目　　录

前　言
第一章　通用网关接口（CGI）简介.....1
1.1　WWW 是如何工作的.....1
1.2　CGI 是如何工作的.....3
1.3　标准输入(STDIN)和标准输出(STDOUT).....4
1.4　和 CGI 有关的 Web 服务器上的目录.....6
1.5　CGI 编程的特点.....6
1.6　CGI 的未来.....7
第二章　SSI 的使用.....10
2.1　SSI 简介.....10
2.2　SSI 命令简介.....11
2.3　SSI 简单示例.....12
2.4　有关 SSI 使用的 Web 服务器的文件设置.....13
2.5　config 命令的使用.....14
2.6　include 命令的使用.....18
2.7　fsize 命令的使用.....18
2.8　echo 命令的使用.....19
2.9　exec 命令的使用.....21
2.10　if 命令的使用.....22
2.11　email 命令的使用.....23
2.12　odbc 命令的使用.....24
2.13　SSI 命令的安全问题.....25
2.14　SSI 命令使用实例.....26
2.14.1　给主页添加动态欢迎信息.....26
2.14.2　对将来某一日期进行倒记时.....28
2.14.3　给 Web 网页添加有关最后访问者的信息.....30
第三章　CGI 编程的基本原理.....33
3.1　HTML 表单（Form）.....33
3.2　第一个 CGI 程序.....35
3.3　MIME 类型.....38
3.4　表单（Form）的制作.....38
3.4.1　TEXTAREA.....39
3.4.2　SELECT.....39
3.4.3　INPUT.....39
3.5　传送表单数据的两种方法：GET 和 POST.....42

3.6 URI 译码（URI Encoding）.....45
3.7 URI 解码.....47
3.8 利用 CGI 程序生成 HTTP 头信息.....54
第四章 环境变量的使用.....57
4.1 和 Web 服务器状态有关的环境变量.....57
4.2 和浏览器的请求头有关的环境变量.....58
4.3 和浏览器状态有关的环境变量.....61
4.4 SSI 环境变量和 CGI 环境变量之间的区别.....61
4.5 环境变量 PATH.....64
4.6 在 CGI 程序中使用环境变量.....64
第五章 深入 CGI 编程.....70
5.1 CGI 编程语言的选择.....70
5.1.1 UNIX SHELL.....70
5.1.2 Perl.....71
5.1.3 TCL.....72
5.1.4 C 语言.....72
5.1.5 C++.....73
5.1.6 Visual Basic.....73
5.2 CGI 程序的基本框架.....74
5.2.1 程序的初始化准备工作.....74
5.2.2 数据处理.....75
5.2.3 程序处理结果的输出.....75
5.2.4 程序的结束.....76
5.3 CGI 编程的注意事项.....76
5.3.1 程序的可维护性.....76
5.3.2 程序的安全性.....76
5.3.3 CGI 程序和系统平台的无关性.....77
5.3.4 CGI 程序和 Web 服务器的无关性.....77
5.3.5 CGI 程序输出的一致性.....78
5.4 CGI 程序的测试.....78
5.5 CGI 程序的出错处理.....80
第六章 利用现有 CGI 程序库资源.....81
6.1 CGI-UTIL 库.....81
6.2 CGIC.....92
6.3 GD 库.....93
6.4 CGI4VB.....95
6.5 其他一些有关 CGI 的网络资源.....103
第七章 CGI 编程实例.....105
7.1 C 语言编程实例.....105

7.1.1 下拉式菜单链接选择器.....105
7.1.2 主页意见箱.....110
7.1.3 贵宾留言簿.....119
7.1.4 给需要身份认证的 Web 资源增加用户.....128
7.1.5 主页计数器.....137
7.2 Visual Basic 编程实例.....140
7.2.1 用 VB 编写的 CGI 程序处理中文信息.....140
7.2.2 主页意见箱.....141
7.2.3 贵宾留言簿.....149
第八章 客户端用户状态信息的保存.....156
8.1 利用环境变量 PATH_INFO 和 QUERY_STRING.....156
8.2 利用表单的隐含文本框.....157
8.3 使用隐含文本框的 CGI 程序实例.....157
8.4 HTTP Cookie.....164
8.4.1 什么是 HTTP Cookie.....164
8.4.2 HTTP Cookie 工作原理.....165
8.4.3 HTTP Cookie 使用时的一些限制.....166
8.4.4 用 CGI 程序设置和读取 HTTP Cookie.....166
8.5 使用 HTTP Cookie 的计数器.....168
第九章 动态网页的制作技术.....172
9.1 普通 CGI 程序.....172
9.2 Client Pull.....174
9.3 Server Push.....175
9.4 外部程序.....177
第十章 Web 网页和数据库的连接.....178
10.1 VB 数据库 CGI 程序.....179
10.1.1 VB 的数据库编程能力.....179
10.1.2 用 VB 编写处理数据库的 CGI 程序.....179
10.2 IDC.....187
10.2.1 IDC 简介.....187
10.2.2 IDC 文件和 HTX 文件.....187
10.2.3 用 IDC 实现贵宾留言簿.....190
10.3 ASP.....192
10.4 Livewire.....195
10.4.1 Livewire 简介.....195
10.4.2 用 Livewire 实现贵宾留言簿.....196
第十一章 Image Map 的使用.....199
11.1 什么是 Image Map.....199
11.2 Web 服务器内置的 Image Map.....199

11.3 客户端浏览器内置的 Image Map 200
11.4 用 CGI 程序实现 Image Map 202
第十二章 Windows CGI 规范 211
12.1 Windows CGI 的工作原理 211
12.2 CGI 数据文件 212
12.3 Win-CGI 程序的输出 215
12.4 Win-CGI 编程 215
附录 A MIME 类型 216
附录 B 服务器的状态码 219

第一章　通用网关接口（CGI）简介

利用通用网关接口（CGI），Web 服务器可以执行一些外部的小程序，可以将这些外部小程序所产生的输出结果和 Web 服务器所管理的静态文本、图像和声音融合在一起传给相应的 Web 浏览器。Web 服务器和 CGI 程序的结合大大增强了整个万维网（World Wide Web，WWW）的信息提供和交互能力。

通用网关接口（CGI）只是规定了一个标准的接口规范，只要遵守这个标准规范，程序开发人员就可以利用各种编程工具，例如 Perl、C、Fortran、Visual Basic 等进行 CGI 编程。利用 CGI 程序，Web 服务器可以收集 Web 页面中表单内的信息，可以对特定的 Web 网页进行访问计数，可以根据浏览器的请求对数据库进行检索和更新。

本章将简要介绍 CGI 的一些基本概念。读完本章后，读者将对 CGI 如何工作的全貌有一个基本的了解。

本章主要讨论以下问题：

- WWW 是如何工作的；
- CGI 是如何工作的；
- 标准输入和标准输出；
- Web 服务器上和 CGI 程序有关的目录；
- CGI 编程的特点；
- CGI 的未来。

1.1　WWW 是如何工作的

目前流行全球的因特网（Internet）热有很大一部分是由 WWW 的盛行而引起的，WWW 的实现使得以前基于字符方式的 Internet 信息交流方式进入了图形和多媒体时代。

WWW 其实是一个基于超文本的信息查询和交互系统，最初是由欧洲粒子物理实验室为满足世界各地的研究人员交流信息而开发的。它采用超文本和多媒体技术，将不同文件通过关键字建立链接，为用户提供一种非线性的查询方式。在一个超文本中，一个关键字链接着另一个与该关键字有关的文件，该文件既可以在同一台主机上，也可以在 Internet 的另一台主机上。同样，被链接的文件也可以是一个超文本文件。超文本文件可以把不同类型的文件，如文本、声音、图像和视频等连接起来。即使是一个超文本文件，也可以包含各种声音、图像和视频等。这样，借助于这类超文本文件的链接，人们可以把各类信息组织起来，形成一个遍及全球的万维网 WWW，其工作方式如图 1-1 所示。

如果您已经使用过 Windows 应用程序的帮助文件，那么实际上您已经和超文本打过交道并享受到超文本所带来的优越性了。

发生在计算机网络上的一切活动都必须遵守一定的协议，WWW 也不例外。Web 服务器和浏览器间所要遵守的协议是超文本传输协议（HTTP）。网络协议通常是分层次的，一层网络协议建立在下一层网络协议之上并利用下一层网络协议所提供的功能来实现。事实上关于 WWW 的超文本传输协议是位于整个国际互联网协议栈的最上层即应用层，如图

1-2 所示。

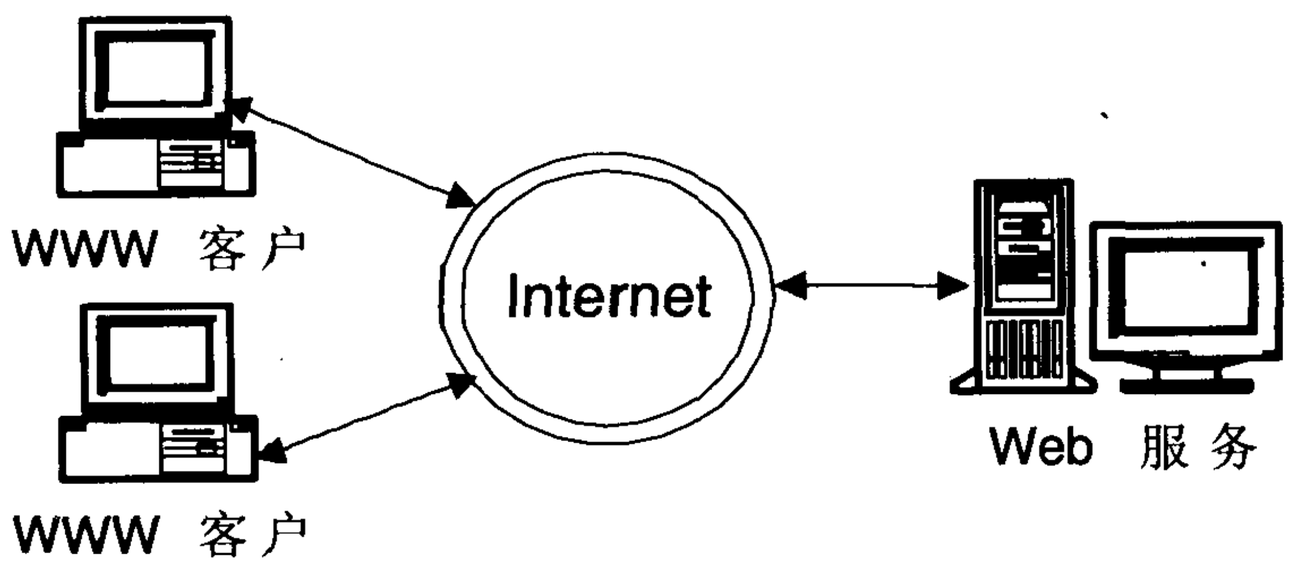

图 1-1　WWW 的工作方式

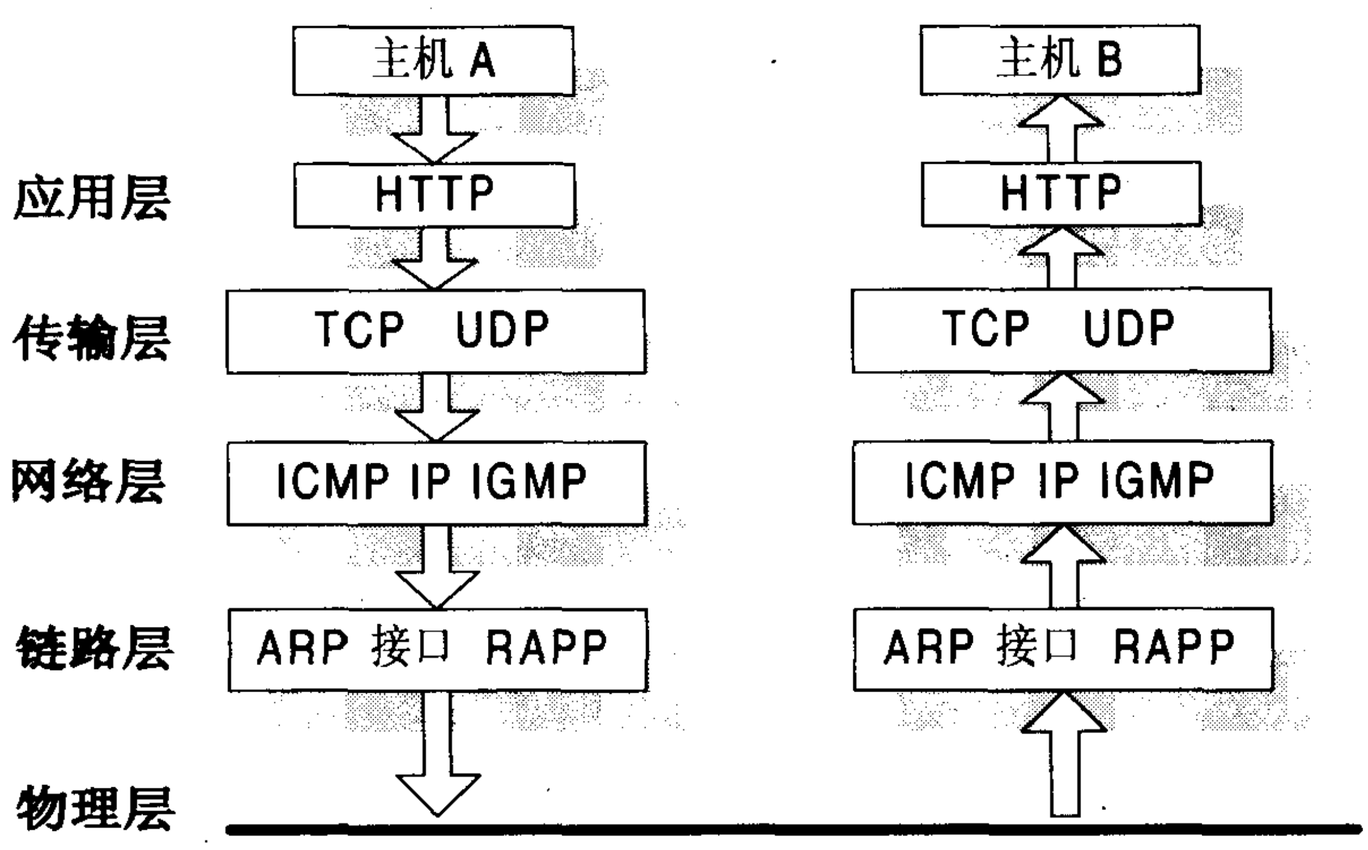

图 1-2　WWW 的协议栈

WWW 采用客户/服务器的工作方式。客户/服务器的计算模型在现今的计算机技术领域已经被应用得非常广泛，WWW 更是一个成功的例子。一般来说，Web 浏览器为客户端，负责提供用户界面，向用户显示服务器所提供的信息。而服务器负责生成查询结果，并把查询结果传送给客户端。

当您激活了 Web 浏览器并准备访问 WWW 上的一个站点时，在客户端的 Web 浏览器和 Web 服务器间将会发生下列事件。

（1）Web 浏览器根据您提供的 URL 和相关的 Web 服务器接触。

URL（Universal Resource Locator）是一个非常重要的有关 WWW 的概念。有了 URL，Web 浏览器才知道要到哪里去、如何到达目的地以及到达目的地以后又干些什么。从技术的角度来看，URL 是用已有的协议来表示 Internet 上存在对象的 URI（Universal Resource Indentifier）的一种形式，它通常具有以下的结构：

http: // <host> : <port> / <path> ? < search-part>

其中 http 表示超文本传送协议，<host>为服务器的 Internet 域名，<port>为服务器的 TCP 端口，如果在 URL 中没有指明端口，浏览器将默认为 80。

（2）如果能和 Web 服务器联系上，浏览器将把整个 URL 传送给 Web 服务器。

（3）Web 服务器将根据 URL 所提供的路径查询 Web 浏览器所需要的资源，并将该资源传送给 Web 浏览器。

（4）Web 服务器中断和浏览器的链接。

（5）Web 浏览器显示 Web 服务器传回的 Web 页面。

Web 浏览器将根据收到文件中的 HTML（超文本描述语言）代码显示 Web 页面。HTML 代码描述了 Web 页面在浏览器窗体中的显示方式。它是一种描述性语言。例如为了使问候语“Good Morning”能以斜体方式出现在浏览器窗体中，就应该用 HTML 代码将它标记为“<I> Good Morning </I>”。<I>表示接下来的文本为斜体，</I>表示斜体部分的结束。

通过上面的描述我们现在已经了解到一个 Web 页面是如何被服务器提交的。此时理解超文本链接的工作方式就变得简单了。当用鼠标点取了一个超文本链接时，浏览器就根据有关超文本链接的 HTML 语句中的 URL 和另一个服务器或超文本文件发生联系。

1.2　CGI 是如何工作的

通用网关接口（CGI）是 Web 服务器和您编写的外部程序之间的一个接口。CGI 可以使您编写的程序处理 WWW 上客户端送来的表单和数据并对此作出某种反应。这种反应可以是 HTML 文件、图片、声音、视频等可以在浏览器窗体上出现的任何数据。通用网关接口有两个含义。首先它为客户端用户通过 WWW 获取 Internet 上的其他服务，例如 Archie、WAIS、Database 等提供了一个接口，虽然不能直接在 WWW 上访问这些服务，但是您可以通过一个 CGI 程序和 Web 服务器访问它们。其次它就是您编写的程序和 Web 服务器间的接口标准。您所编写的处理客户端有关信息的程序必须遵循这个标准。

本节将讨论 CGI 是如何在客户端的 Web 浏览器（如 Netscape Navigator）和 Web 服务器（如 NSCA HTTPd）间工作的。从上一节对 WWW 的介绍可知，当客户端的浏览器向 Web 服务器请求一个 HTML 文件时，服务器在收到请求后就去寻找这个文件并将找到的文件传送给客户端。而当客户端的请求是一个 CGI 程序时，Web 服务器将激活客户端所请求的 CGI 程序。此时 Web 服务器所起的作用就好像是客户端和 CGI 程序间的中间人，如图 1-3 所示。CGI 程序的工作一般可以分为以下几个步骤。

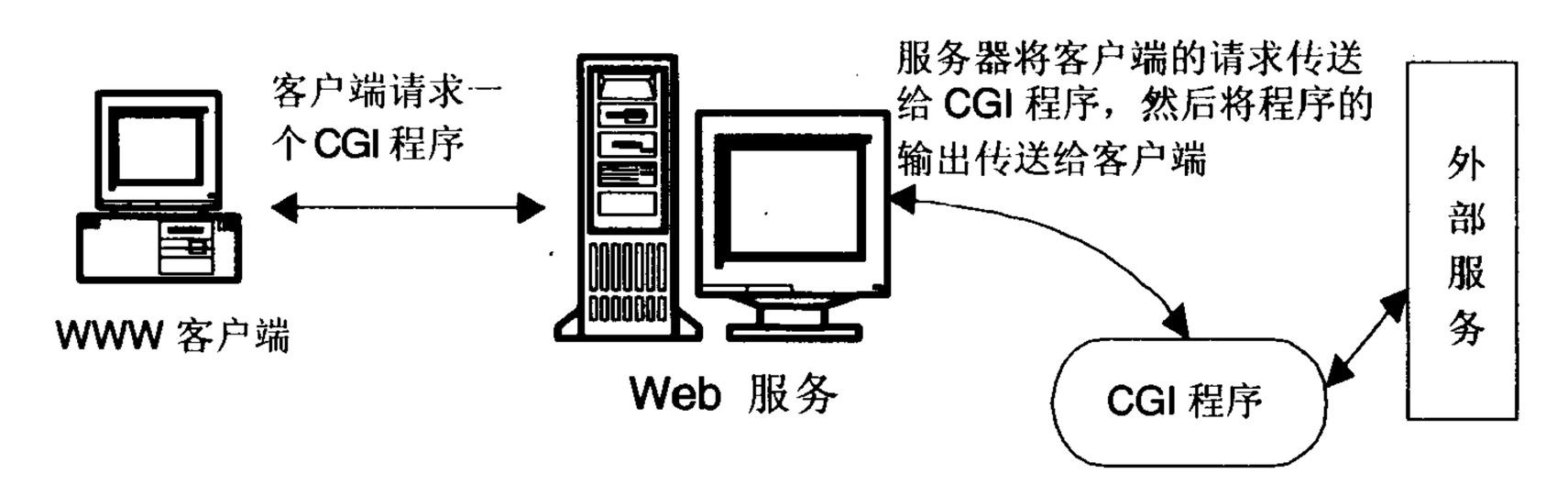

图 1-3　CGI 的工作原理

1. 客户端发出请求

首先是客户端的浏览器发出一个请求给 Web 服务器。如果这个请求是一个普通的文件（例如 HTML 文件、GIF 或 JPEG 文件），Web 服务器就将文件直接传送给客户端的浏览

器。如果这个请求是一个 CGI 程序或外部应用，Web 服务器将激活相应的 CGI 程序。例如客户端传送来的请求可能是对一个数据库的检索，此时一个相应的 CGI 程序将被 Web 服务器激活，这个 CGI 程序将根据客户端要求的准则去检索相应的数据库，并通过 Web 服务器将结果传送给客户端。

2. Web 服务器激活 CGI 程序

当 Web 服务器发现客户端浏览器传送来的请求不是一个普通文件而是一个 CGI 程序时，Web 服务器就要激活这个 CGI 程序并执行它。

在这个 CGI 程序被执行前，Web 服务器要为该 CGI 程序设置一些环境变量。这些环境变量被服务器用来向 CGI 程序传递一些非常重要的信息，例如当前 Web 服务器的状态、谁发出的调用等等。Web 服务器为 CGI 程序所设置的环境变量的使用和正常的环境变量的使用没有任何区别。当 CGI 程序运行结束时，Web 服务器为它设置的环境变量也随着消失。每个被激活的 CGI 程序都有自己唯一的一组环境变量。事实上对于一个非常繁忙的 Web 服务器，可能同时会有同一个 CGI 程序的很多个进程在运行，这时每个 CGI 程序的进程都有自己的运行环境，互不影响。

在以后的章节本书将继续讨论通用网关接口的环境变量，在此处只要了解到这些环境变量是存在的并包含有可以被 CGI 程序读取的重要信息就行了。

3. CGI 程序对客户端的请求做出反应

CGI 程序通过环境变量或其他途经读取 Web 服务器所提供的客户端的输入数据并对这些数据进行处理，如果有必要 CGI 程序还要和一些外部的服务例如 Archie、WAIS 和 Database 等发生联系。最后 CGI 程序将通过 Web 服务器将处理结果传送给客户端。CGI 程序可以根据需要产生各种类型的数据。

4. Web 服务器将 CGI 程序的处理结果传送给客户端

Web 服务器收到了 CGI 程序的处理结果后将对 CGI 程序的输出进行检查，如有必要，将为 CGI 程序的输出结果加上一些 HTTP 协议所必须的 HTTP 头信息。检查完成后将把 CGI 程序的输出传送给客户端。

5. Web 服务器中断和客户端浏览器的连接

6. Web 浏览器将 CGI 程序的输出显示在浏览器的窗体上

从以上的介绍可以看出，CGI 程序的工作要比普通的 WWW 工作要复杂一些。

1.3 标准输入(STDIN)和标准输出(STDOUT)

一般来说大部分情况下 CGI 程序都是一些小应用程序，编写 CGI 程序并不比编写普通的程序难。初次接触 CGI 程序的人都觉得它很神秘，它怎么就可以处理客户端的表单并进行数据库的检索的呢？在这里最重要的是理解，是对通用网关接口规范 CGI 的理解。Web 服务器和 CGI 程序之间不仅仅通过环境变量和 Web 服务器交换信息，大多数的数据交流都是通过标准输入和标准输出进行的。

STDIN 和 STDOUT 是表示标准输入和标准输出的两个文件句柄。一般主机系统所产生的进程都拥有这两个文件句柄，并且都已经被打开。一个程序可以通过向标准输出（STDOUT）写实现向屏幕输出信息，通过标准输入（STDIN）读从键盘获取信息。一个

程序如果不从键盘而从其他地方获取输入信息，它的标准输入（STDIN）就被重新定向；一个程序如果不向屏幕而是向其他地方输出信息，那么它的标准输出也被重新定向。

例如在 UNIX 系统下敲入命令 ls，进程 ls 将把它的输出写到标准输出即终端屏幕上。如果敲入命令 ls > file1，屏幕上将看不到任何信息，因为通过特殊的 shell 字符“>”，进程 ls 的标准输出被定向到名称为 file1 的文件上，以前在屏幕上输出的信息这次将被写到文件 file1 中去。

一个程序的标准输入/输出（STDIN/STDOUT）也可以和另一个程序的标准输出/输入（STDOUT/STDIN）相连，这通常被称为管道连接。例如在 UNIX 系统下敲入命令 ls | sort，就可以将命令 ls 的标准输出和命令 sort 的标准输入相连了，命令 sort 执行时将把命令 ls 的输出作为它的输入。

对于一个 CGI 程序来说，当 CGI 程序被服务器激活以后，它的标准输入（STDIN）被连接到 Web 服务器的标准输出（STDOUT）上，而 CGI 程序的标准输出（STDOUT）则被连到服务器的标准输入（STDIN）上，这就形成了完整的管道循环。从 CGI 程序这一方来看，它从标准输入读取信息也就是从 Web 服务器的标准输出读信息，而它向标准输出写信息也就是向 Web 服务器的标准输入写信息。

Web 服务器一般将客户端传送来的信息放在它的标准输出和相关环境变量中，而 CGI 程序则从环境变量和它的标准输入（也就是 Web 服务器的标准输出）获取所需的信息，程序的最终输出结果则被写向它的标准输出（也就是 Web 服务器的标准输入）。Web 服务器将从它的标准输入（也就是 CGI 程序的标准输出）获取 CGI 程序的输出结果并将它传送给客户端。客户端、Web 服务器和 CGI 程序之间的信息交流如图 1-4 所示，显然，Web 服务器就像是客户端和 CGI 程序间的中介人。

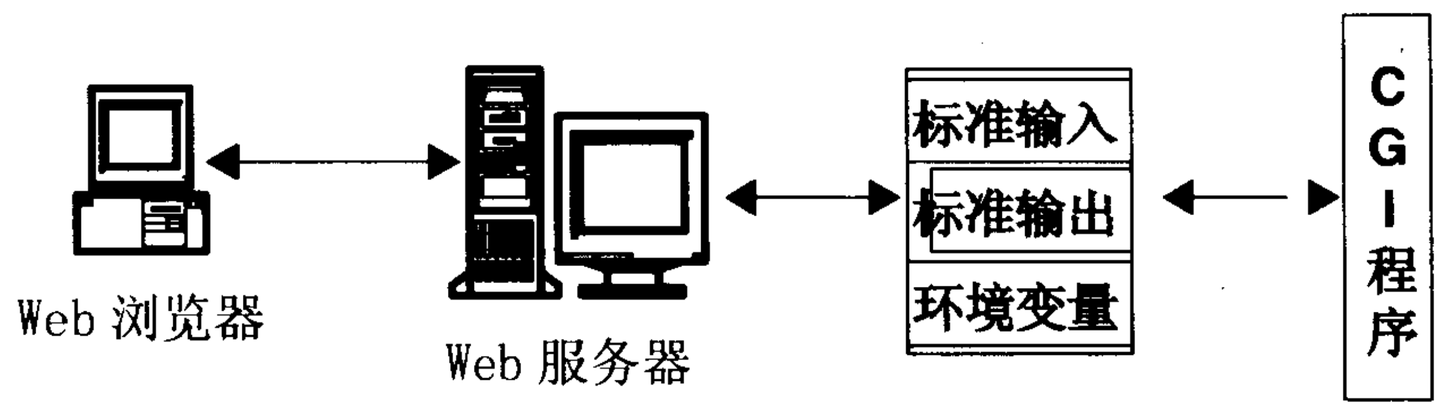

图 1-4　CGI 程序和 Web 浏览器间的信息交流

Web 服务器、CGI 程序间的这种标准框架在 UNIX 系统下和微软视窗环境中的字符方式下可以工作得非常好，因为此时系统所产生的所有进程都可以存取标准输入和标准输出。但对于微软视窗环境图形方式下的程序就不行了，这是由于它们无法存取标准输入和标准输出。为了解决这一问题，微软在 WIN32 系统中创建了另一类型的标准输入和标准输出，程序可以通过调用 WIN32 API 函数在图形方式下存取标准输入和标准输出，不过这就意味着使用这类标准输入和标准输出的 CGI 程序都必须是 32 位的。

微软视窗环境下的其他一些 Web 服务器（例如 Website）则使用另外一种特殊的技术，即利用 INI 文件来实现 Web 服务器和 CGI 程序间的数据交流。采用这种被称为 WIN-CGI 规范编写的 CGI 程序通常只能在部分 Web 服务器上运行。本书将在以后的章节对其进行介绍。一般支持 WIN-CGI 的 Web 服务器将客户端的输入和有关的状态信息写入一个 INI

文件中，而 CGI 程序则从该 INI 文件获取相关信息。

在进行 CGI 编程时，您只要使您的 CGI 程序从标准输入和环境变量中获取客户端提供的信息，把要传送给客户端的输出结果写入标准输出，剩下的信息传递工作 Web 服务器将自动为您完成。

1.4 和 CGI 有关的 Web 服务器上的目录

通常，一个 Web 服务器拥有两个目录树（一个目录树由一个根目录和根目录下的所有子目录组成）。为了将 WWW 资源和 Web 服务器的管理资源分开，对于 UNIX 系统来说，存放 WWW 资源的目录树通常叫 document root，存放服务器系统管理资源的目录树通常叫 server root。

Web 服务器识别 CGI 程序有两种方式。当它发现客户端送来的请求是一个特殊目录下的文件时，就认为该文件是一个 CGI 程序，或者当它发现客户端传送来的请求是带有特殊后缀名（例如 .CGI）时，就认为它是一个 CGI 程序。两种方式的使用取决于 Web 服务器本身和服务器管理员（Webmaster）的个人习惯。相对应于第一种方式，CGI 程序通常被放在 document root 目录树的一个子目录中。比如 document root 树的根目录为 /var/www/docs/ ，所有的 CGI 程序位于目录 /var/www/docs/cgi-bin/ 中，这个目录通常被称为 CGI 目录。相对应于第二种方式，所有位于 /var/www/docs/ 及其子目录下带后缀名 .CGI 的文件都是 CGI 程序。由于 CGI 程序可以存取系统资源，为了安全和管理上的方便，大多数服务器管理员都使用第一种方式，第二种方式即使能用也不采纳。

现今的很多 Web 服务器都支持多 CGI 目录，即可以在目录树 document root 下定义多个 CGI 目录。如果 Web 服务器还支持虚拟服务器，您还可以定义多个 document root 目录树，每个 document root 目录树将对应一个虚拟服务器。

在目录树 sever root 下包含了所有 Web 服务器管理员（Webmaster）和系统管理员（System Administrator）需要负责的有关 Web 服务器的管理资源。虽然只有服务器管理员和系统管理员才能对它们进行设置和改动，但了解它们对 CGI 编程还是有很大帮助的。

在 server root 目录树下通常有 log 和 conf 两个子目录（各种 Web 服务器的名称都大同小异），在 log 目录下存放的都是有关服务器运行的各种日志文件，和 CGI 编程有关的文件一个是 error.log 文件，该文件记录了服务器运行时所发生的错误以及客户端对服务器所采取的一些错误行为，这个文件对于 CGI 程序的调试有一定的帮助作用。另一个文件是 access.log 文件，它详细记录了客户端对服务器资源的访问和使用，可以利用该文件编制一些对 Web 服务器受访问的情况进行分析和统计的 CGI 程序。在 conf 目录下则存放了一些服务器的配置文件，CGI 程序的正确运行还有赖于这些文件的正确设置。目前大多数商业 Web 服务器都提供了方便的图形菜单来管理这些文件。

1.5 CGI 编程的特点

由于 CGI 程序的运行环境和普通程序的运行环境有很大的不同，和普通编程相比，CGI 编程有着自身的特点。

首先，CGI 程序必须考虑程序运行时的安全性。您编写的 CGI 程序是准备放在 Web

服务器上供大家使用的。也许会有成千上万的人使用这个程序，在这当中也许会有很多好奇心特别强的人，他们十分关心您编写的 CGI 程序的安全漏洞并千方百计地想寻找这样的漏洞，在找出这些漏洞以后他们就有可能攻击您的系统。您不能阻止他们这么做，您唯一能做的是在 CGI 程序中消除这些漏洞。对于那些用解释性语言如 Perl 编写的 CGI 程序，其源代码比用编译性语言编写的 CGI 程序的源代码更容易在网上获得，对于这类程序更应该注意安全性的问题。

其次，CGI 程序获取信息和输出信息的方式和普通程序不一样。

在编写 CGI 程序之前，您必须了解 Web 页面中的超文本链接或表单中的数据是如何被传给 CGI 程序的。在设计 CGI 程序时首先要设计好 Web 页面上的客户端用户界面。Web 页面上的数据传送给 CGI 程序的方法有两种，您必须对这两种方法的优缺点有所了解。

而且，WWW 上存在着各种不同类型的 Web 浏览器，例如 Netscape、MSIE 和文本方式的 Lynx 等等。有的浏览器除了支持标准的 HTML 规范外还对其做了自己的扩充，您编写的 CGI 程序必须注意各类浏览器之间甚至同类浏览器的不同版本之间的区别。

最后，不像普通的程序，CGI 程序是工作在无状态的运行环境中。这种无状态的运行环境有两个特征：（1）不提供永久性的上下文信息，即在会话连接期间不保存上下文信息；（2）系统可能会拥有同一 CGI 程序的多个进程，这样，一旦 Web 服务器响应了客户端一个运行 CGI 应用程序的请求，Web 服务器在 CGI 程序运行结束后将立刻切断与发出请求的客户端的连接。如果这个连接重新建立，上一次连接的状态信息将彻底丢失。同时同一个 CGI 程序可能会有多个拷贝在系统中运行，这在某些情况下也会带来问题。

例如您正在建立一个拥有多个网页的 Web 站点，其中的一页可能会在其他网页上产生变化，除非采取一些特殊的步骤，否则在某一网页上发生的变化在下次存取它时将看不到这些变化。所以您的 CGI 程序必须采取一些特殊的方法来跟踪上一次发生的变化，一个最普通的方法是让 CGI 程序将最后一次的变化信息存入一个文件中，不过这个方法有个缺陷，可能会有很多个客户同时调用 CGI 程序，CGI 程序将无法确定是谁调用了它。

解决问题的另一个方法是使用 HTML 表单的隐藏文本框。CGI 程序将这一次的状态信息传送给客户端 Web 页面的隐藏文本框。当客户端再次调用其他网页时，CGI 程序就可以从隐藏文本框中获取上一次的状态信息。当然这一方法也有缺陷，虽然 Web 网页将不向客户端用户展示隐藏文本框的内容，但用户却可以用浏览器的 View Source 的菜单命令看到它的内容，这就使得客户端用户可以改变传送给 CGI 程序的数据。使用 HTTP Cookie 则是一个更好的办法，本书的第八章还将详细讨论这个问题。

运行在无状态环境中的 CGI 程序可能同时会被多个浏览器唤醒。一个 CGI 程序可能会修改某个文件，而当该 CGI 程序同时有多个拷贝在运行时，就可能出现有多个进程试图对同一个文件进行读写的情况。此时如果不采取一些特殊的保护措施，某些数据就有可能丢失。当两个进程同时打开一个文件时，最先关闭文件的进程所保存的信息总是被最后关闭文件的进程所保存的信息所覆盖。为了解决这个问题，可以设计一个特殊的句柄数据库用来锁住正在进行读写操作的文件，而此时其他 CGI 程序的进程只能等待文件的有关读写操作完成以后才能打开文件。

1.6　CGI 的未来

WWW 的出现为许多公司提供了巨大的商业机会，为了占领市场的最高点以使自己处

于有利的位置，很多大公司都纷纷推出自己的新技术，例如 Java、Javascript 和 VBscript 等等。这就使得 WWW 的发展日新月异，从而不可避免要对 WWW 上最早出现的 CGI 产生一定的冲击。

Java 语言是太阳微系统公司（Sun Microsystem）推出的一种和平台无关的编程语言。Java 源程序被编译以后产生的并不是执行代码而是一种中间代码，该中间代码在执行时需要 Java 虚拟机的解释才能运行。正是由于 Java 虚拟机的存在才使得 Java 语言具有和系统平台的无关性。WWW 只是 Java 语言应用的一个方面。通常以 Java applet 的形式出现。

Javascript 是 Netscape 公司和太阳微系统公司联合推出的一种脚本编程语言，可以将它看成 Java 语言的一个子集。用 Javascript 编写的小程序可以被包含在 Web 网页内并在浏览器上执行，这就使得 Web 网页不用和服务器打交道就可以具有很多动态效果。

VBscript 是微软推出的一种和 Javascript 类似的脚本编程语言，不过比 Javascript 更简单、更容易使用。它是 Visual Basic 的一个子集，熟悉 Visual Basic 的编程人员可以很快就掌握它。

Java applet、Javascript 和 VBscript 可以实现以前很多用 CGI 才能实现的功能。但它们和 CGI 程序仍然有着本质的不同。Java applet、Javascript 和 VBscript 所编写的程序是先被传送到客户端然后在客户机上运行，而 CGI 程序是在服务器上运行以后再将结果传送给客户端。这就决定了 Java applet、Javascript 和 VBscript 还不能完全代替 CGI 程序。

由于 Java applet、Javascript 和 VBscript 充分发挥了客户机的潜力，降低了服务器的负担，在实际 CGI 编程中能用 Java applet、Javascript 和 VBscript 的地方要尽量使用它们而不用 CGI，最好能将二者结合起来使用。例如您的某一 Web 网页是利用 HTML 表单向 Web 服务器上的数据库插入或更新记录。若这一功能都用 CGI 程序来实现，CGI 程序首先要检查输入数据的正确性。若不正确，则向客户端用户返回一个错误信息，若正确则插入数据。其实完全可以用 Javascript 或 VBscript 编写的小程序在客户端就完成用户输入数据的正确性检查，如果用户输入的数据不正确，就不能调用 CGI 程序更新数据库。显然，这样做较前者可以缩短网络的延迟，降低对网络带宽的使用要求和服务器的负担。第七章的主页意见箱就是一个 CGI 程序和 Javascript 程序联合使用的例子。

目前还有一种完全可以取代 CGI 的技术，即 ISAPI。一个 ISAPI 应用程序被某一 Web 网页激活后将完全驻留在内存中以等待其他客户端对它的调用，这就避免了在 CGI 的运行环境中为产生并发进程所付出的额外开销，缩短了服务器的响应时间。和 CGI 程序不同，ISAPI 应用程序和 Web 服务器装载在同一地址空间，这就使得 ISAPI 程序不用借助于环境变量和标准输入/输 出就可以和 Web 服务器通信。当然，这样的高性能是有条件的，所有的 ISAPI 程序都必须是安全的多线程程序，而一个 ISAPI 程序发生错误时可能会引起整个 Web 服务器的崩溃。ISAPI 应用程序只能运行在微软的 Web 服务器上或其他厂商支持 ISAPI 的服务器上。

NSAPI 则是 Netscape 公司推出的和 ISAPI 类似的技术。NSAPI 应用程序也只能运行在 Netscape 公司的系列 Web 服务器或其他支持 NSAPI 的 Web 服务器上。

虽然 ISAPI 或 NSAPI 程序和 CGI 相比具有很高的性能，但开发它们需要用到专门的知识和工具，掌握它们对一个普通 Web 开发人员并不是一件容易的事情。而且 ISAPI 或 NSAPI 只能运行于特定的 Web 服务器平台。而 CGI 就不同了，一般开发人员，不管他是

精通 C、Perl、Fortran、Visual Basic 等语言还是 Delphi、PowerBuilder 等开发工具，只要熟悉了 CGI 体系的规范，很快就可以开始 CGI 编程并享受到制作动态网页的乐趣。开发人员甚至可以不用编写一些具体的 CGI 程序细节，因为在 WWW 上可以很容易找到满足他要求的 CGI 程序或 CGI 程序库。

由于 CGI 的易学性、易用性和跨服务器平台性，它还将在 WWW 上占有一席之地。

第二章　SSI 的使用

在第一章中，我们已经了解了 CGI 体系的工作原理。事实上，不用进行任何编程工作您就可以制作出一些简单的动态 Web 网页，例如可以在您的 Web 网页上自动显示 HTML 文件的最后修改时间而不用每次都去手工修改，可以给您的每一个 Web 网页自动加上本站点特有的标志，也可以在您的 Web 网页中自动包含一个文件，这些都要依赖于 SSI 的使用。SSI 是英文 Server Side Include 的缩写。本章将详细介绍各种 SSI 命令的使用。

2.1　SSI 简介

SSI 的命令标识符和 HTML 的命令标识符相类似。这些 SSI 命令标识符被包含在普通的 HTML 文件当中。当这类 HTML 文件被 WWW 上的客户端用户调用时，SSI 命令就会被 Web 服务器执行，而它们的输出结果则被 Web 服务器放在原来 HTML 文件中 SSI 命令标识符所在的地方。

SSI 是一把双刃剑，它既可以减少制作动态网页的工作量，也可以加重 Web 服务器的负担。当 Web 服务器收到的请求是一个普通的 HTML 文件时，服务器所要做的仅仅是找到这个文件，再加上一些必要的 HTTP 响应头信息后将它传给客户端。当 Web 服务器收到的请求是一个 CGI 程序时，服务器所要做的是找到这个 CGI 程序并执行它，最后将 CGI 程序的输出结果加上一些必要的 HTTP 头信息后传给客户端的浏览器。而当 Web 服务器收到的请求是一个包含 SSI 命令的 HTML 文件时，首先它要找到这个文件，找到以后它还要一行一行的分析这个文件以把 SSI 命令找出来。然后才是找出来的 SSI 命令被执行，命令执行的输出结果被放在原来 SSI 命令标识符所在的地方。最终形成的 HTML 文件再被服务器加上一些必要的 HTTP 头信息后传送给客户端的浏览器。这也是为什么在客户端的浏览器上用“View　Source”命令看不见 SSI 命令标识符的原因。

显然，在 Web 网页中使用 SSI 要大大加重服务器的负担，同样这也意味着客户端在请求一个包含 SSI 命令的 HTML 文件时等待的时间要比请求一般的 HTML 文件所需等待的时间要长。Web 服务器所需处理的包含 SSI 命令的 HTML 文件越多，整个服务器的负担就越重，服务器对外界的反应也就越慢。当然，这并不能成为不使用 SSI 的原因，Web 页面的开发人员应该在动态网页制作的方便性和服务器的负担这二者之间寻找一个折衷的平衡点。

现在市场上很多 Web 服务器产品都支持 SSI，但由于 SSI 的使用会大大增加服务器的负担，许多 Web 服务器的管理员在决定是否使用 SSI 时都很谨慎。它们可能会禁止 SSI 在服务器上的使用，也可能只允许有限制的使用 SSI。

当 SSI 功能被打开以后，Web 服务器只有在事先知道 HTML 文件中包含 SSI 命令的情况下才对该文件进行检索以找出 SSI 命令并执行之。一般来说，Web 服务器是通过文件的后缀名来了解文件中是否包含 SSI 命令的。包含 SSI 命令的 HTML 文件通常都具有特殊的后缀名，后缀名的具体形式取决于该 Web 服务器的管理员，缺省情况下一般是 .shtml、.shtm 或.sht。

