

## 第29章 cgi 脚本

现在差不多每个人的PC上都安装了Web服务器，在这样一本关于shell编程的书中似乎很有必要包含一章关于cgi脚本的内容。

本章包含以下内容：

- 基本cgi脚本。
- 使用服务器端内嵌(Server Side Includes,SSI)。
- get方法。
- post方法。
- 创建交互式脚本。
- 能够自动重载Web页面的cgi脚本。

运行Web服务器并不一定需要有网络环境，可以在本地主机上运行它。这里，我们假定你已经安装了Web服务器(apache、Cern等等)以及浏览器(Netscape、Internet Explorer等等)。另外，该服务器应当允许运行cgi脚本。一般来说缺省值是禁止运行cgi脚本的，要运行，只要将配置文件中相应的一行注释掉即可。后面我们会更详细地讨论这一问题。

如何安装并配置Web服务器已经超出了本书的讨论范围，不过我认为只需20分钟就可以安装并运行一个Web服务器。本章中的例子运行于apache Web服务器下，我所使用的浏览器为Netscape。

本章不打算深入探讨有关HTML或Web的细节问题，因为市面上已经有大量关于这方面的书籍。另外，如果要深入探讨HTML的话，还要花费数章的笔墨。

### 29.1 什么是Web页面？

Web页面或文档是包含有HTML标记的文件。当浏览器连接到一个Web页面上时，浏览器就会根据相应的HTML标记来显示该页面。Web页面中可以含有非常丰富的信息，它可以包含指向其他页面的链接、各种色彩、高亮标题、各种字体、直线、表格，还可以包含图像和声音。

Web页面可以分为两类：动态的页面和静态的页面。静态的页面是用于显示信息或下载文件。而动态的页面是交互型的，它们可以按照你所提供的信息产生相应的结果。动态页面还可以用于显示实时变化的信息，如股票价格，或用于完成某些监视任务。如果想要执行这种类型的处理，就需要编写脚本。

如果一个Web服务器能够交换信息脚本，那么它必须支持一种被称为公共网关接口的协议，即大家所熟悉的cgi。

### 29.2 cgi

cgi是一种规范，它规定了获取信息的脚本如何从服务器中取得信息或向服务器中写入信息。这种脚本或cgi脚本可以用任何语言来实现。最为流行的是Perl语言，不过你将会发现，

也可以用普通的 shell脚本来实现（见图 29-1）。

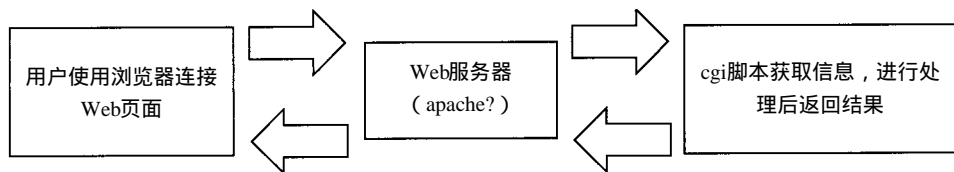


图29-1 浏览器和服务可以通过cgi来交换信息

## 29.3 连接Web服务器

可以使用统一资源定位符 (URL) 连接 Web 服务器。URL 包含两部分信息：

- 协议。
- 地址和数据。

其中，协议包括 http、ftp、mailto、file、telnet 和 news。这里我们只关心 http 协议 (超文本传输协议)。

地址一般是 DNS 域名或服务器主机名，也可以是 IP 地址。其他数据可以是你所要访问文件的实际路径名。

所有的连接都基于 TCP 协议之上，缺省的端口号为 80。

如果 Web 服务器在你的本地主机上，而相应的主页为 index.html，那么可以使用下面的 URL：

```
http://localhost/index.html
```

一般来说，index.html 是缺省下载的文件，即该页面是你的 Web 服务器的缺省页。这样，你可以只输入如下的 URL：

```
http://localhost/
```

## 29.4 cgi 和 HTM 脚本

当浏览器发出下载页面的请求时，Web 服务器将会对收到的 URL 进行分析。如果其中含有 cgi-bin，服务器将打开一个连接，通常是连接相应 cgi 脚本的管道。该 cgi 脚本所有的输入输出都将通过该管道。如果该 cgi 脚本用于显示 Web 页面，那么它的输出中必须要包含必要的 HTML 标记，这样该页面才能够按照服务器所能够理解的格式被显示出来，因此我们有必要了解一些 HTML 的知识。Web 服务器将该页面返回给发出请求的浏览器显示出来。表 29-1 列出了一些常用的 HTML 标记。

### 29.4.1 基本 cgi 脚本

所有的 cgi 脚本都应当位于 Web 服务器的 cgi-bin 目录中，不过在不同的服务器中该目录会有所不同。可以通过查看配置文件 srm.conf 中 ScriptAlias 一段来改变该目录的位置，并允许该服务器运行 cgi 脚本。所有的脚本文件名都应以 .cgi 做后缀。而其他 Web 页面都位于 html 或 htdocs 目录下，并且带有 .html 后缀。所有的脚本都应具有这样的权限。

```
chmod 755 script.cgi
```

所有 Web 页面连接的缺省用户身份为 nobody，不过可以通过配置 httpd.conf 文件来改变这

一设置。尽管我曾经说过本章并不是关于 Web 服务器配置的，不过正好可以借这个机会检查一下 passwd 文件，看看 nobody 用户的登录是否被禁止。如果想防止任何用户以 nobody 的身份从某一个物理端口上登录，只要在 /etc/passwd 文件中该用户口令域的第一个字符前面加入一个星号 \* 即可。

表29-1 基本HTML标记

<HTML></HTML>	HTML 文档的开头和结束
<HEAD></HEAD>	标题部分的开头和结束
<TITLE></TITLE>	主题的开头和结束
<BODY></BODY>	页面部分的开头和结束
<Hn></Hn>	不同层次的标题，1 代表最大的字体
<P></P>	段落的开头和结束
 	换行
<HR>	水平线
<PRE></PRE>	预定义格式文本的开头和结束，其中的所有跳格、行都保持原样。
<B></B>	黑体
<I></I>	斜体
<OL></OL>	有序号的列表
<A HREF=url>link</A>	超文本链接
<FORM></FORM>	表单
METHOD	post 或 get 方法
ACTION	地址
<INPUT...>	数据输入
NAME	变量名
SIZE	以字符计的文本框的宽度
TYPE	复选框、单选框、复位、提交
<SELECT...>	下拉式菜单
NAME	变量名
SIZE	要显示的列表的项数
<OPTION VALUE>	将用户选择的值返回给 NAME 变量
</SELECT>	结束列表框

如果脚本不能正常工作，应当首先查看错误日志，因为所有的错误都记录在这些日志中。如果使用 apache 作为 Web 服务器，那么相应的日志文件位于 /etc/httpd/logs 或 /usr/local/apache/logs 目录下，这取决于 Web 服务器的安装路径。cgi 脚本可以在命令行方式下运行测试，当然这时只能看到文本形式的输出，但是这种输出结果有助于调试该脚本。

现在我们就来创建一个 cgi 脚本。把下面的内容输入到一个名为 firstpage.cgi 的文件中，并保存在 cgi-bin 目录下。不要忘记该文件应当具有权限 755。

```
$ pg firstpage.cgi
#!/bin/sh
# firstpage.cgi
# display a page with text
echo "Content-type: text/html"
echo ""
echo "<HTML>"
echo "<H1><CENTER> THIS IS MY FIRST CGI PAGE</CENTER></H1>"
echo "<HR>"
echo "<H2><CENTER>STAND-BY TO STAND-TO!</CENTER></H2>"
echo "</HTML>"
```

如你所知，第一行表示 shell 解释器的路径。第一个 echo 命令行告诉服务器这是一个 MIME 题头；第二行 echo 命令行用于显示一个空行。如果在 MIME 题头后面没有一个空行，cgi 脚本的输出将无法正确显示。

接着，echo 命令输出了一个 <HTML> 标记，它告诉浏览器整篇文档应以 HTML 格式显示。HTML 文档可以显示从最大的 <H1> 到 <Hn> 的若干种不同字体。为了不至于费眼，<H6> 是通常可接受的最小字体。这里我们把文字居中，这样看起来更舒服一些。接下来，我们显示一条水平线。最后，我们用 <H2> 字体和 <CENTER> 标记居中显示了这样一行：“Stand-By To Stand-To”，该脚本的输出以 </HTML> 标记结束。

