",
cgibi n=>$cgibin_dir,htdocs=>$htdocs_dir);

# Change ownership and access permissions for site dirs.
system("chown -R $USER.$GROUP $thesite_dir");
system("chmod -R $PERMISSION $thesite_dir");

# Write log file
open(FP, ">$LOG_FILE")||die "Can't write to log file.\n";
print FP "Date: $date created www.$site [$htdocs_dir]\n";
close(FP);

# Done!
exit 0;
sub createNamedFile{
#
# Purpose: create the DNS record file in BIND file format
#

# Get passed parameters
my $file = shift;
my $domain = shift;
my $database = shift;
my $line;

# Open the DNS record file for writing
Open (OUT,">$file") || die "Can't write $file\n";

# Open the named template for reading
open (FP, $NANED_TEMPLATE_FILE) || die "Can't open
$NAMED_TEMPLATE_FILE\n";

# Read each line in the template file and replace
# the <DOMAIN> tag with user specified site name
# which is passed to $domain variable
#
while($line=<FP>){
    $line =~ s/<DOMAIN>/$domain/g;
    print OUT $line;
```

```
}

# Close all files opened here
close(FP);
close(OUT);

# Open the named.boot file and add the site in the
# file to setup primary DNS for it.
open(FP,">$NAMED_ETC_FILE") || die "Can't open
$NAMED_ETC_FILE\n";

# Write the record
print FP "primary\t\t$domain\t\t$database\n";

# Close the file
close(FP);
}

sub createIndexFile{
#
# Purpose: create a dummy index.html file in the
# new site's DocumentRoot directory.

# Get the arguments passed to this sub routine
my $htdocs = shift;
my $domain = Shift;
# Write the index.html file.
open(FP, ">$htdocs/index.html") || die "Can't write index.html\n";
print FP <<INDEX_PAGE;
<HTML>
<HEAD> <TITLE> $domain </TITLE></HEAD>
<BODY BGCOLOR="white">
<CENTER> This is $domain </CENTER>
</BODY>
</HTML>
INDEX_PAGE

# Close file
close(FP);
}

sub createVirtualHostConf{
#
# Purpose: create the virtual host configuration
# using the template file and store the configuration
# in the httpd.conf file.
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

```
# Get all parameters passsed to this sub routine
my %params = @_;

# Temp. variable
my $line;

# Open httpd.conf for appending data
open(OUT, ">$HTTPD_CONF_FILE") || die "Can't open
$HTTPD_CONF_FILE\n";

# open victual host template file for reading
open(FP, $VIRTUAL_HOST_TEMPLATE) || die "Can't open
$VIRTUAL_HOST_TEMFLATE\n";

# Read each line of template file and replace
# special tags with values
while($line =<FP>) {
    $line =~ s/<DOMAIN>/$params{domain}/g;
    $line =~ s/<CGI-BIN-DIR>/$params{cgibin}/g;
    $line =~ s/<HTDOCS-DIR>/$params{htdocs}/g;
    $line =~ s/<WWW-SITE>/$params{Website}/g;
    print OUT $line;
    }

# Close all files
  Close(FP);
  Close(OUT);
    }

sub syntax{
#
# Purpose: to display script's syntax.
#
print <<SYNTAX;

makesite <virtual Internet domain>

Example: makesite nitec.com

Do not confuse the domain name with host name such as
www.nitec.com.
this script will create www.<virtual Internet domain> host
automatically.

SYNTAX
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

```
# Exit after showing the syntax.
exit 0;
}
```

此脚本可以运行如下：

```
./makesite 客户域名
```

例如：

```
./makesite client-a.com
./makesite client-b.org
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

这些命令将为 client-a.com 和 client-b.org 创建 DNS 和 Apache 配置。DNS 配置文件存放在 /var/named/client-a.db 和 /var/named/client-b.db。Apache 配置存放在 /www/apache/conf/httpd.conf 文件里。此脚本也为每个客户站点生成相应的目录结构。例如，client-a.com 将会有一个如图 19-7 所示的目录结构。