和 CGI 不同，目前 SSI 还没有一个统一的 Internet 文档对它进行描述，各个 Web 服务器开发商在开发服务器产品时都对 SSI 命令集做了不同的增减，这就给 SSI 的使用带来了一定的麻烦，读者在使用 SSI 命令之前必须参考 Web 服务器的有关资料。

2.2 SSI 命令简介

SSI 命令的使用非常简单，即使如此，在使用 SSI 命令时也必须注意按照正确的语法使用它们。否则，Web 网页可能会出现一些想不到的结果，也许 Web 服务器会将这些错误的 SSI 命令当成它不知道的命令不去管它，更糟糕的情况是 Web 服务器将这些错误的 SSI 命令当成普通的 HTML 语句在 Web 页面上输出。

SSI 命令实际上是 HTML 命令符“<!--#， -->”的一个扩充，这并不是偶然的。由于继承了 HTML 命令符“<!--， -->”的一些特性，当一个包含 SSI 命令的网页被移植到一个不支持 SSI 的 Web 服务器上时，SSI 命令此时将不会影响 Web 网页的美观，包含在 HTML 命令符“<!--#， -->”之间的 SSI 命令将不会被执行，SSI 命令本身也不会被当成 HTML 语句传给客户端的浏览器。

HTML 的命令符“<!--， -->”具有如下的形式：

<!-- 任何文本（可以是 Javascript 或 VBscript 编写的在 Web 网页上执行的小程序）-->

开始符“<!--”和结束符“-->”定义了一个 HTML 执行体。

SSI 命令和 HTML 命令符的语法结构非常相似。一般出现在 HTML 文件中的每个 SSI 命令都具有如下的形式：

<!--#命令符 命令变量="命令变量值" -->

命令符表示了所要执行的 SSI 命令，“命令变量="命令变量值"”则向 SSI 命令提供了该命令运行时所需要的参数。

注意“<!--#”和命令符之间不能有空格，表示 SSI 命令的命令符必须小写，在“-->”之前必须留有一个空格。赋给命令变量的值必须用英文双引号（"#"）引起来。比如下面就是对 SSI 命令 echo 的正确使用：

<!--#echo var="DATE_GMT" -->

表 2-1 列出了目前所有的 SSI 命令。只有 Questar Microsystems 公司的 Web 服务器产品才支持表中的所有 SSI 命令，所以在使用 SSI 命令时要注意服务器是否支持该 SSI 命令。本章的以后各节将对一些常用的 SSI 命令的使用进行详细的介绍。

表 2-1 SSI 命令

SSI 命令	命 令 说 明
echo	在 Web 网页中显示一些 SSI 变量的值
include	在 Web 网页中“include”命令所在的位置插入另一个文件
fsize	在 Web 网页中显示某一文件的大小值
flastmod	将某一文件的最后更改时间插入到 Web 网页中
exec	执行系统命令或 CGI 程序并将输出结果插入到 Web 网页中
config	对 SSI 命令的输出格式进行设置

SSI 命令	命　令　说　明
odbc	对支持开放数据库互联的数据库进行查询
email	当一个 HTML 文件被存取或一个表单被上交时可以用来发送电子函件
if	为 SSI 命令的执行提供条件
goto	和 label 联合使用，可以跳过某些 SSI 命令而不执行
label	为 SSI 命令 goto 提供一个目的地
break	可以在任何一点终止 HTML 文件

2.3　SSI 简单示例

为了向 Web 服务器的访问者显示您的 Web 网页的信息更新程度，通常在服务器的每一 Web 网页上要加上该 Web 网页的最后更改时间。如果某一网页的信息经常被更新，您每次都得修改该网页的“最后修改时间”，这实在不是一件令人愉快的事，而且有时您也会忘记修改这个“最后修改时间”。使用 SSI 命令“flastmod”，您就可以自动在您的 Web 网页的最后插入“最后修改时间”，就不用每次都去修改这个“最后修改时间”了。

图 2-1 所示就是运用 SSI 命令“flastmod”的一个例子。该 Web 网页上的“最后修改时间”并不是手动修改而是用 SSI 的命令“flastmod”生成的。但您客户端用 Netscape 的浏览器 Navigator 的“View Source”命令看到的却是一个不包含 SSI 命令的普通 HTML 代码，这是因为该文件在传给客户端之前 SSI 命令已经被它的输出结果所替代。

图 2-1 利用 SSI 命令“flastmod”自动向 Web 网页插入“最后修改时间”

以下是用 Netscape 的浏览器 Navigator 的“View Source”命令看到的该网页的 HTML 代码：

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
  <TITLE>SSI 命令的使用</TITLE>
  <META NAME="Author" CONTENT="">
  <METANAME="GENERATOR" CONTENT="Mozilla/3.01Gold (Win95; I) [Netscape]">
</HEAD>
<BODY>
<H1>欢迎光临我的主页</H1>
<P><HR WIDTH="100%"><BR>
<FONT SIZE=+2><A HREF="zys.htm">我的个人简历</A></FONT></P>
<P><FONT SIZE=+2><A HREF="love.htm">我的个人爱好</A></FONT></P>
<P><FONT SIZE=+2><A HREF="friend.htm">我的朋友</A></FONT></P>
<P><FONT SIZE=+2><A HREF="java.htm">我的 Java Applet</A></FONT></P>
<P><HR WIDTH="100%">最后修改时间：97 年 08 月 18 日<BR>
</P>
</BODY>
</HTML>
```

但事实上，存放在 Web 服务器上的该 Web 网页上的显示“最后修改时间”的一行代码的真实面貌如下所示：

```
<P><HR WIDTH="100%">最后修改时间：<!--#config timefmt="%y 年%m 月%d 日" -->
<!--#flastmod file="sample1.shtml" --><BR>
```

在这里用了两个 SSI 命令，后一个 SSI 命令“flastmod”用来插入文件 sample1.shtml 的最后修改结果，前一个 SSI 命令“config”用来定义后一个 SSI 命令“flastmod”的输出形式。

2.4　有关 SSI 使用的 Web 服务器的文件设置

对于一些商业 Web 服务器产品，例如微软和 Netscape 的 Web 服务器产品，打开 SSI 功能非常简单，可以借助于浏览器通过一个图形菜单进行操作。而对于一些共享 Web 服务器软件，例如 NCSA、CERN 和 Appache 等，则要手动修改有关 Web 服务器的配置文件。

对于和 NCSA HTTPd 服务器兼容的 Web 服务器，通常要修改 access.conf 和 srm.conf 这两个配置文件。

在 access.conf 文件中使用 option 关键字可以定义服务器哪些目录下的 HTML 文件和子目录可以使用 SSI。语句“Option Includes”可以使某一目录下的 WWW 资源能够使用 SSI 功能，而语句“Option IncludesNoExec” 则可以使某一目录下的 WWW 资源能够使用 SSI 功能但不能使用 SSI 功能的 exec 命令。例如文件 access.conf 中的以下设置语句将使目录“/usr/local/etc/httpd/htdocs”下的所有子目录和 HTML 文件都可以完全使用 SSI 命令。

```
<Directory /usr/local/etc/httpd/htdocs>
  Options  Includes
</Directory>
```

文件 srm.conf 则定义了 Web 服务器上使用 SSI 命令的 HTML 文件的后缀名。例如文件 srm.conf 中的如下语句定义了使用 SSI 的 HTML 文件的后缀名为 .shtml:

```
AddType  text/x-server-parsed-html .shtml
```

2.5　config 命令的使用

在上一节中我们已经看到了 SSI 命令“config”的使用。SSI 命令“flastmod”缺省的输出形式是西文格式，如果不用“config”命令规范“flastmod”的输出，将无法在中文网页中使用 SSI 自动插入更改时间的功能，除非您能忍受在一行中文句子中夹杂着几个英文缩写单词。

一般“config”命令用来定义各种 SSI 命令的输出形式。如果想改变日期的表达形式，对系统命令的输出结果进行加工，定义出错信息，以 KByte 而不是以 Byte 来反映文件的长度，都必须用到 SSI 的命令“config”。虽然“config”并不在 Web 网页中输出任何东西，但对“config”的正确使用将使 Web 网页更美观、更友好。

“config”命令和其他的 SSI 命令类似，通常具有如下的形式：

```
<!--#命令符 命令变量="命令变量值" -->
```

表 2-2 列举了“config”命令的一些用法。

表 2-2　config 命令的使用

命令符	命令变量	命令变量值
config	errmsg	任意文本字符串
config	sizefmt	bytes 或 abbrev
config	timefmt	表 2-4 中的取值
config	cmdecho	ON 或 OFF
config	cmdprefix	任意文本字符串或 HTML 标识符
config	cmdpostfix	任意文本字符串或 HTML 标识符
config	onerr	表 2-3 中的取值

SSI 的各种命令在执行中都可能发生一些错误，此时 Web 服务器会在网页上显示一些出错信息，一般这些信息都是英文的。如果 Web 网页是中文页面，此时您当然希望用中文的出错信息去替代英文的，这时就要用到“config”命令：

```
<!--#config errmsg="您所定义的出错信息" -->
```

对于“config”命令，关于出错信息还有一个更灵活、更强大的使用方法，即可以利用“config”定义一些事件：

```
<!--#config onerr="表 2-3 中列举的事件" -->
```

当 SSI 命令的执行出现错误时这些事件会被触发，SSI 会转而去执行这些事件，表 2-3 列举了这些事件。

表 2-3　“config”定义出错信息时用到的一些事件

事件名称	事　件　说　明
goto	跳转到 label 所定义的地方，执行相应的 SSI 命令
print	输出自定义的文本或 HTML 标识符，使用方式为 print <text>
error	输出当前的错误信息
break	终止 HTML 对客户端的传送
errorbreak	输出当前的错误信息，然后终止 HTML 对客户端的传送
printbreak	输出自定义的文本或 HTML 标识符，然后终止 HTML 对客户端的传送

当“config”的命令变量为“sizefmt”时，“config”命令就会影响 SSI 命令“fsize”的输出。“fsize”可以返回某一文件的长度，此时“config”命令决定这一长度值是以 Byte 方式输出还是以 Kbyte 方式输出。当“sizefmt”的取值为“bytes”时，按 Byte 方式输出，若取值为“abbrev”时，则按 Kbyte 方式输出。

“config”还有一个非常有用的功能，就是定义诸如“echo”、“flastmod”等 SSI 命令的日期时间输出格式。您可以仅仅以星期几的方式显示日期，也可以以 GMT 的完整格式显示日期，由于“config”的灵活性，甚至可以在 Web 网页上用各种语言来显示时间。本章的第 3 节对此已经有所介绍。此时“config”以如下的方式使用：

```
<!--#config timefmt="表 2-4 中的各种格式符与文本串的取值的组合"  -->
```

例如：

```
<!--#config timefmt="%A, %B %d, %Y"  -->
```

或

```
<!--#config timefmt="%y 年%m 月%d 日" -->
```

表 2-4 列举了“config”命令所使用的定义日期输出格式的格式符。

表 2-4　“config”命令所使用的定义日期输出格式的格式符

格式符	格　式　符　说　明
%a	用英文名称缩写表示当前是星期几。
%A	用英文全称表示当前是星期几。
%b	月份的英文名称缩写
%B	月份的英文全称
%c	事先定义的日期和时间格式
%d	以十进制数（0~31）表示一个月中的第几天
%m	以十进制数（0~12）表示一年中的月份数
%U	以十进制数显示当前是一年当中的第几周，星期天是第一周的第一天
%W	以十进制数显示当前是一年当中的第几周，星期一是第一周的第一天
%w	以十进制数显示当前是星期几，从 0 开始，0 表示星期天
%x	事先定义的日期格式，不包含时间
%y	以十进制数（00~99）表示的当前世纪的当前年份数
%Y	以十进制数表示的当前年份数，比如 1997

格式符	格 式 符 说 明
%H	以十进制数（00~23）显示当前的时间中的钟点数
%I	以十进制数（01~12）显示当前的时间中的钟点数
%j	以十进制数（001~366）显示当前是一年当中的第几天
%M	以十进制数显示当前的时间中的分钟数
%P	根据当前时间在日期时间格式前加 am 或 pm
%S	以十进制数显示当前的时间中的秒数
%X	事先定义时间的格式，不包含日期
%Z	时区的名称或缩写

使用 SSI 的“cmd”命令可以在 Web 网页中执行一些系统的命令。当在命令行执行这些系统命令时，终端屏幕上一般都有输出。“config”的命令变量为“cmdecho”时，可以用“config”来控制这些命令是否在 Web 网页上显示出来。当命令变量“cmdecho”的取值为“ON”时则显示，取值为“OFF”时则不显示。具体的运用示例如下：

```
<!--#config cmdecho="ON" -->
```

如果还想对系统命令在 Web 网页上的输出进行一些设置，可以使用命令变量“cmdprefix”和“cmdpostfix”。“cmdprefix”用于在系统命令的每一行输出信息前加上一些文本字符串和 HTML 标识符，“cmdpostfix”则用于在每一行输出信息后加上一些文本字符串和 HTML 标识符。

例如在 HTML 文件包含下列 SSI 命令：

```
<!--#config cmdecho="ON" -->
<!--#config cmdprefix="<BR> "-->
<!--#exec cmd="ping 127.0.0.1 -w 20000" -->
```

在 Web 网页上这些 SSI 命令所在的地方将会出现如下的输出信息：

```
Pinging 127.0.0.1 with 32 bytes of data:
Reply from 127.0.0.1: bytes=32 time<10ms TTL=32
Reply from 127.0.0.1: bytes=32 time<10ms TTL=32
Reply from 127.0.0.1: bytes=32 time<10ms TTL=32
Reply from 127.0.0.1: bytes=32 time<10ms TTL=32
```

如果不在每行输出信息前加 HTML 标识符，Web 网页上的 Ping 命令的输出信息则具有如下的形式：

Pinging 127.0.0.1 with 32 bytes of data: Reply from 127.0.0.1: bytes=32 time<10ms TTL=32 Reply from 127.0.0.1: bytes=32 time<10ms TTL=32 Reply from 127.0.0.1: bytes=32 time<10ms TTL=32Reply from 127.0.0.1: bytes=32 time<10ms TTL=32

最后，在如下的 HTML 文件中列举了“config”命令的一些用法，此时该 Web 网页的输出结果如图 2-2 所示。

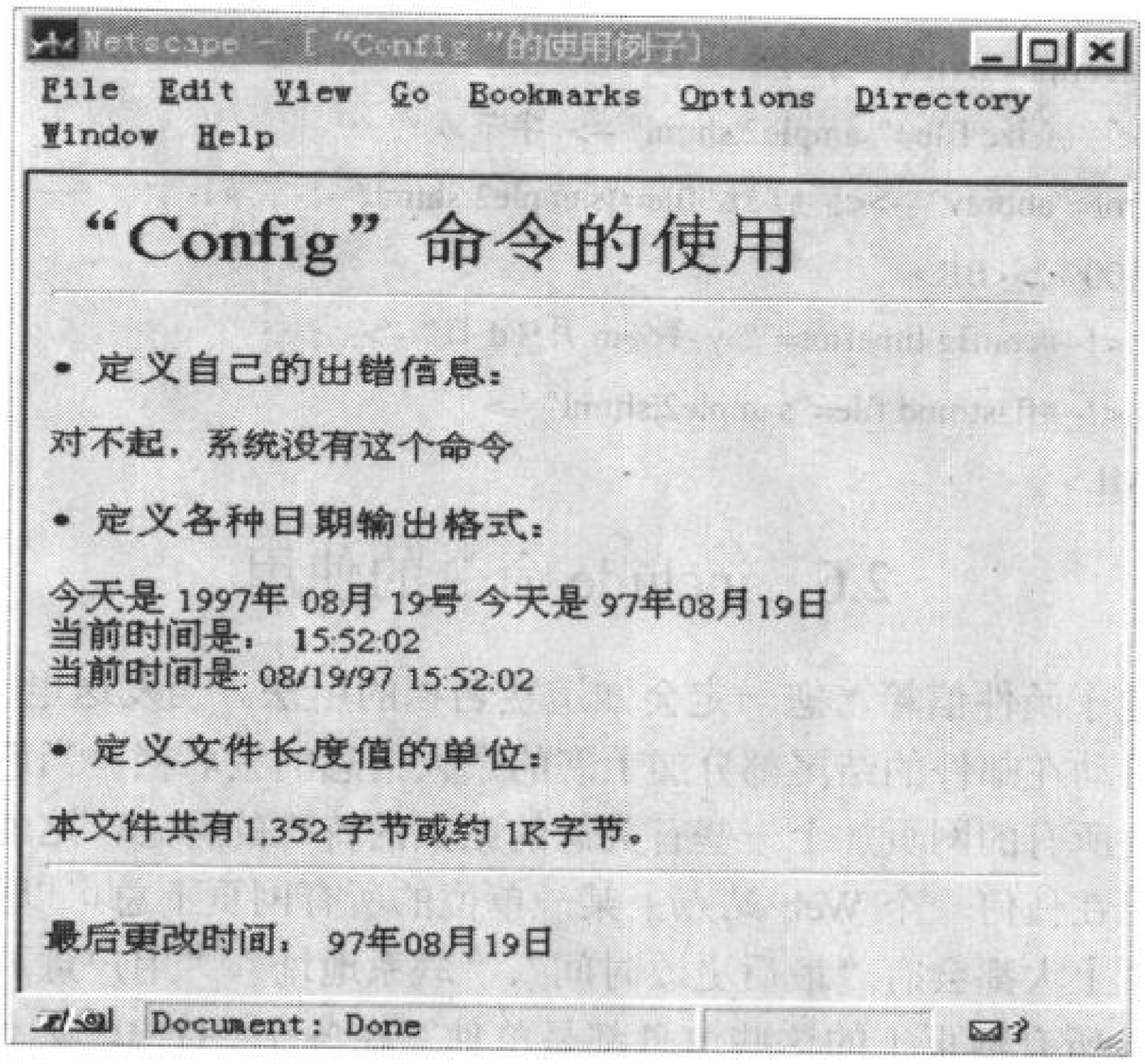

图 2-2　config 命令的使用

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">
<HTML>
<HEAD>
   <TITLE> "Config" 的使用例子</TITLE>
</HEAD>
<BODY>
<P><FONT SIZE=+3> "Config" 命令的使用</FONT>
<HR WIDTH="100%"></P>
<LI><FONT SIZE=+1>定义自己的出错信息: </FONT></LI>
<p><!--#config errmsg="对不起，系统没有这个命令" -->
<!--#exec cmd="ls -l" -->
<P><LI><FONT SIZE=+1>定义各种日期输出格式: </FONT></LI>
<P>今天是 <!--#config timefmt="%Y" --><!--#echo var="DATE_LOCAL" -->年
<!--#config timefmt="%m" --><!--#echo var="DATE_LOCAL" -->月
<!--#config timefmt="%d" --><!--#echo var="DATE_LOCAL" -->号
今天是<!--#config timefmt="%y 年%m 月%d 日" -->
      <!--#echo var="DATE_LOCAL" --><BR>
当前时间是: <!--#config timefmt="%H:%M:%S" -->
            <!--#echo var="DATE_LOCAL" --><BR>
当前时间是: <!--#config timefmt="%c" -->
            <!--#echo var="DATE_LOCAL" --></P>
<LI><FONT SIZE=+1>定义文件长度值的单位: </FONT></LI>
```

```
<P><!--#config sizefmt="bytes" --></P>
<P>本文件共有<!--#fsize file="sample2.shtml" --> 字节或约
<!--#config sizefmt="abbrev" --><!--#fsize file="sample2.shtml" -->字节。
<HR WIDTH="100%"><BR>
最后更改时间: <!--#config timefmt="%y 年%m 月%d 日" -->
              <!--#flastmod file="sample2.shtml" -->
</BODY></HTML>
```

2.6　include 命令的使用

如果您拥有电子函件信箱，您一定会知道签名档的用法。当发送电子函件时，相应的电子函件软件会自动在邮件的结尾部分加上您的个人信息的签名档。当您在制作 Web 网页时，可能也会想给所有的网页加上一些有关您的站点的特殊的标志，这就好像是 Web 网页的签名档。事实上在任何一个 Web 站点上某一单位的所有网页上总可以找到一些共同的东西，例如一般网页上大都会有“最后更改时间”、“联系地址”、“用户意见箱”、“首页地址”等一些条目。如果所有网页上的这些信息都是单独写在各个 HTML 文件中，当某一天需要对这些类似签名档的信息进行修改时，就必须对所有包含这类信息的网页进行修改，工作量之大是显而易见的。如果能把这些共同的东西都放在一个文件中，服务器在向客户端传送 Web 网页时自动包含这个文件，则在对这些类似签名档的信息进行修改时就只需修改这一个文件即可。

SSI 的命令“include”就提供这种功能。例如 HTML 文件中的 SSI 命令：

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

就将当前目录下的文件 filename.html 插入到 HTML 文件中该 SSI 命令所在的地方。如果命令变量为“file”，则命令变量值只能是当前目录或当前目录下子目录中的文件。

“include”命令还有另外一种使用方法，即使用“virtual”命令变量：

```
<!--#include virtual = "/full pathname/filename.html" -->
```

此时 SSI 命令将从 Web 服务器上 HTML 文件目录树的根目录及其子目录下寻找所要插入的文件。

注意不能用 SSI 命令“include”向 HTML 文件中插入 CGI 程序。虽然如此，但可以用它向该 HTML 文件插入另一个包含 SSI 命令的 HTML 文件，当然这个被插入的 HTML 文件可以被用来执行一个或多个 CGI 程序。

2.7　fsize 命令的使用

SSI 命令“fsize”可以用来向 Web 网页中插入某一文件所具有的字节数。这一文件可以是服务器上的其他文件，也可以是本身包含这个 SSI 命令的 HTML 文件。“fsize”命令的使用和“include”命令的使用相类似，都有两个命令变量，并且含义也是相同的：

```
<!--#fsize file = "filename.html" -->
<!--#fsize virtual = "/full pathname/filename.html" -->
```

假设您在某一 Web 网页上放置了一些精美的图片，例如是有关飞机的、动物的或者是

您个人的生活照片。我们知道，图片的精美程度是和字节数成正比的。当一个万维网浏览者在浏览这一网页时，这一网页被下载的速度可能会很慢，甚至会不能忍受，有些浏览者就会因此而放弃浏览您的网页。在这种情况下您就可以使用 SSI 命令“fsize”，您可以为 Web 网页上的精美图片做一些只能够反映概貌的小图片并把它们放在网页上，在这些拥有较少字节数的小图片周围再放置 SSI 命令“fsize”用以指出拥有那些完整细节的图片的字节数，这些小图片可以通过超文本链接指向它们所代表的精美图片。这样一来，这一 Web 网页下载的速度将会得到提高，而网页的浏览者也可以根据每一图片的字节数作出是否下载该图片的决定。

2.8　echo 命令的使用

echo 命令的使用和其他 SSI 命令类似，echo 的命令变量只有一个，即“var”。echo 命令用于向 Web 网页插入一些变量的值。这些变量有两类，一类是环境变量，一类是用 POST 方法激活包含 echo 命令的 HTML 文件的网页上表单元素所存储的变量。我们知道，Web 服务器在执行一个 CGI 程序之前要为它设置一些环境变量，这些环境变量存储了客户端和服务器本身的一些重要信息，Web 服务器在执行包含 SSI 命令的 HTML 文件之前也要为这个 HTML 文件设置同样的环境变量，echo 命令所能显示的环境变量如表 2-5 所列。如果仅仅是为了在 Web 网页上显示一些环境变量的值，这时可以使用 SSI 的 echo 命令而不用进行 CGI 编程。

表 2-5　SSI 命令 echo 的常用环境变量

echo 的常用环境变量	环境变量的说明
DOCUMENT_NAME	当前所用文件名称，包含路径
DOCUMENT_URI	当前所用文件名称，使用相对路径
DATE_LOCAL	当前日期和时间
DATE_GMT	当前日期和时间的格林尼治形式
LAST_MODIFIED	当前所用文件的最后修改日期和时间
REMOTE_ADDR	客户端的 IP 地址
SERVER_SOFTWARE	Web 服务器所用的软件类型和版本号
SERVER_NAME	Web 服务器的 Internet 域名
GATEWAY_INTERFACE	当前 Web 服务器所用的通用网关接口规范的版本号
SERVER_PROTOCOL	当前 Web 服务器所用 HTTP 协议的版本号
SERVER_PORT	服务器用于监听浏览器请求标头的 TCP 端口
REMOTE_HOST	客户端的 Internet 域名
AUTH_TYPE	对客户端用户进行身份验证的方法
REMOTE_USER	用户用来进行身份验证的用户名
CONTENT_TYPE	浏览器传送数据的方法
CONTENT_LENGTH	当浏览器用 POST 方法传送数据时所传送数据的字节数
HTTP_ACCEPT	客户端浏览器所能接受的数据类型

echo 的常用环境变量	环境变量的说明
REFERER	当前浏览器所浏览的上一网页的 URL
FROM	浏览器用户的电子函件地址
FORWARDED	浏览器所使用的 Proxy 服务器的域名
ACCEPT_LANGUGE	浏览器所希望接受的语言的 ICO 代码

echo 命令的使用实例如下：

```
<!--#echo var="REMOTE_ADDR" -->
<!--#echo var="建议" -->
```

第一个例子是用来在 Web 网页中显示环境变量 REMOTE_ADDR 的值，它代表客户端的 IP 地址。第二个例子是用来显示某一表单中变量“建议”所存储的值。

注意，为能用 echo 命令存取表单中的变量，包含 echo 命令的 HTML 文件必须像 CGI 程序一样被该表单用 POST 方法激活。当用 echo 命令所存取的环境变量为空值时，echo 将在 Web 网页上输出字符串“none”。

以下包含 SSI 命令的 HTML 文件可以在 Web 页面上输出有关环境变量的值，该 HTML 文件所生成的 Web 页面如图 2-3 所示，其 HTML 文件的内容如下：

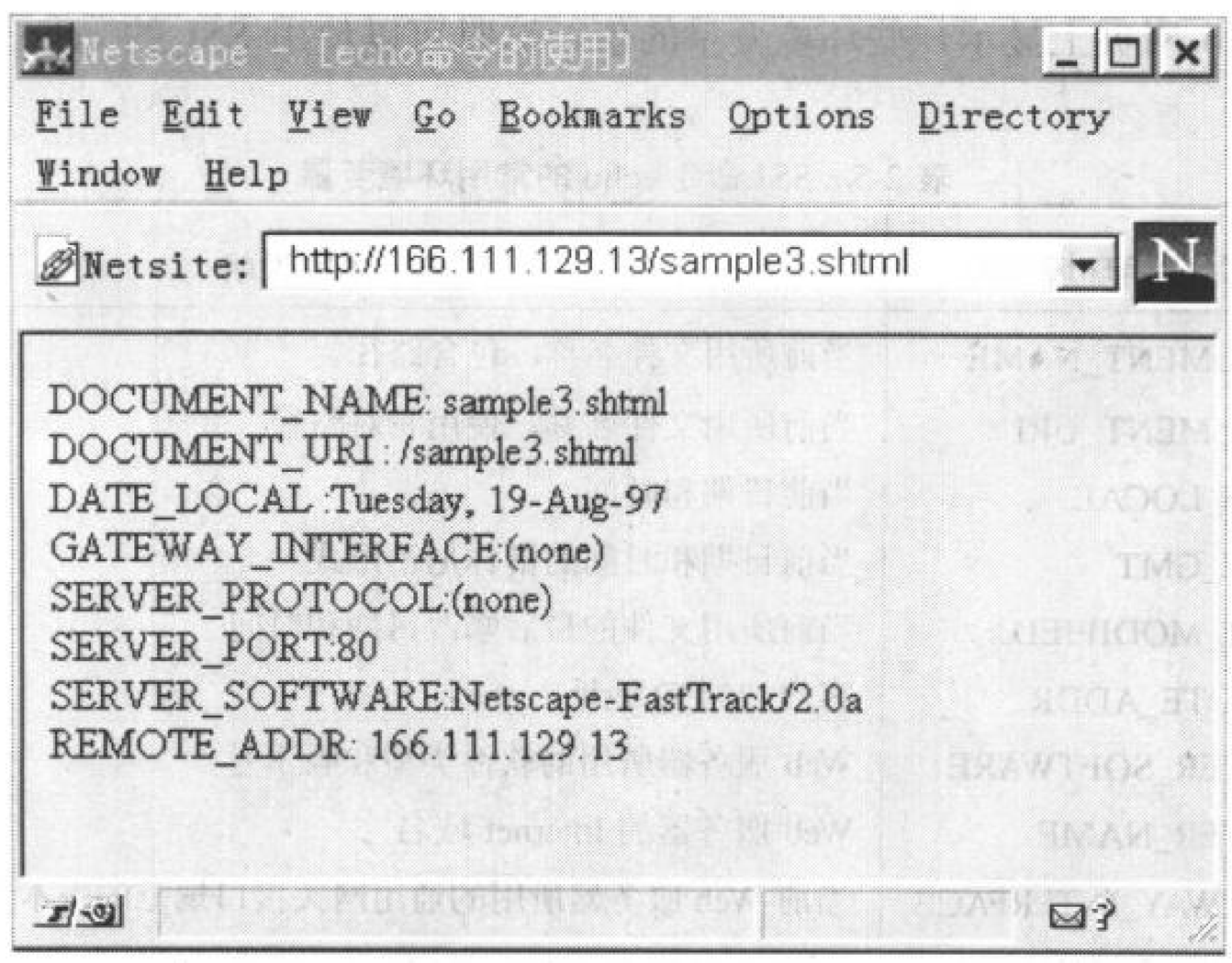

图 2-3　用 echo 命令在 Web 网页上输出环境变量

```
<HTML><HEAD><TITLE>echo 命令的使用</TITLE></HEAD>
<BODY>
<P><P> DOCUMENT_NAME : <!--#echo var="DOCUMENT_NAME" -->
<BR>DOCUMENT_URI :<!--#echo var="DOCUMENT_URI" -->
<BR>DATE_LOCAL : <!--#echo var="DATE_LOCAL" -->
<BR> GATEWAY_INTERFACE : <!--#echo var="GATEWAY_INTERFACE" -->
<BR>SERVER_PROTOCOL : <!--#echo var="SERVER_PROTOCOL" -->
```

```
<BR> SERVER_PORT : <!--#echo var="SERVER_PORT" -->
<BR> SERVER_SOFTWARE :<!--#echo var="SERVER_SOFTWARE" -->
<BR> REMOTE_ADDR : <!--#echo var="REMOTE_ADDR" --></P>
</BODY></HTML>
```

2.9　exec 命令的使用

exec 命令可以在 Web 网页中执行一些系统命令或命令行程序，这些命令的输出最后被放在 HTML 文件中 exec 命令所在的位置。exec 命令也可以在 Web 网页中自动执行 CGI 程序，而通常 CGI 程序是需要万维网上的客户端用户按一下类似“Submit”的按钮才能执行的。由于 SSI 的命令 exec 赋予了网页制作者很多权力，处理不当也有可能给系统造成安全上的问题。因此一些 Web 服务器的管理员虽然将服务器的 SSI 功能打开了，但常常关闭 SSI 的 exec 命令的使用，如图 2-4 所示。

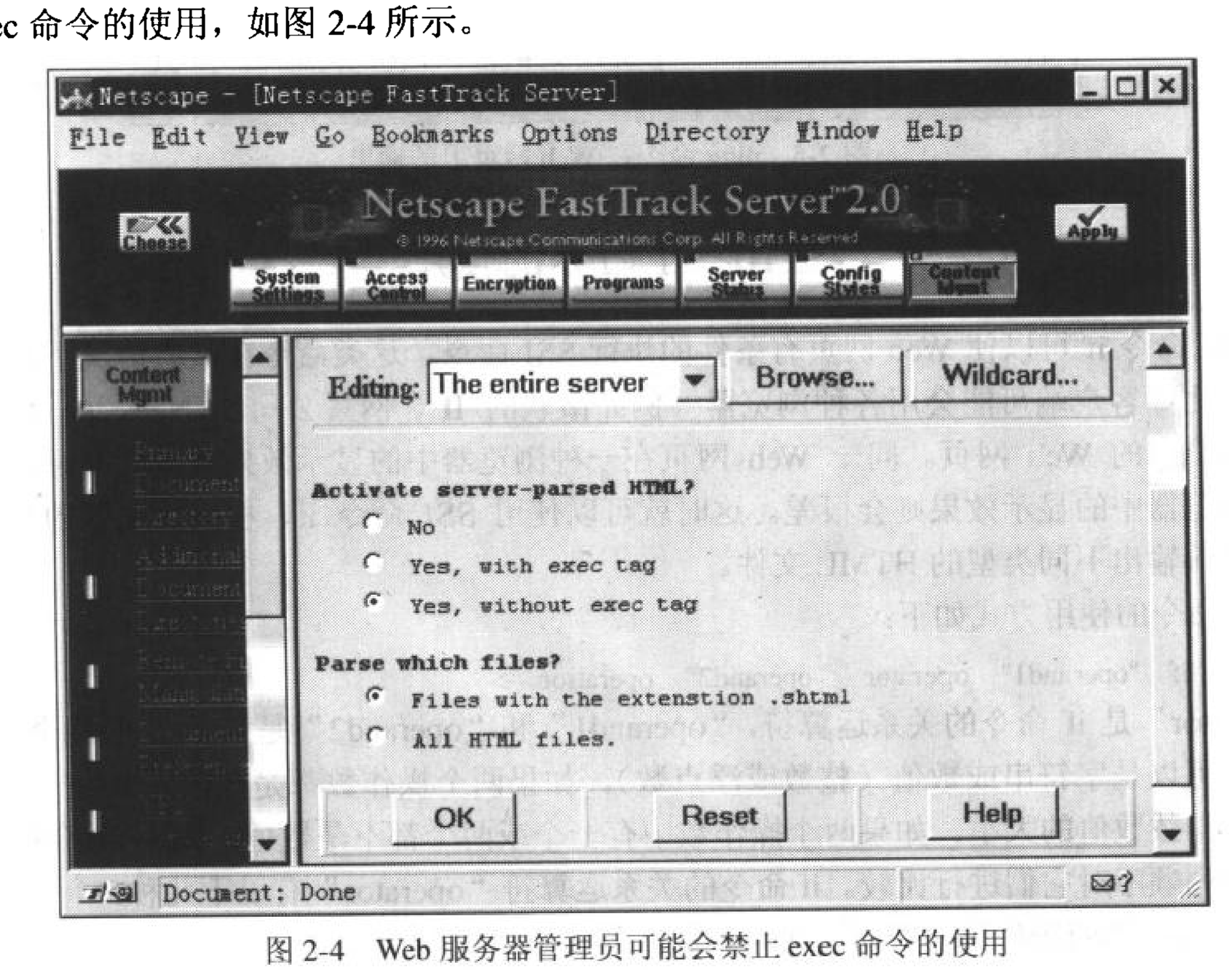

图 2-4　Web 服务器管理员可能会禁止 exec 命令的使用

用 exec 调用系统或外部命令，命令变量要用“cmd”，如果调用的是 CGI 程序，命令变量要用“cgi”，以下是两个分别调用 UNIX SHELL 命令“ls -l”和 CGI 程序的示例：

```
<!--#exec cmd="ls -l" --></P>，  <!--#exec cgi="/cgi-bin/count.cgi" --></P>
```

如果不想让系统、外部命令或 CGI 程序的输出在 Web 网页上显示出来，可以在 exec 命令前使用一个 config 命令进行设置：

```
<!--#config cmdecho="OFF" -->
```

当然也可以用 config 命令的其他功能对系统或外部命令的输出进行一些简单的格式设置，详见 2.4 节 SSI 命令 config 的使用。

图 2-5 显示了 UNIX 系统命令“ping　127.0.0.1”在 Web 网页上的输出内容。

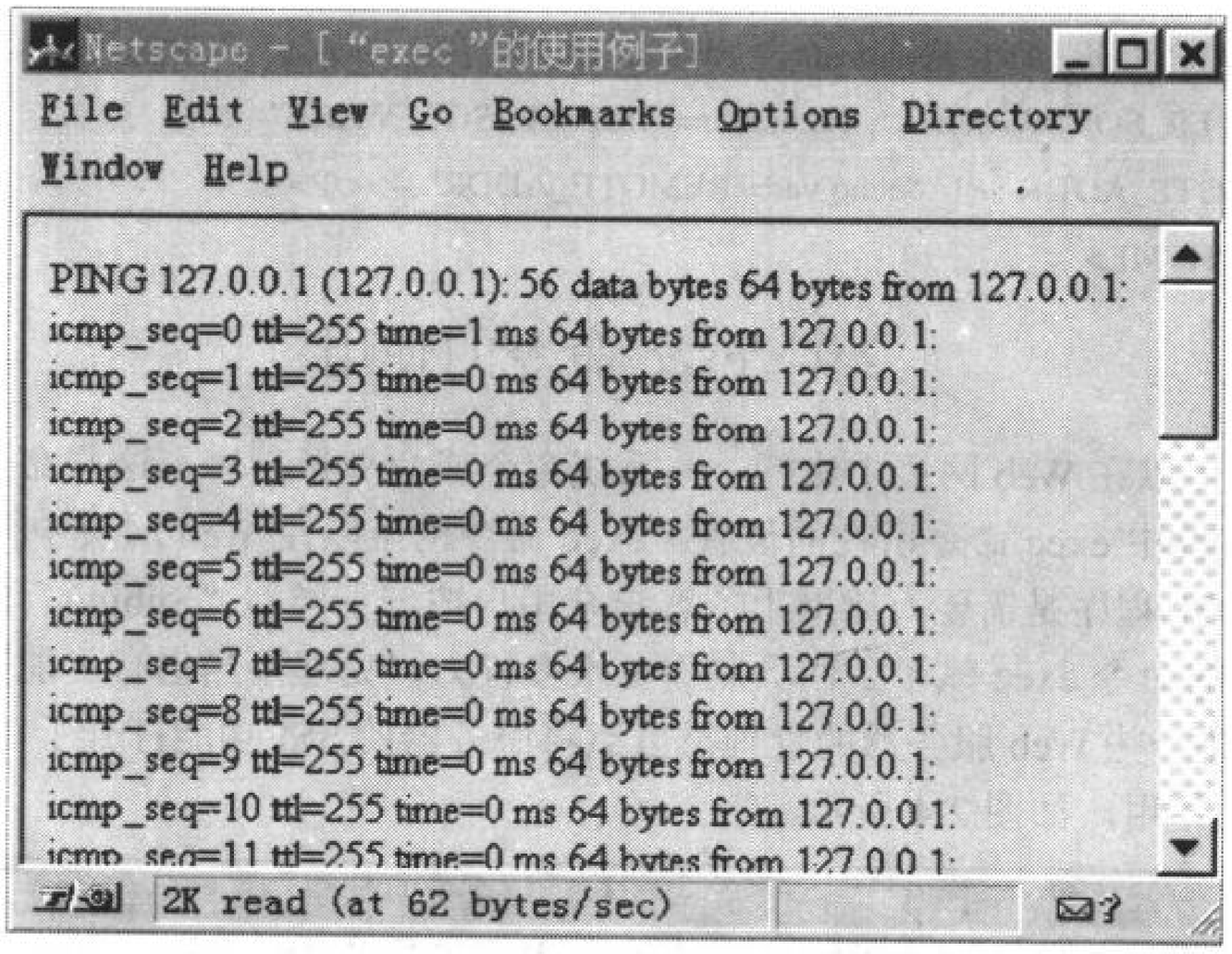

图 2-5　ping 命令在 Web 网页上的输出

2.10　if 命令的使用

SSI 命令 if 可以使 Web 网页有条件的执行 SSI 命令。这类命令在某些情况下是非常有用的。比如客户端可能会用各种浏览器（例如微软的 IE、网景公司的 Netscape Navigator 等）浏览您的 Web 网页。同一 Web 网页在一种浏览器中的显示效果可能会很好，但在另一种浏览器中的显示效果则会很差。这时就可以使用 SSI 命令 if，根据浏览器的不同类型向客户端输出不同类型的 HTML 文件。

if 命令的使用方式如下：

```
<!--#if "operand1" operator "operand2" operation -->
```

“operator”是 if 命令的关系运算符，“operand1”和“operand2”是 if 命令的两个操作数，操作数可以是字符串或数值（整数或浮点数）。如果两个操作数都是数值，则 if 命令比较两个操作数数值的大小。如果两个操作数中有一个或两个都不是数值，则 if 命令根据字符串的排列顺序对它们进行比较。if 命令的关系运算符“operator”有以下几种：

- =：当两操作数相等时，关系表达式的值为“真”。
- !=：当两操作数不相等时，关系表达式的值为“真”。
- <：当操作数 1 小于操作数 2 时，关系表达式的值为“真”。
- >：当操作数 1 大于操作数 2 时，关系表达式的值为“真”。
- !<：当操作数 1 不小于操作数 2 时，关系表达式的值为“真”。
- !>：当操作数 1 不大于操作数 2 时，关系表达式的值为“真”。
- hasstring：当操作数 2 表示的字符串包含在操作数 1 中时，关系表达式的值为“真”。

if 命令中的“operation”是当 if 命令中的关系表达式为真时 if 命令所要执行的操作，主要有以下几种：

- 'goto ' <label >：跳转到 label 所定义的地方，执行相应的 SSI 命令。

- 'print' <text>：在 Web 网页上输出由 text 定义的文本或 HTML 标识符。
- 'error'：输出由 config 命令定义的错误信息。
- 'break'：终止 Web 网页向客户端的输出。
- 'errorbreak'：终止 Web 网页向客户端的输出，并输出由 config 命令定义的错误信息。
- 'printbreak'：终止 Web 网页向客户端的输出，并输出由 text 定义的文本。

下面的示例说明了 if 命令的使用，当关系运算的结果为真时将跳到 Web 网页中由 SSI 命令“label”定义的标号为“testlabel”的地方：

```
<!--#if "A" != "B" goto testlabel --> <p>文本 A
<!--#label ="testlabel " --> <p>文本 B
```

由于一个包含 SSI 命令的 HTML 文件可以被一个表单用 POST 的方法激活，因此可以用 if 命令对表单提供的变量进行比较、判断以决定下一步是否进行 Web 网页的输出或 SSI 命令的执行：

```
<!--#if "&&姓名&&" = "" printbreak  "<P>请填入您的姓名"  -->
```

以下的 SSI 命令可以根据客户端浏览器的不同类型（Netscap Navigator 或 MSIE），输出网页的不同部分，这通常可以使您的 Web 网页在各种浏览器中都具有相同的输出效果：

```
<!--#if "&&HTTP_USER_AGENT&&" hasstring "MSIE" goto msielabel -->
<P>适合于 Netscape Navigator 浏览器浏览的网页
<!--#goto ="defaultlabel" --><!--#label ="mosaiclabel" -->
<P>适合于 MS IE 浏览器浏览的网页
<!--#label ="defaultlabel" -->
```

2.11 email 命令的使用

为了将客户端在 Web 网页上表单中填取的建议信息取出，通常要进行 CGI 编程。这个 CGI 程序处理表单中的信息，并将它们以各种方式比如电子函件的形式传送给需要它们的用户。而利用 SSI 提供的命令 email，您可以不用进行 CGI 编程就实现这个功能。

当一个包含 email 命令的 HTML 文件被浏览或被普通 HTML 件中的表单以 POST 的方法激活时，就会向所定义的用户发送一个电子函件。email 命令的使用和其他 SSI 命令的使用相同。email 命令共有以下几个命令变量：