如果忘记了某一个结束标记，不要紧——你会很快发现这一点，因为在你试图浏览该页面时，一些标记将会在浏览器中显示出来。

现在如果想浏览该页面，可以在浏览器的 URL 框中输入这样一行：

```
http://your_server/cgi-bin/firstpage.cgi
```

输入时用实际的服务器名来替代上面一行中的 your\_server。

如果你的机器已经联网，而你得到“DNS 查找失败”的错误提示，那可能是由于浏览器在 Internet 上查找你刚刚编写的页面。不妨查看一下浏览器的连接选项；它可能设置为忽略本机代理，在此处键入主机名并重新运行浏览器即可。

图29-2显示了我们刚刚编写的页面。

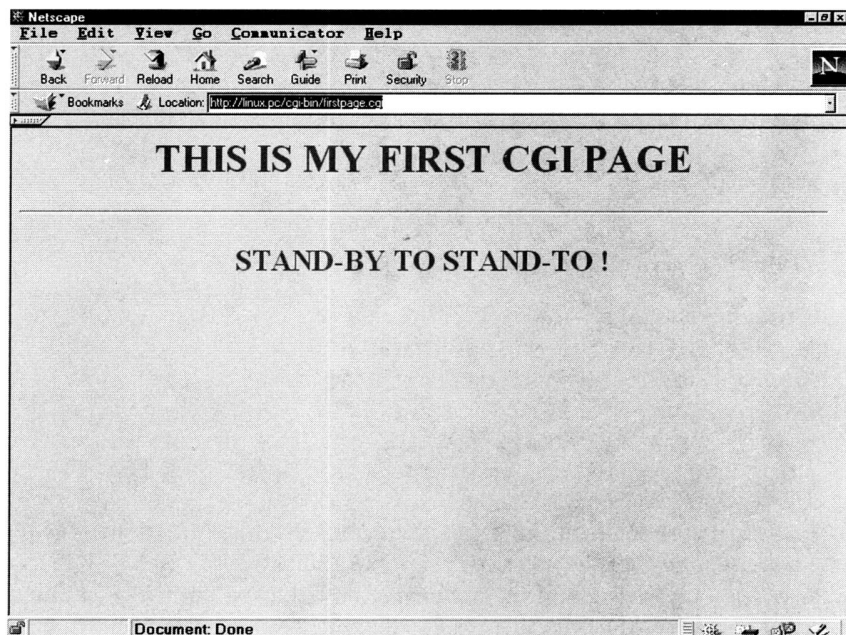


图29-2 脚本firstpage的输出

#### 29.4.2 显示shell命令输出

现在我们在脚本中加上一条 shell 命令，这样就可以在浏览器中显示该命令的输出。

我们将显示当前登录的用户数，这通过将 who 命令的输出经管道传递给 wc 命令就可以实现。还将显示当前的日期。

```
$ pg pagetwo.cgi
#!/bin/sh
# pagetwo.cgi
# display a page using the output from a unix command
MYDATE=`date +%A" "%d" "%B" "%Y`
USERS=`who |wc -l`
echo "Content-type: text/html"
echo ""
echo "<HTML>"
echo "<H1><CENTER> THIS IS MY SECOND CGI PAGE</CENTER></H1>"
echo "<HR>"
echo "<H2><CENTER>$MYDATE</CENTER></H2>"
echo " Total amount of users on to-day is :$USERS"
echo "<PRE>"
if [ "$USERS" -lt 10 ]; then
    echo " It must be early or it is dinner time"
    echo " because there ain't many users logged on"
fi
echo "</PRE>"
echo "</HTML>"
```

在这个脚本的开头，我们取得了当前的日期和连接数。日期居中显示。变量 `USERS` 也被显示出来。通过一个 `if` 语句来判断用户数是否少于 10，如果是，则显示这样的信息“现在还早或现在是晚餐时间”。

标记 `<PRE>` 保留其间的所有空格和跳格。如果希望显示系统命令的输出，如用 `df` 命令显示文件系统的情况，或只是使用简单的 `echo` 命令，那么应当使用 `<PRE>` 标记。在本例中我本来没有必要使用 `<PRE>` 标记，但我想我还是应该及早介绍它的这种应用，这样即使读者今后遇到这样的问题，也不会感到迷惑。要显示该页面，可以使用如下的 URL：

`http://your_server/cgi-bin/pagetwo.cgi`

输入时用实际的服务器名来替代上面一行中的 `your_server`。

图29-3显示了刚才编辑的页面。

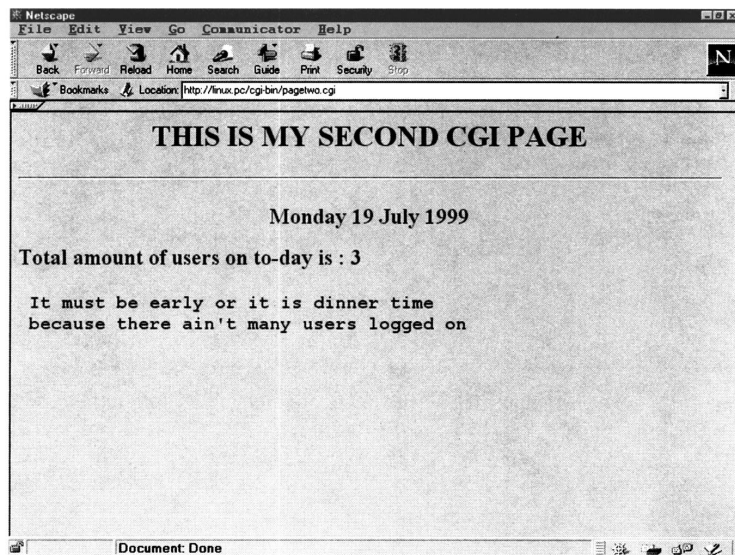


图29-3 脚本pagetwo.cgi的输出



### 29.4.3 使用SSI

使用cgi脚本产生一个Web页面并显示少量信息有点杀鸡用牛刀。例如我们用 cgi脚本产生了一个页面而仅仅是为了显示当前日期。如果我们能够把 cgi脚本嵌入到HTML文档中,让该脚本的输出显示在普通的页面中岂不更好?这样做是可行的,下面我们就将讨论这一内容。

为了内嵌cgi脚本,我们必须使能服务器端内嵌 (SSI)。这样,在显示一个页面时,它将会把SSI命令替换为相应命令或脚本的输出。一些含有服务器和命令信息的特殊环境变量也是全局可见的。

为了让服务器能够知道替换文档中的相应 SSI命令,需要使能 SSI,应当去掉配置文件中含有server-passed的一行开头的注释符,在apache上是这样的:

```
AddHandler server-passed.shtml
```

```
AddType text/html shtml
```

需要重新启动 Web服务器软件,使用 kill -1 命令使该服务器重新读入配置文件。含有 SSI 的页面应当具有后缀.shtml,而不是.html。

### 29.4.4 访问计数器

现在让我们来创建一个能够显示访问次数的页面。你肯定见过类似的页面:“你是第n个访问本站点的用户”。我们还将显示该页面的最新更改时间。

不要忘记将下面的脚本放置在 cgi-bin目录中,起名为hitcount.cgi。

```
$ pg hitcount.cgi
#!/bin/sh
# hitcount.cgi
# hit page counter for html <cgi>
# counter file must be chmod 666
counter=../cgi-bin/counter
echo "Content-Type: text/html"
echo ""
read access < $counter
access=`expr $access + 1`
echo $access
echo $access >$counter
```

如你所见的,该脚本读入 ../cgi-bin/counter文件中的值并将其赋给变量 access,将该变量加1,显示该变量,最后将新的值写回到 ../cgi-bin/counter文件中。

现在我们创建一个名为 counter的文件。我们只需在其中放置一个初始值,这里我们取 1。于是,操作步骤为:创建一个名为 counter的文件并写入1(不要加引号),然后保存并退出。

由于每个用户都需要使用该文件,文件属主、同组用户及其他用户都应对其具有读和写的权限。

```
$ chmod 666 counter
```

现在还需要在Web根目录下(该目录通常是放置所有其他HTML文档的地方,一般是htdocs或html目录)创建一个相应的.shtml文件。下面就是该文件,千万不要忘记带.shtml:

```
$ pg main.shtml
<!-- main.shtml>
<!-- this is a comment line>
```

```
<HTML>
<H4> Last modified: <!--#echo var="LAST_MODIFIED" -->
</H4>
<HR>
<H1><CENTER> THE MAY DAY OPERATIONS CENTER </H1>
<H2>Stand-by to Stand-to
<HR>
This page has been visited <!--#exec cgi="/cgi-bin/hitcount.cgi"--> times
</CENTER>
</H2>
<HR>
</HTML>
```