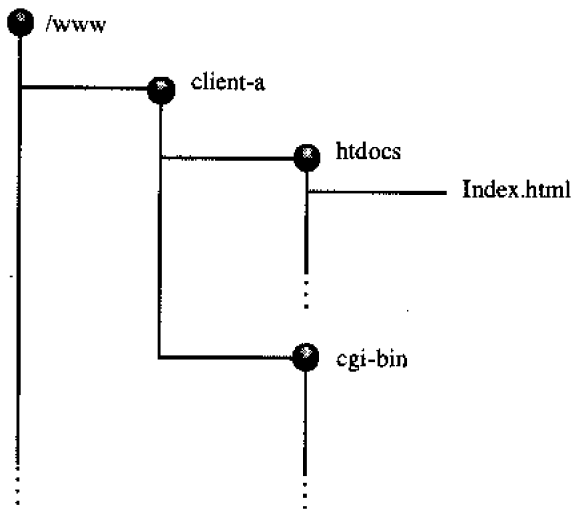

图 19-7 www.client-a.com 站点的目录结构

makesite 脚本也将生成一个 Index.html，这使测试新生成的站点变得很容易。为每个新客户域运行 makesite 脚本后，重新启动域名服务器，并把生成的 Apache 配置（即 /www/apache/conf/httpd.conf 文件里新的<VirtualHost>容器部分）添加到每个 Web 服务器的 httpd.conf 文件里。例如，当运行 makesite client-a.com 时，列表 19-11 所显示的 Apache 配置被增加到/www/apache/conf/httpd.conf 文件里。

列表 19-11：/www/apache/conf/httpd/conf 文件

```
# Domain Configuration for www.client_a.com
#
<VirtualHost REPLACE-THIS-WITH-WEB-SERVER-IP-ADDR>
ServerAdmin webmaster@www.client-a.com
DocumentRoot /www/client-a/htdocs
#ScriptAlias /cgi-bin/ /www/client-a/cgi-bin/
```

```
ScriptAlias /cgi-bin/ /www/xcnews/cgi-bin/
Alias /cgi-data/ /www/cgi-data/
ServerName www.client-a.com
ErrorLog logs/cerror.log
TransferLog logs/www.client-a.com.access.log
</VirtualHost>
#
# End of Domain Configuratin for www.client-a.com
```

超星阅览器提醒您：
使用本复制品
需遵相关知识产权！

下一步，必须把列表 19-11 的部分从/www/apache/conf/httpd/conf 文件中拷贝出来粘贴到 www1.xcnews.com 和 www2.xcnews.com Apache 服务器的 httpd.conf 配置文件里。注意 REPLACE-THIS-WITH-WEB-SERVER-IP-ADDR 必须用相应的 IP 地址替换。例如，当把这配置段加到 www1.xcnews.com Apache 服务器的 httpd.conf 时，要用 206.177.175.61 替换 REPLACE-THIS-WITH-WER-SERVER-IP-ADDR。

注意，在客户域的新目录被分发到 Web 服务器之前，不要重新启动 Apache Web 服务器。这又把我们带到了这一项目的文件分发问题。

19.6.1　使用 rdist 分发文件

rdist 程序能够在多台主机上保持文件的一致的拷贝。它使用 rcmd 函数或远程 shell（rsh）访问每台目标计算机。

使用 rdist 的最容易的方法是在所涉及的所有主机上建立公共的账号，并在目标系统（www1 和 www2）上建立 .rhost 文件，这样在域名服务器上的此公共用户就被允许运行 rsh 会话过程。为了此目的，在涉及到的三个系统上创建称为 httpd 的用户。在两个 Web 服务器系统上的 httpd 用户的主目录下创建一个.rhosts 文件。此文件只包含一行，如下：

```
ns.xcnews-lan.com
```

此文件必须由 root 超级用户拥有，每个其他用户只有读权限。这允许 ns.xcnews-lan.com 上 httpd 用户在每个 web 服务器系统上运行远程 shell 会话。注意，使用 ns.xcnews-lan.com 主机就是为了此目的，这是因为内部 LAN 需要使用 rdist 传输。下一步是为 rdist 生成一个 distfile 文件，它是一个文本文件，其中包含指示 rdist 如何执行文件分发任务的指令。列表 19-12 显示了一个 distfile 文件：rdist-distfile。