- fromhost：定义用来发送电子函件的 SMPT 服务器。
- tohost：定义接受电子函件的主机。
- fromaddress：定义发送方的电子函件地址。
- toaddress：定义接受方的电子函件地址。
- message：定义所要传送的邮件内容。
- subject：定义所要传送的邮件内容的主题信息。
- sender：定义 SMPT 服务器管理员的电子函件地址。
- replyto：定义了接受方使用 replyto 命令时所使用的电子函件地址。
- cc：定义了可以接受该电子函件的另外一些电子函件地址。

- inreplyto：定义了发送电子函件时 inreplyto 域所包含的信息。
- id：定义了发送电子函件时 id 域所包含的信息。

以上的命令变量只有 fromhost、tohost、fromaddress 和 toaddress 是必须的，其他都是可选的。比如，Web 页面上有这样一个表单，它利用文本输入框定义了如下一些变量：name、subject、suggestion、email-address 等分别存储客户端用户输入的各种信息。在该表单用 POST 方法激活的 HTML 文件中，可以用如下的 email 命令将以上客户端用户输入的信息传送给电子函件地址为 zzz@mail.yourunit.com 的用户：

```
<!--#email fromhost= "www.albert.com" tohost="mailbox.yourunit.com"
subject="&&subject&&" message="&&suggestion&&"
fromaddress="&&email-address&&"
toaddress="zzz@mail.yourunit.com" -->
```

2.12　odbc 命令的使用

如果您想通过 Web 网页实现对 ODBC 数据库的查询和更新，可以使用 SSI 的命令 odbc。当然，使用 odbc 命令是有条件的，您只能在微软的视窗系统中支持 SSI 命令 odbc 的 Web 服务器产品中使用它。

ODBC 是微软推出的一个有关存取数据库数据的标准，即开放数据库互联（Open DateBase Connect）。通过使用 ODBC，存取数据库数据的外部程序可以不用直接和各种数据库打交道，而是借助于一个 ODBC 管理器去操作它们。这样程序编制者就可以不用了解各类数据库的细节而只需了解 ODBC 的原理。现在支持 ODBC 标准的数据库产品越来越多，将来 UNIX 系统上的数据库产品也可能支持 ODBC 标准。当然，使用 ODBC 需要付出一定的代价，由于在外部程序和数据库之间多了一个环节 ODBC，程序的运行性能将会受到影响。

odbc 命令具有 SSI 命令通用的格式：

```
<!--#obdc　命令变量="命令变量值" -->
```

对于 odbc 命令，有四个命令变量可以使用：debug、connect、statement 以及 format。

命令变量 debug 的使用可以使 Web 服务器的 SSI 引擎进入调试模式。在调试模式下服务器可以对 SSI 引擎和 ODBC 引擎发生的错误进行详细的诊断以使网页制作者找出发生错误的原因。odbc 命令运行时发生的错误将在 Web 网页上该命令所在的位置显示出来。debug 命令变量只能用在 Web 网页的开发阶段使用，当在 Web 服务器上正式发布网页时应该将这条 SSI 命令去掉。在调试模式下 SSI 引擎将在 Web 网页上输出一些警告信息以表明现在正处于调试模式，这类警告信息不会影响 SSI 引擎对其他 Web 网页上 SSI 命令的处理。debug 命令变量的使用格式如下：

```
<!--#obdc　debug = "<debugstring> " -->
```

其中“<debugstring>”可以是任意文本字符串，该文本字符串是留给将来的 SSI 标准使用的。

在对一个数据库进行查询和更新之前，必须和该数据库建立连接。connect 命令变量用来在 Web 网页中和一个 ODBC 管理器中的数据源建立连接，其使用格式如下：

```
<!--#obdc  connect = "<datasource>, <username>, <password >" -->
```

其中“<datasource>”、“<username>”、“<password >”分别是 ODBC 管理器中数据源的名称、数据源的用户以及数据源用户用于建立连接的密码。

命令变量 statement 用于向 ODBC 管理器中的数据源提供一个 SQL 语句以实现对该数据源的查询或更新，其使用格式如下：

```
<!--#obdc  statement = "<SQLstatement >" -->
```

其中“<SQLstatement >”可以是任意的符合 ODBC 标准和 SQL 标准的 SQL 语句。

命令变量 format 用于定义 SSI 引擎向 Web 网页返回的查询数据在页面上的输出格式，其使用格式如下：

```
<!--#obdc  format = "<outputformat >" -->
```

其中“<outputformat >”用来定义输出格式，它的使用和 C 语言的 printf 语句中的格式说明有些类似。“<outputformat >”由任意文本串（包括 HTML 标识符）和“%S”组成，“%S”则代表数据库中的字段。

下面用示例说明 odbc 命令的使用。假设要和 Web 网页建立连接的 ODBC 数据源的名称为 AAA，用户名 zjy，密码是 zjy8，我们希望查询该数据源的“paper”表，并在页面上显示“论文题目”、“第一作者”以及“发表时间”等字段的值，每行显示一个记录。以下是完成这些工作的 SSI 命令：

```
<!--#obdc  connect= "AAA, zjy, zjy8 " -->
<!--#obdc  format = "<P>论文题目: %S, 第一作者: %S, 发表时间: %S " -->
<!--#obdc  statement = "SELECT 论文题目, 第一作者, 发表时间 FROM  paper" -->
```

可以利用客户端的 HTML 表单动态生成 SQL 语句以执行更灵活的查询。当然包含 odbc 命令的 HTML 文件必须被这个表单以 POST 方式激活。如果数据库不支持 ODBC 或您喜欢直接和数据库本身打交道，就要用到 CGI 编程，在本书以后的章节将专门讨论这个问题。

2.13 SSI 命令的安全问题

由于 SSI 命令可以在 Web 网页内执行系统命令或 CGI 程序，许多 Web 服务器管理员出于安全上的考虑常常关闭服务器的 SSI 功能。有些 Web 服务器管理员即使允许 Web 网页执行 SSI 命令也禁止 exec 命令的使用，因为他们认为由于 exec 能够执行系统命令，将给系统带来更多的安全问题。

实际上 SSI 命令的使用所带来的安全问题并不比 CGI 给系统带来的安全问题要多。虽然 SSI 的 exec 命令可以执行部分系统命令，但 CGI 程序同样也可以调用系统命令。如果一个 Web 服务器管理员允许用户在服务器上执行 CGI 程序，那他也就没有必要禁止用户对 SSI 命令的使用。使用 SSI 的一个主要目的就是不用进行 CGI 编程而创作一些简单的动态网页。

只要对 CGI 程序的使用做了正确的设置，SSI 命令的使用通常不会带来安全上的问题。事实上 SSI 命令的使用给服务器带来的一个主要问题是加重了服务器的负担。由于服务器要对包含 SSI 命令的 HTML 文件进行搜索以找出 SSI 命令，服务器的工作量比处理普通 HTML 文件要增加许多。不过这个问题对于万维网上大多数具有较好配置的 Web 服务器并不是很突出。

其实微软在它的 Web 服务器产品 IIS3.0（Internet Information Server）中也推出了一项类似 SSI 的技术 ASP（Active Server Page）。这项技术也就是把一些用 VBscript 或 Javascript 等脚本语言编写的小程序放在后缀名为 ASP 的文件中。服务器在向客户端传送 ASP 文件时，首先要对整个文件进行搜寻以找出这些小程序并执行它们，程序的输出结果被放在 ASP 文件中相应的小程序所在的位置，最后整个网页被传给客户端。微软在万维网上的站点（*www.microsoft.com*）的很多网页都是用 ASP 文件实现的。从这我们也可以看出 SSI 的生命力。

2.14　SSI 命令使用实例

本章对常用的 SSI 命令都做了介绍，最后本节向读者介绍几个在 Web 页面上使用 SSI 命令的简单示例。

2.14.1　给主页添加动态欢迎信息

本例将介绍如何在 Web 站点的主页中加入一些动态的欢迎信息。比如我们希望在站点的首页出现如下的欢迎信息：

晚上好！来自 xxx.xxx.xxx.xxx 的朋友！

欢迎光临！您是本主页的第 xxxxx 个访问者！

其中 xxx.xxx.xxx.xxx 是客户端的 IP 地址，xxxxx 是对本网页的访问统计，实现后的 Web 网页如图 2-6 所示。

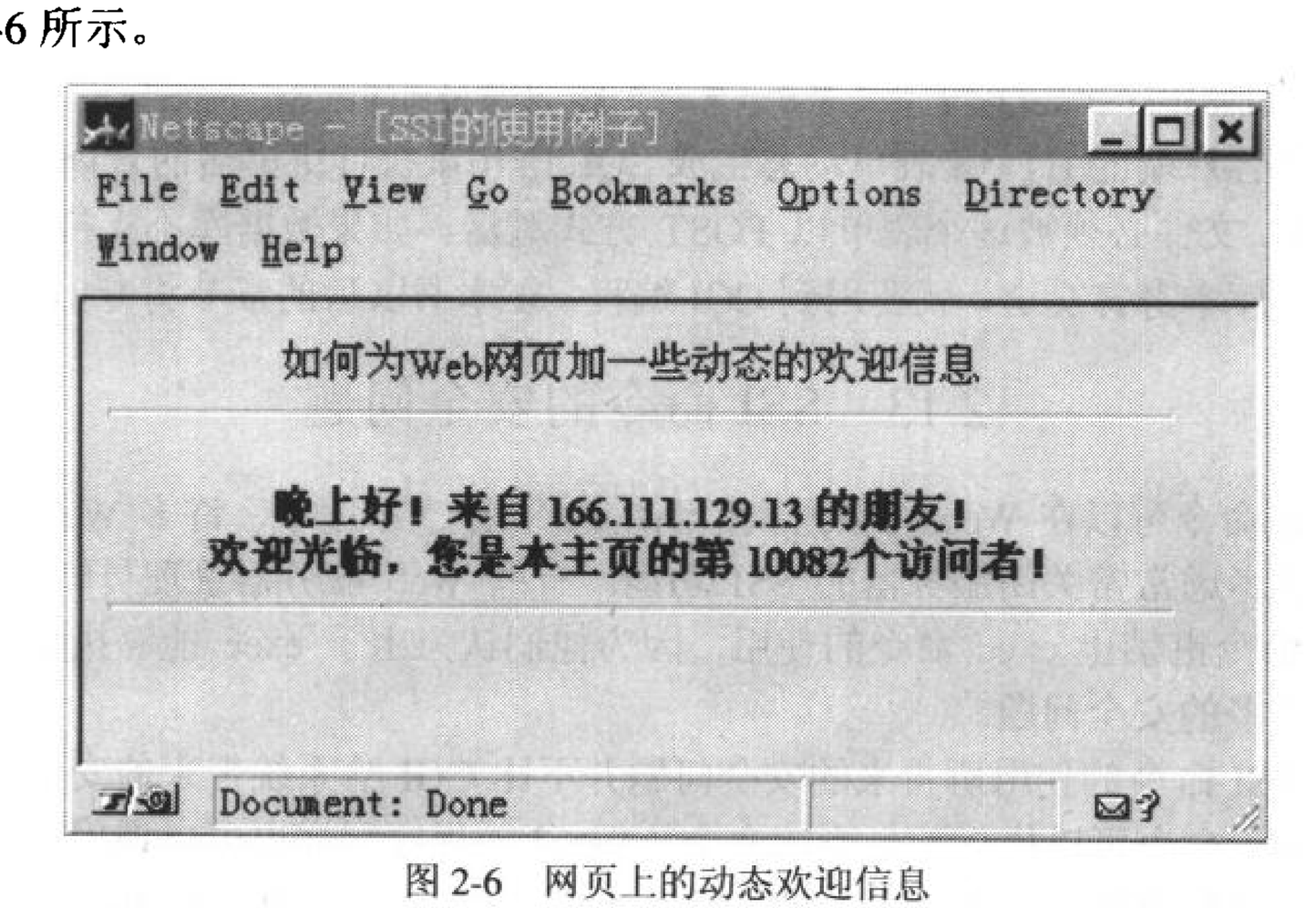

图 2-6　网页上的动态欢迎信息

用 SSI 来实现这段动态信息是非常方便的。可以用 SSI 的命令 echo 显示环境变量 REMOTE_HOST 的方式在 Web 页面上输出客户端的 IP 地址。目前大多数网页都是利用 CGI 程序动态生成 GIF 图像的方式显示对网页的访问的人次，而在这里本例则用了一个更简单的方法，首先用任何语言（本例是 C 语言）编写一个小程序（外部命令），这个程序每次执行时从某一文件中读取一个整数，把这个整数加一后继续存在同一文件中并将它的新值在屏幕上输出。用 SSI 的 exec 命令在 Web 页面中执行这个程序，就可以实现对该 Web 页面的访问计数。在实现这段动态信息时，本例还运用了一个 Javascript 小程序，该小程序

可以根据客户端的当地时间动态显示“早上好”、“中午好”以及“晚上好”的问候语。

实现图 2-6 所示的 Web 网页的 HTML 源代码如下：

```
<HTML><HEAD>
  <TITLE>SSI 的使用例子</TITLE>
  <META NAME="GENERATOR" CONTENT="Mozilla/3.01Gold (Win95; I) [Netscape]">
</HEAD><BODY>
<P><CENTER>如何为 Web 网页加一些动态的欢迎信息
<HR><P><B>
<SCRIPT Language="JavaScript"><!--
document.write(" <CENTER>")
day = new Date()
hr = day.getHours()
if ((hr > 0) && (hr < 4 ) )
document.write("晚上好！ ")
if ((hr>=4) && (hr<8))
document.write("早上好！ ")
if ((hr>=8) && (hr<11))
document.write("上午好！ ")
if ((hr >=11)&&(hr<14))
document.write("中午好！ ")
if ((hr >=14)&&(hr<18))
document.write("下午好！ ")
if ((hr>=18) && (hr<24))
document.write("晚上好！ ")
document.write("来自 ")
//--></SCRIPT>
<!--#echo var="REMOTE_HOST" --> 的朋友！
<BR>欢迎光临，您是本主页的第<!--#config cmdecho="ON" -->
<!--#exec cmd= "count.exe count.dat " -->个访问者！ </CENTER>
<HR>
</BODY></HTML>
```

对 Web 页面的访问进行计数的外部程序的 C 语言源代码则如下所示：

```
/* count.c */
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/file.h>
int main(int argc, char * argv[ ])
{
```

```
    FILE * counter;
    int count;
    int fln;
    fln = open(argv[1], O_RDONLY);
    flock(fln, LOCK_EX);
    counter = fopen(fln, "r");
    if (counter == NULL)
      {
        printf("无法打开计数器文件\"%s\"\n",argv[1]);
        exit(0);
      }
    fscanf(counter, "%d", &count);
    fclose(counter);
    count = count + 1;
    printf("%d", count);
    fln = open(argv[1], O_WRONLY);
    flock(fln, LOCK_EX);
    counter = fopen(fln, "w");
    if (counter == NULL)
      {
        printf("无法打开计数器文件 \"%s\" ", argv[1]);
        exit(0);
      }
    fprintf(counter, "%d\n", &count);
    fclose(counter);
    exit(0);
  }
```

由于 count.exe 可能会被多个用户同时访问，在对计数器文件 count.dat 进行读写时应该对文件加锁以防止其他用户同时对它进行读写。本程序可以在 UNIX 环境下运行，如果要使本程序在 DOS 或 Windows 95 下运行，本程序的加锁部分需要修改。

2.14.2　对将来某一日期进行倒记时

在 Web 页面上对某一日期（如元旦、农历新年等）进行以一天为单位的倒计时可以给您的页面增加动感和趣味，有时还会给人留下深刻的印象。以下是可以对澳门回归祖国的日期进行倒计时的 C 程序语言源码：

```
/* leftdays.C */
#include <stdio.h>
#include <time.h>
void main()
```

```
{
 struct tm   today;
 time_t   now;
 long   days;
 time(&now);
 today = *localtime(&now);
 mktime(&today);
 if ((today.tm_mon == 11) && (today.tm_mday==20)&&(today.tm_year==99))
 {
  printf("<BR><FONT COLOR=\"#8080FF\">热烈庆祝澳门回到祖国怀抱！ ");
  printf("<\/FONT>/n");
 }
  else {
        days =0;
        while ( (today.tm_mon   != 11)||
              (today.tm_mday !=20 )||(today.tm_year!=99) )
         {
              days++;
              today.tm_mday = today.tm_mday + 1;
              mktime(&today);
         }
        printf("<CENTER>");
        printf("<BR><FONT COLOR=\"#8080FF\" SIZE=\"+2\">");
        printf("<BR>距离中华人民共和国对");
        printf("<BR>澳门特别行政区恢复行使主权");
        printf("<BR>还有 <B><FONT SIZE=\"+3\" COLOR=\"ff0000\">");
        printf("%i<\/FONT><\/B> 天！ ",days);
        printf("<\/FONT>");
        printf("<\/CENTER>");
    }
 fflush(stdout);
 return;
}
```

在 HTML 文件中使用如下的 SSI 命令，就可以在 Web 网页上对澳门回归祖国的日期进行倒计时了。该 HTML 文件所产生的 Web 页面如图 2-7 所示。

```
<html><head>
<title>用 SSI 对澳门回归进行倒计时</title></head><body>
<P><center><h2><!--#config cmdecho="ON" --> <!--#exec cgi="/cgi-bin/dj.exe " -->
</h2></center>
```

```
</body></html>
```

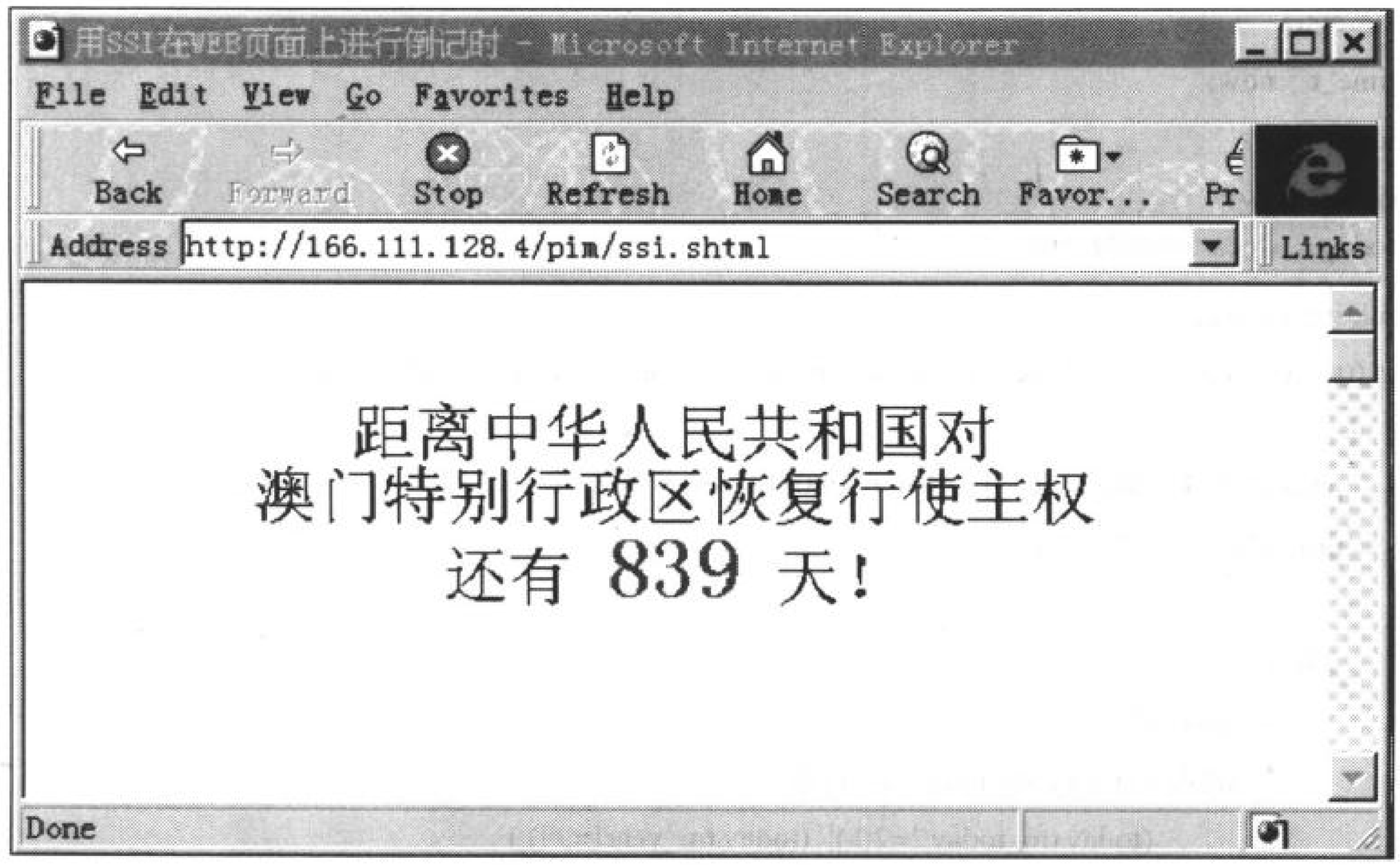

图 2-7　Web 页面上的倒计时

2.14.3　给 Web 网页添加有关最后访问者的信息

如果经常使用 TELNET 软件或光顾 BBS 站点，您常常可以看到有关最后登陆者的信息，事实上利用 SSI 也可以为 Web 网页加上一些有关网页最后访问者的信息。以下 C 语言程序可以将最近一次访问 Web 网页的客户的 IP 地址和时间存入一个文件，并将文件上一次记录的信息在标准输出中显示出来。

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
int main(void)
{
    FILE *in, *out;
    char ltime[30],host[30];
    char *ntime,*nhost;
    char year[10],month[10],day[10],hour[10],min[10];
    struct tm    today;
    time_t    now;
    int i;
    time(&now);
    today = *localtime(&now);
    mktime(&today);
```

```
if ((in = fopen("/usr/people/pim/public_html/lasthost.dat", "rt")) == NULL)
 {
     fprintf(stderr, "Cannot open input file.\n");
     return 1;
 }
 i=0;
 while ((host[i]=fgetc(in))!=' ')
 {
   i++;
 }
 host[i]='\0';
 i=0;
 while (!feof(in))
  {
     ltime[i]=fgetc(in);
     i++;
   }
 ltime[i-1]='\0';
 printf("对本网页的最近一次浏览\n");
 printf("<p>\n");
 printf("来自：  %s   <p>访问时间：   %s", host, ltime);
 fclose(in);
 if((nhost = getenv("REMOTE_HOST"))==NULL)
    {
    nhost = getenv("REMOTE_ADDR");
    }
 if ((out = fopen("/usr/people/pim/public_html/lasthost.dat", "wt"))== NULL)
    {
          fprintf(stderr, "Cannot open output file.\n");
          return 1;
    }
    fprintf(out,"%s %d 年%d 月%d 日%d 时%d 分\n",nhost,today.tm_year+1900,
          today.tm_mon+1,today.tm_mday,today.tm_hour,today.tm_min);
          fclose(out);
          return 0;
}
```

由于程序只是使用了数据文件 lasthost.dat 记录最后浏览者的信息，没有必要在进行读写操作时对文件 lasthost.dat 加锁。在 HTML 文件中使用如下语句，即可得到图 2-8 所示的 Web 页面。

```
<html><head>
<title>在页面上显示最近一次访问者</title>
</head><body>
<P><center><!--#config cmdecho="ON" -->
<!--#exec cmd="/usr/people/pim/public_html/lasthost.exe" -->
</center> </bordy></html>
```

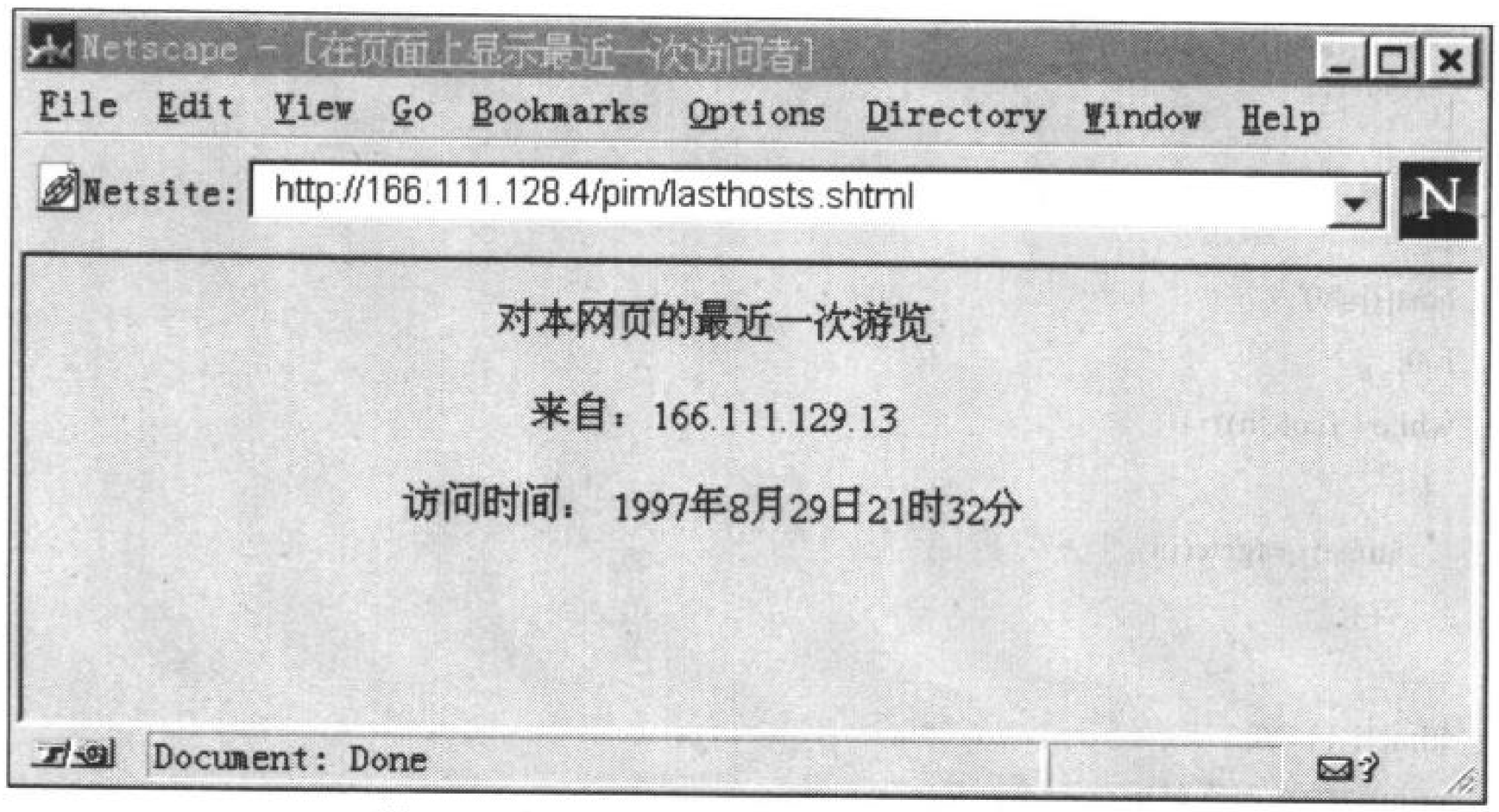

图 2-8　在 Web 页面上显示最后访问者的信息

一般来讲，使用 SSI 的一个主要原因就是因为它可以不用进行 CGI 编程就可以制作出一些简单的动态网页。如果为了实现某一功能而在网页上使用了大量的很复杂的 SSI 命令，本书还是建议您使用 CGI 编程，因为大量使用 SSI 命令的网页维护起来非常不方便。

第三章　CGI 编程的基本原理

您现在可能已经浏览过很多 Web 网页，也许您已经开始着手建立自己的网页。除了文本，您可以为网页加入声音、视频和图片。但如果一个网页仅仅拥有这些，那它只能是一个 WWW 上的电子读物，它还没有充分发挥 Web 系统的潜力。使用功能比 SSI 更强大的 CGI 程序，将使您的 Web 网页具有动态特性和交互式能力。从本章开始，本书将向读者介绍 CGI 编程的基本知识。考虑到目前大部分程序员都使用 C 语言和 Visual Basic 语言编程，本章和本书以后各章所列举的 CGI 程序，都尽量同时利用这两种语言实现。本章主要讨论的内容有：

- 超文本描述语言 HTML 的表单（Form）；
- MIME 类型；
- 如何利用表单的各种元素收集数据；
- 如何将表单收集的数据传给 CGI 程序；
- URI 译码（encoding）和解码（decoding）；
- CGI 程序对表单数据的处理。

3.1　HTML 表单（Form）

超文本描述语言(HTML)表单是 WWW 从客户端向 Web 服务器上的 CGI 程序传送数据的基本方法。编写了 CGI 程序以后，您通常还要为您的 CGI 程序编制它的用户界面，也就是通常所说的 HTML 表单，事实上表单的制作一般要先于 CGI 程序完成。既然 HTML 表单被当成 CGI 程序的界面，它的设计就应满足一般的人机界面原则。

由于 HTML 表单常常和 CGI 程序联系在一起，本书将同时介绍 HTML 表单的制作和 CGI 程序的编制。首先本节将列举一个最简单的表单和与之相联的一个最简单的 CGI 程序。生成如图 3-1 所示的只包含一个 Submit 按钮的 Web 页面的 HTML 源码如下：

```
<html> <head>
<title> 最简单的表单 </title>
</head>
<body>
<h3>一个仅包含 Submit 按钮的表单 </h3>
<FORM Method="GET" Action="/cgi-bin/cgi1.exe">
<input type="submit" >
</FORM>
<hr noshade>
<h1> The HTML required for this FORM </h1>
<table border = 10><td>
<xmp>
<FORM Method="GET" Action="/cgi-bin/cgi1.exe">
```

```
<input type="submit">
</FORM>
</xmp>
<tr>
</table>
</body> </html>
```

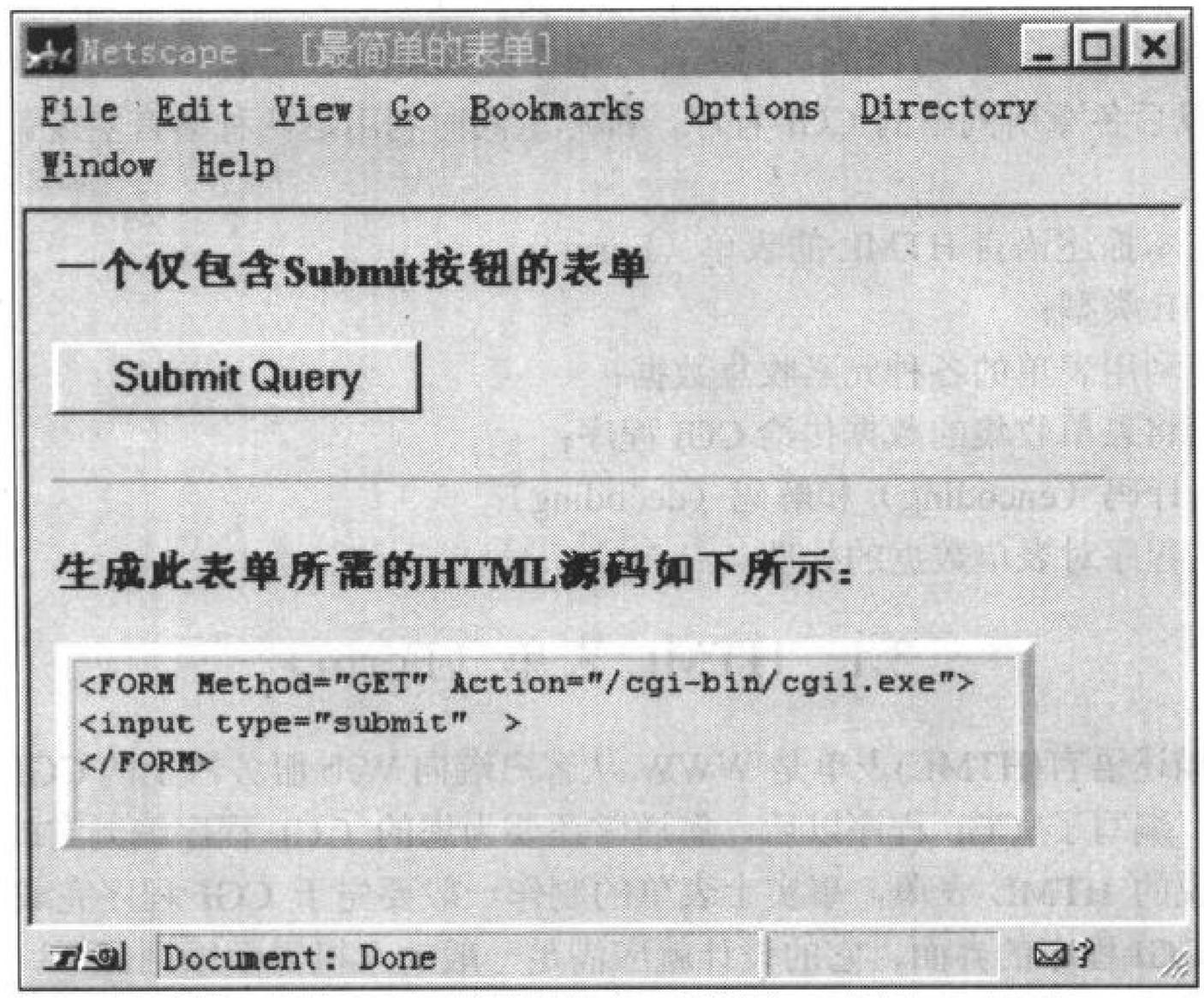

图 3-1　最简单的 HTML 表单

HTML 语句不区分大小写，但在“<”符号后不能有空格，否则在 Web 页面上将会有非常奇怪的输出结果。

这个简单的表单没有传递任何数据给 CGI 程序，在这里它所起的作用就好像是用一个 HTML 的超级链接或 SSI 命令激活 CGI 程序。在这个 Web 页面的 HTML 源码中，“<FORM Method="GET" Action="/cgi-bin/cgi1.exe">”定义了该表单将用 GET 方法激活 Web 服务器上/CGI-BIN/子目录中的 CGI 程序 CGI1.EXE。一般，HTML 的 FORM 标识符具有如下的使用形式：

```
<FORM METHOD= "GET" | "POST" ACTION="URL"
        ENCTYPE= "application/x-www-form-urlencoded">
```

可以用这条 HTML 语句定义表单的三个属性：METHOD、ACTION 和 ENCTYPE。

METHOD 属性定义了用来把表单中的数据传送给 ACTION 所定义的 URL 的方法。目前能使用的有两种：GET 和 POST。缺省是 GET，但现在 POST 基本上是表单的首选方法。GET 方法将数据放在请求头信息中 URL 的后面传给服务器，服务器再将它放在环境变量 QUERY_STRING 中。POST 方法则将数据单独作为一个整体放在请求头信息的后面传给服务器，服务器再将它放在 CGI 程序的标准输入中。

ACTION 属性定义了服务器上用来处理表单数据的外部程序的 URL，通常这个 URL 可以是一个 CGI 程序或包含 SSI 命令的 HTML 文件。即使表单不包含任何数据，ACTION 所定义的 URL 也可以被表单激活。如果没有定义 URL，被表单所激活的 URL 就是包含该表单的 HTML 文件的 URL。

ENCTYPE 属性定义了用来对表单数据进行译码的 MIME 类型。缺省值是 application/x-www-form-urlencoded。由于目前只有缺省值是可用的，在定义 FORM 的 HTML 语句中没有必要将它列出。

</FORM>表示 Web 页面上一个表单的结束，<FORM>和</FORM>必须成对使用，在它们中间可以加入表单的各种元素，一个 Web 页面可以拥有多个表单。

注意，用来定义 CGI 程序的 URL 后面还可以包含路径信息（Path Information），可以利用路径信息向 CGI 程序提供一些变量信息。例如一个 Web 页面有很多表单，每个表单都使用相同的 CGI 程序，但您希望每个表单处理服务器上不同的数据库。这时您可以利用 URL 中的路径信息，通过它告诉 CGI 程序数据库所在的位置。以下是一个运用实例：

```
<FORM Method="POST" Action="/cgi-bin/database.cgi/database1/">
```

3.2　第一个 CGI 程序

本书第一章介绍了 CGI 程序和客户端用户交流信息的基本方式。本节将为图 3-1 所示的表单制作一个简单的 CGI 程序，该程序被表单激活以后将动态产生一个新的网页，并在该 Web 网页上输出一些简单的信息。由该 CGI 程序产生的 Web 网页如图 3-2 所示，程序的 C 语言源码如下所示：

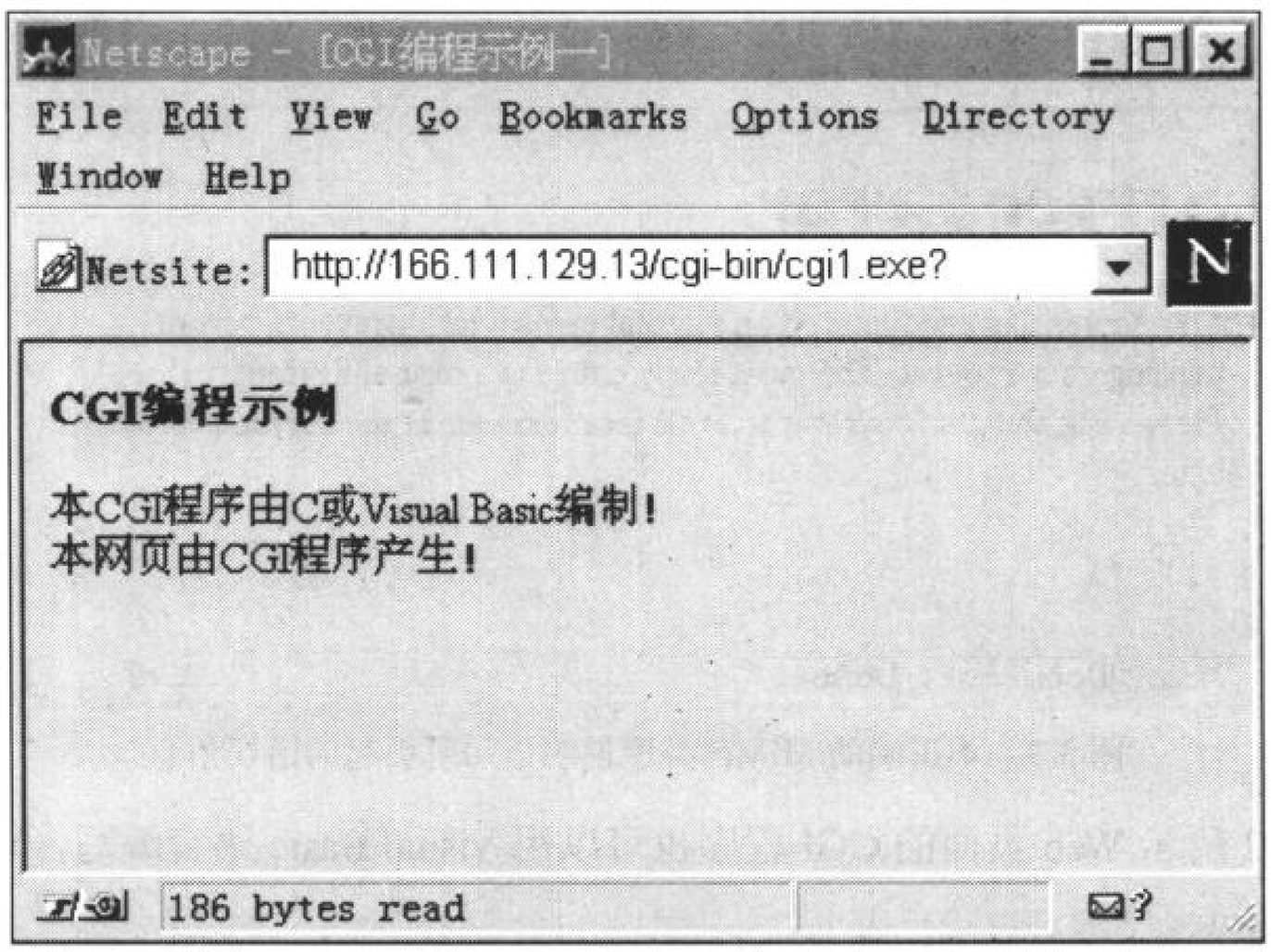

图 3-2　CGI 程序所产生的 Web 页面

```
#include <stdio.h>
main( )
{
```

```
char mimeType[ ]="text/html";
fprintf(stdout, "Content-type: %s%c%c", mimeType, 10, 10);
fprintf(stdout, "<HTML><HEAD>\n");
fprintf(stdout, "<TITLE>CGI 编程示例一</TITLE></HEAD>\n");
fprintf(stdout, "<BODY><H3>CGI 编程示例</H3>\n");
fprintf(stdout, "<P>本 CGI 程序由 C 或 Visual Basic 编制！ \n");
fprintf(stdout, "<BR>本网页由 CGI 程序产生！ \n");
return 0;
}
```

注意 CGI 程序所产生的任何信息都应该向标准输出（STDOUT）中输出，以上源码中的第四行用来向 Web 浏览器说明 CGI 程序向客户端输出信息的 MIME 类型（比如是 HTML 文件还是一个 GIF 文件），以便于浏览器对这些信息作出正确的反应，字符串“Content-type: text/html\n\n”表示该 CGI 程序所输出的是一个 HTML 文件，在该字符串中两个回车符“\n\n”缺一不可，否则 CGI 程序就不会被执行，Web 网页上将会出现图 3-3 所示的信息。任何 CGI 程序在向客户端传送任何信息之前都必须用类似第四行的语句说明信息的类型。第五到第十行则用来向 Web 网页输出一些 HTML 语句。本程序的源码编译以后可以在微机和 UNIX 主机上运行。

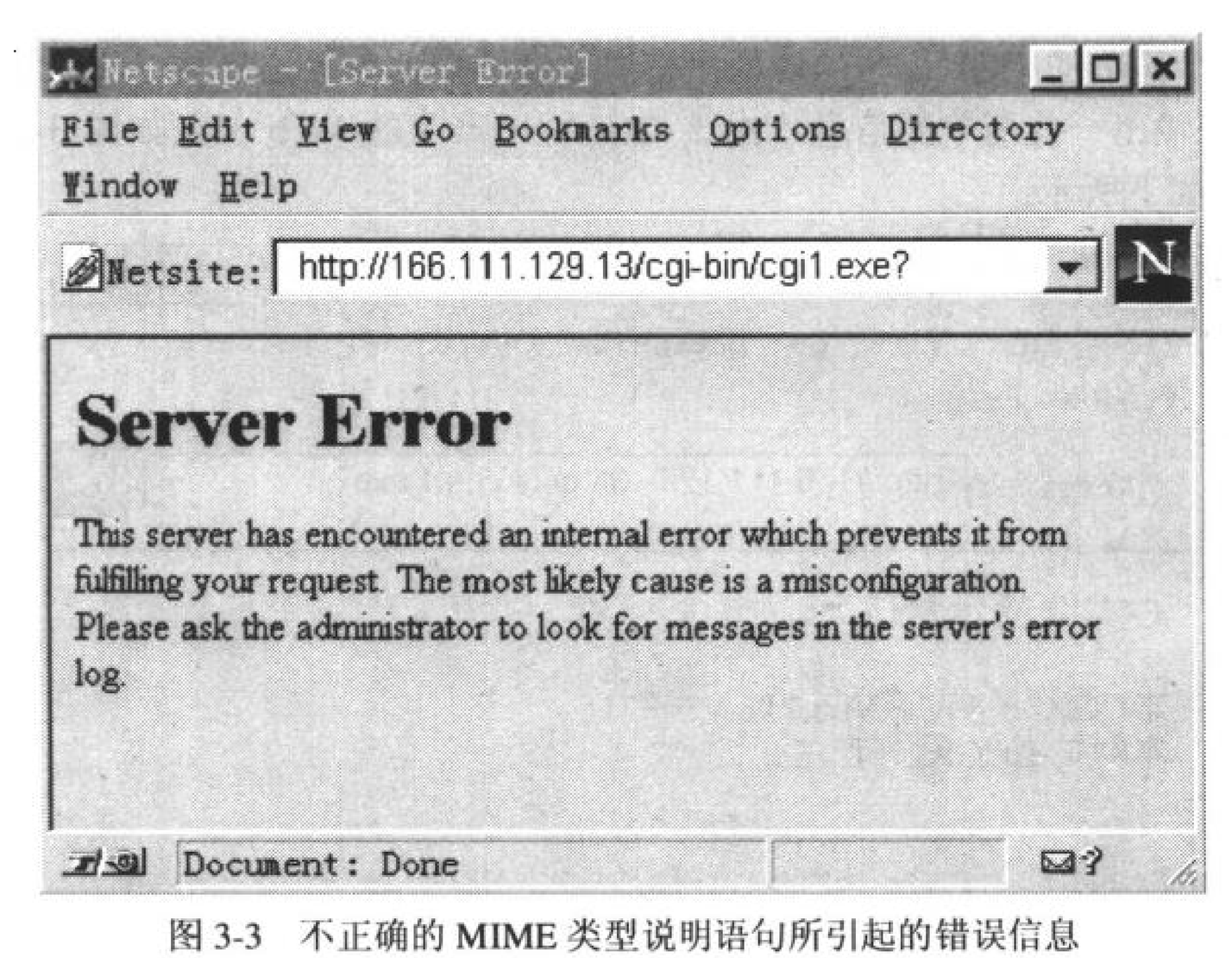

图 3-3　不正确的 MIME 类型说明语句所引起的错误信息

产生图 3-2 所示 Web 页面的 CGI 程序也可以用 Visual Basic 语言编写，VB 源码如下：