在使用SSI时LAST\_MODIFIED变量及其他变量是全局可见的。可以通过访问 apache的Web站点(www.apache.org)得到所有在使用SSI时全局可见的特殊变量的详细描述。

我们来看下列的SSI命令：

**This page has been visited <!--#exec cgi="/cgi-bin/hitcount.cgi"--> times**

它的一般形式为：

```
<!--#command argument="value"-->
```

在本例中，cgi脚本hitcount是这样运行的：

命令是exec。

参数为cgi。

其中的value是要运行的脚本。

我已经改变了配置文件，把这个页面作为主页，而不是 index.html。不过你仍然可以通过在URL中使用全路径来访问index.html页面。

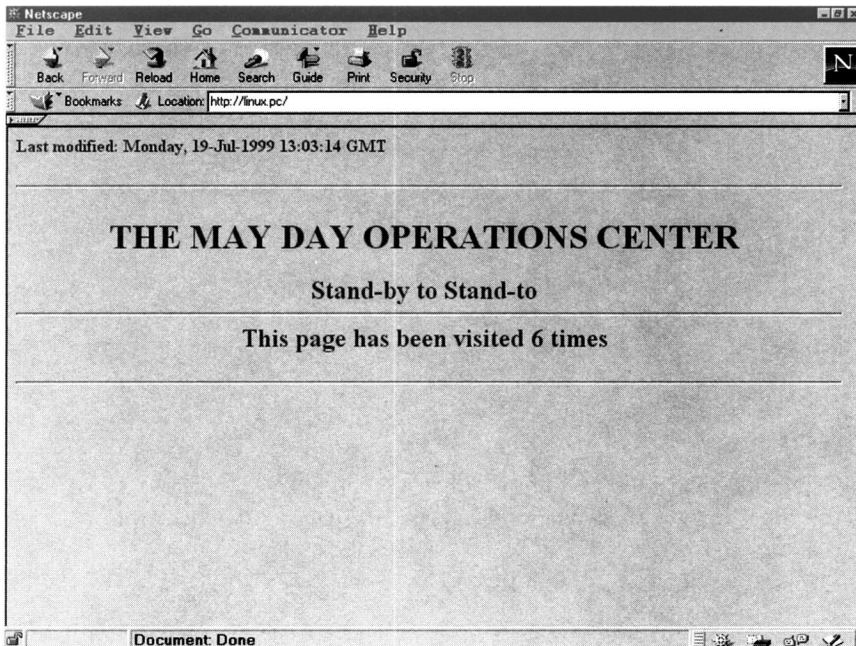


图29-4 含有点击计数器的页面

如果想改变缺省页，可以编辑 srm.conf 文件。该文件中含有这样一行：

```
DirectoryIndex
```

你会看到文件名 index.html 显示在该条目中。把它改成所希望的缺省页即可。不要忘记关闭并重新启动 Web 服务器以使该改动生效。

要访问该脚本，可以在浏览器的 URL 框中输入：

```
http://<server_name>/main.shtml
```

或

```
http://<server_name>
```

(如果该页面是缺省页的话)。

图 29-4 显示了刚才创建的页面；可以刷新该页面观察计数器的增加。注意，LAST\_MODIFIED 变量的值也被显示出来。

可以通过在 cron 中加入一行，每天夜里向 counter 文件中写入 1 来复位该计数器的值。

#### 29.4.5 使用一个链接来显示当前 Web 环境变量

在执行 cgi 脚本时，有若干环境变量可用。可以通过 env 或 set 命令看到绝大多数的变量。我们在 main.shtml 文档中创建一个指向相应 cgi 脚本的链接，该脚本能够显示这些变量。

下面是我们使用的链接：

```
<A HREF="/cgi-bin/printenv.cgi">Environment</A>
```

其中 A HREF 是链接标记的头。相应的地址包含在双引号中。单词 Environment 是要显示出来的单词，用户将点击这里。</A> 是链接标记的尾。

下面就是 main.shtml 文档：

```
$ pg main.shtml
```

```
<HTML>
<!-- this is a comment line>
<!-- main.shtml>
<H4> Last modified: <!--#echo var="LAST_MODIFIED" -->
</H4>
<HR>
<CENTER>
<H1> THE MAY DAY OPERATIONS CENTER </H1>
<H2> Stand-by to Stand-to
<HR>
This page has been visited <!--#exec cgi="/cgi-bin/hitcount.cgi"--> times
<HR>
To see your environment settings just click
  <A HREF="/cgi-bin/printenv.cgi" >here</A>
</CENTER>
</H2>
<HR>
</HTML>
```

下面是被调用的脚本 printenv.cgi，它使用 env 命令显示环境变量。在这里，我们使用 <PRE> 标记来保持该命令输出中的空格和跳格不变。

```
$ pg printenv.cgi
```



```
#!/bin/sh
# printenv.cgi
# print out the Web server env using the command env settings,
echo "Content-type: text/html"
echo ""
echo "<HTML><PRE>"
env
echo "</PRE></HTML>"
```