列表 19-12：rdist-distfile

```
#
# Distfile for rdist
#
# This is used to distribute files from ns.xcnews-lan.com
# to www[12].xcnews-lan.com systems
#
# $Author$ (kabir@nitec.com)
# $Version$
# $Date$
```

```
# $Id$

# List all the hosts that need to be updated.
# The list is created using user@hostname entries where each
# entry is separated by a white-space character.
#
HOSTS = (httpd@www1.xcnews-lan.com httpd@www2.xcnews-lan.com)

# List the directories that needs to be updated.
#
FILES = (/www)

# list the directories that need to be excluded from
# the update process.
EXCLUDE_DIR = (/www/cgi-data/ /www/apache /www/secured)

# Here are the commands:
# Install all directories listed in FILES for all hosts
# listed in HOSTS except for the directories that are
# listed in EXCLUDE_DIR
#
${FILES} -> ${HOSTS}
install;
except ${EXCLUDE_DIR} ;
```

超星阅览器提醒您：
使用本复制品
请尊重相关知识产权！

这是一个很简单的 distfile 文件。它定义了 HOSTS 变量，有两条目作为它的值：httpd@www1.xcnews-lan.com 和 httpd@www2.xcnews-lan.com。它告诉 rdist 在 www1.xcnews-lan.com 和 www2.xcnews-lan.con 两个主机上都使用 httpd 用户进行连接。下一个变量 FILES 定义了 rdist 分发的文件和目录。因为在 ns.xcnews.com 上的试用区是/www，所以它的值只有一项。

第三个变量是 EXCLUDE_DIR。它列出了分发时需要排除在外的所有文件和目录。在本例中，你看到的这个变量的值很重要。第一个目录，/www/cgi-data/，是所有 CGI 脚本用来写它们的数据的数据目录。这个目录将通过 NFS 输出给 Web 服务器主机。因此不需要通过 rdist 将它拷贝到每个 Web 服务器上。/www/apache 目录是 makesite 写 /www/apache/conf/httpd.conf 文件的地方。它不需要被拷贝，因为每个 Web 服务器在本地 /www/apache 目录里有自己专用的 Apache 配置。最后一个值是/www/secured，它用作安全服务器的文档根目录，不需被拷贝到 Web 服务器上。文件的剩余部分说明了一个简单的命令：

```
${FILES}->${HOSIS}
install;
except ${EXCLUDE_DIR};
```

这条命令取得 FILES 变量指出的所有文件的目录，并把它安装到由 HOSTS 变量指定

的主机上。它也告诉 rdist 将 EXCLUDE_DIR 变量指定的文件和目录排除在外。为了以用户 httpd 运行 rdist，在命令行键入下面的命令：

```
/usr/bin/rdist  -p  /usr/sbin/rdistd  -o  remove, quiet  -f
/usr/local/rdist/ rdist-distfile
```

-p 选项指定 rdist 需要的 rdistd 程序的路径。-o 选项指出后面跟着一个或多个选项，——在此处是 remove 和 quiet。remove 选项告诉 rdist 删除掉在目标系统的目标目录里发现的任何其他文件，这个选项为在每个 Web 服务器上维持与试用区一致的副本提供了容易的方法。quiet 选项告诉 rdist 在操作期间尽可能保持安静。最后一个选项-f，标明了 distfile 文件的位置。

超星阅览器提醒您
使用本复制品
请尊重知识产权！

为了减少在运行此命令时的人为错误，创建了称为 rdistribute.sh 的 sh 脚本。它如列表 19-3 所示。

列表 19-13：rdistribute.sh 脚本

```
#!/bin/sh
#
# This sccipt runs rdist to update Web servers via the
# non-routable lan xcnews-lan.com. The script is run
# by cron at a fixed interval.
#
# /etc/rc.d/rc.local starts the script to clean up
# left-over tempfiles that might have been left
# at shutdown. This process also removes the
# log file.
#
# $Author$ (kabir@nitec.com)
# $Version$
# $Id$
# $Date$
# $Status
##################################################################
# If the srcipt is called with an argument then
case "$1" in
boot )
# since the argument is 'boot' the script is being
# called at system start-up so remove all old lock
# files and logs.
echo -n "Cleaning up rdistribute.sh tmp files:"
   rm -f /tmp/rdist.lck
      rm -f /tmp/rdist.log
echo "complete."
exit 0;
;;

# since the argument is 'restart' the script
```

```
# needs to clean up as if the system just booted.
restart)
$0 boot
;;

esac
# if the lock file exists then don't do anything.
if [ -f /tmp/rdist.lck ]; then
exit 0
fi

# Otherwise create the lock file using /bin/touch
/bin/touch /tmp/rdist.lck

# Run rdist
/usr/bin/rdist -p /usr/sbin/rdistd -
oremove, nochkgroup, nochkmode, nochkowner, quiet -f
/usr/local/rdist/rdist_distfile

# Remove the lock file
rm -f /tmp/rdist.lck

# Write the time and date in the log file
echo 'date' >> /tmp/rdist.log

# Exit the script
exit 0
```