```
Option Explicit
' 声明程序中要用到的两个 Windows API 函数，一个用于处理标准输入输出文件句柄，一个用于向文件中写信息
Declare Function GetStdHandle Lib "kernel32"  (ByVal nStdHandle As Long) As Long
Declare Function WriteFile Lib "kernel32"  (ByVal hFile As Long, ByVal lpBuffer As String,
```

```
        ByVal nNumberOfBytesToWrite As Long, lpNumberOfBytesWritten As Long, lpOverlapped As Any) As
Long
    ' 定义标准输入输出文件句柄
    Public Const STD_OUTPUT_HANDLE = -11&
    Public hStdOut   As Long    '  标准输出文件句柄
    Sub Main( )
    ' 调用系统函数获取标准输入输出文件句柄
    hStdIn = GetStdHandle(STD_INPUT_HANDLE)
    hStdOut = GetStdHandle(STD_OUTPUT_HANDLE)
    ' 调用 OutPut 函数向标准输出 STDOUT 中输出有关信息
    OutPut "Content-type: text/html" & vbCrLf
    ' vbCrLf 表示换行符，类似 C 语言中的 "\n"
    ' 以上 OutPut 语句用来说明 CGI 程序所要传送信息的 MIME 类型
    OutPut "<HTML><HEAD>"
    OutPut "<TITLE>CGI 编程示例 </TITLE></HEAD>"
    OutPut "<BODY><H3>CGI 编程示例</H3>"
    OutPut "<P>本 CGI 程序由 C 或 Visual Basic 编制！ "
    OutPut "<BR>本网页由 CGI 程序产生！ "
    OutPut "</BODY></HTML>"
    End Sub
    Sub OutPut(s As String)
    '-----------------------------------------
    ' 定义一个向标准输出写信息的函数
    '-----------------------------------------
    Dim lBytesWritten As Long
    s = s & vbCrLf
    WriteFile hStdOut, s, Len(s), lBytesWritten, ByVal 0&
    End Sub
```

由于只有 WIN32 系统才提供处理标准输入输出的系统函数，本程序在 Windows 3.1 下不能运行，在 Windows 3.1 编写 CGI 程序要用 WIN-CGI 规范，本书将在以后的章节详细讨论这个问题。同样，用 VB 编写的 CGI 程序中用来表示 MIME 类型的字符串"Content-type: text/html" 必须跟有两个回车符。

非常遗憾，Visual Basic 在向 Web 网页输出中文信息或处理表单返回的中文信息时会有很多问题，比如中文信息丢失、不能对表单传来的中文信息正确解码（decoding）等等，本书的第七章将讨论如何处理这个问题。C 语言则不存在这个问题。

从本节完成同样功能的两个 CGI 程序的源码我们可以看出，这两个 CGI 程序主要完成以下的工作：

- 向浏览器说明所要传送信息的类型；
- 向浏览器传送 HTML 语句。

3.3 MIME 类型

本章的 3.2 节部分经常提到“MIME 类型”这个词，理解“MIME 类型”对于编写各种类型的 CGI 应用程序是非常重要的。

MIME（Multipurpose Internet Mail Extensions）实际上是一个技术规范，它定义了七种在国际互联网 Internet 上传送的基本数据类型：Text、Multipart、Message、Application、Image、Audio 和 Video，每种基本类型又可以分成几种子类型。比如 Text 可以分成 html、Text、Postscript 和 plain Text，Image 可以分成 gif 和 jpeg 等。Internet 文档 RFC1341、RFC1521 和 RFC1522 对此作了详细的说明。最初，MIME 被用于电子函件系统，它使得基于 Internet 的电子函件系统除了文本信息之外还可以传送和接收各种非文本信息，例如声音、图片、视频等。在 MIME 没有出现以前，如果要向对方发送一个执行文件，您首先必须用一个软件将这个执行文件转换成一个 ASCII 文件，接收方收到这个 ASCII 文件后，再用相应的软件将它还原成二进制的执行文件。有了 MIME 后，您就可以使用“attachment”直接将这个执行文件发送给对方。

由于 MIME 使用的方便性和灵活性，它最终被万维网系统采纳用来定义 Web 服务器传送的信息类型和 Web 浏览器可以接收的信息类型，目前 MIME 已成为 Web 服务器和 Web 浏览器互相沟通的一种重要机制。

当您准备浏览某一 Web 站点的网页时，在您的 Web 浏览器发给 Web 服务器的请求信息中将包含该 Web 浏览器所能处理的 MIME 类型。同样，在 Web 服务器发送给 Web 浏览器的信息中，也必须首先说明该信息的 MIME 类型，以便于浏览器对接收到的信息作出相应的处理。例如，若 Web 服务器传送的是一 Web 网页，在服务器向 Web 浏览器发送的头信息中，必须包含如下的字符串：

```
Content-type: text/html
```

如果传送的是一 GIF 图片，则应包含如下的字符串：

```
Content-type: image/gif
Content-transfer-encoding: BINARY
```

当客户端的请求是一个 CGI 程序时，Web 服务器并不知道 CGI 程序输出信息的 MIME 类型，CGI 程序在向客户端输出有关信息之前必须告诉 Web 服务器其所输出信息的 MIME 类型，再由 Web 服务器转告给 Web 浏览器。这也就是上节 CGI 程序中必须向标准输出写入 MIME 类型的原因。附录 A 列出了常见的 MIME 类型。

3.4 表单（Form）的制作

通常，在开始编写 CGI 程序前都要为它制作一个界面，即在 Web 网页上制作一个将用于为相应 CGI 程序收集数据的表单（Form）。HTML 语言为表单的制作提供了如下的元素：

- TEXTAREA：在表单内定义客户端可以输入多行信息的文本框。
- SELECT：在表单内定义一个弹出式菜单或滚动框，客户端可以从中选取符合自己要求的可选项。

● INPUT：定义其他表单中其他各种可接受客户端输入信息的表单元素。

通过对这些元素的组合使用，Web 网页的开发者可以很容易制作出丰富多彩的表单。

3.4.1 TEXTAREA

通过<TEXTAREA>，可以为表单加入一个客户端可以输入多行信息的文本框。缺省时这个文本框有四行高，40 个字符的宽度，可以用<TEXTAREA>的属性 ROWS 和 COLS 改变文本框的大小。如果想让文本框在客户端没有输入信息以前出现一些缺省文本，可以在 HTML 文件中的<TEXTAREA>和</TEXTAREA>之间加入这些缺省字符。

除了 ROWS 和 COLS 属性，<TEXTAREA>还有一个 NAME 属性。该属性定义了用来存取文本框中用户输入数据的变量名。三个属性中只有 NAME 属性是必须的。例如如下的 HTML 语句将在表单内定义一个输入文本框：

```
<TEXTAREA NAME="suggestion" ROWS=6 COLS=40>缺省文本</TEXTAREA>
```

3.4.2 SELECT

<SELECT>可以以弹出式菜单或滚动列表框的形式在 Web 网页上为用户提供一组选择。和<TEXTAREA>一样，<SELECT>也有一个 NAME 属性用来存取客户端作出的选择。除了 NAME 属性，<SELECT>还有 MULTIPLE 属性和 SIZE 属性。当<SELECT>具有 MULTIPLE 属性即<SELECT MULTIPLE>时，用户可以作出多种选择。当 SIZE 为 1 时，<SELECT>以弹出式菜单的形式出现，当 SIZE 大于 1 时，<SELECT>以列表框的形式出现，在<SELECT>和</SELECT>之间，可以用<OPTION>定义可选项。以下 HTML 语句为表单定义了一个弹出式菜单：

```
<SELECT NAME="Operate System" SIZE=1>
  <OPTION>Win31
  <OPTION>Win95
  <OPTION>NT4.0
  <OPTION>UNIX
</SELECT>
```

3.4.3 INPUT

<INPUT>定义了其他各种可以在表单内使用的可接受客户端输入信息的手段，比如单行文本框、密码输入框、单选钮、复选框、Submit 按钮以及 Reset 按钮。通常它有以下属性：

● NAME：用来存储用户输入的数据，除了 Submit 和 Reset 按钮，其他类型的<INPUT>元素都必须具有 NAME 属性。NAME 属性在 CGI 程序中表现为变量名，通过它 CGI 可以知道数据是属于表单的哪个部分。

● SIZE：用来定义单行文本框和密码输入框的宽度（以字符为单位）。

● MAXLENGTHNAME：定义了单行文本框和密码输入框所能接收的最大字符数。通常在利用 CGI 在 Web 网页上实现数据库的录入操作时要用到这一属性。

● VALUE：对于单行文本框和密码输入框，它定义了它们在网页上显示的缺省文本，对于单选钮和复选框，定义了当这些按钮或复选框被选中时应该传给服务器的值。对于 Submit 和 Reset 按钮，定义了网页中此类按钮上应该显示的文本。

一般<INPUT>标识符具有如下的使用形式：

<INPUT TYPE ="表单元素类型" >

表 3-1 列出了可以由<INPUT>定义的各种表单元素。

表 3-1　<INPUT>所定义的各种表单元素

表单元素类型	简　单　说　明
Checkbox	复选框 Checkbox 只能有两种状态：选或不选。只有在被选择以后，该复选框所包含的数据（VALUE 值）才能被传给 CGI 程序
Hidden	Hidden 的表单元素在 Web 网页上通常是看不见的，它被 CGI 程序用来储存 Web 浏览器和 Web 服务器某一次连接的状态信息
Image	该元素和 Submit 按钮所起的作用一样，只是在 Web 页面上用一幅图片取代了 Submit 按钮
Password	和单行文本输入（Text）框类似，但用户键入的字符将以“*”或空格的形式在 Web 网页上显示出来。传给 CGI 程序的仍然是用户键入的字符串
Radio	单选钮(Radio)必须成组使用，它可以让用户从一组选择项中选出符合自己要求的一种。同一组单选钮具有相同的 NAME 属性
Text	单行文本输入框，您可以通过 SIZE 属性设置它在 Web 网页上的尺寸，也可以通过 MAXLENGTHNAME 属性设置它所能接收的最大字符数
Reset	该按钮用来清除表单中所有元素的输入值，并把它们设置为缺省值
Submit	缺省时，该按钮在 Web 页面上显示字符串 Submit Query，该字符串的值可以通过 VALUE 属性改变。当客户端选择了这个按钮以后，浏览器就向 Web 服务器发出请求信息以请求表单（Form）的 Action 属性所定义的 URL，同时，表单包含的所有数据也被传给这个 URL。不像<INPUT>的其他元素，该按钮的 NAME 属性是可选的。如果使用了 NAME 属性，该属性所具有的值（VALUE）同样会被传给 CGI 程序。因此当一个表单内有多个 Submit 按钮时，CGI 程序可以利用 NAME 属性确定是哪一个 Submit 被客户端选择了

HTML 表单由表 3-1 列举的各种表单元素组成。作为示例，图 3-4 所示的表单使用了多种表单元素，其 HTML 源码如下：

```
<html><head>
<title>个人情况调查表</title></head>
<body>
<form method="POST"action="/cgi-bin/table.exe">
<table width="80%">
<tr><td>您的姓名：</td><td>您的职务：</td></tr>
<tr><td><input type="text" size="20" name="cname" cols="60" align="left"></td>
 <td><input type="text" size="20" name="ZhiWu" cols="60" align="left"></td></tr>
<tr><td>您的电话号码：</td><td>您的 E_mail 地址：</td></tr>
```

```
<tr><td><input type="text" size="25" name="Phone" cols="60" align="left"></td>
 <td><input type="text" size="30" name="EMAIL" cols="60" align="left"></td></tr>
<tr><td colspan=2>您的工作单位: </td></tr>
<tr><td colspan=2><input type="text" size="50" name="cunit" cols="60" align="left"></td></tr>
</table>
<p>您的职称: <br>
  教授 <input type="radio" name="ZhiCheng" value="教授" align="left">
  研究员 <input type="radio" name="ZhiCheng"  value="研究员" align="left">
  副教授 <input type="radio" name="ZhiCheng" value="副教授" align="left">
  副研究员 <input type="radio" name="ZhiCheng" value="副研究员" align="left">
  高级工程师 <input type="radio" name="ZhiCheng" value="高级工程师" align="left">
  工程师 <input  type="radio" name="ZhiCheng" value="工程师" align="left">
  讲师 <input type="radio" name="ZhiCheng" value="讲师" align="left"></p>
<p>您的简历: <br>
 <textarea name="Cresume" rows="4" cols="60" size="100" align="left">
 </textarea></p>
<table width="60%">
<tr> <td><input type="submit" name="SUBMIT" value=" 上交材料 "align="left"></td>
 <td><input type="reset" name="CLEAR" value=" 重新填写 " align="left"></td>
</tr></table>
</form>
</bordy></html>
```

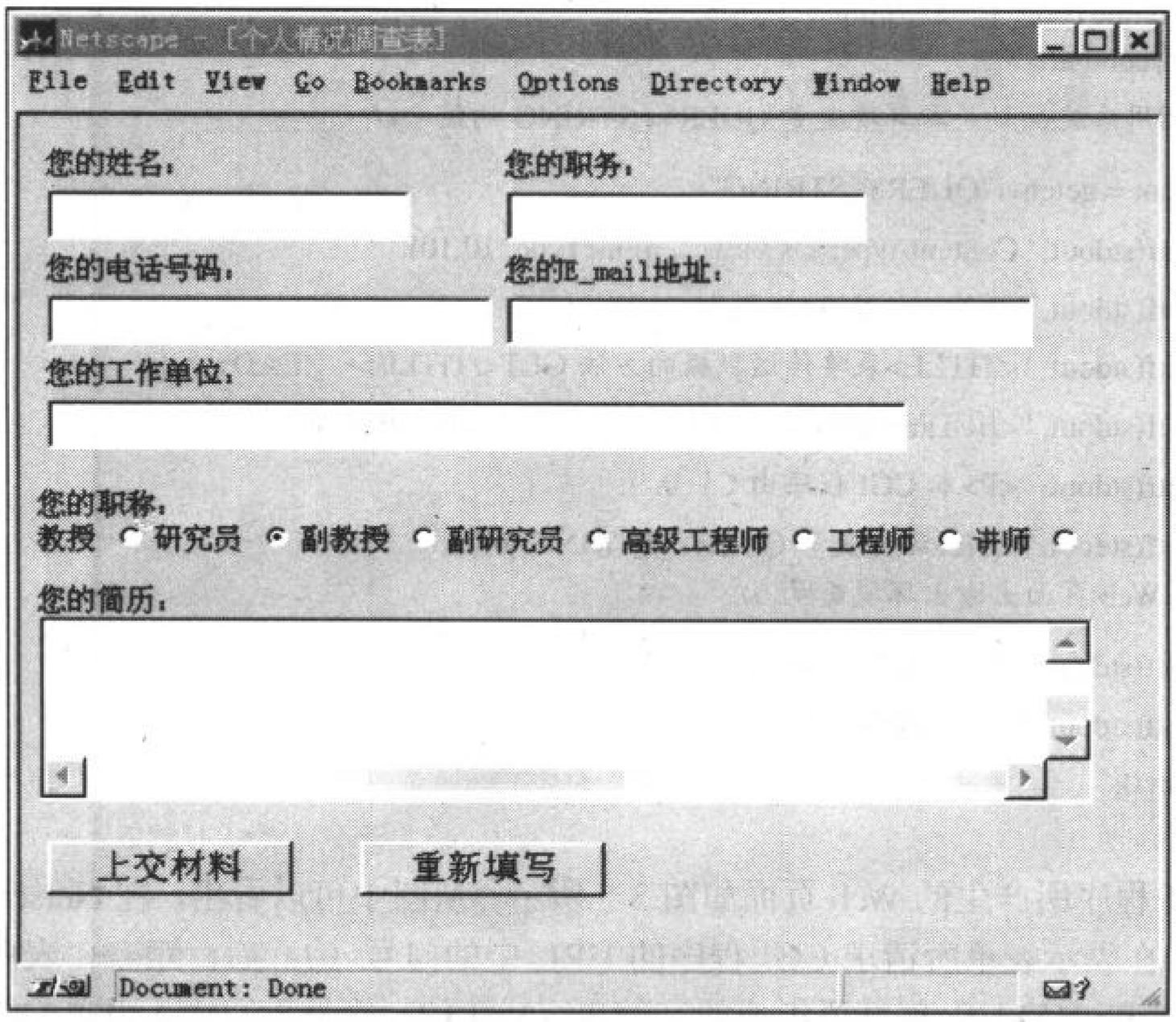

图 3-4　Web 网页上的表单

如果一个页面上有很多个输入文本框，页面在正常文本（Normal Text）模式下将很难对齐，一般都要使用预定义文本（Preformatted Text）模式，也就是将有关表单的 HTML 语句包含在标识符<PRE>和</PRE>之间。但在这种模式下页面的文本显示效果不是很好，在某一版本的浏览器上可能显示相当好，但换了一种版本后或浏览器类型后页面的显示效果就大不一样。浏览器运行平台的不同也会影响由 PRE 所定义的文本的显示。

这时可以在正常文本模式下使用 HTML 的表格（Table），将表单中的文本和文本输入框分别放入表格的单元格中，这时您就可以在各种情况下都得到满意的并且是相同的显示效果。

除非 Web 网页上的表单只有一个文本输入框，否则表单都必须拥有一个 Submit 按钮。当表单只有一个简单的文本输入框时，则可以省略 Submit 按钮。这时客户端在完成输入工作以后，只要按下 Enter 键，输入数据就会被浏览器传送给 CGI 程序。

3.5　传送表单数据的两种方法：GET 和 POST

通常有两种方法向 CGI 程序传送数据，即 GET 和 POST。GET 是缺省的方法。对于图 3-4 所示的表单，我们分别用 GET 和 POST 这两种方法传送数据。对于 GET 方法，本节用 C 语言编写了一个 CGI 程序，该 CGI 程序将储存在环境变量 QUERY_STRING 中的信息读出并在没有做任何处理的情况下向 Web 网页输出，程序的源代码如下：

```
#include <stdio.h>
#include <stdlib.h>
main( )
{   char mimeType[ ]="text/html";
    char *content=" ";
    /* 调用系统函数获取环境变量 QUERY_STRING 的值   */
    content = getenv("QUERY_STRING");
    fprintf(stdout, "Content-type: %s%c%c", mimeType, 10,10);
    fprintf(stdout, "<HTML><HEAD>\n");
    fprintf(stdout, "<TITLE>表单传送数据的方法 GET</TITLE></HEAD>\n");
    fprintf(stdout, "<BODY><H3>表单传送数据的方法 GET</H3>\n");
    fprintf(stdout, "<P>本 CGI 程序由 C！ \n");
    fprintf(stdout, "<BR>环境变量 QUERY_STRING:\n");
   /* 在 Web 页面上输出环境变量 */
    fprintf(stdout, "<P>%s\n", content);
    fprintf(stdout, "</BODY></HTML>\n");
    return 0;
}
```

该 CGI 程序所产生的 Web 页面如图 3-5 所示。从图中可以看出，在 Netscape 的 Netsite 菜单条内，在表示表单所请求 CGI 程序的 URL 中的问号（?）后，跟着由表单各个元素的 NAME 属性和 VALUE 属性所组成的字符串。VALUE 就是我们输入表单的数据。NAME

和 VALUE 之间用“=”连接，不同表单元素的数据之间用“&”分开。Web 页面上所输出的环境变量 QUERY_STRING 的值和 URL 中问号后的字符串相同。

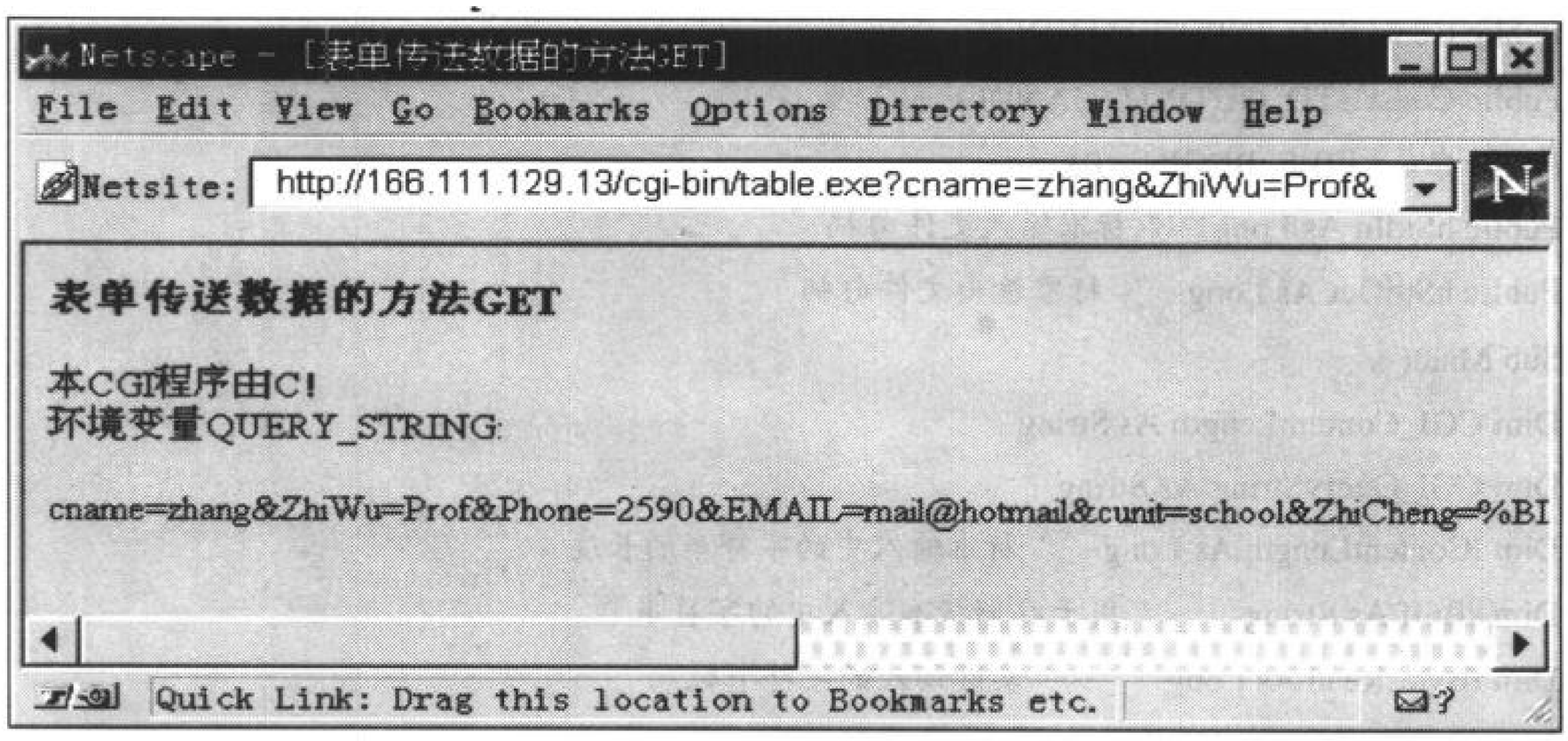

图 3-5　使用 GET 方法时 CGI 程序所产生的 Web 页面

使用 POST 方法时 CGI 程序所产生的 Web 页面如图 3-6 所示，该 CGI 程序用 Visual Basic 编写，它将标准输入中的数据读出并在未做任何处理的情况下向 Web 页面输出。

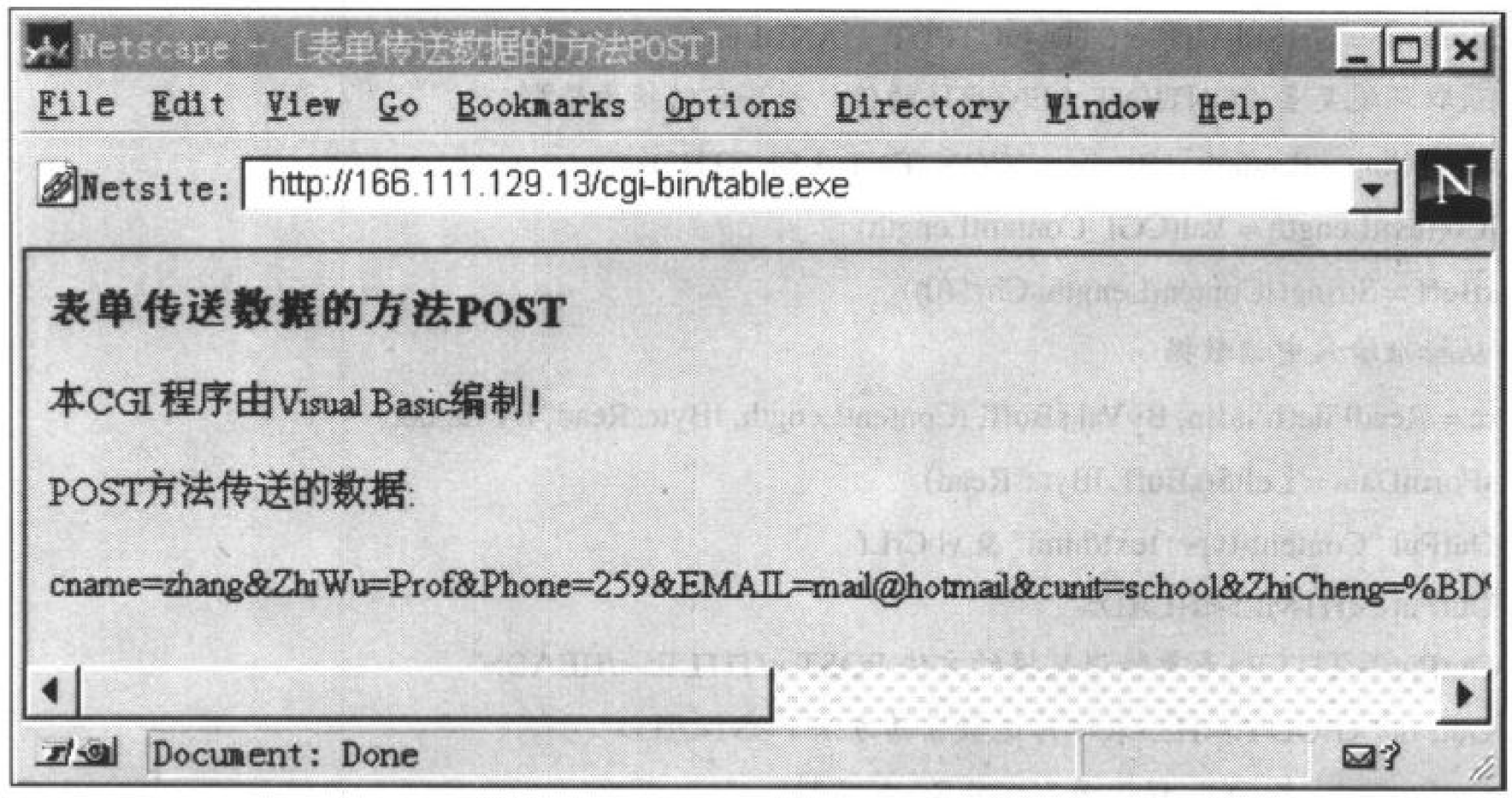

图 3-6　使用 POST 方法时 CGI 程序所产生的 Web 页面

程序的源代码如下：

```
'声明程序中要用到的三个 Windows API 函数，一个用于处理标准输入输出文件句柄，另两个用于读写文件
Declare Function GetStdHandle Lib "kernel32"  (ByVal nStdHandle As Long) As Long
Declare Function ReadFile Lib "kernel32"  (ByVal hFile As Long,  lpBuffer As Any,
    ByVal nNumberOfBytesToRead As Long,  lpNumberOfBytesRead As Long,
    lpOverlapped As Any) As Long
Declare Function WriteFile Lib "kernel32"  (ByVal hFile As Long, ByVal lpBuffer As String,
```

```
    ByVal nNumberOfBytesToWrite As Long, lpNumberOfBytesWritten As Long,
    lpOverlapped As Any) As Long
Public Const STD_INPUT_HANDLE = -10&
Public Const STD_OUTPUT_HANDLE = -11&
Public Const FILE_BEGIN = 0&
Public hStdIn As Long    ' 标准输入文件句柄
Public hStdOut As Long    ' 标准输出文件句柄
Sub Main( )
Dim CGI_ContentLength As String
Dim CGI_QueryString As String
Dim lContentLength As Long    ' 标准输入中的字符串的长度
Dim sBuff As String        ' 用于存储标准输入中的字符串
Dim lBytesRead As Long        ' 实际读入的字符个数
Dim rc As Long
Dim sFormData As String
'调用系统函数生成标准输入输出文件句柄
hStdIn = GetStdHandle(STD_INPUT_HANDLE)
hStdOut = GetStdHandle(STD_OUTPUT_HANDLE)
'获取环境变量 CONTENT_LENGTH 的值，并将它转换为整型
CGI_ContentLength = Environ("CONTENT_LENGTH")
lContentLength = Val(CGI_ContentLength)
sBuff = String(lContentLength, Chr$(0))
'从标准输入中读数据
rc = ReadFile(hStdIn, ByVal sBuff, lContentLength, lBytesRead, ByVal 0&)
sFormData = Left$(sBuff, lBytesRead)
OutPut "Content-type: text/html" & vbCrLf
OutPut "<HTML><HEAD>"
OutPut "<TITLE>表单传送数据的方法 POST </TITLE></HEAD>"
OutPut "<BODY><H3>表单传送数据的方法 POST</H3>"
OutPut "<P>本 CGI 程序由 Visual Basic 编制！  "
OutPut "<P>POST 方法传送的数据: "
OutPut "<P>" & sBuff
OutPut "</BODY></HTML>"
End Sub
Sub OutPut(s As String)
'-----------------------------------------
'  定义一个向标准输出写信息的函数
'-----------------------------------------
Dim lBytesWritten As Long
```

```
s = s & vbCrLf
WriteFile hStdOut, s, Len(s), lBytesWritten, ByVal 0&
End Sub
```

图 3-6 中，在 Netscape 的 Netsite 菜单条内，URL 仅表示了所请求的 CGI 程序位置，后面没有客户端在表单上的输入信息，因为这些输入信息是在浏览器发完所有的头信息后才作为一个整体传送给服务器的。由图还可以看出，从标准输入中读出的字符串和从环境变量 QUERY_STRING 读出的字符串具有相同的形式。

一般，当用 GET 方法传递数据时，浏览器首先将表单中用户输入的数据 URI 译码（URI Encoding）成“NAME = VALUE”的形式，各个“NAME = VALUE”对之间用“&”分开，NAME 和 VALUE 就是表单中各种元素的属性，其中 VALUE 属性存放用户的输入数据。完成 URI 译码以后，浏览器将这些被译码以后的数据放在由表单的属性 ACTION 定义的 URL 后面传给 Web 服务器，两者之间用“？”分开。服务器最终将被译码以后的客户端输入数据放在环境变量 QUERY_STRING 中，CGI 程序可以通过存取该环境变量的值获取客户端用户的输入。GET 方法的一个主要缺陷是用它所传送的数据量包括 URL 在内不能超过 1024 个字符。如果表单包含的数据很多而又采用 GET 方法传递，就有可能造成数据的丢失。

对于 POST 方法，浏览器则是在向服务器传递了所有的 HTTP 头信息后，再将 URI 译码以后的表单数据单独作为一个整体传送给服务器，服务器接收以后再将它放在标准输出也就是 CGI 程序的标准输入中。从理论上说，用 POST 方法向 Web 服务器传送的数据量没有大小的限制。POST 方法和 GET 方法都采用相同的译码方法。

3.6　URI 译码（URI Encoding）

以下是上一节的 CGI 程序从标准输入或环境变量 QUERY_STRING 中读出的字符串的全貌，字符串中每一对（NAME=VALUE）代表了表单中的输入信息：

```
cname=zhang&ZhiWu=Prof&Phone=259&EMAIL=mail@hotmail&cunit=school
&ZhiCheng=%BD%CC%CA%DA&Cresume=resume&SUBMIT=+%C9%CF%BD
%BB%B2%C4%C1%CF+
```

从以上字符串中我们不难发现，一些输入信息发生了变化。比如表单中的信息“讲师”在字符串中则成了“%BD%CC%CA%DA”这四个十六进制数，而 Submit 按钮的 VALUE 属性中的空格则变成了“+”。

这就是 URI 译码（URI Encoding）的效果。由于 Web 服务器和浏览器对于一些特殊的字符不能正确处理，Web 服务器和浏览器之间可能会由此而产生某种程度的误会，因此在数据被传送之前，浏览器都要对表单内客户输入的数据中的特殊字符进行 URI 译码。

例如 Web 系统用“=”分解 NAME 和 VALUE 属性，用“&”分隔不同的输入数据，如果在表单的输入数据中包含这些特殊的字符，并且表单的数据在传送给 Web 服务器前不做任何处理，Web 服务器将无法知道哪一个“=”、“&”是用户输入的，哪一个是浏览器加的。在由表单属性 ACTION 定义的 URL 中，也可能出现一些特殊的字符，当在 CGI 程序的名称中和路径信息（Path Information）中出现像“=”、“&”和“？”时，都会影响数据

的正确传送。

URI 译码就是将 Web 服务器所不能正确处理的特殊字符转换成它的十六进制数的形式，比如将“%”转换成“%25”、“=”转换成“%3D”等等。这些特殊的字符通常被称为 Web 系统的保留字符，表 3-2 列出了这些保留字符。

表 3-2　Web 系统的保留字符

保留字符	保留字符的十六进制形式
Tab	09
Space	20
“	22
(	28
)	29
,	2C
.	2E
;	3B
:	3A
<	3C
>	3E
@	40
[	5B
\	5C
]	5D
^	5E
‘	60
{	7B
|	7C
}	7D
~	7E

Web 系统将汉字当成特殊的字符，对它也要进行 URI 译码，通常在 Web 系统上不论是用 GET 方法还是用 POST 方法传送的数据都要进行 URI 译码。CGI 程序要想处理表单传送来的数据，还必须对浏览器 URI 译码过的数据进行解码，理解 URI 译码对于我们进行 CGI 编程是非常重要的。一般 URI 译码的步骤如下：

（1）浏览器将所传送的数据根据表单所包含的元素分解成“NAME=VALUE”的形式，NAME 和 VALUE 分别是表单元素的属性，其中 VALUE 属性中存储客户端在表单中输入的数据，如果客户端没有输入数据，则 VALUE 存储的是表单定义的缺省值，如果缺省值也没有定义，则 VALUE 值为空；

（2）表单中各元素的各个“NAME=VALUE”对被浏览器用“&”连接起来；

（3）VALUE 属性中存放的数据若含有空格，则被转换成“+”；

（4）URL 和输入数据中所包含的 Web 系统的保留字符必须被译码成它们的十六进制

数的形式。

（5）被译码后的字符被表示成一个“%”加上它们的十六进制数形式（%HH）。

3.7 URI 解码

CGI 程序从环境变量 QUERY_STRING 或标准输入中读入的数据是经过浏览器 URI 译码的，在使用这些数据以前还必须对它们进行 URI 解码。解码的目的就是将数据还原成客户在 Web 页面上输入时的形式。本章 3.6 节介绍了 URI 译码的过程，URI 解码和它正相反，一般它有以下的工作要做：

（1）从浏览器用 GET 或 POST 方法所传送来的数据中找出代表各个表单元素所储存数据的“NAME=VALUE”对。

（2）VALUE 属性中存放的数据若含有“+”，则被转换成空格。

（3）将 VALUE 属性中存放的数据中的十六进制数“%HH”转换成相应的字符。

虽然实现 URI 解码的程序的编写比较简单，但使用别人提供的一些自由共享程序库将可以使您把注意力更好的集中在对客户端输入数据的最后处理上。以下 CGI 程序（table.c）可以将表单的“NAME=VALUE”对分割开，最后在 Web 页面上按照用户最初输入的形式输出这些数据对。程序中定义了一个结构类型的数组（entries）用来存储表单的各个 NAME 属性和 VALUE 属性值，并使用了 NCSA 的 util.c 所提供的函数 makeword( )、fmakeword( )、unescape_url( )和 plustospace( )来完成对输入字符串的分割和 URI 解码。 本程序编译后在 Windows 95 和 UNIX 系统下均可以运行。

```
/*table.c */

#include <stdio.h>
#include <stdlib.h>
typedef struct
{
    char *name;
    char *val;
} entry;
entry entries[1000];
char *makeword(char *line, char stop);
char *fmakeword(FILE *f, char stop, int *len);
char x2c(char *what);
void unescape_url(char *url);
void plustospace(char *str);
main( )
{
    register int x,m=0;
    int cl;
    /*输出 MIME 类型*/
```

```
    printf("Content-type: text/html%c%c",10,10);
    if(strcmp(getenv("REQUEST_METHOD"),"POST"))
    {
        printf("本 CGI 程序必须使用 POST 方法来传递数据！\n");
        exit(1);
    }
/*根据环境变量获取标准输入中字符串的长度*/
    cl = atoi(getenv("CONTENT_LENGTH"));
    for(x=0;cl && (!feof(stdin));x++)
    {
     m=x;
     entries[x].val = fmakeword(stdin,'&',&cl); /*从标准输入中读入 NAME=VALUE 对*/
     plustospace(entries[x].val);      /*将字符'+'转换为空格*/
     unescape_url(entries[x].val);     /*对传来的数据进行 URI 解码*/
     entries[x].name = makeword(entries[x].val,'='); /*将 NAME 和 VALUE 分开*/
     }
    printf("<TITLE>如何处理表单送来的数据</TITLE></HEAD>\n");
    printf("<H1>表单中的数据</H1>");
    printf("您输入了以下数据:<p>%c",10);
    printf("<ul>%c",10);
    for(x=0; x <= m; x++)
    printf("<li> <code>%s = %s</code>%c",entries[x].name, entries[x].val,10);
    printf("</ul>%c",10);
  }
/*本函数用于根据指定的字符分割字符串*/
char *makeword(char *line, char stop)
   {
     int x = 0,y;
     char *word = (char *) malloc(sizeof(char) * (strlen(line) + 1));
     for(x=0;((line[x]) && (line[x] != stop));x++)
          word[x] = line[x];
     word[x] = '\0';
     if(line[x]) ++x;
     y=0;
     while(line[y++] = line[x++]);
     return word;
    }
/*本函数用于从标准输入中读取各个 NAME=VALUE 对*/
char *fmakeword(FILE *f, char stop, int *cl)
```

```
{
    int wsize;
    char *word;
    int ll;
    wsize = 10240;
    ll=0;
    word = (char *) malloc(sizeof(char) * (wsize + 1));
    while(1)
    {
        word[ll] = (char)fgetc(f);
        if(ll==wsize)
        {
            word[ll+1] = '\0';
            wsize+=1024;
            word = (char *)realloc(word,sizeof(char)*(wsize+1));
        }
        --(*cl);
        if((word[ll] == stop) || (feof(f)) || (!(*cl)))
        {
            if(word[ll] != stop) ll++;
            word[ll] = '\0';
            return word;
        }
        ++ll;
    }
}
char x2c(char *what)
 {
    register char digit;
    digit = (what[0] >= 'A' ? ((what[0] & 0xdf) - 'A')+10 : (what[0] - '0'));
    digit *= 16;
    digit += (what[1] >= 'A' ? ((what[1] & 0xdf) - 'A')+10 : (what[1] - '0'));
    return(digit);
}
/*本函数用于将字符串中的十六进制数恢复成 ASCII 字符*/
void unescape_url(char *url)
    register int x,y;
    for(x=0,y=0;url[y];++x,++y)
    {
```