图29-5显示了该页面。

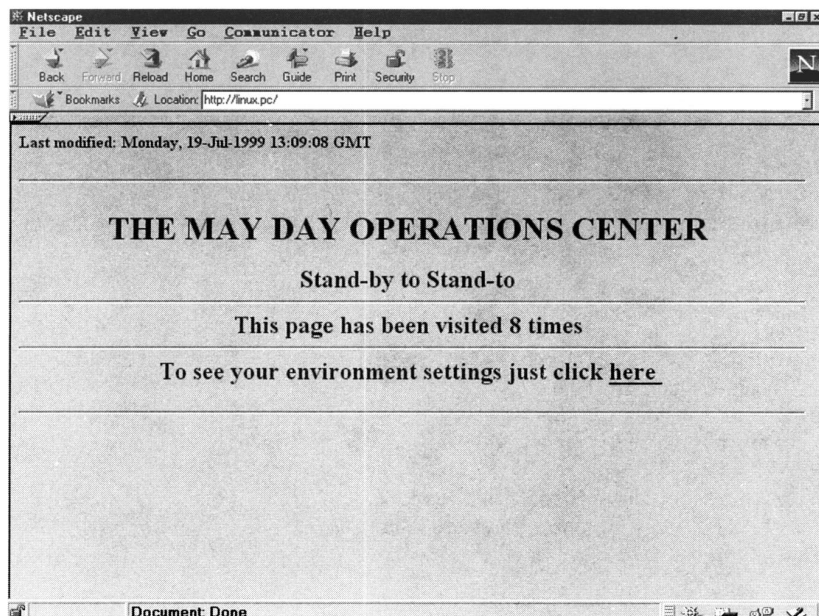


图29-5 含有显示环境变量链接的页面

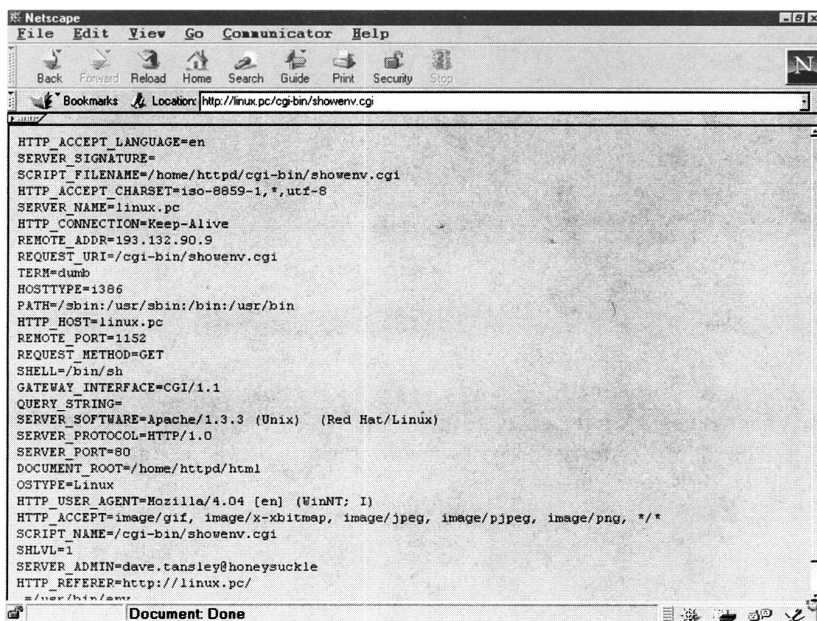


图29-6 显示当前环境变量的页面

当你点击该链接时，将会显示出当前的环境设置（见图29-6）。你在自己机器上所看到的可能与此有所不同。在运行不同的脚本时，一些相应的环境变量值将会随之改变。

#### 29.4.6 其他常用的环境变量

表29-2列出了最常用的cgi环境变量。其中有些变量可以用 `env` 或 `set` 命令显示出来。

表29-2 常用的cgi Web服务器变量

DOCUMENT_ROOT	Web服务器的主目录，是放置HTML文档的地方
GATEWAY_INTERFACE	cgi的版本
HTTP_ACCEPT	可接受的各种 MIME 类型
HTTP_CONNECTION	缺省的HTTP连接
HTTP_HOST	本地主机名
HTTP_USER_AGENT	客户端浏览器
REMOTE_HOST	远程主机
REMOTE_ADDR*	远程主机的IP地址
REQUEST_METHOD	传递信息的方法
SCRIPT_FILENAME	cgi脚本的绝对路径
SCRIPT_NAME	cgi脚本的相对路径
SERVER_ADMIN	Web服务器管理员的邮件地址
SERVER_NAME	服务器的主机名、DNS或IP地址
SERVER_PROTOCOL	连接所使用的协议
SERVER_SOFTWARE	Web服务器软件名
QUERY_STRING	get方法所传递的数据
CONTENT_TYPE	MIME类型
CONTENT_LENGTH	post方法所传递的字节数

\*严格来讲，这是你连接至 Internet 的网关地址。

为了显示这些变量，可以把它们放在一个 `cgi` 脚本中，这样，在需要使用某个变量值时随时都可以调用该脚本。

```
$ pg envcgi.cgi
#!/bin/sh
# envcgi.cgi
# print out the web server env
echo "Content-type: text/html"
echo ""
echo "<HTML><PRE>"
echo "CGI Test ENVIRONMENTS"
echo "SERVER_SOFTWARE = $SERVER_SOFTWARE"
echo "SERVER_NAME = $SERVER_NAME"
echo "GATEWAY_INTERFACE = $GATEWAY_INTERFACE"
echo "SERVER_PROTOCOL = $SERVER_PROTOCOL"
echo "SERVER_PORT = $SERVER_PORT"
echo "REQUEST_METHOD = $REQUEST_METHOD"
echo "HTTP_ACCEPT = $HTTP_ACCEPT"
echo "PATH_INFO = $PATH_INFO"
echo "PATH_TRANSLATED = $PATH_TRANSLATED"
echo "QUERY_STRING = $QUERY_STRING"
echo "SCRIPT_NAME = $SCRIPT_NAME"
echo "REMOTE_HOST = $REMOTE_HOST"
echo "REMOTE_ADDR = $REMOTE_ADDR"
echo "REMOTE_USER = $REMOTE_USER"
echo "AUTH_TYPE = $AUTH_TYPE"
```

```
echo "CONTENT_TYPE = $CONTENT_TYPE"
echo "CONTENT_LENGTH = $CONTENT_LENGTH"
echo "</PRE></HTML>"
```

## 29.5 get和post方法简介

到现在为止，我们只是向屏幕上输出。要想从用户那里得到信息，我们需要使用表单，这也是cgi为什么如此流行的原因。你需要有获得用户输入的能力。有了表单就可以显示文本框、下拉式列表框和单选框。

在用户通过键入或选择向表单输入了一些信息之后，他可以点击发送按钮将这些信息发送给某个脚本，在这里是cgi脚本。我们需要使用get或post方法来收集这样的信息。

### 29.5.1 get方法

任何表单的缺省操作都是get方法。get方法是从静态HTML页面获取文件的方法。

当用户点击“提交”按钮时，用户选择的信息将以编码字符串的形式附加在服务器 URL 的后面。服务器环境变量 QUERY\_STRING保存了编码字符串。变量 REQUEST\_METHOD保存了该表单所使用的方法。

#### 1. 创建一个简单的表单

现在让我们来创建一个简单的表单，在 main.shtml文档中创建一个指向booka.cgi脚本的链接。

在main.shtml文档中上一次创建的链接后面增加这样一行：

```
<BR>
Basic form using GET method <A HREF="/cgi-bin/booka.cgi" >Form1</A>
```

现在就按照下面的内容创建booka.cgi文件，不要忘记把它放在cgi-bin目录中。

```
$ pg booka.cgi
#!/bin/sh
# booka.cgi
echo "Content-type: text/html"
echo ""
echo "<HTML>"
echo "<BODY>"
# call booka_result.cgi, when user hits sent!
echo "<FORM action=\"/cgi-bin/booka_result.cgi\" METHOD=GET>"

echo "<H4> CGI FORM</H4>"
# text box, input assigned to variable name 'contact'
echo "Your Name: <INPUT NAME=contact SIZE=30><BR><BR>"
# drop down menu selection assigned to variable name 'film'
echo "<SELECT NAME=film>"
echo "<OPTION>-- Pick a Film --"
echo "<OPTION>A Few Good Men"
echo "<OPTION>Die Hard"
echo "<OPTION>Red October"
echo "<OPTION>The Sound Of Music"
echo "<OPTION>Boys In Company C"
echo "<OPTION>Star Wars"
echo "<OPTION>Star Trek"
```

```

echo "</SELECT>"
# drop down menu selection assigned to variable name 'actor'
echo "<SELECT NAME=actor>"
echo "<OPTION>-- Pick Your Favourite Actor --"
echo "<OPTION>Bruce Willis"
echo "<OPTION>Basil Rathbone"
echo "<OPTION>Demi Moore"
echo "<OPTION>Lauren Bacall"
echo "<OPTION>Sean Connery"
echo "</SELECT>"
echo "<BR><BR>"
# check box variable names are 'view_cine' and 'view_vid'
echo "Do you watch films at the..<BR>"
echo "<INPUT TYPE='Checkbox' NAME=view_cine> Cinema"
echo "<INPUT TYPE='Checkbox' NAME=view_vid> On Video"
echo "<BR><BR>"
# input assigned to variable name 'textarea'
echo "Tell what is your best film, or just enter some comments<BR>"
echo "<TEXTAREA COLS='30' ROWS='4' NAME='textarea'></TEXTAREA>"

echo "<BR><INPUT TYPE=Submit VALUE='Send it'>"
echo "<INPUT TYPE='reset' VALUE='Clear Form'>"

echo "</FORM>"
echo "</BODY>"
echo "</HTML>"

```

我们在上述表单的操作项中设定：一旦用户按下“发送”按钮，脚本 booka\_result.cgi 将被执行。我们现在讨论 get 方法。

上面的表单将会显示两个文本框、两个下拉式列表框和一个复选框。

输入姓名的文本框宽度为 30 个字符，相应的输入被赋给变量 contact。

第一个下拉式列表框让用户选择最喜爱的电影，选择项被赋给变量 film。

第二个下拉式列表框让用户选择最喜爱的演员，选择项被赋给变量 actor。

用户可以通过点击选择某一个复选框或两个都选。相应的逻辑值分别保存在变量 view\_cine 和变量 view\_vid 中。如果用户选择某一个复选框，那么相应的变量的值为 on。

区域型文本框允许用户输入超过一行的文本，而不是像标准文本框那样只能输入一行（在本例中是 4 行，每行宽 30 个字符），所有的输入都将赋给变量 textarea。

发送数据时需要使用 submit 作为输入类型。用户可以点击 clear 按钮清除当前的表单。

创建如下的 cgi 脚本，命名为 booka\_result.cgi 并将其保存在 cgi-bin 目录下。

```

$ pg booka_result.cgi
#!/bin/sh
# booka_result.cgi
# print out the web server env for a get
echo "Content-type: text/html"
echo ""
echo "<HTML><PRE>"
echo "<PRE>"
echo " Results from a GET form"
echo "REQUEST_METHOD : $REQUEST_METHOD"
echo "QUERY_STRING   : $QUERY_STRING"
echo "</PRE></HTML>"

```

上面的脚本显示了变量 QUERY\_STRING 和 REQUEST\_METHOD。变量 QUERY\_STRING 保



现在我们在该表单中输入一些信息并发送(见图29-8)。点击“发送”按钮就会看到图29-9所示的页面。由于字符串太长，变量QUERY\_STRING只能显示出一部分。下面是该字符串的全部。

contact=David+Tansley&film=The+Sound+Of+Music&actor=Bruce+Willis&view\_cine=on&view\_vid=on&textarea=%21%22%A3%A3%24%25%24%25%5E\*%5E%26\*%28%29\*%28%29%28\*%0D%0AHow%27s+that+%21%21