超星阅览器提醒您：
使用本系统产品
请尊重相关知识产权！

这个脚本很巧妙。它通过使用一个锁文件探测 rdistribute.sh 程序在运行过程中的进度，它能告诉什么时候前一个 rdistribute.sh 处理已经开始并且在继续中。在多服务器上有大量的文件需要更新时会看到这种情况。此脚本接受一个参数 boot，用于清除锁文件和在引导过程中生成的 log 文件。此脚本应当从/etc/rc.d/rc.local 中调用如下：

```
/usr/local/rdistribute.sh  boot
```

也可通过在/etc/crontab 中增加 cron 项定时运行此脚本。例如，为了每隔 10 分钟运行一次脚本，在/etc/crontab 中添加如下的 cron 项：

```
0,10,20,30,40,50 * * * *  httpd  /usr/local/rdistribute.sh
> /dev/null
```

cron 守护进程将以 httpd 用户每隔 10 分钟运行一次脚本。

19.6.2 在内部网络上使用 NFS

到此为止，已设置了 DNS 和文件分发，但 Web 服务器仍未被重新启动。在重新启动 Web 服务器前，CGI 数据区/www/cgi-data 目录必须要以 NFS 方式安装到 Web 主机上，这样才能使 Web 服务功能完全。为了可以在 Web 服务器上使用 CGI 数据目录

（/www/cgi-data/），要将域名服务器主机设置成 NFS 服务器，而将 Web 服务器设置成 NFS 客户。

19.6.2.1　设置 NFS 服务器

NFS 服务器必须运行一个程序：portmapper （也叫做 portmap 或 rpc.portmap）。它一般由 rc 脚本启动。为检查 portmapper 是否已经运行，使用下面的命令：

```
ps auxw | grep portmap
```

使用本复制品请尊重相关知识产权!

它显示出在 RedHat Linux 下，portmapper 由/etc/rc.d/rc3.d/S40portmap 脚本（即，/etc/rc.d/init.d/portmap.init 脚本）自动启动。因此对于 XC News 不需要手工启动它。

下一步是修改/etc/exports 文件，告诉系统什么文件系统或目录需要输出给 NFS 客户。因为 XC News 只需/www/cgi-data 目录输出给 Web 服务器，所以在 ns.xcnews-lan.com 主机上的输出（export）文件看起来如下：

```
/www/cgi-data www1.xcnews-lan.com(rw) www2.xcnews-lan.com(rw)
```

这一行告诉 NFS 服务器 www1.xcnows-lan.com 和 www2.xcnows-lan.com 可以读写/www/cgi-data 目录。

对于不同的 UNIX 版本，输出文件的语法也许不一样。

下一个需要运行的程序是 mountd（rpc.mountd）和 nfsd（rpc.nfsd）。这两个程序也由在/etc/rc.d/rc3.d 中的 rc 脚本自动启动。然而，无论何时改变/etc/exports 文件，都需要告诉这两个程序，由一个称为 exportfs 的脚本重新启动这两个程序，命令如下：

```
exportfs
```

如果系统上没有 exportfs，那么可用下面的脚本替代：

```
#!/bin/sh
killall -HUP /usr/sbin/rpc.mountd
killall -HUP /usr/sbin/rpc.nfsd
echo re-exported file systems
```

这个脚本使用大部分 Linux 如 RedHat 上有的 killall 程序。如果没有此程序，总可以运行一个 ps 命令，找到这些进程的 PID，然后对每个进程手工执行 kill -HUP 命令。现在，确保 mountd 和 nfsd 都正常地运行，运行一个称为 rpcinfo 的程序如下：

```
rpcinfo  -p
```

输出类似如下的结果：

```
program    vers    proto    port
100000      2      tcp      111      rpcbind
100000      2      udp      111      rpcbind
100005      1      udp      635      mountd
100005      2      udp      635      mountd
100005      1      tcp      635      mountd
100005      2      tcp      635      mountd
```