```
            if((url[x] = url[y]) == '%')
            {
                url[x] = x2c(&url[y+1]);
                y+=2;
            }
        }
        url[x] = '\0';
    }
    void plustospace(char *str) /*本函数用于将字符'+'转换为空格*/
    {
        register int x;
        for(x=0;str[x];x++) if(str[x] == '+') str[x] = ' ';
    }
```

客户端调用该 CGI 程序的 Web 页面和用户在其上的输入如图 3-7 所示，其 HTML 源码如下：

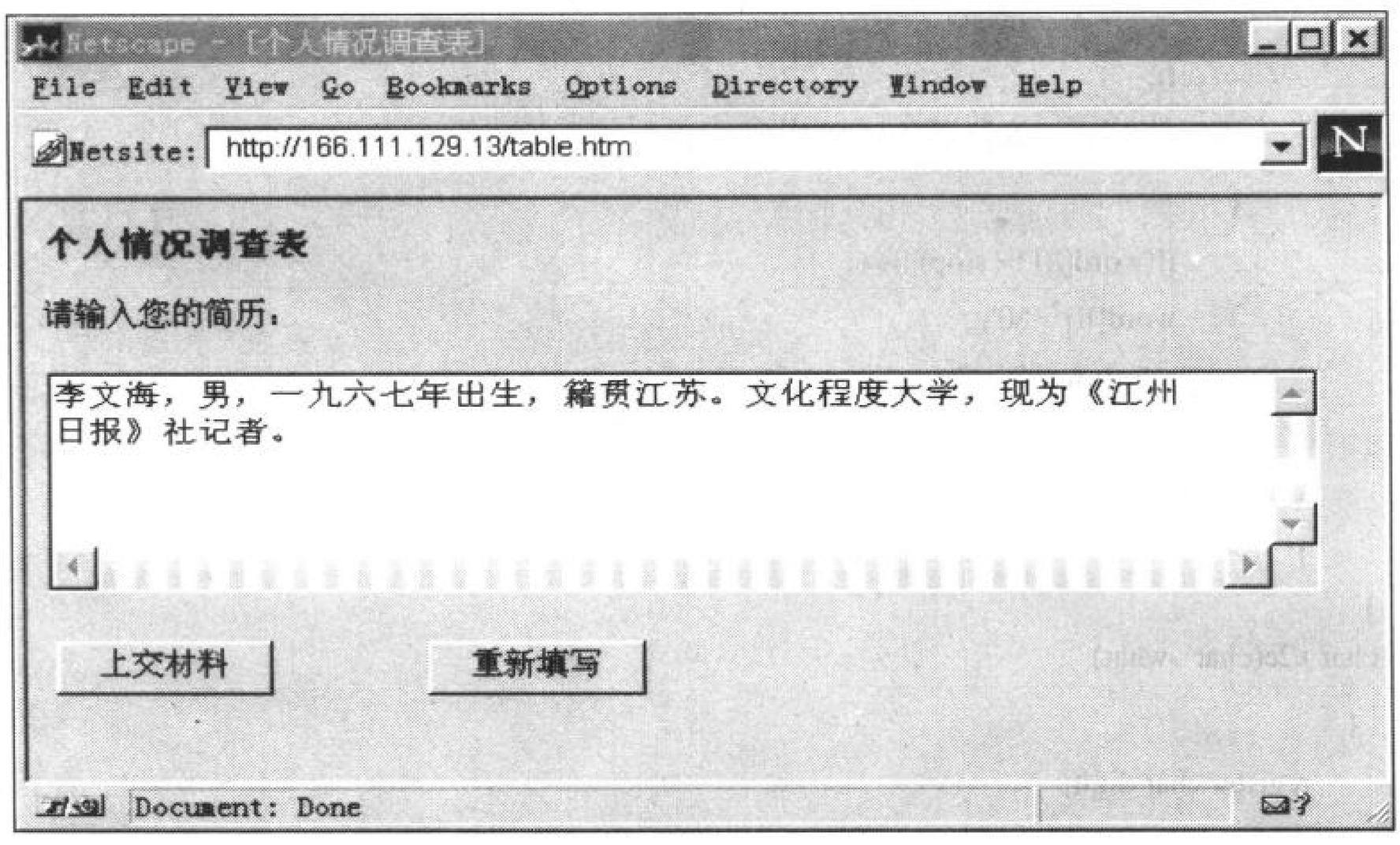

图 3-7　调用 CGI 程序 table.c 的表单

```
<html>
<head><title>个人情况调查表</title></head>
<body>
<h3>个人情况调查表</h3>
<P>请输入您的简历:
<form action="/cgi-bin/table.exe" method="POST">
<textarea name="简历" rows="4" cols="60" size="100" align="left"></textarea></p>
<table width="60%">
```

```
<tr>
<td><input type="submit" name="SUBMIT" value=" 上交材料 "align="left"></td>
<td><input type="reset" name="CLEAR" value=" 重新填写 " align="left"></td>
</tr>
</table>
</form>
</bordy></html>
```

本节 CGI 程序 table.c 所产生的 Web 页面如图 3-8 所示。

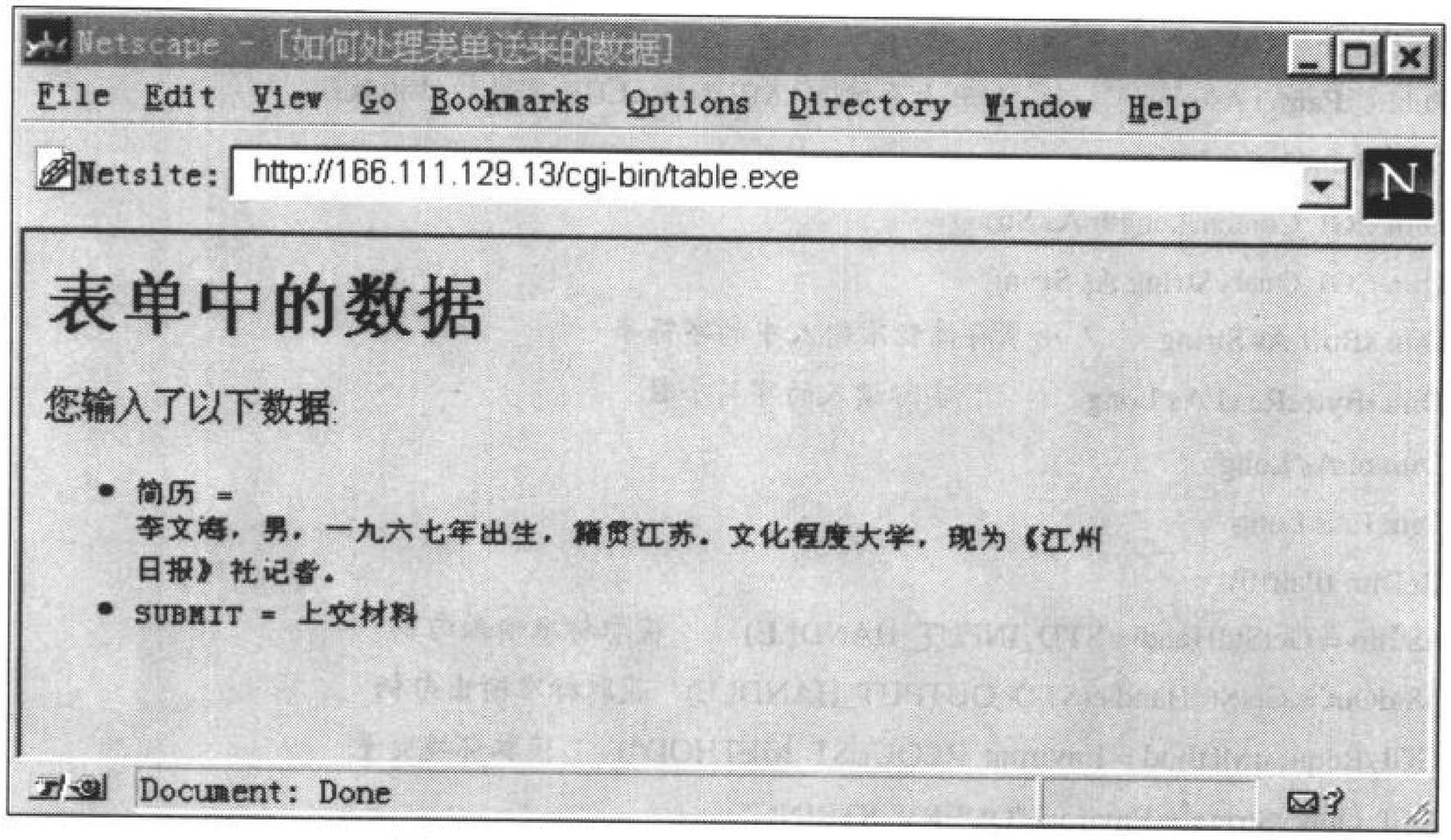

图 3-8　CGI 程序 table.c 所生成的 Web 页面

能够完成 table.c 的功能的用 Visual Basic 语言编写的 CGI 程序的源代码如下所示，该程序调用了三个 WIN32 API 函数来处理标准输入输出，程序的结构和 table.c 类似。

```
' table.bas
' 声明程序中要用到的三个 Windows API 函数，一个用于处理标准输入输出文件句柄，另两个用于读写文件
Declare Function GetStdHandle Lib "kernel32"  (ByVal nStdHandle As Long) As Long
Declare Function ReadFile Lib "kernel32"  (ByVal hFile As Long, lpBuffer As Any,
    ByVal nNumberOfBytesToRead As Long, lpNumberOfBytesRead As Long,
    lpOverlapped As Any) As Long
Declare Function WriteFile Lib "kernel32"  (ByVal hFile As Long,
    ByVal lpBuffer As String, ByVal nNumberOfBytesToWrite As Long,
    lpNumberOfBytesWritten As Long, lpOverlapped As Any) As Long
Public Const STD_INPUT_HANDLE = -10&
Public Const STD_OUTPUT_HANDLE = -11&
Public Const FILE_BEGIN = 0&
```

```
Public hStdIn As Long    ' 标准输入文件句柄
Public hStdOut As Long    ' 标准输出文件句柄
Public sFormData As String  ' 客户端传送来的没有经 URI 解码的数据
Public lContentLength As Long      ' 标准输入中字符串的长度
Public CGI_RequestMethod As String
Type pair
   Name As String
   Value As String
End Type
Public tPair( ) As pair            ' 用于存储 NAME 和 VALUE 属性的结构数组
Sub Main( )
Dim CGI_ContentLength As String
Dim CGI_QueryString As String
Dim sBuff As String      ' 用来存储标准输入中的字符串
Dim lBytesRead As Long        ' 实际读入的字符个数
Dim rc As Long
Dim I As Long
ReDim tPair(0)
hStdIn = GetStdHandle(STD_INPUT_HANDLE)    ' 获取标准输入句柄
hStdOut = GetStdHandle(STD_OUTPUT_HANDLE) ' 获取标准输出句柄
CGI_RequestMethod = Environ("REQUEST_METHOD")  ' 获取环境变量
CGI_QueryString = Environ("QUERY_STRING")
CGI_ContentLength = Environ("CONTENT_LENGTH")
lContentLength = Val(CGI_ContentLength)
sBuff = String(lContentLength, Chr$(0))
' 获取标准输入中的字符串，并对其进行 URI 解码
If CGI_RequestMethod = "POST" Then   ' 获取用户的输入
    sBuff = String(lContentLength, Chr$(0))
    rc = ReadFile(hStdIn, ByVal sBuff, lContentLength, lBytesRead, ByVal 0&)
    sFormData = Left$(sBuff, lBytesRead)
    StorePairs sFormData
Else If CGI_RequestMethod = "GET" Then
StorePairs CGI_QueryString
End If
OutPut "Content-type: text/html" & vbCrLf
OutPut "<HTML><HEAD>"
OutPut "<TITLE>如何处理表单送来的数据"
OutPut "</TITLE></HEAD><BODY>"
OutPut "<H3>表单中的数据</H3>"
```

```
OutPut sFormData
OutPut "<P>您输入了以下数据:"
OutPut "<ul>"
For I = 1 To UBound(tPair)
 OutPut "<li> <code>" & tPair(I).Name & "=" & tPair(I).Value & "</code>"
Next I
OutPut "</BODY></HTML>"
End Sub
Sub OutPut(s As String)
'-----------------------------------------
'定义一个向标准输出写信息的函数
'-----------------------------------------
Dim lBytesWritten As Long
s = s & vbCrLf
WriteFile hStdOut, s, Len(s), lBytesWritten, ByVal 0&
End Sub
Sub StorePairs(sData As String)
' 本程序用于解码用户的输入，并将其存入一个全局结构数组中
Dim pointer As Long
Dim n As Long        ' name/value 对的数量
Dim delim1 As Long       ' "=" 的位置
Dim delim2 As Long       ' "&" 的位置
Dim lastPair As Long          '   数组 tPair(# )的大小
Dim lPairs As Long
lastPair = UBound(tPair)
pointer = 1
Do
   delim1 = InStr(pointer, sData, "=")
   If delim1 = 0 Then Exit Do
   pointer = delim1 + 1
   lPairs = lPairs + 1
Loop
If lPairs = 0 Then Exit Sub
ReDim Preserve tPair(lastPair + lPairs) As pair
pointer = 1
For n = (lastPair + 1) To UBound(tPair)
    delim1 = InStr(pointer, sData, "=")
    If delim1 = 0 Then Exit For
    tPair(n).Name = UrlDecode(Mid$(sData, pointer, delim1 - pointer))
```

```
    delim2 = InStr(delim1, sData, "&")
        If delim2 = 0 Then delim2 = Len(sData) + 1
    ' value is between the "=" and the "&"
    tPair(n).Value = UrlDecode(Mid$(sData, delim1 + 1, delim2 - delim1 - 1))
    pointer = delim2 + 1
Next n
End Sub
'函数 UrlDecode( )用于 URI 解码，将字符串中的“+”转换成空格
Public Function UrlDecode(ByVal sEncoded As String) As String
Dim pointer As Long
Dim pos As Long
If sEncoded = "" Then Exit Function
' 将"+" 转换成空格
pointer = 1
Do
    pos = InStr(pointer, sEncoded, "+")
    If pos = 0 Then Exit Do
    Mid$(sEncoded, pos, 1) = " "
    pointer = pos + 1
Loop
' 将十六进制数转换成相应的字符
pointer = 1
Do
    pos = InStr(pointer, sEncoded, "%")
    If pos = 0 Then Exit Do
    Mid$(sEncoded, pos, 1) = Chr$("&H" & (Mid$(sEncoded, pos + 1, 2)))
    sEncoded = Left$(sEncoded, pos) & Mid$(sEncoded, pos + 3)
    pointer = pos + 1
Loop
UrlDecode = sEncoded
End Function
```

注意，由于用到了三个 WIN32 API 函数，本程序必须使用 32 位版本的 VB 编译器编译。

3.8　利用 CGI 程序生成 HTTP 头信息

通常，Web 服务器收到一个浏览器发来的某一 URL 请求后，将根据在服务器上的搜寻结果，向浏览器发回一个 HTTP 响应头（Response Head）。这个 HTTP 响应头中的信息将告诉浏览器所请求的资源是否存在，当前服务器所用的协议、版本号、所传送信息的 MIME

类型和服务器所使用的软件等一些信息，以使浏览器能对随后将要收到的信息作出正确的处理，最后服务器才将浏览器所请求的资源传送给浏览器。以下就是一个 HTTP 响应头的具体形式：

```
HTTP/1.0 200 OK
Date: Mon, 02 Oct 1996 11:11:32 GMT
Server: NCSA/1.0a6
Content-type: text/html
Last-Modified: Mon, 04 Sep 1996 17:42:40 GMT
```

该响应头的第一行用来告诉浏览器 Web 服务器所用的是 HTTP 协议，版本号是 1.0，当前服务器的状态码是“200”，状态描述符是“OK”，这一行一般具有如下的形式：

```
PROTOCOL/Version_Number   Status_Code   Status_Description
```

附录 B 中详细列出了服务器所返回的各种状态码（Status Code）和状态描述符（Status Description）。

HTTP 响应头的第二行表示服务器发出该响应头的时间，时间必须是格林尼治时间，一般具有以下三种形式：

```
Sun, 06 Nov 1996 06:15:10 GMT
Sunday, 06-Nov-96 06:15:10 GMT
Sun Nov 6 06:15:10 1996
```

HTTP 响应头的第三行用来告诉浏览器 Web 服务器所用的软件，第四行是服务器所传信息的 MIME 类型，第五行则是服务器所传送信息的最后修改时间。

实际上服务器所产生的 HTTP 响应头还有一行，即用来将 HTTP 响应头和客户端所需信息分开的空行，这一空行是必不可少的。

当浏览器的请求是一个CGI程序时，Web服务器并不知道CGI程序所输出结果的MIME类型。因此本章所列举的所有 CGI 程序在向 Web 网页输出信息之前都要说明所输出信息的 MIME 类型，即向标准输出（STDOUT）中写入字符串“Content-type: text/html”。服务器将根据 CGI 程序所声明的 MIME 类型产生一个完整的 HTTP 响应头 。为了将标准输出（STDOUT）中表示 MIME 类型的字符串和 CGI 程序所产生的其他信息区分开，CGI 程序在它们两者之间通常也要加一空行，这也就是为什么 CGI 程序在输出字符串“Content-type: text/html”时一定要在其后加两个回车符“\n\n”的原因。

事实上为了减轻 Web 服务器的负担，缩短程序的响应时间，可以让 CGI 程序自己生成完整的 HTTP 响应头，而此时服务器所要做的仅仅是将 CGI 程序的输出结果直接传送给浏览器。本章第二节处理图 3-1 所示表单的 CGI 程序可以改为如下的形式以生成完整的 HTTP 响应头：

```
#include <stdio.h>
#include <stdlib.h>
 main( )
 {
    char mimeType[ ]="text/html";
    char *serversoftware;
```

```
    char *lastmodified;
    char *currenttime;
/*为了生成完整的头信息，获取有关环境变量的值*/
    serversoftware = getenv("SERVER_SOFTWARE");
    lastmodified = getenv("LAST_MODIFIED");
    currenttime = getenv("DATE_GMT");
/*生成完整的 HTTP 响应头 */
    fprintf(stdout, "HTTP/1.0 200 OK%c",10);
    fprintf(stdout, "Date: %s%c",currenttime,10);
    fprintf(stdout, "Server: %s%c",serversoftware,10);
    fprintf(stdout, "Content-type: %s%c", mimeType, 10);
    fprintf(stdout, " Last-Modified: %s%c%c",lastmodified,10,10);
/*向浏览器输出有关信息*/
    fprintf(stdout, "<HTML><HEAD>\n");
    fprintf(stdout, "<TITLE>表单传送数据的两种方法</TITLE></HEAD>\n");
    fprintf(stdout, "<BODY><H3>表单传送数据的两种方法</H3>\n");
    fprintf(stdout, "<P>本 CGI 程序由 C 或 Visual Basic 编制！ \n");
    fprintf(stdout, "<BR>本网页由 CGI 程序产生！ \n");
    fprintf(stdout, "</BODY></HTML>\n");
    return 0;
  }
```

为了让服务器知道它不必为 CGI 程序生成 HTTP 响应头 ，所有能够产生完整 HTTP 响应头的 CGI 程序的名称都必须具有前缀“NPH-”或“nph-”，比如 nph-cgi1.exe。

如果 HTTP 响应头含有以下字符串：“Location: URL”，则当浏览器收到这一响应头信息后，将立刻向新的 URL 发出请求。例如以下的 CGI 程序可以在收到浏览器的调用后使浏览器转向另一个 URL：http://www.edu.cn。

```
#include <stdio.h>
main( )
 {
    char *where="http://www.edu.cn";
    fprintf(stdout, " Location: %s%c%c",where,10,10);
    return 0;
 }
```

这一功能是非常有用的，比如您有一文件让万维网上的游客下载，如果这个文件非常有用，前来下载的客户可能会非常多。这时您可以将这一文件的拷贝放在几个服务器上。当然您可以将指向这几个拷贝的 URL 的超文本链接放在以前的主页上，客户自己决定从哪处下载。不过使用 HTTP 响应头中的 Location 信息，您可以在这几个 URL 之间平均的分配用户。实现的方法就是编写一个 CGI 程序，该 CGI 程序将随机地从这几个 URL 中选取一个并将它用 HTTP 响应头发给客户。这样在主页上您只需要一个指向该 CGI 程序的链接。

第四章　环境变量的使用

可能您已经开始了 CGI 编程或阅读过别人写的 CGI 程序，您会发现任何 CGI 程序都离不开环境变量的使用。在上一章中我们可以看到，通过环境变量，CGI 程序可以知道用户请求 CGI 程序的方法是 GET 还是 POST，用户总共输入了多少字节的信息。事实上环境变量能提供的信息不仅仅是这些。

当 Web 服务器为一个 CGI 程序收集信息以准备一个运行环境时，储存在环境变量内的不仅仅是和该 CGI 程序直接有关的信息，基本上 Web 服务器和浏览器一次连接的所有状态信息都包含在环境变量内。根据环境变量所包含信息的来源，环境变量可以分为以下三类：和 Web 服务器状态有关的环境变量、和浏览器状态有关的环境变量以及和浏览器发出的请求头（Request Header）有关的环境变量。虽然一个 CGI 程序不可能用到所有的环境变量，但了解每一个环境变量的意义和用途对于 CGI 编程却是非常重要的。本章将从以下几个方面来介绍环境变量的使用：

- 环境变量的分类；
- 环境变量的意义；
- 环境变量的使用实例；
- CGI 和 SSI 环境变量之间的不同。

4.1　和 Web 服务器状态有关的环境变量

当一个 CGI 程序被 WWW 上某一客户端用户请求而被 Web 服务器激活时，Web 服务器会为它准备一个运行环境，设置环境变量。一般每一个 CGI 程序的运行环境都是独一无二的，但对于运行环境中和 Web 服务器状态有关的环境变量，每一个 CGI 程序却是一样的。通常和 Web 服务器状态有关的环境变量有以下几类：

- GATEWAY_INTERFACE

环境变量“GATEWAY_INTERFACE”存储了当前 Web 服务器所用的通用网关接口 CGI 规范的版本号，该环境变量所包含的字符串具有以下的格式：

CGI/revision　number

例如 CGI/1.1，目前 CGI 最新的版本号就是 1.1，有关 CGI 规范的最新标准可以在 Web 站点 http://hoohoo.ncsa.uiuc.edu/cgi/上找到。

- SERVER_ADMIN

环境变量“SERVER_ADMIN”则说明了当前 Web 服务器管理人员或技术支持人员的电子函件信箱。当您在调试 CGI 程序遇到了无法解决的问题或者想了解 Web 服务器的有关设置情况时，可以用这个电子函件信箱和 Web 服务器的管理员联系和请教。

- SERVER_NAME

环境变量“SERVER_NAME”则说明了当前 Web 服务器的 Internet 域名，例如：www.yourhost.com，如果服务器没有域名，该环境变量存储的是当前服务器的 IP 地址。该环境变量的值应该和浏览器发来的请求头中 URL 所包含的服务器域名相同。

● SERVER_SOFTWARE

环境变量“SERVER_SOFTWARE”说明了当前 Web 服务器所用的软件类型和版本号。通过对这个环境变量的使用，您可以大致了解服务器对 SSI 的支持情况等信息。

● SERVER_PORT

环境变量“SERVER_PORT”说明 Web 服务器在哪一个端口（PORT）监听浏览器发来的请求并对此作出反应。服务器缺省的“SERVER_PORT”值是“80”，如果 Web 服务器使用一个非 80 端口监听浏览器发来的请求，浏览器必须知道这个端口值，并在请求头的 URL 中包含这个端口值，例如：http://www.yourhost.com : 100。同样，SERVER_PORT 的值也必须和 URL 中的端口值一致。

● SERVER_PROTOCOL

环境变量“SERVER_PROTOCOL”说明了当前 Web 服务器处理浏览器发来的请求时所用的协议类型和版本号，例如“HTTP/1.0”。

由于这些和 Web 服务器状态有关的环境变量对于每个 CGI 程序进程基本上都是一样的，通常在 CGI 程序中很少使用它们。但是如果您希望编写的 CGI 程序可以在各种 Web 服务器软件下运行，或 CGI 程序运行在虚拟 Web 服务器上，您就有可能要用到这些环境变量。所谓虚拟 Web 服务器，是指在同一台机器上同时运行几个 Web 服务器进程，每个服务器具有不同的 IP。

4.2　和浏览器的请求头有关的环境变量

当客户端请求浏览 WWW 上某一资源时，浏览器将根据客户输入的 URL 生成一个请求头（Request Head），并将该请求头发送给相应的 Web 服务器。如果 Web 服务器找到了客户端请求的 URL，服务器将回送一个响应头给浏览器。如果客户端请求的是一个 CGI 程序，服务器还将根据所收到的请求头的内容设置部分环境变量的值。

和有关服务器状态的环境变量不一样，这类环境变量的值不仅和具体的 CGI 程序有关，而且还和传送数据的客户端有关。上一章中 CGI 程序经常用到的环境变量 QUEY_STRING、REQUEST_METHOD 和 CONTENT_LENGTH 就属于此类。

● AUTH_TYPE

环境变量 AUTH_TYPE 说明了 Web 服务器验证客户端使用服务器上 CGI 程序的权利的方法，AUTH_TYPE 的值通常是 Basic，因为现在“Basic”所定义的验证方法是目前网上主要的验证方法。当使用 Basic 验证方法时，客户端在请求服务器上需要进行身份验证的资源时，服务器将发送一个验证响应头（WWW-Authenticate response header）给浏览器，通知客户端进行身份验证，浏览器将通过一个验证请求头（Authorization request header）把用户输入的用户名和密码传给服务器。只有通过了身份验证，服务器才会将相应的资源传给客户端。环境变量 AUTH_TYPE 的另外两种可能的取值是 PGP 和 PEM，它们代表了另外两种验证方法。当浏览器所请求的资源不需要身份验证时，环境变量 AUTH_TYPE 的值为 NULL。

● CONTENT_LENGTH

当 Web 页面使用 POST 方法向服务器上的 CGI 程序传递数据时，环境变量

CONTENT_LENGTH 的值为 CGI 程序的标准输入中经过 URI 程序译码的字符串的字节数。由于服务器在将浏览器传来的经过 URI 译码的字符串放入相应 CGI 的标准输入后并没有接着再放一个文件终止符 EOF，所以如果不使用环境变量 CONTENT_LENGTH 提供的字节数，CGI 程序将无法准确读出标准输入中的字符串。

● CONTENT-TYPE

环境变量 CONTENT-TYPE 定义了跟在浏览器发送的请求头后的浏览器所传送的数据类型。如果没有数据传送，该环境变量的值为空。当 Web 页面的表单用 POST 方法向服务器上的 CGI 程序传送数据时，该环境变量的值是“application/x-www-form-urlencoded”。

● REQUEST_METHOD

环境变量 REQUEST_METHOD 存储了浏览器发出的请求头中有关 HTTP 的信息，可能的取值有 Get、Post、Head、Put、Delete、Link 和 Unlink。其中只有两种取值 Get 和 Post 与 CGI 程序有关。

当该环境变量的取值是 Get 时，Web 页面上的表单将用 Get 方法向 CGI 程序传送数据，表单中的数据在 URI 译码完成后被放在请求头中 URI 的后面传给服务器，服务器最终将这些数据放在环境变量 QUERY_STRING 之中。当该环境变量的取值是 Post 时，表单中的数据则被服务器放在 CGI 程序的标准输入中，经过 URL 译码后的数据的字节数被存入环境变量 CONTENT_LENGTH 中。除非不需要从 Web 页面上获取信息，一般任何 CGI 程序都需要使用 REQUEST_METHOD 这个环境变量。

环境变量 REQUEST_METHOD 的其他取值由于不常使用，对于它们的意义本书将不作说明。如要了解，可以参考 WWW 资源“http://www.w3.org/Protocols/HTTP/Methods.html”。

● PATH_INFO

当客户端发出的请求头中的 URL 是针对服务器上的一个 CGI 程序，并且 URL 中在 CGI 程序和 URL 最后的 Query String 之间包含有数据时，环境变量 PATH_INFO 将被服务器设置，其值就等于 CGI 程序和 Query String 之间的数据。对于如下的 URL：

http://www.host.com/cgi-bin/aaa.cgi/doc/a?xx=11

http://www.host.com/cgi-bin/aaa.cgi 表示了 CGI 程序在万维网上的位置，“？”后的字符串为环境变量 QUERY-STRING 的值，“/doc/a ”将被设置成环境变量 PATH_INFO 的值。

在某些情况下环境变量 PATH_INFO 是非常有用的。例如您有一个 CGI 程序需要处理服务器上的多个文件。您可以利用环境变量 PATH_INFO 将不同文件的位置传递给 CGI 程序：

http://www.host.com/cgi-bin/aaa.cgi/doc/b?xx=11

环境变量 PATH_INFO 的值此时是/doc/b，它代表了 CGI 程序所需文件的相对路径，事实上 PATH_INFO 可以传递任何类型的数据给 CGI 程序，不过它通常用来向 CGI 程序传递某些文件的路径信息。

● PATH_TRANSLATED

环境变量 PATH_TRANSLATED 中的值是服务器根据环境变量 PATH_INFO 和 DOCUMENT_ROOT 的值综合得出的。它存储了 PATH_INFO 所代表的相对路径的绝对路径。CGI 程序在 Web 服务器上的移动不会影响环境变量 PATH_TRANSLATED 的使用，当

CGI 程序被移动到另一个 Web 服务器时，可能会影响该环境变量的使用。

● QUERY_STRING

浏览器发给 Web 服务器的 URL 中问号“？”后的字符串全部被服务器设置为环境变量 QUERY_STRING 的值。当 Web 页面的表单用 GET 方法传送数据时，浏览器自动将表单中经过 URI 译码后的数据放在所请求的 CGI 程序的 URL 中问号“？”的后面。当然，您也可以利用在超文本链接后加上问号“？”和所要传送的经过 URI 译码后的数据向相应的 CGI 程序传递信息。

使用环境变量 QUERY_STRING 向服务器上的 CGI 程序传递信息有两个主要的缺点。首先是您向 CGI 程序传递的数据将在浏览器的 Location 文本框中显示出来。其次，环境变量 QUERY_STRING 所能存储的字节数受 Web 服务器的输入缓冲区大小的限制。当浏览器向服务器所传送的字节数超过了服务器的输入缓冲区所能容纳的字节数时就有可能造成输入数据的丢失。每种 Web 服务器的输入缓冲区所能接收的最大字节数大都不同。不过在使用此种方法传递数据时至少应使所传递的数据不超过 1024 字节。

● REMOTE_ADDR

环境变量 REMOTE_ADDR 存储了访问 Web 服务器上 CGI 程序的客户端的 IP 地址。

● REMOTE_HOST

环境变量 REMOTE_HOST 存储了访问 Web 服务器上 CGI 程序的客户端的 Internet 域名。如果客户端没有域名，该环境变量的值将为客户端的 IP 地址或为空。

● REMOTE_IDENT

当客户端的用户名称被浏览器用 IDENT 方法传递给 Web 服务器时，环境变量 REMOTE_IDENT 则存放客户端的用户名称。IDENT 是一种用来验证使用服务器上 CGI 程序的客户端用户身份的协议。只有当 Web 服务器和客户端浏览器同时支持 IDENT 时，才可以在 CGI 程序中使用该环境变量。

● REMOTE_USER

当客户端存取了 Web 服务器上需要进行身份验证的 CGI 程序时，该环境变量将保存客户端使用 CGI 程序时所输入的用户名。如果被激活的 CGI 程序不需要身份验证，该环境变量将为空。

● SCRIPT_FILENAME

环境变量 SCRIPT_FILENAME 存储了包含全部绝对路径的被激活的 CGI 程序的文件名，如果您的 CGI 程序使用相对路径，那就没有必要在程序中使用该环境变量。

● SCRIPT_NAME

环境变量 SCRIPT_NAME 存储了被激活的 CGI 程序的文件名以及它的相对路径（相对于 Web 服务器文件目录树的根目录）。如果您想用 CGI 程序向客户端返回一个 Web 页面，在该 Web 页面上还有一个指向该 CGI 程序的链接。这个环境变量是非常有用的，您可以用如下的语句实现：

```
char * scriptname, *servername;

servername=getnev("SERVER_NAME");
scriptname=getenv("SCRIPT_NAME");
printf("<a href=http:\\%s%s > CGI 程序<\a>\n", servername,scriptname)
```

如果您不使用环境变量而直接在 printf 语句中使用 CGI 程序的名称和相对路径，当 CGI 程序的名称和路径发生改动时，CGI 程序就不能正确执行。

● REQUEST_LINE

该环境变量存储了浏览器发给 Web 服务器的请求头中有关 HTTP 请求的一行字符串，例如：GET http://www.host.com/index.html HTTP/1.0

4.3　和浏览器状态有关的环境变量

最后一类环境变量和客户端用来浏览网页的浏览器有关，它们存储了一些有关浏览器和使用浏览器的用户的信息。表 4-1 列出了这些变量。

表 4-1　和浏览器状态有关的环境变量

环境变量	环境变量的说明
HTTP_ ACCEPT	说明了浏览器本次请求所能接收的数据类型
HTTP_ REFERER	该变量存储了浏览器浏览当前网页前所请求上一个网页的 URL。该变量所起的功能类似于浏览器的 BACK 按钮
HTTP_ USER_AGENT	该变量存储了浏览器软件的类型和版本号
HTTP_ ACCEPT_ENCODING	说明了浏览器所支持的 Encoding 类型
HTTP_ ACCEPT_LANGUAGE	说明了浏览器希望接受的语言的 ISO 代码
HTTP_ AUTHORIZATION	该变量存储客户端用来进行身份验证的用户名
HTTP_ CHARGE_TO	对客户端对网络资源的使用进行记费（留待将来使用）
HTTP_ FROM	存储客户端用户的电子函件地址
HTTP_ IF_MODIFIED_SINCE	该变量存储了一个日期，只有当服务器上浏览器所请求的资源比该日期更新时该资源才被下载

表 4-1 所列出的环境变量中前三个使用频率最高。利用表 4-1 中的环境变量，您可以给 CGI 程序增加很多实用的功能。例如每种浏览器虽然都支持标准的 HTML 语言，但也都对它们做了自己特有的扩充。您可以在 CGI 程序中利用环境变量 HTTP_ USER_AGENT 的值确定客户端使用的浏览器的类型，根据不同的浏览器的类型向客户端传送不同的网页，充分发挥各种浏览器的潜力而又不致于引起兼容性的问题。您还可以利用环境变量 HTTP_ REFERER 的值给 Web 网页加上一个类似浏览器的 BACK 按钮功能的链接和按钮。

4.4　SSI 环境变量和 CGI 环境变量之间的区别

SSI 环境变量和 CGI 环境变量之间的区别通常和具体的服务器产品和版本有关。

一般来说，环境变量和具体的执行进程有关。当一个命令行程序被执行时，它会继承操作系统为它设置的一些环境变量，当一个 CGI 程序被 Web 服务器激活时，它会继承 Web 服务器为它设置的一些环境变量。当用一个 SSI 命令在 Web 页面上执行一个命令行程序时，由于开始是被 Web 服务器激活，它当然可以存取由 Web 服务器设置的环境变量。同时作

为一个命令行程序，它又可以继承系统为命令行程序设置的环境变量。这种情况出现在早期的 Web 服务器产品中，即当使用 SSI 的 exec 执行一个外部命令行程序时，SSI 的环境变量和 CGI 的环境变量有着很大的不同，它还包括系统为它设置的环境变量。

随着 Web 技术的发展，Web 服务器、浏览器和 CGI 程序用来交换信息的环境变量越来越多，为了节省宝贵的输入内存缓冲区，目前绝大多数 Web 服务器产品不再为 SSI 在输入缓冲区中准备系统曾经为命令行程序设置的环境变量。因此现在 SSI 的环境变量和 CGI 的环境变量已经没有什么区别了。事实上存取系统的环境变量对于我们制作动态网页并没有多大的帮助。

本章和本书的第二章介绍了 SSI 和 CGI 所能存取的环境变量，但对于各种不同类型的 Web 服务器产品，则没有一个统一的标准，对于 SSI 变化就更大。不过各类产品一般都是向下兼容，版本越高所能存取的环境变量也越多。如果手头没有 Web 服务器的手册，您可以通过执行一些 SSI 命令或 CGI 程序来了解服务器所支持的环境变量。

例如若 Web 服务器的运行平台是一个 UNIX 系统，通过调用如下的 HTML 文件，可以了解服务器所支持的 SSI 的环境变量。图 4-1 就是由该命令所生成的 Web 网页，它显示了目前 Internet 上最流行的共享 Web 服务器软件 Apache 1.3a1 所支持的 SSI 环境变量。

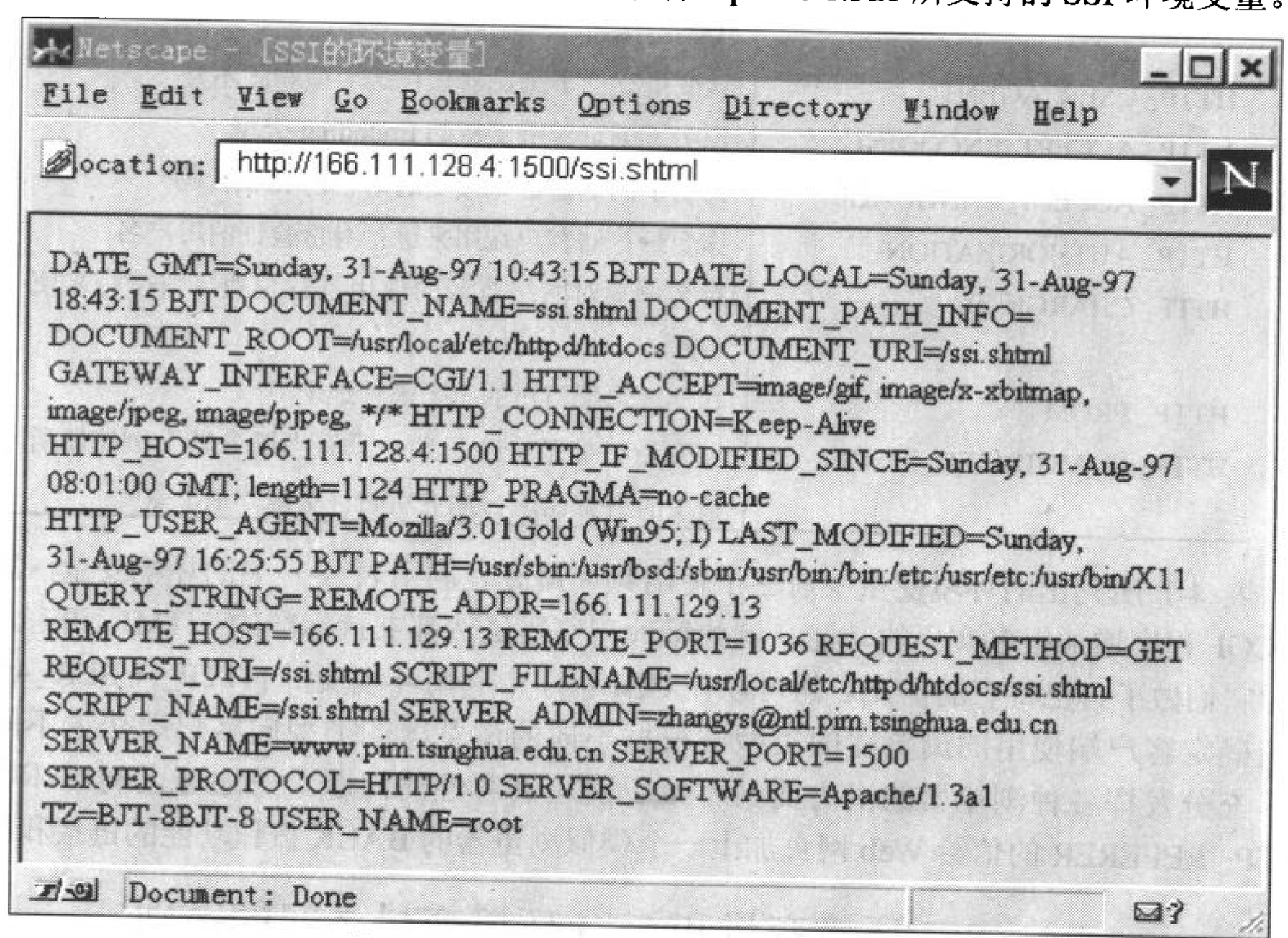

图 4-1　Apache 1.3a1 所支持的 SSI 环境变量

```
<HTML><HEAD>
    <TITLE>SSI 的环境变量</TITLE>
</HEAD><BODY>
<!--# exec cmd="env" -->
</BODY></HTML>
```

为了显示同样一个服务器产品 Apache 1.3a1 所支持的 CGI 环境变量，本书使用了一个由 Apache 自备的 CGI 程序 printenv，printenv 可以输出 Web 服务器所支持的 CGI 环境变量。通过调用如下的 HTML 文件，可以在 Web 页面上显示 Apache 1.3a1 所支持的 CGI 环境变量。该 SSI 命令所产生的 Web 页面如图 4-2 所示。

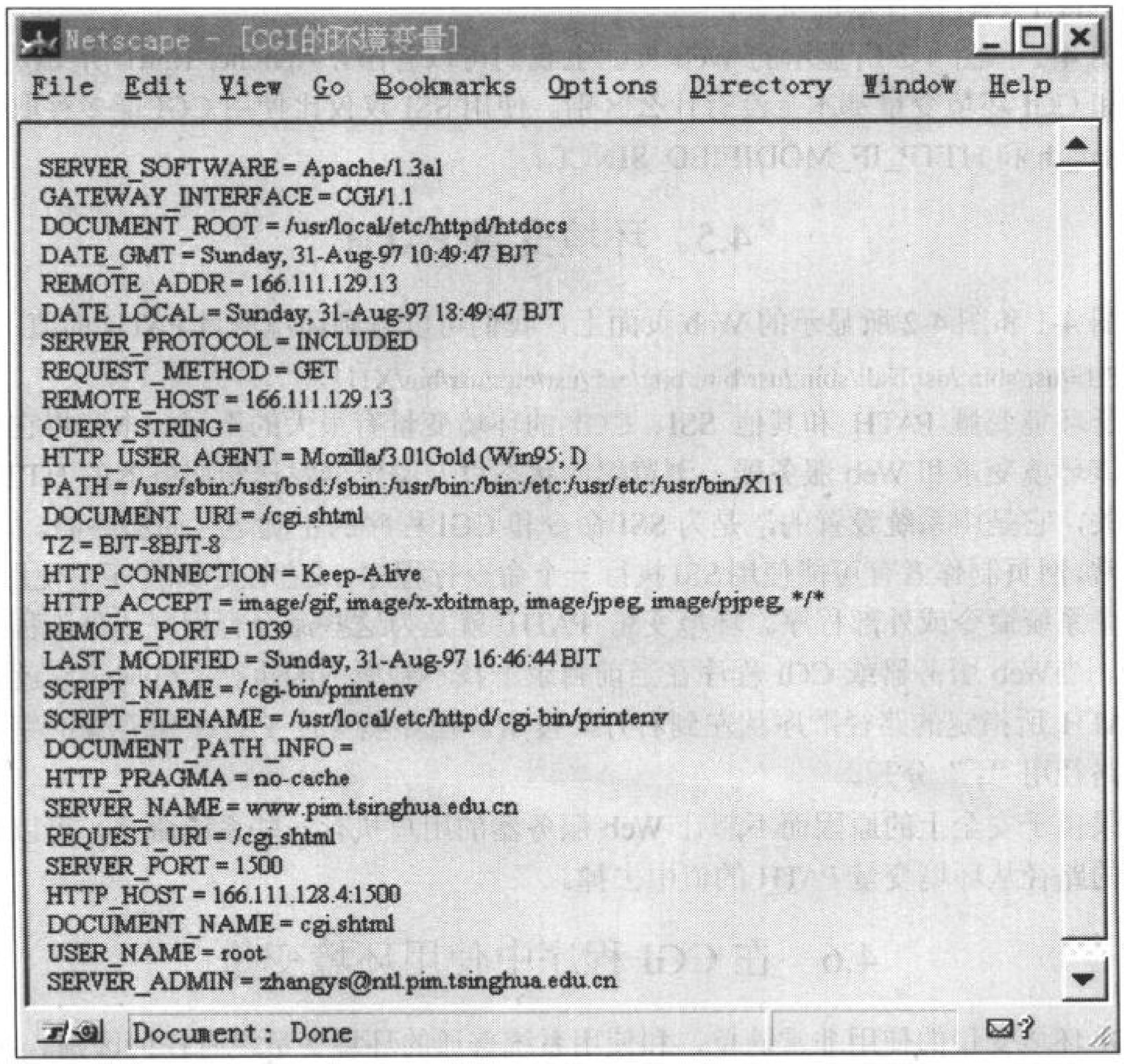

图 4-2　Apache 1.3a1 所支持的 CGI 环境变量