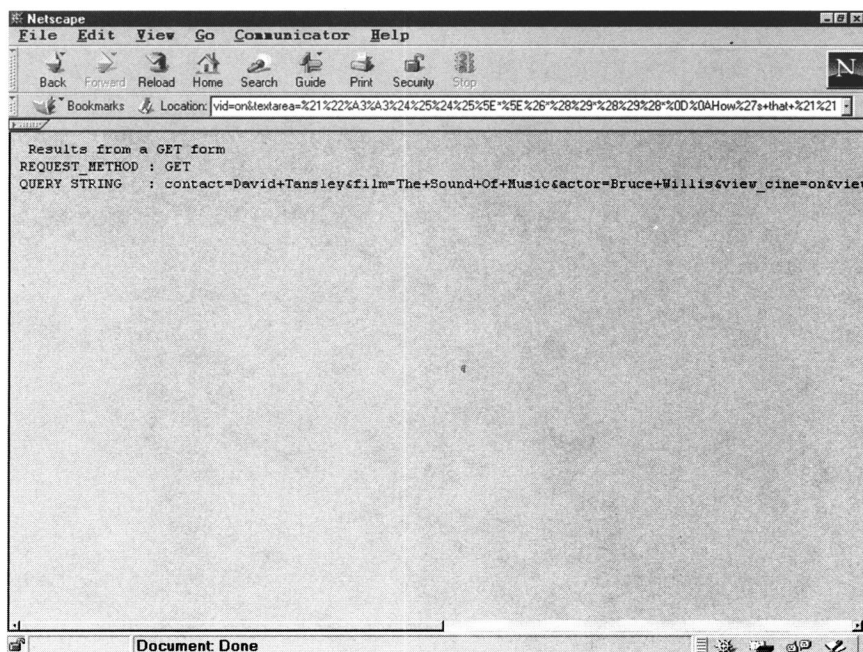


图29-9 该表单所发送的信息被回显出来

从我们所发送的信息可以知道，QUERY\_STRING变量中应包含下列信息：

变 量	值
contact	David Tansley
film	The Sound Of Music
actor	Bruce Willis
view_cine	选中，则该值为 ‘ on ’
view_vid	选中，则该值为 ‘ on ’
textarea	!"比%\$%^&*(*)(* How's that!!

## 2. 对编码字符串解码

当用户点击提交按钮时，相应的信息被赋给了变量 `QUERY_STRING`，这些信息是以下面的格式编码的：

所有的空格用+来替代。

所有的值域用&隔开。

所有的值和相应域用=隔开。

所有的符号和一些特殊字符用% xy的形式表示，其中xy是该字符的16进制ASCII码。看一下QUERY\_STRING变量就知道，在textarea变量中含有很多这样的字符。

cgi协议声明，所有采用% xy形式表示的特殊字符（其中xy为16进制数）都被转换为对应的ASCII字符。这种16进制字符包括特殊字符 &、%、+、=、（、）及所有ASCII码超过127的其他特殊字符。例如字符“（”应为%29。

这种16进制字符产生于文本框，用户可能会在这些地方输入这样的字符。不过，它们也可以出现在下拉式列表框中。

为了解码相应字符串，我们应当：

将所有的&替换为换行。

将所有的+替换为空格。

将所有的=替换为空格。

将所有的%xy替换对应的ASCII字符。

在完成上述转换之后，我们可能需要访问某个变量，这样就可以根据用户发送的信息来进行某些处理。解码只是所有工作的一部分，尽管这是最繁重的一部分。如果想访问这些变量，可以使用eval命令。

下面的脚本将会完成所有必要的转换并能够访问每一个变量。该脚本的注释丰富，应该能够理解。

```
$ pg conv.cgi
#!/bin/sh
# conv.cgi
# decode URL string
echo "Content-type: text/html"
echo ""
echo "<HTML><PRE>"
# display the method and the coded string
echo "Method      : $REQUEST_METHOD"
echo "Query String : $QUERY_STRING"
echo "<HR>"
# use sed to replace all & with tabs
LINE=`echo $QUERY_STRING | sed 's/&/ /g'`

for LOOP in $LINE
do
    # split the fields up into NAME and TYPE
    NAME=`echo $LOOP | sed 's=/ /g' | awk '{print $1}'`
    # get the TYPE now replace all = with spaces and %hex_num with \xhex_num
    # replace all + with spaces
    TYPE=`echo $LOOP | sed 's=/ /g' | awk '{print $2}' | \
sed -e 's/%(\|)\|\\x/g' | sed 's+/ /g'`
    # use printf it does all the hex conv for you, display the variables
    printf "${NAME}=${TYPE}\n"
    # now assign the fields across to VAR ready to eval them, so
    # we can then address individual fields, double backslash needed
    # in case the fields have spaces in them
    VARS=`printf "${NAME}=\${TYPE}\n"`
    eval `printf $VARS`
done
echo "<HR>"
# keep using printf to preserve special chars...if any
printf "Your name is          : $contact\n"
printf "Your choice of film is:   : $film\n"
printf "Your choice of actor is    : $actor\n"
printf "You watch films at the cinema : $view_cine\n"
printf "You watch films on video    : $view_vid\n"
printf "And here are your comments : $textarea\n"
echo "</PRE>"
echo "</HTML>"
```

你可能已经注意到，这里使用 printf 命令来回显所有的变量——这是因为使用它更简单。

printf命令和echo命令的功能相似，但它可以完成所有 16进制字符的转换。需要注意的是，在使用printf命令时，它不会自动换行，必须要在每一行的末尾使用 \n来换行。QUERY\_STRING变量中的 16进制字符是以%xy的形式来表示的，我们只要使用 sed把它们转换成\xnn的格式(其中nn为16进制数)，printf命令就可以自动完成相应的转换。有简单的办法我们何必要使用复杂的呢？

把上面的脚本存为conv.cgi，并放在cgi-bin目录下。现在我们只要对脚本booka.cgi做小小的改动，就能够在表单提交时执行脚本 conv.cgi而不是booka\_result.cgi。下面是做改动的一行：

```
<FORM action="/cgi-bin/conv.cgi" METHOD=GET>
```

现在如果我们重新提交该表单，它将执行脚本 conv.cgi，我们将会得到如图 29-10所示的结果。

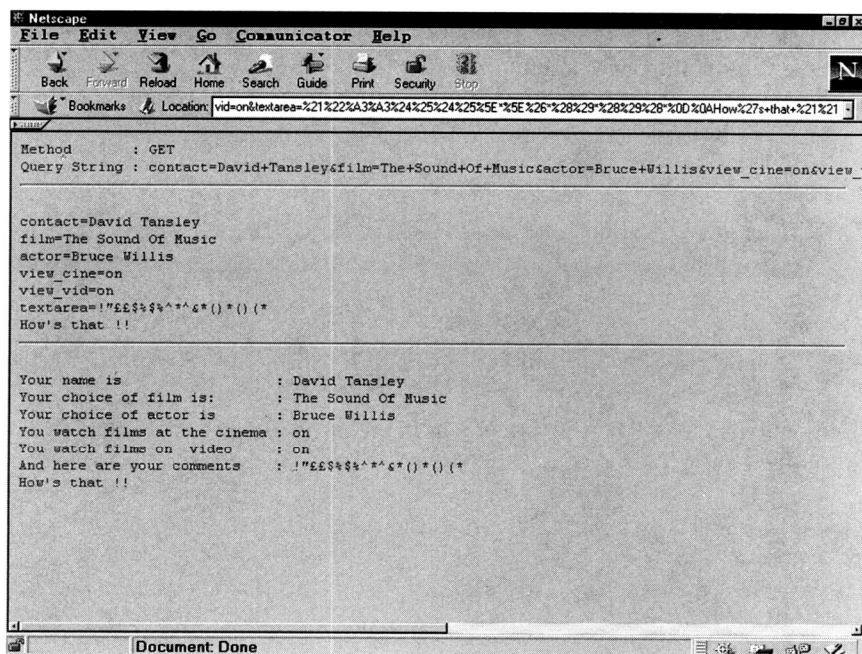


图29-10 完全解码的表单输入数据

现在我们把该字符串转换成为一种更可读的形式，这样我们可以对其做进一步的处理。

在使用表单时，get方法是缺省的方法。根据所处的环境，使用get方法有两个潜在的问题。在发送信息时，整个编码字符串都附加在服务器 URL的后面，这样所发送的信息可以从 URL框中看到。可能你会认为这没什么大不了的，如果你是通过网络发送公司或个人的信息，这就是一个值得注意的问题了。

如果表单具有很多的输入域，那么 QUERY\_STRING变量将会变得很长。大多数人在使用cgi时都用post作为表单输入的方法。在下一小节中，我们将对 post方法进行介绍。

### 29.5.2 post方法

post方法的字符串编码方式与get方法相同。它们所不同的是获取数据的方法，post方法是

从标准输入读入的。如果想用 post 方法来发送数据，只要把表单操作语句中的 get 替换为 post 即可。

```
<FORM action="/cgi-bin/conv.cgi" METHOD=POST>
```

变量 CONTENT\_LENGTH 中含有使用 post 方法发送的总字节数。我们从标准输入读入该字符串，然后进行与 get 方法一样的转换。在输入的字节数等于变量 CONTENT\_LENGTH 的值时，将会停止读入。

只需要改动表单处理脚本中的相应语句就可以产生一个通用的解码脚本。为了从标准输入中读入，可以使用 cat 命令。下面是需要在 conv.cgi 脚本中增加的语句，这样它就既可以适用于 get 方法，也可以适用于 post 方法。

```
if [ "$REQUEST_METHOD" = "POST" ]; then
    QUERY_STRING=`cat -`
fi
```

注意 cat 命令后面的横杠 -，它允许 cat 命令从标准输入中读入。

我们只测试 QUERY\_STRING 变量——如果使用 post 方法，那么就用 cat 命令把所有来自于标准输入的字符串赋给该变量。如果使用 get 方法，就无须做这样的操作，只需从该变量中读取信息即可。

将 cgi 脚本 booka.cgi 中的表单操作一行由

```
<FORM action="/cgi-bin/conv.cgi" METHOD=GET>
```

改为：

```
<FORM action="/cgi-bin/conv.cgi" METHOD=POST>
```

我们还将 conv.cgi 脚本中做一些其他修改，这样就可以测试文本框和复选框中输入的值。

下面是修改后的脚本。

```
$ pg conv.cgi
#!/bin/sh
# conv.cgi
# decode URL string
echo "Content-type: text/html"
echo ""
echo "<HTML><PRE>"
# is it post ???
if [ "$REQUEST_METHOD" = "POST" ]; then
    QUERY_STRING=`cat -`
fi

# display the method and the coded string
echo "Method: $REQUEST_METHOD"
echo "Query String : $QUERY_STRING"
echo "<HR>"
# use sed to replace & with tab
LINE=`echo $QUERY_STRING | sed 's/&/ /g'`

for LOOP in $LINE
do
    NAME=`echo $LOOP | sed 's=/ /g' | awk '{print $1}'`
    TYPE=`echo $LOOP | sed 's=/ /g' | awk '{print $2}' | \
sed -e 's/%(\)/\\x/g' | sed 's+/ /g'`
```

```

# use printf it does all the hex conv for you
printf "${NAME}=${TYPE}\n"
VARS=`printf "${NAME}=\${TYPE}\n"`
eval `printf $VARS`
done
echo "<HR>"
if [ "$contact" != "" ]; then
    printf "Hello $contact, it's great to meet you\n"
else
    printf "You did not give me your name ... no comment!\n"
fi

if [ "$film" != "-- Pick a Film --" ]; then
    printf "Hey I agree, $film is great film\n"
else
    printf "You didn't pick a film\n"
fi

if [ "$actor" != "-- Pick Your Favourite Actor --" ]; then
    printf "So you like the actor $actor, good call\n"
else
    printf "You didn't pick a actor from the menu\n"
fi

if [ "$view_cine" = "on" ]; then
    printf "Yes, I agree the cinema is still the best place to watch a
        film\n"
else
    printf "So you don't go to the cinema, do you know what you're missing\n"
fi

if [ "$view_vid" = "on" ]; then
    printf "I like watching videos at home as well\n"
else
    printf "No video!!, you're missing out on all the classics to rent or
        buy\n"
fi

if [ "$textarea" != "" ]; then
    printf " And here are your comments.....OK    $textarea\n"
else
    printf "No comments entered, so no comment !\n"
fi
echo "</PRE>"
echo "</HTML>"

```

注意，我使用了很多 printf 命令，在不需要置换字符串变量时，本可以使用 echo 命令，但为了统一起见，我一律都使用了 printf 命令。

浏览上述表单并使用 post 方法来发送数据：

`http://<server_name>/cgi-bin/booka.cgi`

图29-11显示了我所输入的信息。

在输入一些信息之后，点击“发送”按钮，结果如图 29-12所示。

该脚本检查各个变量，看是否有相应的信息输入。还可以做进一步的改进，检查所有变量的值，如果用户没有在某输入域输入任何信息，那么就重新向用户显示该表单，要求用户重新输入。如果用户提供了填写正确的表单，可以把它们附加在一个文件的末尾，创建一



个微型的数据表。

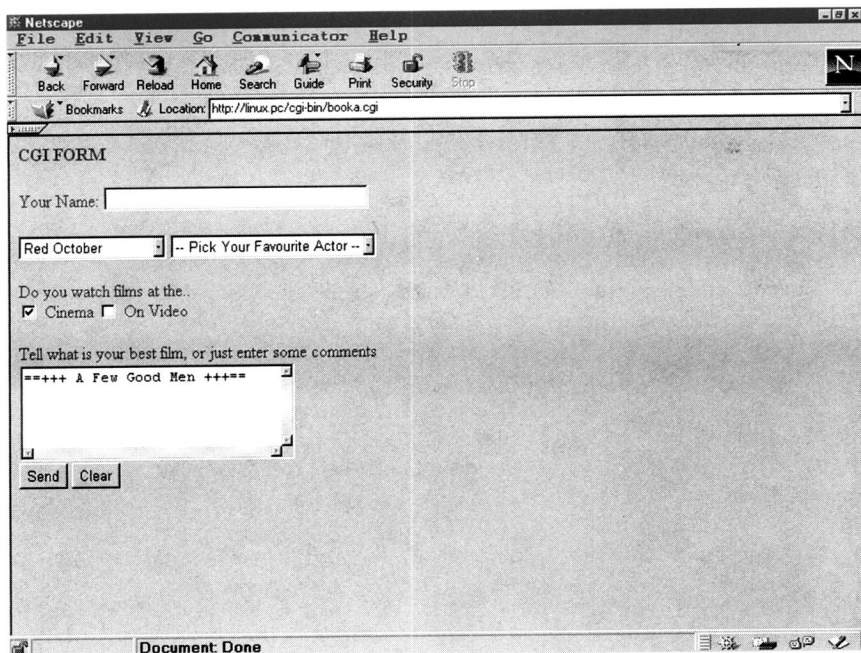


图29-11 使用post方法的cgi表单

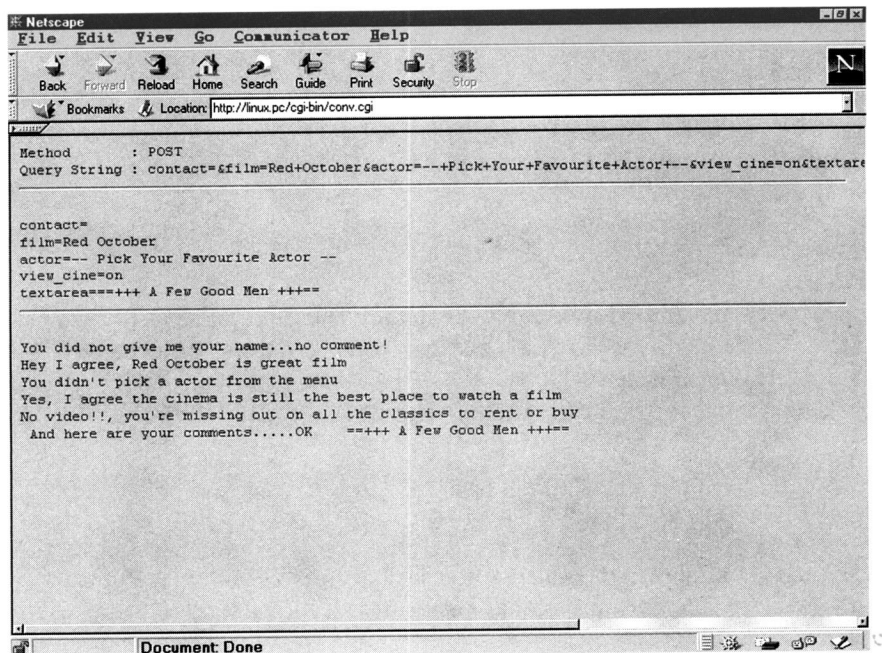


图29-12 使用post方法所获得的数据被完全解码

### 3. 一个更为实用的cgi脚本

我们现在创建一个更为实用的脚本。这里我们虚构一个名为 Wonder Gifts的公司，按照用

户的选择显示该公司的报表。

一个会计文件包含了该公司 1998年每一个季度每一个部门的销售额。该文件中包含下列部门：文具、图书、礼品。

我们的任务是按照用户的选择生成一个报表。用户可以按照季度或部门进行选择。收到用户选择后的处理过程就是，将所选择的部门和季度中所有月份的销售额加在一起。输出可以是屏幕、打印机或两者皆有。