```
<HTML><HEAD>
   <TITLE>CGI 的环境变量</TITLE>
</HEAD><BODY>
<!--# exec cmd="/cgi-bin/printenv" -->
</BODY></HTML>
```

printenv 是一个用 Perl 语言编写的小程序，它的源代码如下所示：

```
#!/usr/local/bin/perl
print "Content-type: text/html\n\n";
while (($key, $val) = each %ENV)
{
print "$key = $val<BR>\n";
}
```

Perl 是一种共享的解释性编程语言，目前可以运行在各种 UNIX 平台和 NT 平台上。该程序的第一行用来告诉 Web 服务器 Perl 解释器的位置，第二行用于输出 MIME 类型，最后两行用于输出环境变量。虽然 printenv 是 Apache 自带的 CGI 程序，只要您的 Web 服务器运行的平台支持 Perl，printenv 都可以运行，使用的时候应使 printenv 的第一行指向您的系统平台上 Perl 解释器的位置。

从图 4-1 和图 4-2 所显示的 Web 页面上我们可以看出，Apache 1.3a1 所支持的 SSI 环境变量和 CGI 环境变量基本上没有什么区别。使用 SSI 仅仅比使用 CGI 能多存取两个环境变量：length 和 HTTP_IF_MODIFIED_SINCE。

4.5　环境变量 PATH

在图 4-1 和图 4-2 所显示的 Web 页面上，我们可以看到环境变量 PATH 的值：

PATH=/usr/sbin:/usr/bsd:/sbin:/usr/bin:/bin:/etc:/usr/etc:/usr/bin/X11

由于环境变量 PATH 和其他 SSI、CGI 的环境变量有很大的不同，本章将它单独列为一节。该环境变量和 Web 服务器、浏览器的状态以及浏览器发送给服务器的 HTTP 请求头信息无关，它是由系统设置的，是为 SSI 命令和 CGI 程序的正确运行而服务的。

通常，网页制作者有可能使用SSI执行一个命令行程序，或者在CGI程序中使用system( )执行一个系统命令或外部程序。环境变量 PATH 就是为这些命令行程序或外部程序指定搜索路径，当Web 服务器或 CGI 程序在当前目录下找不到所需的执行文件时，它就按照环境变量 PATH 所指定的路径顺序从左到右开始搜索。在环境变量 PATH 所存储的字符串中，不同的路径用“:”分开。

如果由于安全上的原因而不想让 Web 服务器的用户执行一些系统命令，可以将这些系统命令的路径从环境变量 PATH 的值中去掉。

4.6　在 CGI 程序中使用环境变量

CGI 环境变量的使用非常简单，和使用系统普通的环境变量没有任何区别。对于 C 语言来说，可以用内部函数:

```
char  *getenv(const char *name )
```

来存取环境变量的值。getenv( ) 函数在 DOS、Windows 95 和 UNIX 下均可以使用，该函数返回指定环境变量的值，如果指定环境变量的值不存在，该函数返回一个空指针。

在 CGI 程序中使用环境变量的一个最重要的目的就是获取客户端用户在 Web 页面上的输入，以下的 C 语言程序代码说明了如何利用环境变量来存取客户端用户在 Web 页面上的输入。

```
char *content, *querystring, *method;
long   length;
method=getenv("REQUEST_METHOD");
if(method=="GET") then content=getenv("QUERY_STRING");
else
 {
      length=atoi(getenv("CONTENT_LENGTH"));
```

```
    content= (char *) malloc(length);
    fread(content, 1, cgiContentLength, stdin);
}
```

以上代码首先根据环境变量 REQUEST_METHOD 的值判断客户端传送数据的方法是 GET 还是 POST，如果是 GET 方法，就直接从环境变量 QUERY_STRING 中读取客户端的输入，如果是 POST 方法，先从环境变量 CONTENT_LENGTH 中读取客户端输入字节数并将其转换为整型，最后根据得到的字节数从标准输入中读出客户端的输入。REQUEST_METHOD、QUERY_STRING 和 CONTENT_LENGTH 是使用频率最高的三个环境变量，除非不需要客户端的输入，否则所有的 CGI 程序都要用到它们。

对于 Visual Basic 语言来说，也可以使用内部函数 Environ({*envstring* | *number*}）来获取环境变量的值。该函数的自变量取值既可以是环境变量的名称，也可以是环境变量在环境变量表中的序号。以下是一个用 Visual Basic 语言编写的 CGI 程序，该 CGI 程序利用函数 Environ 在 Web 页面打印出所有的环境变量。该程序的代码如下：

```
' 声明程序中要用到的两个 Windows API 函数，一个用于处理标准输入输出文件句柄，一个用于向文件中写信息
Declare Function GetStdHandle Lib "kernel32"   (ByVal nStdHandle As Long) As Long
Declare Function WriteFile Lib "kernel32" (ByVal hFile As Long, ByVal lpBuffer As String,
    ByVal nNumberOfBytesToWrite As Long, lpNumberOfBytesWritten As Long,
     lpOverlapped As Any) As Long
Public Const STD_INPUT_HANDLE = -10&
Public Const STD_OUTPUT_HANDLE = -11&
Public Const FILE_BEGIN = 0&
Public hStdOut   As Long    ' 标准输出文件句柄
Sub Main()
Dim BrowserSoftware As String
Dim EnvString(100) As String
Dim rc   As Long
Dim Index As Long, I As Long
Indx = 1
' 运用函数 Environ 获取客户端的浏览器类型
BrowserSoftware = Environ("HTTP_USER_AGENT")
Do
    EnvString(Indx) = Environ(Indx)    ' 获取所有的环境变量
    Indx = Indx + 1
Loop Until EnvString(Indx - 1) = " "
hStdOut = GetStdHandle(STD_OUTPUT_HANDLE) ' 获取标准输出文件句柄
OutPut "Content-type: text/html" & vbCrLf
OutPut "<HTML><HEAD>"
OutPut "<TITLE>CGI 环境变量"
OutPut "</TITLE></HEAD><BODY>"
OutPut "<font size=-1>"
OutPut "<P>客户端的浏览器为:"
```

```
OutPut BrowserSoftware
OutPut "</font>"
OutPut "<ul>"
For I = 1 To Indx - 1
 OutPut "<li> <code>" & EnvString(I) & "</code>"
Next I
OutPut "</BODY></HTML>"
End Sub
Sub OutPut(s As String)
'-----------------------------------------
' 定义一个向标准输出写信息的函数
'-----------------------------------------
Dim lBytesWritten As Long
s = s & vbCrLf
WriteFile hStdOut, s, Len(s), lBytesWritten, ByVal 0&
End Sub
```

CGI 程序所能使用的环境变量，不仅和 Web 服务器有关，而且还和浏览器的类型有关。Web 服务器要根据浏览器发出的请求头设置部分环境变量。有的浏览器发送的请求头包含的信息比较多，有的比较少，因此服务器设置的环境变量会由于浏览器类型的不同而不同，在进行 CGI 编程时应该考虑这个问题。例如同样是网景公司的服务器产品 Netscape FastTrack/2.0a，分别用微软的浏览器 IE4.0 和网景公司的 Navigator/3.01Gold 调用以上用 VB 编写的同一个 CGI 程序，在 Web 页面上输出的环境变量就不一样。CGI 程序所输出的 Web 页面参见图 4-3 和 4-4。

图 4-3　IE 4.0 调用 env.exe 时 env.exe 所输出的环境变量

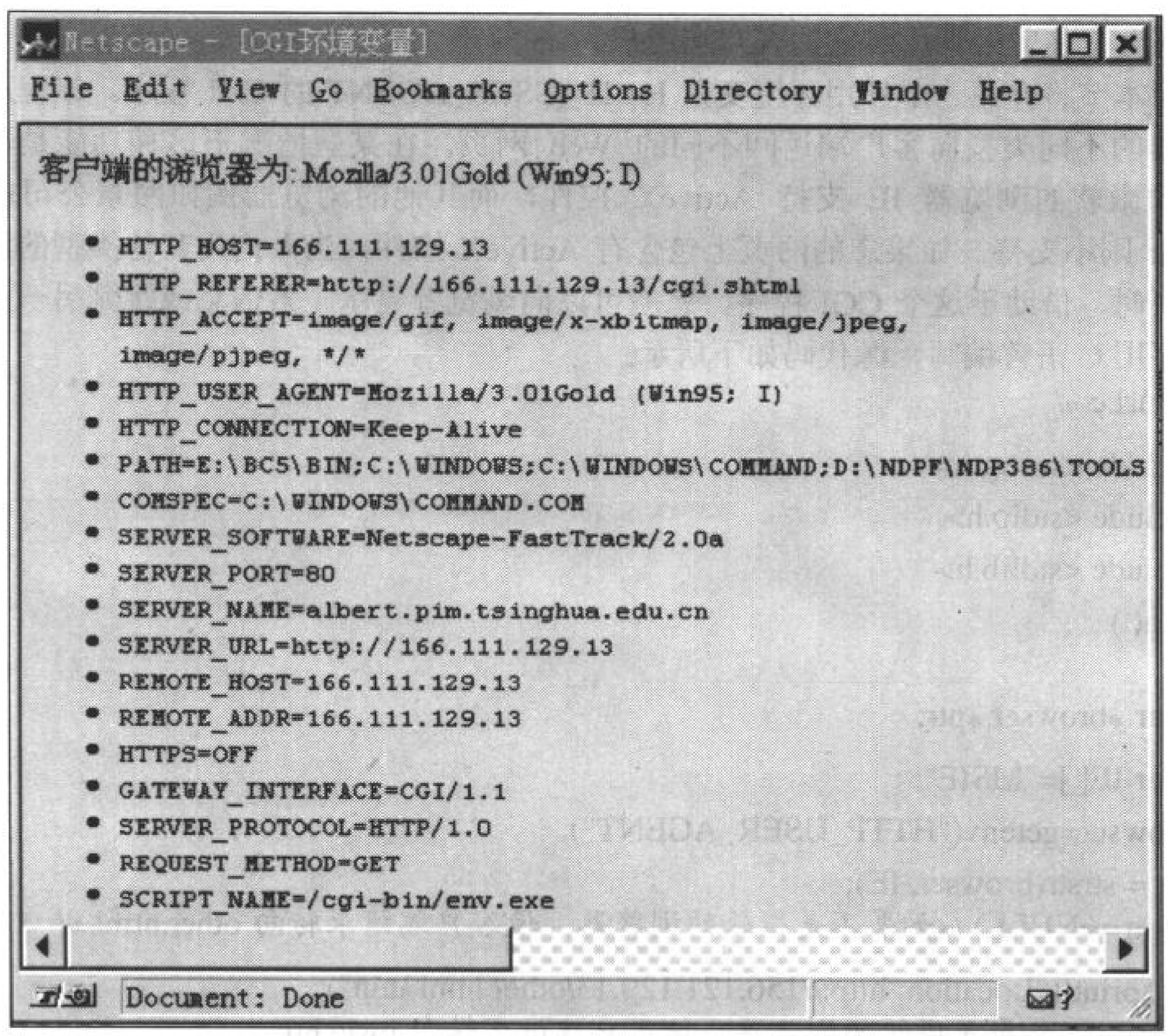

图 4-4　Navigator/3.01Gold 调用 env.exe 时 env.exe 所输出的环境变量

事实上环境变量不仅仅可以用来帮助CGI程序正确读取客户端的输入。在服务器为CGI程序所设置的环境变量中，包含了大量的有关客户端的信息。如果想了解客户端使用的浏览器软件，您可以使用环境变量 HTTP_USER_AGENT，您甚至可以用它编写一个 CGI 程序以统计访问您的 Web 站点的浏览器的类型，这样您就可以比较客观地评价哪一种浏览器的使用用户较多。如果您想了解访问您的主页的人来自世界何方，可以使用环境变量 REMOTE_HOST 或 REMOTE_ADDR。如果您想了解 WWW 上有多少站点的链接指向您的主页，可以使用环境变量 HTTP_ REFERER。客户端在很多情况下是借助某一站点上一网页内的链接到达您的主页的，此时环境变量 HTTP_ REFERER 里就保存着这一网页的 URL。

Web 服务器所设置的环境变量中，有一些储存的是路径和文件名，例如 PATH_INFO 、PATH_TRANSLATED、SCRIPT_FILENAME 和 SCRIPT_NAME 等等。如果 CGI 程序需要处理一些路径和文件名，应该尽量使用这些环境变量去代替具体的名称，这将使您的 CGI 程序在改变存放位置时能继续使用而不用修改源程序。例如一个 CGI 程序生成的 Web 页面要继续调用这个 CGI 程序，在 CGI 程序中您就应该用 SCRIPT_NAME 而不是程序本身的名称来代表它。

当一个 CGI 程序被激活后，Web 服务器会为它设置一系列的环境变量。当这个 CGI 程序运行结束以后，这些环境变量也随着消失。即使过一瞬间这一 CGI 程序又被同样的客户端调用，Web 服务器也要为它重新设置环境变量。除非采取一些特殊的方法，CGI 程序

永远也不会记住曾经调用过它的客户端用户。

最后本节将介绍一个使用环境变量 HTTP_USER_AGENT 的 CGI 程序。该程序根据用户浏览器的不同类型向客户端返回不同的 Web 网页。在某些情况下这种功能是非常重要的。例如微软的浏览器 IE 支持 ActiveX 控件，而其他的浏览器例如网景公司的浏览器 Navigator 则不支持。如果您的网页上包含有 ActiveX 控件，当人们用其他类型的浏览器浏览该网页时，借助于这个 CGI 程序，您就可以向浏览者显示一条信息或跳到另一主页。该 CGI 程序用 C 语言编写，源代码如下所示：

```
/*churl.c */
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
main( )
{
 char *browser,*ptr;
 char IE[ ]="MSIE";
 browser=getenv("HTTP_USER_AGENT");
 ptr = strstr(browser, IE);
 if(ptr==NULL) /*如果不是微软的浏览器，使客户端请求转向 other.html */
     printf("Location: http://156.121.129.13/other.html \n\n");
 else    /*如果是微软的浏览器，使客户端请求转向 msie.html */
     printf("Location: http://156.121.129.13/msie.html \n\n");
 return 1;
 }
```

实现同样功能的用 VB 编写的 CGI 程序源码如下所示：

```
'声明程序中要用到的两个 Windows API 函数，一个用于处理标准输入输出文件句柄，一个用于向文件中写信息
Declare Function GetStdHandle Lib "kernel32"   (ByVal nStdHandle As Long) As Long
Declare Function WriteFile Lib "kernel32"       (ByVal hFile As Long,
    ByVal lpBuffer As String,   ByVal nNumberOfBytesToWrite As Long,
    lpNumberOfBytesWritten As Long,    lpOverlapped As Any) As Long
Public Const STD_OUTPUT_HANDLE = -11&
Public Const FILE_BEGIN = 0&
Public hStdOut As Long    ' 标准输出文件句柄
Sub Main( )
Dim Browser As String
Dim I As Long
hStdOut = GetStdHandle(STD_OUTPUT_HANDLE)
Browser = Environ("HTTP_USER_AGENT")
If InStr(Browser, "MSIE") = 0 Then
```

```
    OutPut "Location: http://166.111.129.13/cgi.shtml" & vbCrLf
Else
    OutPut "Location: http://166.111.129.13/ssi.shtml" & vbCrLf
End If
End Sub
Sub OutPut(s As String)
'-----------------------------------------
' 定义一个向标准输出写信息的函数
'-----------------------------------------
Dim lBytesWritten As Long
s = s & vbCrLf
WriteFile hStdOut, s, Len(s), lBytesWritten, ByVal 0&
End Sub
```

在主页上加上如下的 HTML 语句，就可以让该 CGI 程序发生功效：

```
<A HREF=http://156.121.129.13/cgi-bin/churl.exe> 我的主页 </A>
```

第五章　深入 CGI 编程

本书的第三章、第四章以对一些简单表单的处理为例，逐步介绍了 CGI 编程的基本原理。事实上您现在就可以运用所学的知识开始编写一个复杂的 CGI 应用程序了。CGI 程序一般都具有从上到下的逻辑结构，当一个 CGI 程序被 Web 服务器激活以后就执行相应的任务，任务完成以后这个 CGI 程序也就结束了。即使您使用一些事件驱动的编程语言，如 Visual Basic 编写程序也改变不了 CGI 程序这个面向任务的特性。

本书主要是以 C 语言和 Visual Basic 这两种语言为主来介绍 CGI 编程。虽然由于语言类型和系统平台的不同会使 CGI 编程在细节上会有所不同，但用各种语言实现的 CGI 程序的结构，编程时所要遵循的基本准则却都是相同的。本章将从以下几个方面来讨论 CGI 程序的结构以及 CGI 编程的基本准则：

- 编程语言的选择；
- CGI 程序的基本结构；
- CGI 编程注意事项；
- CGI 程序的测试；
- CGI 程序的出错处理。

5.1　CGI 编程语言的选择

一般来说，只要某种计算机编程语言可以对标准输入输出进行处理，或借助于混合编程可以实现这项功能，它都可以用来进行 CGI 编程。目前市面上的各类编程语言基本上都可以用来进行 CGI 编程。选择 CGI 的编程语言就好比是为完成某项任务而挑选工具，如果您已经掌握了某种工具的使用，并且可以通过这种工具的使用完成任务，您当然没有必要再去为完成任务而学会另一种工具的使用。进行 CGI 编程时应尽量选择自己所熟悉的编程语言，没有必要为了编写 CGI 程序而专门去学习一种语言。

在选择 CGI 编程语言时还应该考虑 Web 服务器的系统平台，有些语言只能在特定的系统平台上运行，例如 Visual Basic 语言适用于 Windows 和 NT 平台，而 UNIX SHELL 语言仅能在 UNIX 系统平台上使用。

选择 CGI 编程语言时另一个需要考虑的是编程语言在万维网上所拥有的 CGI 程序库资源。这些共享程序库提供了大量的 CGI 功能模块，通过对这些功能模块的使用，可以大大减少 CGI 编程的工作量。如果所使用的编程语言的网上资源很少，在进行 CGI 编程时将不得不从字符串的 URI 解码、表单处理等 CGI 编程工作的底层做起。当然，如果您的目的是在网上向人们提供这类语言的共享程序库，那又另当别论。

在进行 CGI 编程时每种语言都会有自己的长处和短处，不会有最好的 CGI 编程语言，但肯定会有最适合于您进行 CGI 编程的语言。本节的以下部分将对目前较流行的 CGI 编程语言进行一些介绍。

5.1.1　UNIX SHELL

SHELL 是 UNIX 的指令翻译器和程序语言工具。在计算机工作时用户实际上就开始和 UNIX 的 SHELL 对话。SHELL 负责解释用户键入的命令并执行它。用户也可以将 SHELL 命令集中保存在一个文件中，就像一个高级语言的程序一样，该文件中的指令集可以完成一个特定的任务。在命令行敲入文件名后，UNIX 的 SHELL 将负责文件中所有指令的解释、执行和连接。这和 DOS 的批处理文件有点像，但功能要强大的多。由于 UNIX SHELL 可以处理标准输入输出以及环境变量，所以 UNIX SHELL 可以用来进行 CGI 编程。

目前在 UNIX 系统上比较流行的 SHELL 有 bsh 和 csh 两种，bsh 是 UNIX 系统的标准 SHELL，和 bsh 相比，csh 在程序语言结构上和 C 语言类似，比较适用于交互式环境。观察/etc 目录下的 passwd 文件，就可以知道自己能使用哪种 SHELL。

在编写一些具有特定功能的小程序和对文本文件进行检索时，UNIX SHELL 使用起来非常简单和有效，这就使得它非常适合于 CGI 编程。UNIX SHELL 的另一个优点是可以使用 UNIX 系统本身所具有的一些命令程序，比如使用 UNIX 的系统命令 cal 就可以在 Web 页面上生成某一月的月历，而若用其他编程语言实现这项功能，可能会需要很多行代码。由于最初的 Web 服务器都使用 UNIX 系统平台，因此在网上可以找到很多有关 UNIX SHELL 的 CGI 编程资源，不过它现在有被 C 语言和 Perl 语言代替的趋势。

UNIX SHELL 主要是通过系统的一系列命令来完成相应的任务，这些命令可以被连接到另一个程序上，这通常会给系统带来安全上的问题。同时，由于 SHELL 是一种解释性语言，黑客们在获取了 SHELL 程序的脚本文件时，可以仔细的研究分析它以找出程序的漏洞并借此攻击系统，如果这个脚本文件还是一个 CGI 程序，那就更危险了。

5.1.2　Perl

和 UNIX 的 SHELL 类似，Perl 也是一种解释性语言。Perl 的英文全称是 Practical Extraction and Report Language，它是一种专门用来处理数据和报表的编程语言。Perl 克服了 C 语言和 UNIX SHELL 语言在某些方面的缺陷。某些功能用 C 语言编写可能会要很多行程序代码，但 Perl 却可以很容易实现。Perl 语言在字符串的处理和对大型文本文件的检索方面尤为突出，而这正是大部分 CGI 程序所要完成的工作。Perl 语言编写的程序也可以用来进行系统管理和维护。

一般可以通过后缀名.pl 来识别 Perl 程序，Perl 程序的执行需要一个解释器，通常在 Perl 程序的第一行要告诉程序系统中 Perl 解释器的位置，否则程序不能正确执行。和 SHELL 语言一样，在 Perl 程序中也可以调用 UNIX 的系统命令，Perl 语言还可以像 C 语言一样编制和使用程序模块。

使用其他编程语言时，您要为购买软件和软件升级支出一定的费用。但 Perl 就不一样了，它是一个完全免费的共享软件，您随时可以在网上得到它的最新版本。经过多年的发展，Perl 已经从 UNIX 系统移植到目前流行的各种系统平台上，比如 MS-DOS、OS/2 Warp、VMS、Windows NT、Windows 95 和 Macintosh 等。使用 Perl 编写的程序经过少量的修改就可以在各种平台上运行，当然这也是一个十分诱人的长处。

可以说，Perl 是目前万维网上使用最广泛的 CGI 编程语言，网上有关 Perl 语言的 CGI 共享资源也是最丰富的。

像其他解释性编程语言一样，当 Perl 程序的脚本文件落入黑客之手时是非常危险的，

除非您在编程时十分注意程序的安全问题，让黑客无机可乘。

虽然 Perl 程序的脚本是一个文本文件，但对一个初学者来说要读懂一个 Perl 程序还是比较困难的。Perl 使用了一些和其他编程语言非常不同的符号、方法，即使是编写 Perl 程序的人，在一段时间不用 Perl 后也可能读不懂他以前编写的程序。

5.1.3 TCL

和 UNIX SHELL 和 Perl 一样，由美国加州伯克力大学的 John Ousterhout 开发出的 TCL 也是一种解释性语言，它的英文全称是 Tool Command Language。和其他解释性语言不一样，TCL 还具有图形功能，利用 TCL 提供的图形开发工具 TK，编程人员可以很容易就开发出一些 X-windows 应用程序。

TCL 语言具有很简单的语法结构，每一行程序都由一个 TCL 命令开始，后面跟着该命令的使用参数。所有 TCL 命令都被 TCL 解释器理解为一个 C 函数，命令的使用参数被理解为该函数的调用参数，可以用 C 语言编写自己的 TCL 命令，也可以用 TCL 命令 proc 编写 TCL 过程。TCL 的解释器实际上也是一组函数，在程序中同样也可以调用这些函数。TCL 程序可以自动加载程序库中的程序，在某些系统平台上还支持加载动态连接库。实现同样功能，TCL 的速度要比 Perl 和 UNIX SHELL 要快一些，但比 C 语言慢。TCL 的安全性比 Perl 也要好一些。

和 Perl 一样，TCL 也是一种免费的编程语言，在网上可以获取它的最新版本，不过目前 TCL 所支持的平台比 Perl 要少得多，现在 TCL 能运行在大多数 UNIX 系统平台、DOS 和 Macintosh 平台上。

虽然使用 TCL 进行 CGI 编程的人在缓慢的增加，但 TCL 在 CGI 编程中的运用还是大大落后于其他各种 CGI 编程语言，在网上所能找到的有关 TCL 的 CGI 共享资源也是较少的。

5.1.4 C 语言

1967 年英国伦敦剑桥大学的 Martin Richards 在 CPL 语言的基础上推出了 BCPL 语言，1970 年美国贝尔实验室的 Ken Thompson 以 BCPL 语言为基础，设计出很简单而且很接近硬件的 B 语言。1973 年贝尔实验室的 D.M. Ritchie 在 B 语言的基础上设计出了 C 语言。最初的 C 语言只是一种描述和实现 UNIX 系统的工作语言，主要还是在贝尔实验室内部使用。一直到 1975 年 C 语言的突出优点才引起了人们的注意。随后 C 语言被移植到各种软硬件平台上，目前它已经成为世界上应用最广泛的编程语言之一。1983 年美国国家标准化协会根据 C 语言问世以来各种版本对 C 语言的扩充和发展，制订了有关 C 语言的最新标准，即 ANSI C。目前各种系统平台上的 C 语言都支持 ANSI C。在用 C 语言编写 CGI 程序时应尽量按 ANSI C 的标准编写代码，这将使 CGI 程序很容易在各种系统平台之间移植。

C 语言简洁紧凑，使用方便、灵活，对程序的语法结构要求不是很严，这就使得编程人员在编程时具有很大的灵活性，可以设计出自己风格的程序。

不像 UNIX SHELL、Perl 和 TCL，C 语言是一种编译语言，源程序代码要被系统的编译器翻译成机器能直接执行的二进制代码，因此用 C 语言编写的 CGI 程序的运行速度要比用解释性语言编写的程序快。使用编译语言的另一个好处是即使 CGI 执行程序落入黑客之

手，他们也无法像分析用解释性语言编写的 CGI 程序那样找出程序中的漏洞。

现在在网上使用 C 语言进行 CGI 编程的人数仅次于 Perl，有关 C 语言的 CGI 程序库网上资源也非常丰富。

由于 C 语言最初是针对系统设计的，这使得 C 语言的字符串处理能力比较差，如果 CGI 程序需要对字符串进行一些复杂的操作，用 C 语言实现起来将比较麻烦。

5.1.5　C++

CGI 编程时另一个经常使用的编译语言是 C++，它是由 C 语言发展而来的。虽然 C++继承了 C 语言的所有优点，例如处理软硬件接口和系统底层的灵活性、语言的紧凑性和强有力的表达式功能，但 C++还是一种完全和 C 语言不同的全新语言，它为编程人员提供了面向对象的工作平台。用 C++进行编程，程序员不仅要学会使用一种新的语言，而且要掌握一种全新的编程方法和思想。

使用 C++进行编程的一个好处是程序员书写的代码可以重复使用，这就可以大大提高程序的开发速度。那些可以重复使用的代码在 C++中被称为类，每种类都有自己的属性和方法，一种类可以继承其他类的方法和属性。编程人员利用很少的代码就可以利用现存的类构造出自己的类或程序，这就是所谓的面向对象的程序设计。利用面向对象的程序设计方法，编程人员在设计和维护大型软件方面可以节省很多精力。如果您手中现在已经有了一个专门处理 CGI 编程的类，那在进行 CGI 编程时是不需要花费太多精力的。

很多系统平台都提供了 C++编译器，和 C 语言一样，C++源代码只要经过少量修改就可以在多种平台上运行。由于 C++是从 C 语言继承发展而来的，大多数用 C 语言编写的程序也可以在 C++环境中运行。

C++处理字符串的能力比 C 语言要强些，但还是比不上 Perl。有关 C++的 CGI 网上共享资源比 C 语言要少。

5.1.6　Visual Basic

自从微软推出视窗操作系统以来，视窗系统软件所具有的图形用户界面一直是程序员所追求的目标。在 Visual Basic 没有出现以前，一个视窗软件中需要大量的十分难懂的代码来处理图形界面，这使得很多非专业的程序员对视窗编程望而却步。Visual Basic 的出现改变了这一状况，使用 Visual Basic，人们可以轻松自如地编写视窗应用软件。目前 Visuai Basic 已经成为视窗系统上拥有程序员最多的语言之一。

Visual Basic 是一个运行于 Windows 3.1、Windows 95 和 NT 系统平台的编程环境。事实上它不仅仅可以用来开发视窗应用软件的图形用户界面。随着对 Visual Basic 这个编程环境的熟悉和了解程度的增加，您能够利用它做的事情也将越来越多。微软的总裁比尔·盖茨曾经说过用 C 语言能够实现的事情 Visual Basic 都能实现，但 Visual Basic 的使用显然比 C 语言要方便的多。

使用 Visual Basic 进行 CGI 编程的一个十分明显的优点是可以充分享受 Visual Basic 内在的数据库编程环境所带来的好处，可以利用 Visual Basic 提供的各种数据库对象方法或第三方厂家提供的动态库，使 CGI 程序实现对外部数据库的查询和更新。这个数据库可以是一个桌面数据库系统，也可以是一个客户/服务器模式的数据库系统。数据库既可以和

Web 服务器在同一台机器上，也可以在不同的机器上。另外，Visual Basic 也具有较强的字符串处理能力。

虽然如此，使用 Visual Basic 编写 CGI 程序还是有一定代价的。它的代码运行速度没有 C 语言快。并且，由于 Visual Basic 自身和视窗系统的紧密结合，用它所编写的程序只能运行在微软的视窗系统上，而目前万维网上大多数 Web 服务器都工作于各种 UNIX 系统平台。在 Web 系统上对多国语言的支持能力 Visual Basic 也没有 C 语言强。

不过随着微软的 NT 服务器在市场上所占份额的不断增加和 Visual Basic 良好的客户服务器框架的编程能力，Visual Basic 也将会成为一种流行的 CGI 编程语言，它在网上的共享 CGI 程序库资源也将越来越多。

表 5-1 比较了各种 CGI 编程语言的一些情况。

表 5-1　CGI 的编程语言的比较

编程语言	UNIX	VMS	NT	WIN95	OS/2	共享资源
UNIX Shell	可以	否	否	否	可以	一般
Perl	可以	可以	可以	可以	可以	丰富
TCL	可以	可以	可以	可以	否	少
C/C++	可以	可以	可以	可以	可以	丰富
Visual Basic	否	否	可以	可以	否	一般

5.2　CGI 程序的基本框架

正像前两章中一些 CGI 程序的例子所显示的那样，一个 CGI 程序的基本框架是非常简单的，它一般都具有从上到下的线性结构，程序只是按照一定的流程执行某种特定的任务，任务完成以后这个进程的生命也就结束了。通常，一个 CGI 程序由以下几部分组成：

- 程序的初始化准备工作；
- 数据处理；
- 输出处理结果；
- 程序结束。

5.2.1　程序的初始化准备工作

一个 CGI 程序被激活以后，它首先要做的事情就是确定系统平台、Web 服务器和客户端浏览器的状态信息、客户端用户的输入方式和输入的数据，否则它将一事无成。这些有关 CGI 程序运行的信息都是通过环境变量和标准输入获得的。任何编程语言编写的 CGI 程序，都可以像普通程序一样使用环境变量和标准输入。一个 CGI 程序的初始化工作通常由以下步骤组成：

（1）获取程序所需要的所有环境变量的值并为它们分配内存空间。

（2）如果环境变量 REQUEST_METHOD 的值是“GET”，则从环境变量 QUERY_STRING 中读取用户的输入。如果环境变量 QUERY_STRING 的值不为空，则对它的值进行 URI 解码。此时 CGI 程序初始化部分的工作完成。

（3）如果环境变量 REQUEST_METHOD 的值是“POST”，则从环境变量 QUERY_STRING 中读取信息。如果环境变量 QUERY_STRING 的值不为空，则对它的值进行 URI 解码。获取环境变量 CONTENT_LENGTH 的值，如果该值不为空，将它转换为整型，并从标准输入中读取相应的字节数。

（4）将从标准输入中得到的字符串根据表单元素的属性分解成变量和变量值的形式。

（5）获取环境变量 CONTENT_TYPE 的值，如果它的值为“application/x-www-form-urlencoded”，则对上一步获得的变量值进行 URI 解码。注意环境变量 CONTENT_TYPE 目前就只有这一种取值，但为了使您的 CGI 程序在将来能够适应通用网关规范和 HTTP 协议的扩充，在 CGI 程序中应该对它的值进行判断。

（6）如果环境变量 REQUEST_METHOD 的值既不是“GET”也不是“POST”，则显示一个错误信息。这一步通常不能省略，因为该环境变量也可能有其他的取值。

（7）释放所有为环境变量分配的内存块。

（8）结束初始化准备工作。

通常完成这部分工作的 CGI 程序代码在网上一些 CGI 共享资源中都可以找到。因为它们对所有的 CGI 程序都是通用的，使用它们将可以大大降低 CGI 编程的工作量。即使是准备亲自动手编写以上这些步骤的代码，这些 CGI 共享资源也还是有参考价值的。

5.2.2 数据处理

一般，所有 CGI 程序的初始化工作都大同小异，但每个 CGI 程序的数据处理过程就大不一样了，这也正是一个 CGI 程序的核心。

一个 CGI 程序可能会忽略客户端的输入而立刻向浏览器传送信息，或者可能根据用户的输入作出一些判断后再向浏览器传送信息。当然，它可能什么也不做，仅仅把用户的输入转送给 Web 系统外的另一个服务器系统，比如某一数据库系统，再从那里接收处理结果并将结果格式化以后传送给客户端的用户。

通常，CGI 程序这部分的编程和普通的程序编程没有什么两样，只是在对文件进行读写操作时要考虑到可能会有很多个客户端用户同时调用该 CGI 程序，同时对文件进行读写操作。这时，CGI 程序在打开文件以后要对文件进行加锁处理，否则，前面用户向文件写入的信息可能会被最后一个用户覆盖掉，从而造成信息的丢失。如果程序只是对文件中的某一部分记录读写，则程序只应对文件中的这一部分而不是全部加锁。

5.2.3 程序处理结果的输出

任何 CGI 程序都必须有输出，虽然并不是每个 CGI 程序的输出都在 Web 页面上显示出来。

一个 CGI 程序的输出通常有两部分，即 CGI 程序所生成的部分或全部 HTTP 响应头信息和 CGI 程序的正常输出（也就是在 Web 页面上显示的部分）。CGI 程序生成的 HTTP 响应头信息主要是用来说明 CGI 程序输出结果的 MIME 类型。一个 CGI 程序可以不向客户端的 Web 页面输出任何信息，但必须输出部分或全部的 HTTP 响应头信息。如果 CGI 程序只输出了部分 HTTP 响应头信息，则 Web 服务器在接到了 CGI 传来的部分头信息以后，将把它处理成一个完整的 HTTP 响应头信息后传送给客户端浏览器。如果 CGI 程序输

出的是完整的 HTTP 响应头信息，CGI 程序将通知 Web 服务器不用在对此进行处理。通知的方式就是在程序的名称前加上前缀“NPH-”或“nph-”，否则，程序运行时将会出错。

在 CGI 程序输出的 HTTP 响应头信息和 CGI 程序的正常输出之间必须有一空行，否则程序运行时也会出错。

以下就是一个 perl CGI 程序的输出部分，它由两部分组成，一部分是 HTTP 响应头信息，一部分是 CGI 程序的处理结果，在这两部分之间有一空行“\n\n ”。

```
print "Content-type: text/html\n\n";
while (($key, $val) = each %ENV)
 {
print "$key = $val<BR>\n";
 }
```

5.2.4 程序的结束

CGI 程序的结束部分和普通程序一样。上了锁的文件应该解锁，打开了的文件应该关闭，分配出去的内存空间应该重新收回。有的系统平台在程序结束时会自动为您做这些事，但有的则不行。为了让 CGI 程序尽可能的在各种平台下都有一定的健壮性，每一个 CGI 程序都应该主动完成这些工作，而且，实现这些工作所需的代码是非常少的。

5.3 CGI 编程的注意事项

由于 CGI 程序的运行环境和使用者与普通程序有很大不同，CGI 编程通常比普通编程要注意更多的事项。

5.3.1 程序的可维护性

CGI 编程时首先要注意的问题是程序的可维护性。虽然这是所有编程工作都应注意的问题，但对于 CGI 编程来说尤其重要。Web 页面经常需要更新，与之相对应的 CGI 程序有时也需要更新，或者想给以前的 CGI 程序增加新的功能，这些都需要修改 CGI 程序的源代码。

在编写 CGI 程序时，每一个子程序和函数的功能都应该用注释行说明，而变量、子程序和函数的取名应尽量和表单元素的 NAME 属性或它本身所具有的含义相一致，比如存取环境变量 QUERY_STRING 的字符串变量名可以取名为“wQueryString”，存取环境变量 CONTENT_LENGTH 的变量名可以取名为“lContentLength”。CGI 程序也应遵循结构化程序设计的一般准则，程序的逻辑结构应清晰明了。程序中应尽可能将一些具体事务的处理交给子程序和函数去完成。如果在 CGI 编程的初期省略了这些工作，过了一两个月当再想修改程序时，您可能要花很大的精力才能读懂以前的程序代码。

5.3.2 程序的安全性

CGI 程序是放在 WWW 上让任何可以接触它的用户使用的，您无法保证 CGI 程序的所有用户都是善意的。安全性是 CGI 编程要考虑的另一个问题。在 CGI 程序中尽量不要让客户端用户具有更改系统平台的某些变量值的能力，客户端用户所能修改的应是那些仅

仅和本次连接有关的环境变量。如果在 CGI 程序中需要使用系统的外部命令或程序（比如在 C 语言中使用 system( )函数），应该尽量让 CGI 程序自己提供命令行参数或变量，如果需要用户提供，在 CGI 程序中应该对用户的输入进行严格的检查。

CGI 程序还应能对用户在表单内的输入信息进行检查，它应能识别一些无用的信息并把它们放弃，比如无效的电子函件地址和电话号码。CGI 程序应该给每一个存取用户输入数据的变量分配合适的字节数，当用户的输入超过这一长度时就拒绝处理。这一点是非常重要的。因为一个专门攻击计算机系统的黑客可能会编写一个类似 Web 浏览器的程序，通过这个程序向您的 CGI 程序发送几十兆甚至上百兆的字节，如果 CGI 程序没有限定存取客户端用户输入数据的变量的内存空间大小，此时这个 CGI 程序就会由于输入缓冲区的溢出而立刻崩溃。更危险的是，CGI 程序的崩溃将会给黑客带来很多攻击系统的机会。

当 CGI 程序编写调试完毕以后，还应对该程序的权限做正确的设置，通常 UNIX 系统平台上的 Web 服务器激活的所有 CGI 进程的属主都是“nobody”，“nobody”的权限在系统上受到很大的限制。UNIX 系统上一个正确的 CGI 程序的权限设置如下所示：

```
-rwx------ 1 nobody user 8836  6 月 13 日  1996 年 table.cgi
```

这将能保证所有的客户端用户都可以执行 CGI 程序，而系统平台上无关的用户却无法执行和阅读该 CGI 程序，这一点对那些用 SHELL 和 Perl 等解释性语言编写的 CGI 程序的安全性尤其重要。

5.3.3　CGI 程序和系统平台的无关性

CGI 编程另一个需要注意的问题是 CGI 程序应尽量保持和系统平台的无关性。目前只有 C 和 Perl 这两种语言能满足这种要求，当然并不是说其他语言就不能用了，因为和系统平台的无关性只是选择 CGI 编程语言的准则之一，选择 CGI 编程语言还要考虑程序的可维护性、语言使用的难易与方便、程序所要完成的任务以及个人的爱好等多种因素。

如果和系统平台的无关性是 CGI 程序要考虑的首选目标，C 语言无疑将是一个最佳的语言。如果在编程时遵循 ANSI C 的标准而不使用那些和具体的系统平台有关的函数，CGI 程序的源代码可以不经修改就在各种系统平台上使用。这还可以给编程者带来另一个好处，即使 CGI 程序最终的运行平台是 UNIX 系统平台，编程人员也可以在 PC 机上调试好了以后，再将程序放到 UNIX 平台上编译，这样就可以充分利用目前 PC 机上有关 C 语言的集成编译环境（如 VC、BC 等）的强大调试功能，弥补 UNIX 系统编译器的缺陷。

当然，只使用 ANSI C 而不使用那些和具体的系统平台有关的函数肯定会使您的 CGI 程序的功能逊色不少，有时在程序中还不得不使用这些函数，此时应该将程序中和具体系统平台有关的代码放在一个函数中。可以为各种平台开发不同版本的函数，这将使 CGI 程序在各种平台间的移植工作量大大减少。

虽然就平台无关性来说 Perl 也是一种可选语言，但各个版本的 perl 语言程序库常有一些差别，这常常给 CGI 编程带来一些麻烦。

5.3.4　CGI 程序和 Web 服务器的无关性

比系统平台无关性更重要的一个准则是 CGI 程序和具体的 Web 服务器产品的无关性，CGI 程序应该在不经修改的情况下就能和同一系统平台上的各种 Web 服务器产品一起正常

工作。如果您想在网上发布一些共享软件或商用软件，就必须考虑这个问题。

由于所有的 Web 服务器都遵循标准的通用网关接口规范，CGI 程序和具体的 Web 服务器产品的无关性实现起来是非常容易的。通常在进行 CGI 编程时要注意以下问题。不要使用服务器所特有的环境变量，即使使用也应该在程序中考虑该环境变量不存在的情况。不要在 CGI 程序中使用有关路径名、文件名和服务器 IP 地址等的具体名称，应该尽量用 Web 服务器设置的环境变量值来代替具体的名称。在 CGI 程序中不应该对 Web 服务器、系统平台以及网络的状态做任何假设。

5.3.5 CGI 程序输出的一致性

最后，CGI 程序还应该考虑它的输出在各种类型浏览器上的显示问题，应尽量使 CGI 程序所生成的 Web 页面在不同平台下的各种浏览器上都有一致的显示效果。通常要做到这一点很困难。同一 HTML 文件在不同浏览器下浏览肯定是有差别的。对一个 CGI 程序来说，不应该使它的输出在某一浏览器下很好看而在另一种浏览器下却很丑陋。必要时 CGI 程序应该根据不同的浏览器类型输出不同的 Web 页面。

总而言之，编写 CGI 程序时应着重注意以下问题，即程序的可维护性、程序的安全性、程序与系统平台和具体的 Web 服务器产品的无关性以及程序的输出在各类浏览器上的显示一致性问题。

5.4 CGI 程序的测试

测试是任何软件编程都必不可少的阶段，有很多业余的软件编程人员都忽略这个阶段。当然如果编写的程序仅仅是给自己使用，这不会有什么问题，因为程序的拥有者会在以后的使用过程中不断地纠正发现的问题，这实际上就是一个测试的过程。但 CGI 程序就不同了，它是要放在万维网上给无数个用户使用的。CGI 程序出了问题，不仅会影响您的 Web 站点的使用，有时还会给您的系统带来安全性的问题。因此一个 CGI 程序在投入使用以前必须经过良好的测试。

CGI 程序是由 Web 服务器激活的，运行时它从 Web 服务器为它设置的环境和标准输入中获取信息，运行结束时将输出结果放入标准输出中并由服务器再传给相应的客户端浏览器。因此 CGI 程序的测试和普通的程序有所区别。在 CGI 程序测试之前必须专门为它构造一个运行环境，即在另外一个系统平台上再建立一个类似的 Web 服务器，如果所编写的 CGI 程序和系统平台无关，也可以将测试用的服务器建在另一种系统平台上。您当然不能在正在对外提供信息发布的 Web 服务器上对 CGI 程序进行测试。

也许 CGI 程序一开始就能顺利运行，但通常情况下它总要遇到一些麻烦。不过幸运的是 Web 服务器会在 CGI 程序运行出错时显示一些错误信息。

Web 服务器显示的错误信息代码一般有两类，即 4xx 系列和 5xx 系列。4xx 的错误通常是由于浏览器发出的请求不合法或浏览器的请求超越了相应的权限引起的。当服务器无法满足浏览器的合理请求时，错误信息的代码就是 5xx。CGI 程序不正确的运行和设置都可能导致这两种错误的发生。由 CGI 程序所引起的 4xx 和 5xx 系列的错误通常有 403、404 和 500。

403 的错误是指服务器理解浏览器的请求但由于某种原因服务器不能满足浏览器的请求。如果是在浏览器请求 CGI 程序时发生这种问题，对于 UNIX 系统来说，通常是 Web 服务器产生的进程的属主（一般是 nobody）对在目录“CGI-BIN”中相应的 CGI 程序没有读和执行的权利，赋予 nobody 读和执行该程序的权利就可以解决这类问题。注意为了安全起见，除了 nobody，任何人都应该没有读和执行 CGI 程序的权利。404 的错误是指服务器没有找到浏览器所请求的 HTML 文件或 CGI 程序。通常这是由于在 HTML 文件中错误书写了 CGI 程序的 URL 或相应的文件、程序放错了地方造成的。

CGI 程序运行时最可能遇到的错误是 500，500 的含义是服务器遇到了一个内部的错误，当然不是服务器而肯定是 CGI 程序有问题。更详细的错误信息可以查阅 Web 服务器的日志文件 error.log。如果是 CGI 程序生成的 HTTP 响应头信息有问题，服务器通常会在 error.log 文件中指出来。引起这种错误的原因通常有两种，一是 CGI 程序在输出头信息后没有接着输出一空行，一是 CGI 程序自己生成了全部必须的 HTTP 响应头信息却没有给程序的名称加上前缀“NPH-”或“nph-”。

对于由于其他原因而引起的 500 错误，Web 服务器将无法具体指出，这通常是 CGI 程序在运行时发生的错误。一般，CGI 程序的错误不会发生在从环境变量和标准输入中读信息的阶段，因为有关这部分的程序代码网上有很多公共的库可以利用，大部分 CGI 程序都使用这类库完成读入数据的任务。即使不使用这类公共的库，由于程序代码的结构比较简单，也可以通过对比很容易找出这部分代码的错误。CGI 程序的错误一般都发生在对读入的数据进行处理的阶段。

如果 CGI 程序有一定的规模，仅仅依靠阅读检查程序的源代码是很难找到这类错误的原因的。这时，我们可以利用一些工具（比如集成编译环境下的调试器）对 CGI 程序进行调试。当然，此时的 CGI 程序不能由 Web 服务器激活而是应当像一个普通程序一样被执行。在执行这个 CGI 程序之前，我们应当模拟 Web 服务器为 CGI 程序设置一个和它出错时基本一样的运行环境。这可以利用设置环境变量的系统命令实现。在 UNIX 系统、NT 系统、Windows 95、DOS 下都有这种命令。比如在 DOS 或 Windows 95 下可以用下列命令为 CGI 程序设置一个仿真运行环境：