我们的表单有两个下拉式列表框，一个用来选择部门，一个用来选择季度。有一个单选框用于选择输出形式。使用单选框时，用户只能选择其中之一。实际上报表只输出到屏幕上，这里的单选框只是一个演示。

下面就是原始的数据文件，它包含以下的域：

部门：年：季度：该季度1至3月份的销售额

```
$ pg qtr_1998.txt
STAT      1998      1st      7998      4000      2344      2344
BOOKS     1998      1st      3590      1589      2435      989
GIFTS     1998      1st      2332      1489      2344      846
STAT      1998      2nd      8790      4399      4345      679
BOOKS     1998      2nd      889       430      2452      785
GIFTS     1998      2nd      9822      4822      3555      578
STAT      1998      3rd      8911      4589      2344      8690
BOOKS     1998      3rd      333       1489      6322      889
GIFTS     1998      3rd      2310      1483      3443      778
STAT      1998      4th      9883      5199      2344      6456
BOOKS     1998      4th      7333      3892      5223      887
GIFTS     1998      4th      8323      4193      2342      980
```

下面是表单脚本。

```
$ pg gifts.cgi
#!/bin/sh
# gifts.cgi .... using POST
echo "Content-type: text/html"
echo ""
echo "<HTML>"
echo "<BODY>"
# gifts_result.cgi is going to process the output from this form
echo "<FORM action=\"/cgi-bin/gifts_result.cgi\" METHOD=POST>"
echo "<P>"
echo "<HR>"
echo "<H1><CENTER>GIFTS Inc <BR>"
echo "QUARTERLY REPORT</H1></CENTER>"
echo "</P><HR>"
echo "Department: <SELECT NAME=dept>"
echo "<OPTION>GIFTS"
echo "<OPTION>STATIONERY"
echo "<OPTION>BOOKS"
echo "</SELECT>"
echo "Quarter End:<SELECT NAME=qtr>"
echo "<OPTION>1st"
echo "<OPTION>2nd"
echo "<OPTION>3rd"
echo "<OPTION>4th"
echo "</SELECT>"
```

```

echo "<BR><BR>"
echo "Report To Go To:<BR>"
echo "<INPUT TYPE='radio' NAME= stdout VALUE=Printer >Printer"
echo "<INPUT TYPE='radio' NAME= stdout VALUE=Screen CHECKED>Screen"
echo "<INPUT TYPE='radio' NAME= stdout VALUE=Both >Both"
echo "<BR><BR><HR>"
echo "<INPUT TYPE=Submit VALUE='Send it'>"
echo "<INPUT TYPE=Reset VALUE='Clear'>"

```

```

echo "</FORM>"
echo "</BODY>"
echo "</HTML>"

```

其中，变量 dept 将保存用户所选择的部门；而变量 qtr 用于保存用户所选择的季度。变量 stdout 可能出现的值有 printer、screen、both，缺省为屏幕（缺省值由关键字 CHECKED 指定）。下面的脚本用于处理所接收到的信息。

```

$ pg gifts_result.cgi
#!/bin/sh
# gifts_result.cgi
# decode URL string
echo "Content-type: text/html"
echo ""
echo "<HTML><PRE>"
# is it post ???
if [ "$REQUEST_METHOD" = "POST" ]; then
    QUERY_STRING=`cat -`
fi
# decode it
# use sed to replace & with tab
LINE=`echo $QUERY_STRING | sed 's/&/ /g'`
for LOOP in $LINE
do
    NAME=`echo $LOOP | sed 's=/ /g' | awk '{print $1}'`
    TYPE=`echo $LOOP | sed 's=/ /g' | awk '{print $2}' | \
sed -e 's/%(\)/\\x/g' | sed 's+/ /g'`
    # use printf it does all the hex conv for you
    VARS=`printf "${NAME}=\\${TYPE}\\n"`
    eval `printf $VARS`
done
echo "<HR>"
echo "<H1><CENTER> GIFTS Inc</CENTER></H1>"
echo "<H2><CENTER> Quarter End Results </CENTER></H2>"
echo "<HR>"
# we need to change the fields name from STATIONERY to STAT to
# search properly
if [ "$dept" = "STATIONERY" ];then
    dept=STAT
fi

# Read in the file qtr_1995.txt
TOTAL=0
while read DEPT YEAR Q P1 P2 P3 P4
do

```

```

if [ "$DEPT" = "$dept" -a "$Q" = "$qtr" ]; then
    TOTAL='expr $P1 + $P2 + $P3 + $P4'
fi
continue
done </home/httpd/cgi-bin/qtr_1995.txt
echo "<H2>"
echo " TOTAL ITEMS SOLD IN THE $dept DEPARTMENT"
echo " IS $TOTAL IN THE $qtr QUARTER"
echo "</H2><HR>"
# where is the report going..
if [ "$stdout" = "Both" ]; then
    echo "This report is going to the printer and the screen"
else
    echo " This report is going to the $stdout"
fi
echo "</PRE>"
echo "</HTML>"

```

该脚本的第一部分和其他使用 post 方法的表单处理脚本没有什么区别。由于这里没有 16 进制字符需要转换 (因为输入域都是预定义的下拉式列表框)，就不需要使用 printf 命令，不过你完全可以使用该命令。这里有意思的部分就是从 qtr\_1998.txt 文件中读入信息的一段。

在 while 循环中分别将原始文件中的几个域赋给变量 DEPT、YEAR、Q、P1、P2、P3、P4。然后比较 \$dept (这个值是由用户给出) 和变量 DEPT 的值；如果匹配，再比较 \$qtr (这个值由用户给出) 和变量 Q 的值，如果也匹配，那么就将这一行中的所有数据相加。

现在我们已经有了表单脚本和处理表单所发送的信息的脚本，现在来试着运行它。在浏览器的 URL 框中输入如下的 URL (或在主页中加入另外一个链接)：

[http://<server\\_name>/cgi-bin/gifts.cgi](http://<server_name>/cgi-bin/gifts.cgi)