```
SET HTTP_HOST=166.111.129.13
SET HTTP_REFERER=http://166.111.129.13/cgi.shtml
SET HTTP_ACCEPT=image/gif, image/x-xbitmap, image/jpeg, image/pjpeg,
SET HTTP_USER_AGENT=Mozilla/3.01Gold (Win95; I)
SET HTTP_CONNECTION=Keep-Alive
SET QUERY_STRING= cname=zhang&ZhiWu=Prof&ZhiCheng=%BD%CC%CA%DA
SET SERVER_SOFTWARE=Netscape-FastTrack/2.0a
SET SERVER_PORT=80
SET SERVER_NAME=albert.pim.tsinghua.edu.cn
SET SERVER_URL=http://166.111.129.13
SET REMOTE_HOST=166.111.129.13
SET REMOTE_ADDR=166.111.129.13
SET HTTPS=OFF
```

```
SET GATEWAY_INTERFACE=CGI/1.1
SET SERVER_PROTOCOL=HTTP/1.0
SET REQUEST_METHOD=GET
SET SCRIPT_NAME=/cgi-bin/mycgi.exe
```

用一个批处理文件执行以上命令，CGI 程序就可以在命令行或集成环境下直接运行而不用由服务器激活了，此时就可以利用各种调试器对程序进行跟踪、调试。

如果客户端向 CGI 传送数据的方式是 POST，还应该编写一个小 CGI 程序，该程序可以将客户端用户在 Web 页面上的输入数据存入一个文件中。被调试的 CGI 程序可以直接从该文件中读入数据或借助于输入重新定向语句：mycgi.exe < cgiinput.dat 来读入数据。

CGI 程序测试时应该注意的另一个问题是应检查 CGI 程序同时被多个用户使用时的运行情况，尤其是当 CGI 程序要对一个文件读写操作或对数据库进行查询更新时。

5.5 CGI 程序的出错处理

CGI 程序测试通过以后，就可以放在服务器上运行了。不过 CGI 程序的测试通过并不表示 CGI 程序在以后的运行过程中不会遇到问题。

此时 CGI 程序遇到的问题通常和程序的结构以及代码的正确性无关。这种错误一种是由于客户端用户错误的输入造成的，另一种是特殊情况下系统函数调用的失败造成的，比如没有成功的打开一个文件。

对于第一类问题，CGI 程序在使用客户端用户的输入之前必须进行合法性检查，CGI 程序应尽可能设想用户错误输入的各种形式，当用户输入了非法的数据时 CGI 程序应能够向用户发回适当的说明信息。

对于第二类问题，CGI 程序在每一次系统函数调用后都应该检查调用是否成功，如果不成功，应该向客户端的 Web 页面返回一些相应的信息。对于 CGI 程序中编程人员编写的函数，也应该有是否成功执行的标志和相应的返回信息。比如，一个用 C 语言编写的 CGI 程序在使用 system( )函数调用系统命令 sendmail 将客户端用户的输入信息传送给 Web 服务器管理员时，应该检查 sendmail 是否成功的执行，如果由于某种原因没有成功，CGI 程序应该建议用户过一段时间再试或利用别的邮件服务器将信息发给管理员。

在 CGI 程序中加入这些处理程序错误的代码，将使您的 CGI 程序对万维网上的浏览者显得更友好和礼貌。

第六章　利用现有 CGI 程序库资源

WWW 上的信息是如此的丰富，基本上任何人都可以在网上找到他所需要的信息资源。对于 CGI 编程来说，网上同样也有很多可以利用的资源，比如具体的 CGI 编程例子、共享的 CGI 程序库等等。利用这些和 CGI 编程有关的共享资源，可以大大降低编程的工作量和时间。即使不准备使用这些资源，阅读它们的代码对您的编程也会有很大的帮助。因此一个 CGI 编程人员尤其是一个初学者，在开始设计 CGI 程序以前应该利用 WWW 上的搜索引擎（例如 yahoo 和 lycos）查找和关键字“CGI”或“CGI library”有关的资源，也许您会有意想不到的收获。

通常，CGI 程序库所提供的函数都是有关 CGI 程序从 Web 页面上获取客户端的输入信息和 URI 解码的，而这也正是所有 CGI 程序所共同的部分。这些程序库的使用将使您可以把主要精力放在 CGI 程序所要解决的具体问题上。

本章将向读者具体介绍几个有关 C 语言和 Visual Basic 的共享程序库。

- CGI-UTIL 库；
- CGIC；
- GD；
- VB-CGI 程序库；
- 其他一些网上资源。

6.1　CGI-UTIL 库

CGI-UTIL 库（cgi-util.h 和 cgi-util.c）是根据 NCSA 提供的“util.c”、“post-query.c”和“query.c”开发而来的。

CGI-UTIL 库的 cgi-util.h 所定义的全局常量“NUM_ENTRIES”用来储存 CGI 程序所获取的客户端用户输入的“NAME=VALUE”对的数目，结构类型的数组 entries[ ]用来存储 CGI 程序获取的这些 NAME 值和 VALUE 值。整型常量“VIA_SHELL”的值为 1 时，CGI 程序将由命令行激活，而“VIA_SHELL”的值为 0 时，CGI 程序则由 Web 服务器激活，这主要是为了程序调试的方便。

cgi-util.c 中的函数 void cgiinit(void)用来完成 CGI 程序的初始化准备工作，正如前一章所介绍的，这部分的工作主要是获取客户端用户的输入并对它门进行 URI 解码。该函数最终将处理后的结果放入结构类型的数组 entries[ ]中以供其他函数的使用。

void getentry(char *buf, char *s) 函数用来根据表单元素的 NAME 属性获取该元素所储存的数据。int getentryasint(char *s) 函数同样是用来根据表单元素的 NAME 属性获取该元素所储存的数据，不过该函数最终将获得的输入数据转换为整型，这主要是用来处理用户的输入为数字时的情况。int getentryyesno(char *s, int def) 函数主要是用来处理 Web 页面上的单选钮和复选框。函数 int dump_no_abort(char * filename)和 void dump(char * filename)可以将某一文件在 Web 页面上显示出来，这主要是用来进行一些出错信息的处理。函数 int goodemailaddress(char * addr)则用来检查客户端用户输入的电子函件地址是否具有正确的格

式。cgi-util.c 的其他函数均来自 NCSA 的“util.c”、“post-query.c”和“query.c”。

以下是一个具体使用 cgi-util 库的 CGI 程序例子，假设该 CGI 程序所要处理的是第三章图 3-4 所示的表单。该 CGI 程序的代码如下所示：

```
/* util-test.c 一个使用 cgi-util 库的 CGI 程序*/
#include <stdio.h>
#include "cgi-util.h"
main( )
{
  char name[50],title[50],phone[20],email[20],unit[50];
  char zhicheng[20],resume[1000];
  cgiinit( );
  getentry(name, "cname");     getentry(title, "ZhiWu");
  getentry(phone, "Phone");     getentry(email, "EMAIL");
  if(!goodemailaddress(email))
    {
      printf("Content-type: text/html\n\n");
      printf("<h2>CGI-UTIL 试验</h2>\n");
      printf("<P>CGI 程序无法理解您输入的电子函件地址: %s\n",email);
      printf("<BR>请输入合法的电子函件地址!\n");
      exit(0);
    }
  getentry(unit, "cunit");
  getentry(zhicheng, "ZhiCheng");
  getentry(resume, "Cresume");
  printf("Content-type: text/html\n\n");
  printf("<h2>CGI-UTIL 试验</h2>\n");
  printf("<P>您的姓名: %s\n", name);
  printf("<BR>您的职务: %s\n", title);
  printf("<BR>您的电话: %s\n", phone);
  printf("<BR>您的电子函件: %s\n", email);
  printf("<BR>您的工作单位: %s\n", unit);
  printf("<BR>您的职称: %s\n", zhicheng);
  printf("<BR>您的简历: <BR>%s\n", resume);
  printf("</bordy></html>");
  exit(0);
}
```

程序编写完毕以后使用如下的编译命令：

```
cc util-test.c cgi-util.c -o program.cgi (UNIX 系统平台)
bcc32 util-test.c cgi-util.c     (Windows 95 平台使用 BC 编译器)
```

就可以得到最终的 CGI 执行程序。在这里我们可以看到利用现成的程序库给 CGI 编程所带来的便利。

cgi-util 是一个非常精炼的库，它的源代码如下所示：

```
/*  cgi-util.h */
typedef struct
{
  char name[128];
  char val[1024];
} entry;
entry entries[1000];
/* entries[ ]用来存储由函数 cgiinit( )找出的 NAME/VALUE 对*/
int NUM_ENTRIES;
int VIA_SHELL;
/* VIA_SHELL 为 1 表示 CGI 程序由命令行激活，为 0 则表示由服务器激活 */
void cgiinit(void); /*该函数用于 CGI 程序的初始化准备工作*/
void getentry(char * buf, char *s);
/*该函数用于确定表单中各个元素存储的输入数据，*s 存储表单元素的 NAME 属性，* buf 则存储相应表单元素的 VALUE 值 */
int getentryasint(char *s);
/*该函数与 getentry 的作用基本相同，只是最后将获取的字符串转换为整型 */
int getentryyesno(char *s, int def);
/*该函数与 getentry 的作用基本相同，只是当表单元素的 VALUE 值是“no”或“off”时，返回 0，当 VALUE 值是“yes”或“on”时返回 1。如果 VALUE 值为空则返回 def */
int dump_no_abort(char * filename); /*将某一文件向标准输出输出*/
void dump(char * filename); /*用于在标准输出上显示错误信息 */
int goodemailaddress(char * addr); /*用于检查 E-mail 地址的合法性*/
/* cgi-util.h 文件结束*/

/* cgi-util.c
  created by Bill Kendrick        kendrick@zippy.sonoma.edu
  http://zippy.sonoma.edu/~kendrick/   based on "post-query.c" and “query.c" by NCSA
  New Breed Software
  April 6, 1996 - January 28, 1997 */

#include <stdio.h>
#ifndef NO_STDLIB_H
#include <stdlib.h>
#else
char *getenv( );
```

```
#endif
#include <string.h>
#include "cgi-util.h"
#define LF 10
#define CR 13
/* 当 CGI 程序被客户端用户用 GET 方法激活时，函数 getword 用于分解字符串*/
void getword(char *word, char *line, char stop)
{
  int x, y;
    for(x = 0; ((line[x]) && (line[x] != stop)); x++)
    word[x] = line[x];
    word[x] = '\0';
  if (line[x])
    ++x;
  y=0;
  while (line[y++] = line[x++]);
}

/* 函数 makeword 用于将 NAME=VALUE 对分解成 NAME 和 VALUE*/
char *makeword(char *line, char stop)
{
  int x,y;
  char *word = (char *) malloc(sizeof(char) * (strlen(line) + 1));
  for(x = 0; ((line[x]) && (line[x] != stop)); x++)
   word[x] = line[x];
  word[x] = '\0';
  if (line[x])
    ++x;
  y=0;
  while (line[y++] = line[x++]);
  return word;
}
/* 函数 fmakeword 可以从标准输入中读出各个 NAME=VALUE 对*/
char *fmakeword(FILE *f, char stop, int *cl)
{
  int wsize;
  char *word;
  int ll;
  wsize = 102400;
```

```
ll=0;
word = (char *) malloc(sizeof(char) * (wsize + 1));
while (1)
  {
    word[ll] = (char)fgetc(f);
    if (ll==wsize)
    {
    word[ll+1] = '\0';
    wsize += 102400;
    word = (char *)realloc(word,sizeof(char)*(wsize+1));
    }
    --(*cl);
    if ((word[ll] == stop) || (feof(f)) || (!(*cl)))
    {
    if (word[ll] != stop)
      ll++;
    word[ll] = '\0';
    return word;
    }
    ++ll;
  }
}
char x2c(char *what)
{
  register char digit;
  digit = (what[0] >= 'A' ? ((what[0] & 0xdf) - 'A')+10 : (what[0] - '0'));
  digit *= 16;
  digit += (what[1] >= 'A' ? ((what[1] & 0xdf) - 'A')+10 : (what[1] - '0'));
  return (digit);
}
/*函数 unescape_url 用于对客户端用户的输入进行 URI 解码*/
void unescape_url(char *url)
{
  register int x,y;
  for (x=0, y=0; url[y]; ++x, ++y)
    {
      if ((url[x] = url[y]) == '%')
      {
      url[x] = x2c(&url[y+1]);
```

```
            y+=2;
            }
        }
    url[x] = '\0';
}
/*函数 plustospace 用于将字符串中的 "+" 转换为空格*/
void plustospace(char *str)
{
    register int x;
    for (x=0; str[x]; x++)
        if (str[x] == '+')
            str[x] = ' ';
}
int rind(char *s, char c)
{
    register int x;
    for (x = strlen(s) - 1; x != -1; x--)
        if(s[x] == c) return x;
    return -1;
}
int getline(char *s, int n, FILE *f)
{
    register int i=0;
    while(1)
        {
            s[i] = (char)fgetc(f);
            if (s[i] == CR)
            s[i] = fgetc(f);
            if ((s[i] == 0x4) || (s[i] == LF) || (i == (n-1))
             {
             s[i] = '\0';
             return (feof(f) ? 1 : 0);
             }
            ++i;
        }
}
void send_fd(FILE *f, FILE *fd)
{
    int num_chars=0;
```

```
    char c;
    while (1)
      {
        c = fgetc(f);
        if(feof(f))
        return;
        fputc(c,fd);
      }
    }
    int ind(char *s, char c)
    {
      register int x;
      for (x=0; s[x]; x++)
        if(s[x] == c)
          return x;
      return -1;
    }
```

/*判断用户输入的命令参数中有没有元字符，函数 escape_shell_cmd 主要是为系统函数 system( )的使用服务的 */

```
    void escape_shell_cmd(char *cmd)
    {
      register int x,y,l;
      l=strlen(cmd);
      for (x=0; cmd[x]; x++)
        {
          if (ind("&;`'\"|*?~<>^()[]{}$\\",cmd[x]) != -1)
          {
           for(y = l + 1; y > x; y--)
             cmd[y] = cmd[y-1];
           l++;
           cmd[x] = '\\';
           x++;
           }
        }
    }
```

/*本程序库以上的函数都是由 NCSA 的“util.c”、“post-query.c”和“query.c”而来的*/

/* cgiinit( )函数用于 CGI 程序的初始化准备工作*/

```
void cgiinit(void)
{
  register int x;
  int cl;
  char * qcl;
  if (getenv("REQUEST_METHOD") == NULL)
    {
      VIA_SHELL = 1;
      NUM_ENTRIES = 0;
    }
  else
    {
      VIA_SHELL = 0;
      if (strcmp(getenv("REQUEST_METHOD"),"POST") == 0)
        {
        if(strcmp(getenv("CONTENT_TYPE"),"application/x-www-form-urlencoded"))
        {
          printf("Content-type: text/html\n\n");
          printf("<h1>Incorrect Type</h1>\n");
          printf("Type must be 'application/x-www-form-urlencoded'.\n");
          exit(0);
        }
      cl = atoi(getenv("CONTENT_LENGTH"));
      for (x = 0; cl && (!feof(stdin)); x++)
        {
          NUM_ENTRIES = x;
          strcpy(entries[x].val, fmakeword(stdin,'&',&cl));
          plustospace(entries[x].val);
          unescape_url(entries[x].val);
          strcpy(entries[x].name, makeword(entries[x].val,'='));
        }
      }
      else if (strcmp(getenv("REQUEST_METHOD"),"GET") == 0)
      {
       qcl = getenv("QUERY_STRING");
       if (qcl == NULL)
         exit(0);
       for (x = 0; qcl[0] != '\0'; x++)
         {
```

```
            NUM_ENTRIES = x;
            getword(entries[x].val, qcl, '&');
            plustospace(entries[x].val);
            unescape_url(entries[x].val);
            getword(entries[x].name, entries[x].val, '=');
        }
    }
    else
    {
     printf("Content-type: text/html\n\n");
     printf("<h1>Unknown method</h1>\n");
     printf("The form was submitted with an unknown method!\n");

     exit(0);
     }
    }
}
/*函数 getentry 用于确定表单中各个元素存储的输入数据，*s 存储表单元素的 NAME 属性，* buf 则
存储相应表单元素的 VALUE 值 */
void getentry(char * buf, char *s)
{
  int x;
  strcpy(buf,"");
  if (VIA_SHELL == 0)
    {
      for (x = 0; x <= NUM_ENTRIES; x++)
      if (strcmp(entries[x].name,s) == 0)
      strcpy(buf,entries[x].val);
    }
  else
    {
      printf("CGI-UTIL: \"%s\" ? ", s);
      fgets(buf, 512, stdin);
      buf[strlen(buf) - 1] = '\0';
    }
}
/*函数 getentryasint 与 getentry 的作用基本相同，只是最后将获取的字符串转换为整型 */
int getentryasint(char *s)
{
```

```
    char buf[512];
    int x, v;
    v = 0;
    if (VIA_SHELL == 0)
      {
        for (x = 0; x <= NUM_ENTRIES; x++)
        if (strcmp(entries[x].name,s) == 0)
        v = atoi(entries[x].val);
      }
    else
      {
        printf("CGI-UTIL: \"%s\" ? ", s);
        fgets(buf, 512, stdin);
        buf[strlen(buf) - 1] = '\0';
        v = atoi(buf);
      }
    return(v);
  }
/*函数 getentryyesno 与 getentry 的作用基本相同，只是当表单元素的 VALUE 值是“no”或“off”时，
返回 0，当 VALUE 值是“yes”或“on”时返回 1。如果 VALUE 值为空则返回 def */
int getentryyesno(char *s, int def)
{
  char temp[512];
  int x, v;
  v = def;
  getentry(temp, s);
  if (strecmp(temp, "yes") == 0 || strecmp(temp, "YES") == 0 ||
      strecmp(temp, "on") == 0 || strcmp(temp, "ON") == 0)
    v = 1;
  else if (strcmp(temp, "no") == 0 || strecmp(temp, "NO") == 0 ||
            strecmp(temp, "off") == 0 || strcmp(temp, "OFF") == 0)
    v = 0;
  return(v);
}
int dump_no_abort(char * filename)
{
  FILE * fi;
  int c;
  fi = fopen(filename, "r");
```

```
if (fi == NULL)
  return(-1);
else
  {
    do
    {
    c = fgetc(fi);
    if (c != EOF)
      fputc(c, stdout);
    }
    while (c != EOF);
    fclose(fi);
    return(0);
  }
}
void dump(char * filename)
{
  if (dump_no_abort(filename) == -1)
    {
      printf("Can't open %s\n", filename);
      exit(0);
    }
}
void error(char * reason)
{
  printf("<h1>Error</h1>\n");
  printf("%s\n", reason);
  exit(0);
}
/*函数 goodemailaddress 用于检查 E-mail 地址的合法性*/
int goodemailaddress(char * addr)
{
  if (strchr(addr, '@') == NULL)
    return 0;
  if (addr[strlen(addr - 1)] == '@' || addr[strlen(addr - 1)] == '.')
    return 0;
  if (strchr(strchr(addr, '@'), '.') == NULL)
    return 0;
    if (strchr(addr, ' ') != NULL)
```

```
    return 0;
  if (addr[0] == '@' || addr[0] == '.')
    return 0;
  return 1;
}
```

6.2 CGIC

CGIC 是网上另一个比较流行的有关 C 语言的 CGI 程序库，读者可以从 WWW 上的站点 http://www.boutell.com/cgic/ 下载它的源程序。该程序库提供的功能比 CGI-UTIL 更多、更丰富，因此在使用它之前还得花一定的时间熟悉它。CGIC 是一个完全符合 ANSI C 标准的程序库，在任何支持 ANSI C 标准的系统平台上都可以使用。用 CGIC 程序库编写的 CGI 程序的编译方法和 CGI-UTIL 类似。

和 CGI-UTIL 不同，CGIC 库本身提供了一个 main( ) 函数，CGIC 的用户编写的程序代码应该放在函数 cgiMain( )中。CGIC 的 main( )函数主要用来完成 CGI 程序的初始化工作、获取客户端用户的输入以及相应的解码工作，最后 main( )函数会调用 cgiMain( )以执行用户编写的程序代码。在初始化准备工作时期，CGIC 将所有已经存在的环境变量值都存入了全局变量，在以后的程序中如要使用环境变量就可以直接使用这些全局变量。

CGIC 还提供了一些调试 CGI 程序的工具，利用 cgiWriteEnvironment( )函数可以将表单中用户输入的数据和相应的环境变量存入一个文件，在 cgiMain( )函数的第一行中调用 cgiReadEnvironment( )函数可以将以前保存的 CGI 程序出错时的运行环境加载到当前的环境中，这就使得 CGI 程序可以在命令行而不是被 Web 服务器激活，这样就可以使用各种各样的调试器对 CGI 程序进行调试。

对于 system( )这个调用系统或外部命令的函数，CGIC 提供了一个更安全的替代函数 int cgiSaferSystem(char *command)。该函数在把客户端用户输入的有关系统命令参数传送给 system( )之前，都要对这些命令参数进行检查，并且认为所有含有元字符的命令参数均为非法。由于在 UNIX 系统平台上含有这些元字符的命令可以激活系统上的其他程序或命令，这一点对 UNIX 系统来说是非常重要的。

通常 Web 网页的作者是通过在网页上加上一组单选钮、复选框或下拉式菜单来让客户端用户从一组可选项中做出选择。如果网页的作者不能确定具体列出多少选择项，可以使用一组具有相同 NAME 属性的表单的 INPUT 元素，CGIC 提供了一个函数 cgiFormStringMultiple( ) 来处理这种情况。CGIC 还可以根据用户的需要自动将表单上的数据转换为整型或者浮点数。

以下是一个具体使用 CGIC 库的 CGI 编程实例，假设该 CGI 程序所要处理的是第三章图 3-4 所示的表单。该 CGI 程序的代码如下所示。

```
/* test.c */
#include <stdio.h>
#include "cgic.h"
int cgiMain( )
```

```
{
char name[81], phone[30], mail[100], unit[240], zhic[100];
char cresume[2000];
    /*获取表单上用户的输入*/
        cgiFormStringNoNewlines("cname", name, 81);
        cgiFormString("Phone", phone, 30);
        cgiFormString("EMAIL", mail, 100);
        cgiFormString("cunit", unit, 240);
        cgiFormString("ZhiCheng", zhic, 100);
        cgiFormString("Cresume", cresume, 100);
        cgiHeaderContentType("text/html");      /*输出 MIME 类型*/
        fprintf(cgiOut, "<HTML><HEAD>\n");
        fprintf(cgiOut, "<TITLE>cgic 测试</TITLE></HEAD>\n");
        fprintf(cgiOut, "<BODY><H1>cgic test</H1>\n");
        fprintf(cgiOut, "您的姓名: %s<BR>\n", name);
        fprintf(cgiOut, "您的电话: %s<BR>\n", phone);
        fprintf(cgiOut, "您的电子函件地址: %s<BR>\n", mail);
        fprintf(cgiOut, "您的工作单位: %s<BR>\n", unit);
        fprintf(cgiOut, "您的职称: %s<BR>\n", zhic);
        fprintf(cgiOut, "您的简历: %s<BR>\n", cresume);
        fprintf(cgiOut, "</BODY></HTML>\n");
        return 0;
}
```

程序编写完毕以后，在 Windows 95 系统上可以采用如下的命令编译：

```
bcc32   test.c cgic.c   (BC 编译器)
```

如果程序复杂，可以建立一个工程项目，充分享受集成编译环境的好处。

6.3　GD 库

和以上介绍的各种 CGI 编程库不同，GD 实际上是一个共享的图形库，在 WWW 上的站点 http://www.boutell.com/gd/可以找到这个图形库的源代码，目前最新的版本是 gd1.2。GD 可以使您的程序以较快的速度用直线、圆弧、文本和各种颜色等生成一个图形，GD 也可以用从其他图像上拷贝的全部或部分图形生成自己的图形，当然也可以用 GD 对现成的图形进行修改。最重要的是，GD 可以将生成的图形保存成 GIF 文件，甚至可以是背景透明和清晰度渐变的 GIF 文件，而 GIF 文件正是 Web 服务器和浏览器之间传送图像的一种标准。这就使得 GD 图形库对于 CGI 编程有很大的实用价值。如果您的 CGI 程序想实时的动态生成图形，而您又对诸如 GIF、JPG 这样的图像文件的结构细节不是很清楚，这时使用 GD 将是最佳的选择。例如一个 CGI 程序每天晚上 12 点对 Web 服务器的访问记录进行统计，并用 HTML 文件将统计结果显示出来。如果在 CGI 程序中使用 GD，就可以用各种

统计图表曲线显示统计结果，此时生成的 Web 页面将比仅仅包含数字的页面要生动活泼的多。使用 GD，您也可以为 Web 页面做一个图形计数器。

和前面介绍的 CGI 程序库一样，GD 也是一个符合 ANSI C 标准的程序库，在各种系统平台上用支持 ANSI C 标准的编译器编译以后都可以使用。以下是一个使用 GD 库的 CGI 程序例子，该程序可以在 Web 页面上生成一幅背景为黑的图像，并从图像的左上角到右下角画一条白线。

```
/* test.c */

#include "gd.h"
#include <stdio.h>
#include <fcntl.h>
int main( )
{
    gdImagePtr im;    /* 声明一幅图像，gdImagePtr 由 “gd.h” 定义*/
    FILE *out;        /*声明一个输出文件句柄 */

    int black;
    int white;
    int result;
    /*将标准输出的打开模式设置成二进制形式*/

    result=setmode(fileno(stdout), O_BINARY);
    if(setmode(fileno(stdout), O_BINARY)==-1)
    {
    printf("Content-type: text/html\n\n");
    printf(“Can not Set mode to binary);
    exit(0);
}
 /* 设置图像的大小*/

    im = gdImageCreate(128, 128);
    /*设置图像的背景颜色为黑色 */

    black = gdImageColorAllocate(im, 0, 0, 0);
    /* 设置画线的颜色为白色*/

    white = gdImageColorAllocate(im, 255, 255, 255);
    /* 从图像的左上角到右下角画一条白线 */
    gdImageLine(im, 0, 0, 127, 127, white);
    /*将 GIF 图像传送给浏览器，声明一个 MIME 类型*/

    printf("Content-type: image/gif\n\n");
    /* 将图像输出到标准输出 STDOUT */

    gdImageGif(im, stdout);
    /*释放图像所用内存空间*/

    gdImageDestroy(im);
```

```
    return 0;
}
```

通常系统是以文本方式从标准输入中读数据或向标准输出写数据。在 DOS、Windows 95 系统下用文本方式无法向标准输出正确传送图像文件，此时应该用 setmode 函数将文本方式改为二进制方式，UNIX 系统下则不存在这个问题。事实上 setmode 函数是 DOS 和 Windows 95 下特有的函数。

在 DOS 或 Windows 95 下利用 GD 库的程序的编译方法和使用 CGIC 库时一样，例如程序简单时可以使用行编译命令：bcc32 test.c gd.c。在 UNIX 系统下则比较复杂，需要使用 makefile 文件。GD 库中包含了一个使用 makefile 文件的例子。对于本节的 test.c 程序，如在 UNIX 系统下编译，makefile 文件的内容如下所示：

```
CC=cc
AR=ar
CFLAGS=-O
LIBS=-L./ -lgd -lm
all: libgd.a test
test: test.o libgd.a gd.h
$(CC) test.o -o test      $(LIBS)
libgd.a: gd.o gd.h
rm -f libgd.a
$(AR) rc libgd.a gd.o
```

图 6-1 显示了 CGI 程序 test.c 在 Web 页面上动态生成的 GIF 图像。

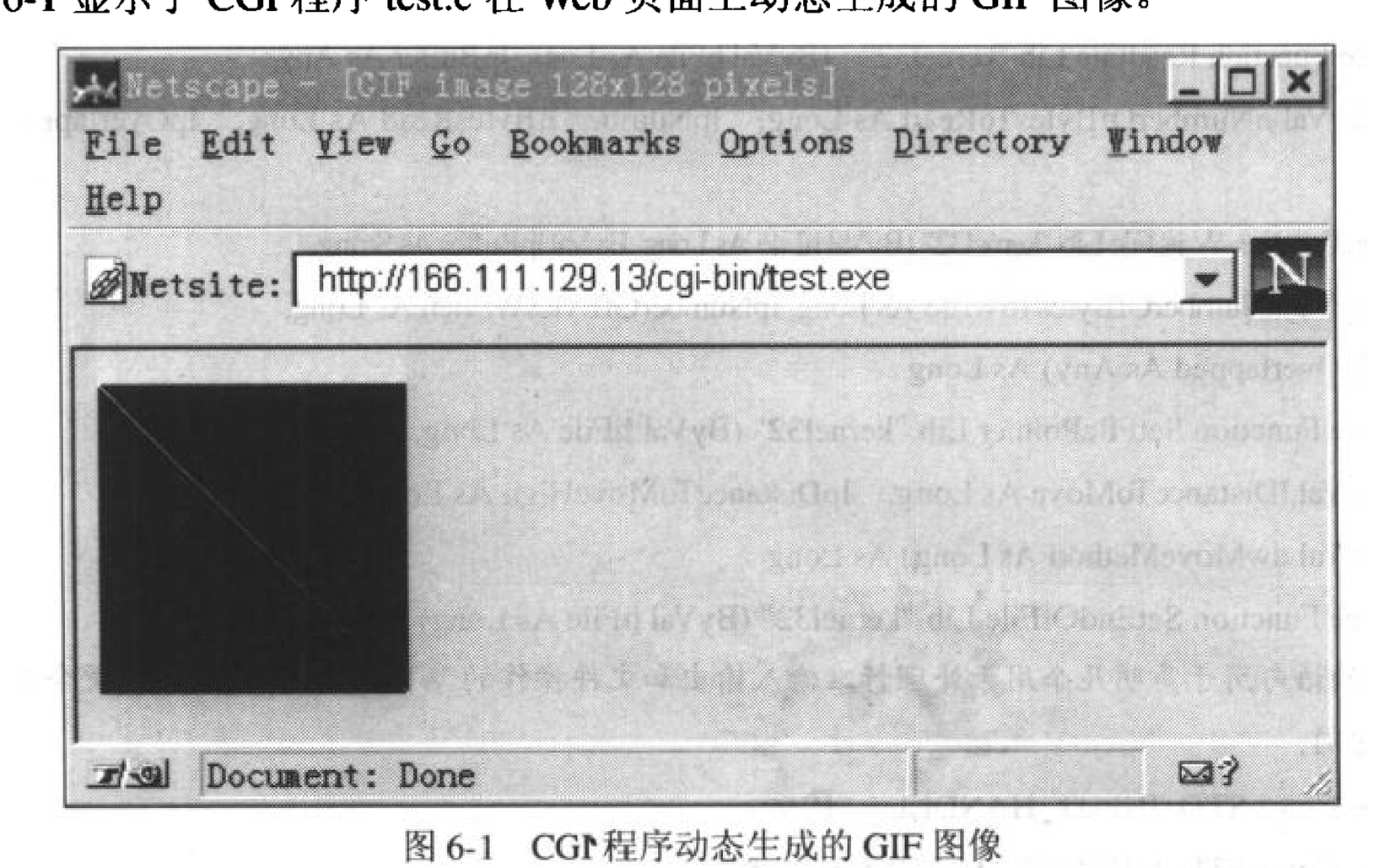

图 6-1　CGI 程序动态生成的 GIF 图像

6.4　CGI4VB

CGI4VB 是一个专门为用 VB 开发 CGI 程序的编程人员服务的一个 CGI 程序库。由于

WIN16 没有为标准输入输出提供文件句柄，CGI4VB 只能在 WIN32 系统下使用。

CGI4VB 实际上提供了一个专门用于 CGI 编程的模块 CGI4VB.BAS，在模块中提供了一些用于 CGI 编程的函数。和 CGIC 类似，CGI4VB.BAS 提供了一些为 CGI 程序进行初始化准备工作的子程序 InitCgi( )、GetFormData( )。这两个函数可以获取所有的环境变量的值和客户端用户的输入并将它们存入全局变量中。CGI4VB.BAS 同时还提供了一个主函数 main( ),CGI 程序运行时主函数在调用一些初始化函数完成准备工作以后将调用 Cgi_Main( ) 函数。函数 Cgi_Main( )是用户自己编写代码的地方。在使用 CGI4VB 时用户可以将自己编写的代码组成一个单独的模块。

虽然 VB 编写的 CGI 程序只能在 Windows 和 NT 平台上使用，速度也没有 C 语言快，但由于 VB 所具有的强大的数据库处理能力和客户服务器模式的编程能力，使用 VB 进行 CGI 编程的人将会越来越多。

用 VB 进行 CGI 编程的关键是使用几个用于标准输入输出处理和读写文件的 WIN32 API 函数。其他的编程原理如环境变量的获取、URI 解码等和 C 语言类似。CGI4VB 的源代码说明了这一点。

```
'-----------------------------------------------------------
' CGI4VB.BAS
'-----------------------------------------------------------
Option Explicit
' Author:   Kevin O'Brien <obrienk@pobox.com>
' <obrienk@ix.netcom.com>
' Declare Function GetStdHandle Lib "kernel32" (ByVal nStdHandle As Long) As Long
Declare Function ReadFile Lib "kernel32"   (ByVal hFile As Long, lpBuffer As Any,
      ByVal nNumberOfBytesToRead As Long,   lpNumberOfBytesRead As Long,   lpOverlapped As Any)
As Long
Declare Function WriteFile Lib "kernel32" (ByVal hFile As Long, ByVal lpBuffer As String,
      ByVal nNumberOfBytesToWrite As Long, lpNumberOfBytesWritten As Long,
      lpOverlapped As Any) As Long
Declare Function SetFilePointer Lib "kernel32" (ByVal hFile As Long,
     ByVal lDistanceToMove As Long,   lpDistanceToMoveHigh As Long,
     ByVal dwMoveMethod As Long) As Long
Declare Function SetEndOfFile Lib "kernel32" (ByVal hFile As Long) As Long
'以上的语句用于声明几个用于处理标准输入输出和文件操作的 WIN32 API 函数，以便今后在 VB 程
序中使用它们。
Public Const STD_INPUT_HANDLE = -10&
Public Const STD_OUTPUT_HANDLE = -11&
Public Const FILE_BEGIN = 0&
' 声明用于存储环境变量值的全局变量
Public CGI_Accept As String
Public CGI_AuthType As String
```

```
Public CGI_ContentLength As String
Public CGI_ContentType As String
Public CGI_GatewayInterface As String
Public CGI_PathInfo As String
Public CGI_PathTranslated As String
Public CGI_QueryString As String
Public CGI_Referer As String
Public CGI_RemoteAddr As String
Public CGI_RemoteHost As String
Public CGI_RemoteIdent As String
Public CGI_RemoteUser As String
Public CGI_RequestMethod As String
Public CGI_ScriptName As String
Public CGI_ServerSoftware As String
Public CGI_ServerName As String
Public CGI_ServerPort As String
Public CGI_ServerProtocol As String
Public CGI_UserAgent As String
Public lContentLength As Long        ' 标准输入中用户输入数据的字节数
Public hStdIn As Long      ' 标准输入句柄
Public hStdOut As Long     ' 标准输出句柄
Public sErrorDesc As String    ' 出错信息
Public sEmail As String    '  Web 服务器管理员的电子函件地址
Public sFormData As String    ' 客户端传送来的经过 URI 译码后的数据
'定义一个结构类型的数组，用于存放表单元素的 NAME 值和 VALUE 值
Type pair
  Name As String
  Value As String
End Type
Public tPair() As pair
'CGI 程序主函数
Sub Main( )
On Error GoTo ErrorRoutine
InitCgi     '完成 CGI 程序的初始化工作，将环境变量值存入全局变量中
GetFormData        ' 获取客户端用户的输入，将结果存入数组 tPair( )
Cgi_Main     ' 用户自己编写代码的子程序，主要用于对客户端传来数据的处理。
EndPgm:
End     ' 程序结束
ErrorRoutine:
```

```
    sErrorDesc = Err.Description & " Error Number = " & Str$(Err.Number)
    ErrorHandler
    Resume EndPgm
End Sub
Sub ErrorHandler( )    ' 本函数用于在 Web 页面上输出出错信息
Dim rc As Long
On Error Resume Next
' 使用 SetFilePointer API 函数将文件指针移到 stdOut 的开始
' 使用 SetEndOfFile API 函数将文件指针移到 stdOut 的最后
rc = SetFilePointer(hStdOut, 0&, 0&, FILE_BEGIN)
SendHeader "Internal Error"
Send "<H1>Error in " & CGI_ScriptName & "</H1>"
Send "The following internal error has occurred:"
Send "<PRE>" & sErrorDesc & "</PRE>"
Send "<I>Please</I> note what you were doing when this problem occurred, "
Send "so we can identify and correct it. Write down the Web page you were "
Send "using, any data you may have entered into a form or search box, "
Send "and anything else that may help us duplicate the problem."
Send "Then contact the administrator of this service: "
Send "<A HREF=""mailto:" & sEmail & """>"
Send "<ADDRESS>&lt;" & sEmail & "&gt;</ADDRESS></A>"
SendFooter
rc = SetEndOfFile(hStdOut)
End Sub
Sub InitCgi( )    '本函数用于完成 CGI 程序的初始化工作，获取标准输入输出文件句柄
hStdIn = GetStdHandle(STD_INPUT_HANDLE)
hStdOut = GetStdHandle(STD_OUTPUT_HANDLE)
sEmail = "YourEmailAddress@Here"
'-----------------------------------------------
' 获取环境变量值并将它们存入全局变量中
'-----------------------------------------------
CGI_Accept = Environ("HTTP_ACCEPT")
CGI_AuthType = Environ("AUTH_TYPE")
CGI_ContentLength = Environ("CONTENT_LENGTH")
CGI_ContentType = Environ("CONTENT_TYPE")
CGI_GatewayInterface = Environ("GATEWAY_INTERFACE")
CGI_PathInfo = Environ("PATH_INFO")
CGI_PathTranslated = Environ("PATH_TRANSLATED")
CGI_QueryString = Environ("QUERY_STRING")
```

```
CGI_Referer = Environ("HTTP_REFERER")
CGI_RemoteAddr = Environ("REMOTE_ADDR")
CGI_RemoteHost = Environ("REMOTE_HOST")
CGI_RemoteIdent = Environ("REMOTE_IDENT")
CGI_RemoteUser = Environ("REMOTE_USER")
CGI_RequestMethod = Environ("REQUEST_METHOD")
CGI_ScriptName = Environ("SCRIPT_NAME")
CGI_ServerSoftware = Environ("SERVER_SOFTWARE")
CGI_ServerName = Environ("SERVER_NAME")
CGI_ServerPort = Environ("SERVER_PORT")
CGI_ServerProtocol = Environ("SERVER_PROTOCOL")
CGI_UserAgent = Environ("HTTP_USER_AGENT")
lContentLength = Val(CGI_ContentLength)
'将环境变量 CGI_ContentLength 转换为整型
ReDim tPair(0)
End Sub

Sub GetFormData(#)
'-----------------------------------------------------------------------------
' 本子程序用于从 STDIN 或环境变量 QueryString 中获取客户端用户的输入
'-----------------------------------------------------------------------------
Dim sBuff   As String      ' 该变量用于存储从 STDIN 中读入的数据
Dim lBytesRead   As Long        ' 存储 ReadFile( ) 函数实际读入的字节数
Dim rc   As Long
' 如果环境变量 RequestMethod 的值是 POST，就从 STDIN 中读数据
如果环境变量 RequestMethod 的值是 GET，就从环境变量 QueryString 中读数据
If CGI_RequestMethod = "POST" Then
    sBuff = String(lContentLength, Chr$(0))
    rc = ReadFile(hStdIn, ByVal sBuff, lContentLength, lBytesRead, ByVal 0&)
    sFormData = Left$(sBuff, lBytesRead)
    If InStr(1, CGI_ContentType, "www-form-urlencoded", 1) Then
        StorePairs sFormData
    End If
End If
StorePairs CGI_QueryString
End Sub

Sub StorePairs(sData As String)
'本子程序将客户端用户的输入分解成 NAME 和 VALUE，并对它们的值进行 URI 解码
```