结果如图 29-13 所示。

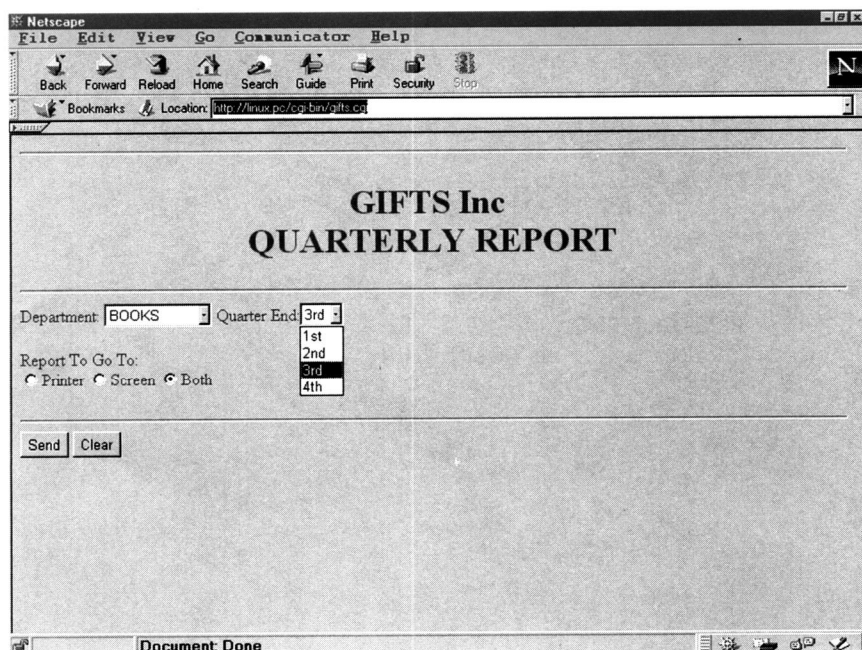


图29-13 选择季度信息供进一步处理



对用户输入信息处理后返回的页面如图 29-14所示。

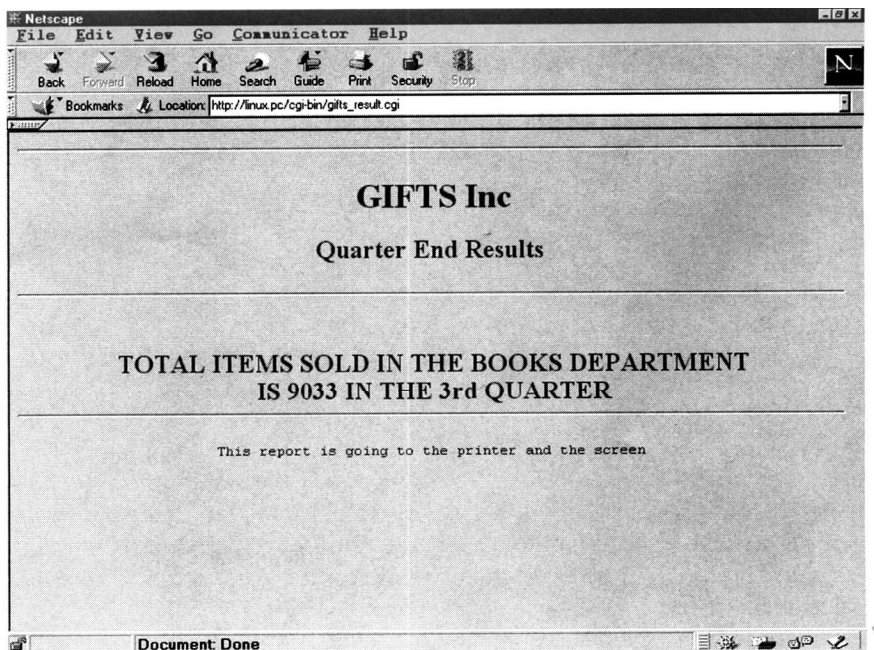


图29-14 处理结果

### 29.5.3 填充列表项

要想使页面成为真正的动态页面，可能需要动态地使用某个文件中的信息来填充列表项，而不是把它们编写进cgi脚本。

在下面的脚本中，下拉式列表框中的选项是由 list文件中的信息填充的，该文件位于 Web 服务器根目录下的temp目录中。在这个脚本中，使用了一个 while循环来读入文件中的内容(一次读入一行)，并将读入的信息填充到列表框选项中。相应的填充语句为：

```
echo "<OPTION>$LINE"
```

被用户选择的项赋给变量 menu\_selection。

下面就是填充下拉式列表框的脚本，这里没有指定表单的操作。

```
$ pg populat.cgi
#!/bin/sh
# populat.cgi
# populate a pull down list from a text file
echo "Content-type: text/html"
echo ""
echo "<HTML>"
echo "<BODY>"
echo "<H4> CGI FORM...populat.cgi..populate pull-down list from a text
file</H4>"
echo "<SELECT NAME=menu_selection>"
echo "<OPTION>-- PICK AN OPTION --"
# read in the file into the list to populate the options
while read LINE
do
```



```

    echo "<OPTION>$LINE"
done < ../temp/list
echo "</SELECT>"
echo "</FORM>"
echo "</BODY>"
echo "</HTML>"

```

#### 29.5.4 自动刷新页面

在使用cgi脚本实现监视或看门狗功能时，如果能够让页面不断自动刷新就方便多了。想要实现这样的功能，需要不断调用自己的脚本或页面。下面的命令将每隔 60秒刷新一次dfspace.cgi脚本。

```

<meta http-equiv="Refresh" content=60;URL=http://linux.pc/cgi-
bin/dfspace.cgi">

```

这里的关键字是Refresh。这样Web服务器就知道应该重载该页面。content=60则表示每次刷新的间隔秒数。只要在URL中包含相应的脚本名就能够不断刷新该页面。

我有好几个监视脚本在实时运行，它们不断轮询网络中的所有的主机，这样我一眼就能够看出每个机器是处在运行状态还是关闭状态。为了更美观，我使用了一个红色的小球和一个绿色的小球来分别代替文字on和off。

下面的脚本在一张表中显示出df命令输出中的两列：磁盘占用情况和文件系统名。

下面的一段只显示了表头。在使用表格显示系统命令输出结果时，需要经过反复调整才能达到令人满意的效果。

```

echo "<TABLE align="center" cellspacing="20" border=9 width="40%"
    cols="2">"
echo "<TH align="center">- Capacity % -</TH>"
echo "<TH align="center">- File System -</TH>"

```

cellspacing设置了表格内框和外框的间距。border则用于控制表格的边框宽度。cols决定了表中所显示的列数。

下面是该脚本的实质部分：

```

df |sed 1d| awk '{print $5"\t"$6}' | while read percent mount
do
    echo "<TR><TD align="center"><B>$percent</B></TD><TD align="center">
        $mount</TD>
    </TR>"
done

```

在上面一段脚本中，使用df命令显示文件系统的情况，并将结果通过管道传递给sed命令，删除其中的题头，然后再通过管道将结果传递给awk命令，取其中的第5、6列，将结果分别赋给变量percent和mount。

TR代表表格行，TD代表表格数据。这是表格中存放信息的地方。

的确，对于监视文件系统来说，60秒刷新一次有些太夸张了，但是如果你正在文件系统间进行大文件的迁移，这个脚本就能够使你掌握每一分钟的最新情况！

下面就是该脚本。

```

$ pg dfspace.cgi
#!/bin/sh
# dfspace.cgi

```

```

echo "Content-type: text/html"
echo ""
# auto refresh every 60 secs
echo "<meta http-equiv='Refresh' content='60;URL=http://linux.pc/cgi-bin/
dfspace.cgi'">"
echo "<HTML>"
echo "<HR>"
echo "<A NAME='LINUX.PC Filesystems'>LINUX.PC Filesystems</A>"

echo "<TABLE align='center' cellspacing='20' border=9 width='40%'
cols='2'>"
echo "<TH align='center'>- Capacity % -</TH>"
echo "<TH align='center'>- File System -</TH>"
# get the output from df, but first filter what we only need!
df | sed 1d | awk '{print $5"\t"$6}' | while read percent mount
do
    echo "<TR><TD align='center'><B>$percent</B></TD><TD align='center'>
    $mount</TD>"
done
echo "</TR>"
done
echo "</TABLE>"
echo "</HTML>"

```

在你浏览器的URL框中输入如下的URL：

`http://<server_name>/cgi-bin/dfspace.cgi`

你就会看到如图29-15所示的输出(在你的机器上所看到的文件系统状态多半与此不同)。

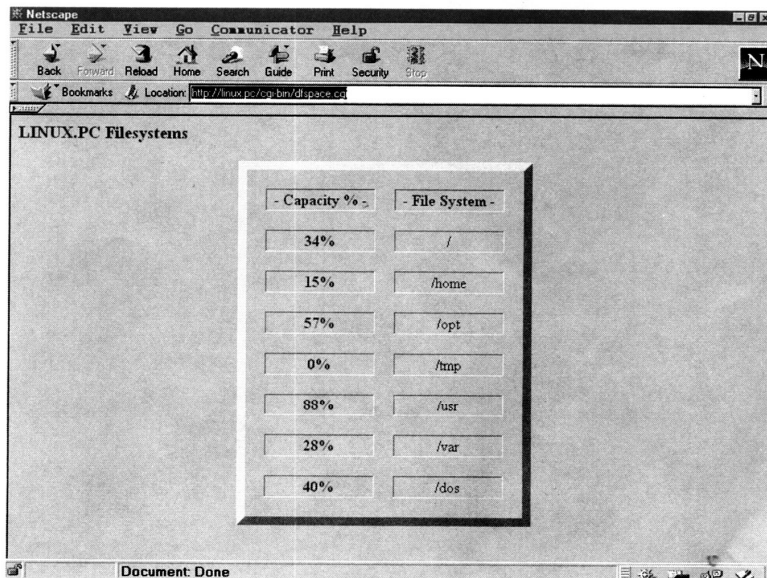


图29-15 使用表格实时显示df命令的输出

## 29.6 小结

使用cgi脚本可以创建更为有趣的用户界面。HTML页面可以作为任何处理过程的前端界面。

我们可以编写用于监视、显示、数据库查询等的脚本。HTML现在已经成为很多应用程序文档的标准格式。