```
Dim pointer      As Long          ' sData position pointer
Dim n   As Long          ' name/value pair counter
Dim delim1   As Long          ' position of "="
Dim delim2   As Long          ' position of "&"
Dim lastPair     As Long          ' size of tPair( ) array
Dim lPairs   As Long          ' number of name=value pairs in sData
lastPair = UBound(tPair)       ' current size of tPair( )
pointer = 1
Do
   delim1 = InStr(pointer, sData, "=")
   If delim1 = 0 Then Exit Do
   pointer = delim1 + 1
   lPairs = lPairs + 1
Loop
If lPairs = 0 Then Exit Sub   'nothing to add
' redim tPair( ) based on the number of pairs found in sData
ReDim Preserve tPair(lastPair + lPairs) As pair
' assign values to tPair( ).Name and tPair( ).Value
pointer = 1
For n = (lastPair + 1) To UBound(tPair)
    delim1 = InStr(pointer, sData, "=") ' find next equal sign
    If delim1 = 0 Then Exit For   ' parse complete
    tPair(n).Name = UrlDecode(Mid$(sData, pointer, delim1 - pointer))
    delim2 = InStr(delim1, sData, "&")
   ' if no trailing ampersand, we are at the end of data
     If delim2 = 0 Then delim2 = Len(sData) + 1
   ' value is between the "=" and the "&"
    tPair(n).Value = UrlDecode(Mid$(sData, delim1 + 1, delim2 - delim1 - 1))
    pointer = delim2 + 1
Next n
End Sub

Public Function UrlDecode(ByVal sEncoded As String) As String
' 本函数可以对某一字符串进行 URI 解码
Dim pointer      As Long
Dim pos      As Long
If sEncoded = "" Then Exit Function
' 将 “+” 转换为空格
pointer = 1
```

```
Do
    pos = InStr(pointer, sEncoded, "+")
    If pos = 0 Then Exit Do
    Mid$(sEncoded, pos, 1) = " "
    pointer = pos + 1
Loop
' 将十六进制数 "%xx" 转换成相应的字符
pointer = 1
On Error GoTo errorUrlDecode
Do
    pos = InStr(pointer, sEncoded, "%")
    If pos = 0 Then Exit Do
    Mid$(sEncoded, pos, 1) = Chr$("&H" & (Mid$(sEncoded, pos + 1, 2)))
    sEncoded = Left$(sEncoded, pos) & Mid$(sEncoded, pos + 3)
    pointer = pos + 1
Loop
On Error GoTo 0
UrlDecode = sEncoded
Exit Function
errorUrlDecode:
If Err.Number = 13 Then
    Err.Clear
    Err.Raise 65001, "Invalid data passed to UrlDecode( ) function."
Else
    Err.Raise Err.Number
End If
Resume Next
End Function

Function GetCgiValue(cgiName As String) As String
Dim n As Integer
'本函数可以获取表单上某一元素所存储的数据
For n = 1 To UBound(tPair)
    If UCase$(cgiName) = UCase$(tPair(n).Name) Then
        If GetCgiValue = "" Then
            GetCgiValue = tPair(n).Value
        Else
            '如果表单上有很多元素具有相同的 NAME 属性，则将这些元素存储的数据全部取出，不同的值之间用";"分开
```

```
            GetCgiValue = GetCgiValue & ";" & tPair(n).Value
        End If
    End If
Next n
End Function
Sub SendHeader(sTitle As String)
'本子程序用于输出 MIME 信息和 HTML 文件的标题
Send "Status: 200 OK"
Send "Content-type: text/html" & vbCrLf
Send "<HTML><HEAD><TITLE>" & sTitle & "</TITLE></HEAD>"
Send "<BODY>"
End Sub

Sub SendFooter( )
' 本子程序用于输出 HTML 文件的结束部分
Send "</BODY></HTML>"
End Sub

Sub Send(s As String)
' 传送给本子程序的字符串被加上一个换行符后写入标准输出
Dim lBytesWritten As Long
s = s & vbCrLf
WriteFile hStdOut, s, Len(s), lBytesWritten, ByVal 0&
End Sub

Sub SendB(s As String)
'本子程序用于向标准输出中写字符串，但字符串后没有加换行符
'本子程序可以用于向标准输出中输入二进制数据，比如图像文件
Dim lBytesWritten As Long
WriteFile hStdOut, s, Len(s), lBytesWritten, ByVal 0&
End Sub
```

以下是一个具体使用 CGI4VB 库的 CGI 编程实例，假设该 CGI 程序所要处理的是第三章图 3-4 所示的表单。编程时用户只需编写自己的 Cgi_Main( )函数并用它取代 CGI4VB 库中的 Cgi_Main( )函数就行了。该 CGI 程序的代码如下所示。

```
Sub Cgi_Main( )
Dim name As String, phone As String, email As String
Dim unit As String, zhicheng As String, cresume As String
SendHeader "CGIVB4 测试"
name = GetCgiValue("cname")
```

```
phone = GetCgiValue("Phone")
email = GetCgiValue("EMAIL")
unit = GetCgiValue("cunit")
zhicheng = GetCgiValue("ZhiCheng")
cresume = GetCgiValue("Cresume")
Send "<BODY><H1>CGI4VB test</H1>\n"
Send "<BR>您的姓名: " & name
Send "<BR>您的电话: " & phone
Send "<BR>您的电子函件地址: " & email
Send "<BR>您的工作单位: " & unit
Send "<BR>您的职称: " & zhicheng
Send "<BR>您的简历: " & cresume
SendFooter
End Sub
```

注意如果 CGI 程序要处理中文信息，必须对本节所介绍的 CGIVB4 模块的 URI 函数进行一些修改，本书的第七章将讨论这个问题。在用 VB 进行 CGI 编程时，应该仅仅使用 VB 的模块，一定不能让窗体出现在 CGI 程序中。

6.5　其他一些有关 CGI 的网络资源

除了本章提到的一些共享程序库外，网上还有很多有关 CGI 编程的资源，诸如文档资料、编程实例、共享软件等等。学会使用这些资源将会很快使您从一个初学者变成一个 CGI 编程专家。您也可以从这些网络资源中了解到 CGI 编程技术、Web 技术的最新发展动态，当然更不用说这些网络资源在节省您的编程工作量和费用上所起的巨大作用了。本节的以下部分将向读者介绍网上部分有关 CGI 编程的资源。

- http://hoohoo.ncsa.uiuc.edu/cgi/：如果您是一个 CGI 编程的初学者，该站点应该就是您在网上的起点。这个站点的有关 CGI 规范标准、CGI 编程基本概念的资料应该是网上最权威的。

- http://www.w3.org/Protocols/HTTP/HTTP2.html：在进行 CGI 编程前，了解 HTTP 协议是必不可少的，尤其是浏览器和 Web 服务器之间互相交换信息的请求头（Request Header）和响应头（Response header），对于 CGI 编程更是具有十分重要的意义。本站点无论是对初学者还是精通者都会有很大的帮助。

- http://www.iftech.com/oltc/Webdev/Webdev0.stm：该站点对制作动态网页的基本技术（CGI、Java、Javascript、ISAPI、Web 页面和 DataBase 的连接）进行了简单的介绍和总结，最后该站点还罗列了一些有关这方面的网络资源。

- http://language.perl.com/index.html：Perl 可以说是万维网上最通用的 CGI 编程语言，在网上可以找到大量的有关用 Perl 进行 CGI 编程的资源。Perl 语言的一个重要优点就是它是完全免费使用的。在该站点您将可以获得 Perl 的最新版本、有关 Perl 语言的一些文档资料以及基本问题解答。如果您使用 Perl 语言编写 CGI 程序，该站点将是您不得不去的一个

地方。

● http://www.perl.hip.com/：该站点提供的 Perl 语言的 WIN32 版本，这将使您可以在 windows 平台上用 Perl 进行 CGI 程序的开发。

● http://www.bio.cam.ac.uk/cgi-lib/：该站点提供的 CGI 编程库 cgi-lib.pl 可以说是网上用来进行 perl-CGI 编程的最经典的程序库。

● http://www-genome.wi.mit.edu/ftp/pub/software/www/cgi_docs.html：该站点为用 Perl 进行 CGI 编程的人提供了一个程序库，相信这可以节省您的编程工作量。

● http://www.iit.edu/~subrbal/MorphCGI/：该站点为 C++编程人员提供了一个编写 CGI 程序的类库。

● http://www.cgi-resources.com/：正像它的域名一样，该站点提供了各种各样的丰富的 CGI 编程网络资源（包括用各种语言编写的 CGI 程序、相关文档资料等等）。由于该站点的分类非常详细，在其上您将可以很快找到所需要的资源。

● http://www.eekim.com/software/cgihtml/：这同样是一个用于 CGI 编程的 C 语言程序库，曾被 1996 年 8 月的《PC Computing》评为当时两个最好的 CGI 编程工具。

● http://solo.dc3.com/：这是 windows 平台上较著名的 Web 服务器产品 WebSite 的作者的主页，在这里您可以找到很多有价值的东西。

● http://www2.primenet.com/~buyensj/ntWebsrv.html：该站点提供了很多利用 NT 平台构筑一个 Internet 服务器的资源，当然也有很多有关动态网页制作技术例如 CGI 和 ASP 等的资料。

● http://Website.ora.com/wsdocs/32demo/windows-cgi.html：如果在 WIN16 平台上想用 VB 开发 CGI 程序，就必须使用 WIN-CGI 技术，本网页对 Windows CGI 1.3a 接口规范做了详细的说明。

● http://sunsite.edu.cn：这是 sun 公司在国内所设的一个镜像站点，在其上也有很多有关 Internet 和 Web 编程方面的资料。

万维网上的信息资源更新很快，每天都有新的站点出现，也有旧的站点消失，而一些站点的网上地址也会更新。因此本章列举的一些站点在本书出版时可能已经关闭不少。对于一个 CGI 编程人员来说，最重要的是他应该知道使用网上的类似 Yahoo 的搜索引擎寻找自己所需要的资源。

第七章　CGI 编程实例

学习 CGI 编程的最好的方法就是仔细研究其他人编写的 CGI 程序代码，同时试着为自己的主页编写 CGI 程序。本章以 C 语言和 Visual Basic 语言为例列举了一些实用的 CGI 程序以及它们在 Web 页面上的运用情况。其中用 C 语言编写的 CGI 程序完全符合 ANSI C 标准，只需经过少量的修改就可以在各种系统平台上使用。

在 Web 技术发展的初期要实现本章所举例子的功能，CGI 编程将是唯一的选择。但在现在，您却可以拥有多种实现方式，例如可以使用服务器或客户端的用 Java、Javascript 和 VBscript 编写的程序等等。本章在这里用 CGI 程序实现这些功能并不等于说 CGI 程序就是最好的实现方式。不过如果从与系统平台和具体的服务器产品无关性这一点出发，CGI 程序也还是一个不错的选择。本章所举的 CGI 编程实例涉及以下几个方面：

- 下拉式菜单链接选择器；
- 主页意见箱；
- 贵宾留言簿；
- 用户身份认证；
- 主页访问计数器。

7.1　C 语言编程实例

7.1.1　下拉式菜单链接选择器

在 Web 页面上安装一个下拉式菜单链接选择器，客户端的用户可以用下拉式菜单选择他想去的地方，这将比在 Web 页面上罗列一大堆超级链接美观多了，同时也可以为您节省宝贵的 Web 页面空间。这种功能可以用 CGI 程序或 Javascript 程序实现。如果是客户端的 Javascript 程序，可能会遇到浏览器的兼容性问题，但 CGI 程序则没有这个问题。

用 CGI 程序实现下拉式菜单链接选择器的基本原理就是读取下拉式菜单被选择项的 VALUE 值。Web 页面上下拉式菜单的每一选择项的 VALUE 值都存有一个指向万维网上某一个 URL 的超级链接。CGI 程序根据读取的 URL 值，生成一个有关 Location 响应头并传给浏览器，浏览器收到 CGI 程序产生的 Location 响应头后，将立刻向新的 URL 位置发出请求。

一个包含下拉式菜单链接选择器的 Web 页面如图 7-1 所示。生成该 Web 页面的 HTML 代码如下所示：

```
<HTML><HEAD>
<TITLE>今天上哪里去</TITLE> </HEAD>
<BODY><CENTER>
<H3>今天上哪里去</H3>
<PRE><FORM ACTION="/cgi-bin/mlink.exe" METHOD="POST">
<SELECT NAME="Location">
```

```
<OPTION VALUE="今天上哪里去"> 今天上哪里去
<OPTION VALUE="http://www.tsinghua.edu.cn"> 清华大学
<OPTION VALUE="http://www.edu.cn"> 中国教育科研网
<OPTION VALUE="http://www.microsoft.com"> 美国微软公司
<OPTION VALUE="http://www.netscape.com"> 美国网景公司
</SELECT>
<input type="submit" name="SUBMIT" value=" 让我们走！  " align="left">
</FORM></PRE>
</BODY></HTML>
```

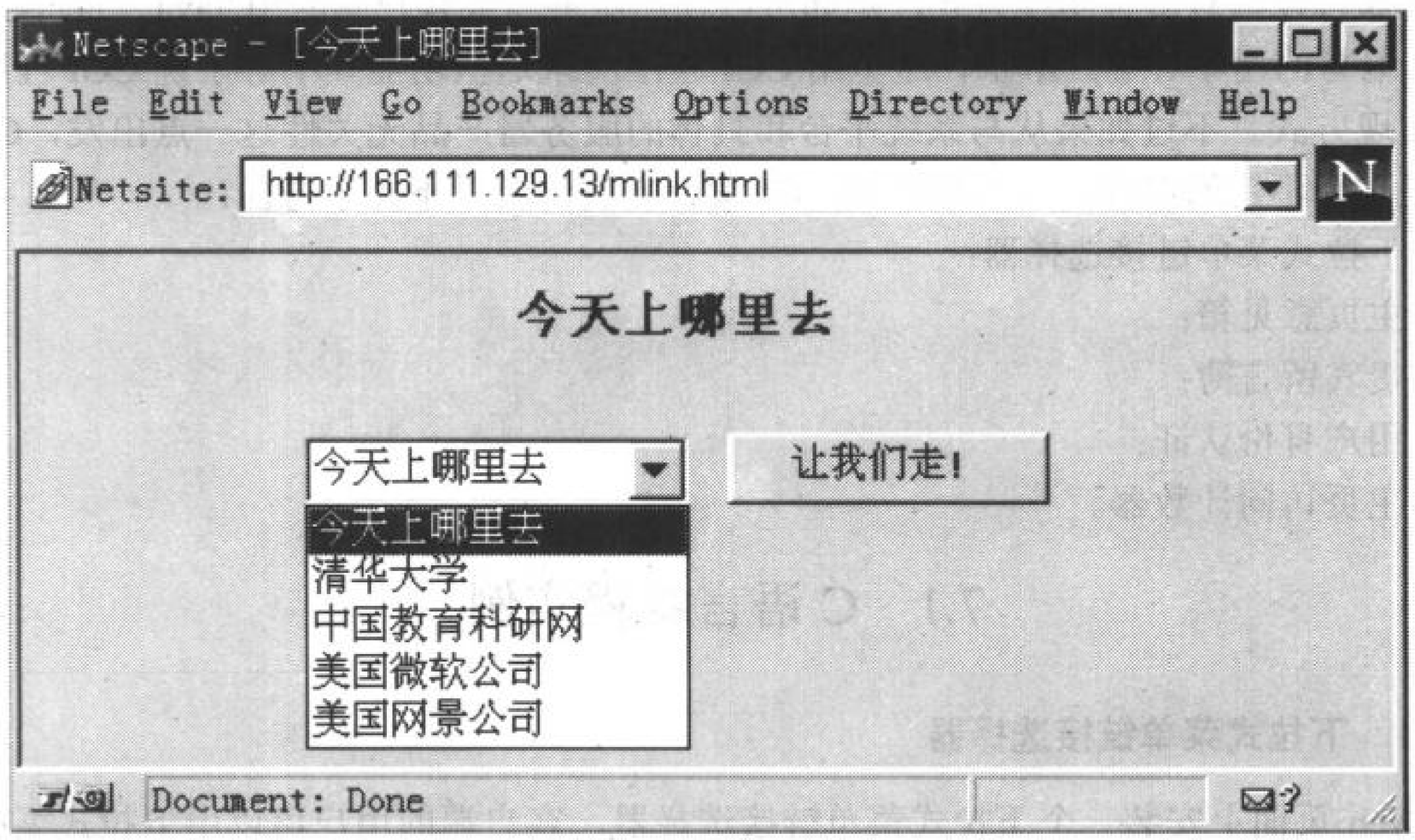

图 7-1　拥有下拉式菜单链接选择器的 Web 页面

处理这个下拉式菜单链接选择器的 CGI 程序代码如下:

```
/* mlink.c */
#include <stdio.h>
#include <stdlib.h>
#define MAX_ENTRIES 10000
typedef struct
 {
     char *name;
     char *val;
 }_entry;
int num_value=0;
entry entries[MAX_ENTRIES];
char *makeword(char *line, char stop);
char *fmakeword(FILE *f, char stop, int *len);
char x2c(char *what);
```

```
void unescape_url(char *url);
void plustospace(char *str);
void getcgivalue(char * buf, char *s);
/*本程序的编写利用了 NCSA 的 "util.c"、"post-query.c" 和 "query.c" 程序*/
main( )
{
    register int x;
    int contentlength;
    char *querystring, *URI;
    /*以下代码可以完成 CGI 程序的初始化准备工作，根据 NCSA 的三个实用程序而来*/
    if(strcmp(getenv("REQUEST_METHOD"),"POST")==0)
    {
      if(strcmp(getenv("CONTENT_TYPE"),"application/x-www-form-urlencoded"))
      {
          printf("Content-type: text/html%c%c",10,10);
          printf("This script can only be used to decode form results. \n");
          exit(1);
       }
       contentlength = atoi(getenv("CONTENT_LENGTH"));
       for(x=0;contentlength && (!feof(stdin));x++)
       {
          num_value=x;
          entries[x].val = fmakeword(stdin,'&',&contentlength);
          plustospace(entries[x].val);
          unescape_url(entries[x].val);
          entries[x].name = makeword(entries[x].val,'=');
       }
      }
    else{
          printf("Content-type: text/html%c%c",10,10);
         printf("Please use Post method.\n");
         exit(1);
       }
    /*以下代码可以获取用户选择的 URL*/
      getcgivalue(URI,"Location");
      if(strcmp(URI,"今天上哪里去")==0)
      {
        printf("Content-type: text/html%c%c",10,10);
        printf("<P>请用网页上的下拉式菜单选择一个您想去的地方!\n");
```

```
        exit(1);
      }
    printf("Location: %s\n\n",URI);  /*向浏览器输出 Location 响应头*/
}
/*以下为本程序使用的 NCSA 的 util.c 所提供的部分函数*/
char *makeword(char *line, char stop)  /*本函数用于分解字符串*/
  {
    int x = 0,y;
    char *word = (char *) malloc(sizeof(char) * (strlen(line) + 1));
    for(x=0;((line[x]) && (line[x] != stop));x++)
        word[x] = line[x];
    word[x] = '\0';
    if(line[x]) ++x;
    y=0;
    while(line[y++] = line[x++]);
    return word;
}
char *fmakeword(FILE *f, char stop, int *cl)  /*本函数用于从标准输入中读取 NAME=VALUE 对*/
{
    int wsize;
    char *word;
    int ll;
    wsize = 102400;
    ll=0;
    word = (char *) malloc(sizeof(char) * (wsize + 1));
    while(1)
    {
        word[ll] = (char)fgetc(f);
        if(ll==wsize)
         {
            word[ll+1] = '\0';
            wsize+=102400;
            word = (char *)realloc(word,sizeof(char)*(wsize+1));
         }
        --(*cl);
        if((word[ll] == stop) || (feof(f)) || (!(*cl)))
        {
            if(word[ll] != stop) ll++;
            word[ll] = '\0';
```

```
                return word;
            }
            ++ll;
        }
    }
char x2c(char *what)   /*本函数用于 URI 译码*/
{
    register char digit;
    digit = (what[0] >= 'A' ? ((what[0] & 0xdf) - 'A')+10 : (what[0] - '0'));
    digit *= 16;
    digit += (what[1] >= 'A' ? ((what[1] & 0xdf) - 'A')+10 : (what[1] - '0'));
    return(digit);
}
void unescape_url(char *url)   /*本函数用于 URI 译码*/
{
    register int x,y;
    for(x=0,y=0;url[y];++x,++y)
    {
        if((url[x] = url[y]) == '%')
        {
            url[x] = x2c(&url[y+1]);
            y+=2;
        }
    }
    url[x] = '\0';
}
void plustospace(char *str)   /*本函数可以将 '+' 转换为空格*/
{
    register int x;
    for(x=0;str[x];x++) if(str[x] == '+') str[x] = ' ';
}
void getcgivalue(char * buf, char *s)   /*本函数用于获取表单元素的输入*/
{
  int x;
  strcpy(buf," ");
  for (x = 0; x <= num_value; x++)
    if (strcmp(entries[x].name,s) == 0)
    strcpy(buf,entries[x].val);
}
```

7.1.2 主页意见箱

完成了 Web 网页的制作以后，通常都希望再给主页增加一个意见箱，这样可以使那些发现了网页缺陷的热心读者能够很方便的转达他们的意见。实现这种功能的途经有两种，一种是借助于客户端浏览器本身所具有的电子函件功能，用 HTML 标识符 mailto 或 Javascript 小程序将用户的意见转发给 Web 服务器管理员。这种方式的使用有一个前提，就是客户端用户的浏览器中有关电子函件部分的设置必须正确，并且用户必须知道一个能使用的 SMTP 电子函件服务器。在主页上实现意见箱的另一种方式就是 CGI 编程，无论客户端的浏览器如何设置，用户是否知道某一个 SMPT 电子函件服务器，CGI 程序都能保证用户的建议被正确传送到服务器管理员那里。当然，此时 Web 服务器发送邮件的服务（对于 UNIX 系统来说就是 sendmail）必须正常运行。

设计一个外观漂亮、整齐的表单是制作意见箱工作的一部分，通常这并不是一件容易的事。就像本书在第三章所说的，此时必须充分利用 HTML 的表格来对 Web 网页上的表单进行排版。一个表格不够就使用多个表格，一个表格的单元格中也可以拥有另一个表格。目前的一些所见即所得的 Web 页面编辑工具将使您很方便的设计出各种美观的表单。使用表格还有另一个好处，它可以使 Web 页面上的表单在各种浏览器的窗体上将有一致的外观。图 7-2 就是一个用表格进行了排版的表单。

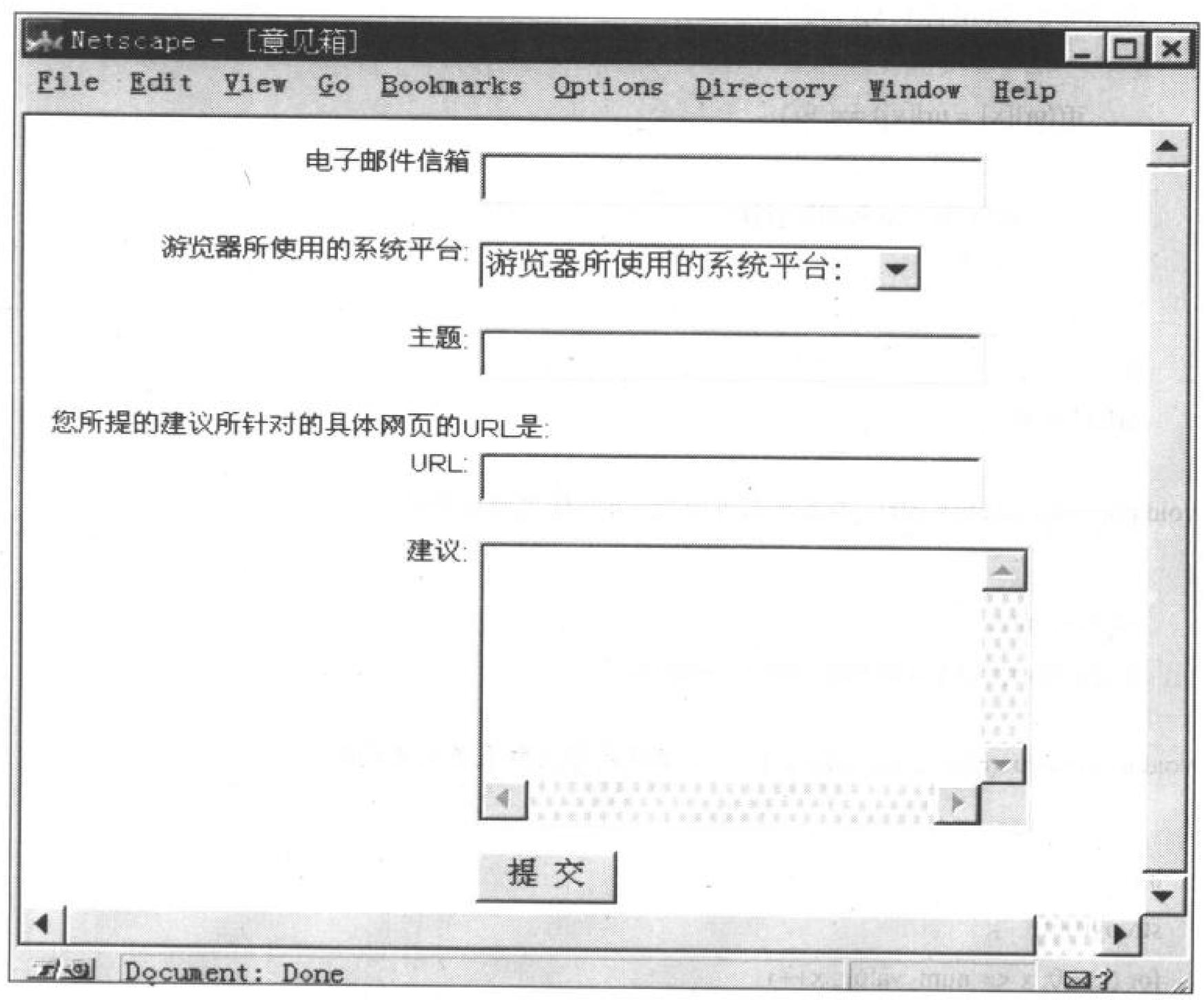

图 7-2　用表格进行了排版的表单

生成图 7-2 所示的 Web 页面的 HTML 语句如下所示：

```
<HTML><HEAD><TITLE>意见箱</TITLE></HEAD>
<BODY BGCOLOR="#FFFFFF">
<CENTER>
<TABLE WIDTH="612" BORDER="0" CELLSPACING="0" CELLPADDING="0">
<TR><TD></TD>
<TD ALIGN="right" VALIGN="top">
<FORM ACTION="/cgi-bin/comment.exe" METHOD=POST>
<TABLE BORDER="0" WIDTH="100%">
    <TR><TD VALIGN="top" ALIGN="right"> <FONT FACE="arial, helvetica" SIZE="-1">电子函件信箱
</FONT></TD>
    <TD><INPUT TYPE="TEXT" SIZE=30 NAME=email MAXLENGTH=50><BR>
    <SPACER TYPE="vertical" SIZE="15"></TD></TR>
    <TR><TD VALIGN="top" ALIGN="right">
    <FONT FACE="arial, helvetica" SIZE="-1">浏览器所使用的系统平台: </FONT>
    </TD>
    <TD><SELECT NAME="SYSTEM">
    <OPTION>浏览器所使用的系统平台:
      <OPTION>MS Windows 3.1
      <OPTION>MS Windows 95
      <OPTION>UNIX
      <OPTION>其他
    </SELECT><BR>
    <SPACER TYPE="vertical" SIZE="15">
    </TD></TR>
    <TR><TD VALIGN="top" ALIGN="right">
    <FONT FACE="arial, helvetica" SIZE="-1"> 主题:</FONT></TD>
    <TD><INPUT TYPE="TEXT" SIZE=30 NAME=URL_Subject><BR>
    <SPACER TYPE="vertical" SIZE="15"></TD>     </TR>
    <TR><TD VALIGN="top" ALIGN="left" COLSPAN="2">
    <FONT FACE="arial, helvetica" SIZE="-1">您所提的建议所针对的具体网页的 URL 是:
    </FONT>
    </TD></TR>
    <TR><TD VALIGN="top" ALIGN="right">
    <FONT FACE="arial, helvetica" SIZE="-1">URL:</FONT></TD>
    <TD><INPUT TYPE="TEXT" SIZE=30 NAME=URL MAXLENGTH=50><BR>
    <SPACER TYPE="vertical" SIZE="15"></TD></TR>
    <TR><TD VALIGN="top" ALIGN="right">
    <FONT FACE="arial, helvetica" SIZE="-1"> 建议:</FONT></TD>
    <TD><TEXTAREA ROWS=6 COLS=25 NAME=URL_Comment>
```

```
</TEXTAREA><BR>
<SPACER TYPE="vertical" SIZE="9"></TD></TR>
<TR><TD></TD>
<TD><INPUT TYPE="SUBMIT" NAME="Submit Comments" VALUE=" 提交 ">
<SPACER TYPE="vertical" SIZE="12"></TD></TR></TABLE>
</TD> </TR>
</TABLE></CENTER>
</BODY></HTML>
```

处理图 7-2 所示表单的 CGI 程序首先将读出客户端用户在表单上的输入，并对用户的输入做一些基本的检查。当用户的输入通过检查以后，CGI 程序就把用户的建议和客户端用户的有关状态信息存入一个 TEMP 文件中，并调用系统（这里是 UNIX 系统）的电子函件服务功能将该文件的内容用电子函件的方式传送给服务器管理员。最后，在成功发送了用户的建议以后 CGI 程序将删除 TEMP 文件。该 CGI 程序的源代码如下所示：

```
/*本程序的编写利用了 NCSA 的 "util.c"、"post-query.c" 和 "query.c" 程序*/
/* comment.c */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX_ENTRIES 10000
typedef struct
{
    char *name;
    char *val;
} entry;
int   num_value=0;
entry entries[MAX_ENTRIES];
char *makeword(char *line, char stop);
char *fmakeword(FILE *f, char stop, int *len);
char x2c(char *what);
void getword(char *word, char *line, char stop);
void unescape_url(char *url);
void plustospace(char *str);
char *getcgivalue(char *s);
main( )
{
    FILE *suggest;   /*临时文件指针 */
    int x,contentlength;
    struct tmtoday;
    time_tnow;
```

```
char *querystring ,*host,*submittime,*navigator,*systems;
char *subject,*email,*comment,*URL， *tn1;
char command[200];  /*所要执行的发送邮件的 UNIX 系统命令*/
/*以下代码用于获取用户发送建议的时间*/
time(&now);
today = *localtime(&now);
mktime(&today);
/*以下代码可以完成 CGI 程序的初始化工作，根据 NCSA 的三个实用程序而来*/
if(strcmp(getenv("REQUEST_METHOD"),"POST")==0)
{
   if(strcmp(getenv("CONTENT_TYPE"),"application/x-www-form-urlencoded"))
     {
     printf("Content-type: text/html%c%c",10,10);
     printf("This script can only be used to decode form results. \n");
     exit(1);
     }
     contentlength = atoi(getenv("CONTENT_LENGTH"));
     for(x=0;contentlength && (!feof(stdin));x++)
     {
       num_value=x;
       entries[x].val = fmakeword(stdin,'&',&contentlength);
       plustospace(entries[x].val);
       unescape_url(entries[x].val);
       entries[x].name = makeword(entries[x].val,'=');
     }
   }
else if(strcmp(getenv("REQUEST_METHOD"),"GET")==0)
{
    querystring = getenv("QUERY_STRING");
    if(querystring == NULL)
      {
       printf("Content-type: text/html%c%c",10,10);
       printf("No query information to decode.\n");
       exit(1);
      }
   for(x=0;querystring[0] != '\0';x++)
      {
       num_value=x;
       getword(entries[x].val,querystring,'&');
```

```
            plustospace(entries[x].val);
            unescape_url(entries[x].val);
            getword(entries[x].name,entries[x].val,'=');
        }
    }
    else{
            printf("Content-type: text/html%c%c",10,10);
           printf("The form was submitted with an unknown method!\n");
        }
    /*以下代码用于获取用户所提的建议和状态信息*/
    navigator=getenv("HTTP_USER_AGENT"); /*获取用户所用浏览器的类型*/
    if((host=getenv("REMOTE_HOST"))==NULL)
    host=getenv("REMOTE_ADDR");
    systems=getcgivalue("SYSTEM");
    email=getcgivalue("email");
    subject=getcgivalue("URL_Subject");
    URL=getcgivalue("URL");
    comment=getcgivalue("URL_Comment");
    if(comment==NULL)  /*检查用户在传送建议以前是否填写了建议*/
    {
        printf("Content-type: text/html%c%c",10,10);
        printf("<P>非常抱歉，您还没有提出宝贵的意见!\n");
        exit(0);
    }
      /*打开一个 TEMP 文件用来保存用户的建议*/
    tn1 = tmpnam(NULL);
    if((suggest=fopen(tn1,"w+t"))==NULL)
    {
      printf("Content-type: text/html%c%c",10,10);
      printf("<P>非常抱歉，系统无法打开暂时文件!\n");
      printf("<P>请您等候片刻以后再试或用别的方式将意见转给我们!\n");
      exit(1);
      }
        fprintf(suggest,"客户端主机为：%s\n",host);
        fprintf(suggest,"发送时间为：%d 年%d 月%d 日%d 时%d 分\n",today.tm_year+1900, today.tm_mon+1,
today.tm_mday,today.tm_hour,today.tm_min);
        fprintf(suggest,"客户端所用浏览器为：%s\n",navigator);
        if((strcmp(systems,"浏览器所使用的系统平台:"))!=0)
        fprintf(suggest,"浏览器所使用的系统平台为：%s\n",systems);
```

```
    if(email!=NULL) fprintf(suggest,"客户端用户电子函件信箱为: %s\n",email);
    if(subject!=NULL) fprintf(suggest,"建议的主题是: %s\n",  subject);
    if(URL!=NULL) fprintf(suggest,"建议所指的主页是: %s\n", URL);
    fprintf(suggest,"客户端所提的建议为: \n\n");
    fprintf(suggest," %s\n\n",  comment);
    fclose(suggest);
/*向客户端用户返回一些状态信息*/
    printf("Content-type: text/html%c%c",10,10);
   printf("<FONT SIZE=\"+2\">");
   printf("<P>请稍候，系统正在传递您的信息.....");
   /*利用系统的 Mail 命令向管理员传送用户的建议*/
  sprintf(command, "\usr\sbin\Mail -s \"关于主页的建议\" zhangys@hotmail.com < %s\n", tn1);
  if(system(command)==-1)
  {
      printf("<P>非常抱歉，系统无法无法传递您的意见!\n");
      printf("<P>请您等候片刻以后再试或用别的方式将意见转给我们!\n");
   }
  unlink(tn1);              /*删除临时文件*/
  printf("<P>非常感谢您对本主页的宝贵意见");
  printf("<P>欢迎您经常光顾本主页！ ");
    printf("<\FONT>");
    return 0;
    }
/*以下为本程序所使用的 NCSA 的 util.c 所提供的部分函数和本书所补充的函数*/
void getword(char *word, char *line, char stop) /*本函数用于分解字符串*/
{
      int x = 0,y;
      for(x=0;((line[x]) && (line[x] != stop));x++)
            word[x] = line[x];
      word[x] = '\0';
      if(line[x]) ++x;
      y=0;
      while(line[y++] = line[x++]);
}
char *makeword(char *line, char stop) /*本函数用于分解字符串*/
{
      int x = 0,y;
      char *word = (char *) malloc(sizeof(char) * (strlen(line) + 1));
      for(x=0;((line[x]) && (line[x] != stop));x++)
```

```
        word[x] = line[x];
    word[x] = '\0';
    if(line[x]) ++x;
    y=0;
    while(line[y++] = line[x++]);
    return word;
}
char *fmakeword(FILE *f, char stop, int *cl) /*本函数用于从标准输入中获取 NAME=VALUE 对*/
{
    int wsize;
    char *word;
    int ll;
    wsize = 102400;
    ll=0;
    word = (char *) malloc(sizeof(char) * (wsize + 1));
    while(1)
        {
        word[ll] = (char)fgetc(f);
        if(ll==wsize)
        {
            word[ll+1] = '\0';
            wsize+=102400;
            word = (char *)realloc(word,sizeof(char)*(wsize+1));
        }
        – (*cl);
        if((word[ll] == stop) || (feof(f)) || (!(*cl)))
        {
            if(word[ll] != stop) ll++;
            word[ll] = '\0';
            return word;
        }
        ++ll;
    }
}
char x2c(char *what) /*本函数用于 URI 译码*/
{
    register char digit;
    digit = (what[0] >= 'A' ? ((what[0] & 0xdf) - 'A')+10 : (what[0] - '0'));
    digit *= 16;
```

```
    digit += (what[1] >= 'A' ? ((what[1] & 0xdf) - 'A')+10 : (what[1] - '0'));
    return(digit);
}
void unescape_url(char *url) /*本函数用于 URI 译码*/
{
    register int x,y;
    for(x=0,y=0;url[y];++x,++y)
    {
        if((url[x] = url[y]) == '%')
        {
            url[x] = x2c(&url[y+1]);
            y+=2;
        }
    }
    url[x] = '\0';
}
void plustospace(char *str) /*本函数用于将 '+' 转换为空格*/
{
    register int x;
    for(x=0;str[x];x++)
        if(str[x] == '+')
          str[x] = ' ';
}
/*以下的函数用于获取表单上某一元素所储存的数据*/
/*当表单元素不含数据时返回 NULL 指针*/
char* getcgivalue(char *s)
{
  int x,length, i=0;
  char *buf;
  for (x = 0; x <= num_value; x++)
  {
    if (strcmp(entries[x].name,s) == 0)
    {
      if((length=strlen(entries[x].val))==0)     return NULL;
      if(( buf=(char *)malloc(length))==NULL)
       {
          printf("Not enough memory to allocate buffer \n");
          exit(0);
       }
```

```
            while((entries[x].val[i])!='\0')
            {
             buf[i]=entries[x].val[i];
             i++ ;
             }
             buf[i]='\0';
        }
    }
    return buf;
    }
```

如果要想在 Windows 95 和 NT 下运行本 CGI 程序，必须对程序中发送邮件的代码按照 Windows NT 或 Windows 95 的方式进行改动。

在本 CGI 程序中对用户的输入进行了检查，如果用户没有输入任何建议，将禁止用户发送任何信息。事实上这一部分工作可以在浏览器上就地完成，没有必要在数据已经传送到服务器上以后再让 CGI 程序去检查，这可以用客户端的 Javascript 实现。以下 Web 页面上所嵌入的 Javascript 小程序将对图 7-2 所示 Web 页面上的用于填写用户建议的文本框进行检查，如果没有填写建议，用户将不能调用此 CGI 程序。

```
……HTML 语句……
<body>
<P><script><!-- hide
function checkForm( Form )
  {
    if ( !checkField( Form.URL_Comment,  "建议" ) )
        return false
    return true
  }
function checkField( Field, Name )
  {
    if ( Field.value == "" )
      {
        errMsg( Field, "\n 您还没有输入您的"+Name )
        return false
      }
    else
      {
        return true
      }
  }
function errMsg( Field, Msg )
```

```
    {
      alert( Msg )
      Field.focus()
      return
    }
// --></script>
……HTML 语句……
<form action="/cgi-bin/comment.exe" method="POST"
        onsubmit="return checkForm(this)">
……HTML 语句……
```

当用户没有填写建议而准备调用 CGI 程序时，在 Web 页面上将会弹出如图 7-3 所示的警告对话框，要求用户填写数据。

图 7-3 Javascript 的警告信息

用 Javascript 编写的小程序可以在客户端对用户的数据进行各种形式的检查，这将部分减轻 Web 服务器的负担。在这里我们可以看到客户端的 Javascript 和服务器方的 CGI 程序互相合作。对于一个 CGI 程序来说，如果它所处理的工作在客户端上可以完成，它就应该尽量将这些工作交给客户端的诸如用 Javascript 和 VBscript 编写的小程序完成。

7.1.3 贵宾留言簿

收集访问者意见的另一种方式就是给主页加一个贵宾留言簿（guests.html）。访问者可以留下自己的建议，也可以看到其他人留下的意见。访问者的建议在加入贵宾留言簿的同时也被发送给服务器管理员。CGI 程序应该对访问者的留言进行最基本的合法性检查。

为了在贵宾留言簿中正确插入访问者的留言，在文件 guests.html 中使用了 HTML 语

句“<! ENDHEAD >”。通常 HTML 标识符“<!”和“>”之间的代码表示一个可以在浏览器上运行的客户端小程序（就像上一小节的 Javascript 小程序），如果浏览器不认识“<!”和“>”之间的代码，浏览器将既不执行也不在 Web 页面上显示这些字符。CGI 程序利用了 HTML 语言的这个特点，用字符串“<! ENDHEAD >”对客户端用户在留言簿中的插入位置进行定位。这种插入方式将保证最后留言的客户端用户的留言出现在留言簿中所需要的位置。

假设客户端用户在贵宾留言簿上的留言由以下的 HTML 语句所产生的表单收集：

```
<html><head><title>贵宾留言簿</title></head>
<body>
<h3>贵宾留言簿测试</h3>
<form action="/cgi-bin/guest.exe" method="post">
您的姓名: <input type="text" name="name"><br>
您的 E-mail 信箱: <input type="text" name="email"><br>
您的主页的 URL: <input type="text" name="URL"><br>
您的建议: <br> <textarea name="URL_Comment" rows=4 cols=30></textarea><br>
您来自: <input type="text" name="from"><br>
<input type="submit" value=" 留言 ">
</form>
</body></html>
```

处理该表单的 CGI 程序的代码则如下所示：

```
/*guestbook.c*/
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX_ENTRIES 10000
typedef struct
{
    char *name;
    char *val;
} entry;
int   num_value=0;
entry entries[MAX_ENTRIES];
char *makeword(char *line, char stop);
char *fmakeword(FILE *f, char stop, int *len);
char x2c(char *what);
char *getcgivalue(char *s);
void getword(char *word, char *line, char stop);
void unescape_url(char *url);
```

```
void plustospace(char *str);
void filecopy(FILE * from, FILE * to);
FILE * cgifopen(char * filename, char   * mode);
main( )
{
    FILE *suggest,*tempto,*tempfrom;
    int x, contentlength;
    char *querystring, *host, *navigator;
    char *name, *email, *comment, *URL, *from, tempstring[10240], *tn1, *tn2;
    char command[200];
/*以下代码用于获取用户发送建议的时间*/
    struct tm   today;
    time_t   now;
    time(&now);
    today = *localtime(&now);   /*获取客户端用户留言的时间*/
    mktime(&today);
    /*以下代码用于完成 CGI 程序的初始化准备工作*/
    if(strcmp(getenv("REQUEST_METHOD"),"POST")==0)
    {
      if(strcmp(getenv("CONTENT_TYPE"),"application/x-www-form-urlencoded"))
      {
      printf("Content-type: text/html%c%c",10,10);
      printf("This script can only be used to decode form results. \n");
      exit(1);
      }
      contentlength = atoi(getenv("CONTENT_LENGTH"));
      for(x=0;contentlength && (!feof(stdin));x++)
      {
        num_value=x;
        entries[x].val = fmakeword(stdin,'&',&contentlength);
        plustospace(entries[x].val);
        unescape_url(entries[x].val);
        entries[x].name = makeword(entries[x].val,'=');
      }
    }
  else if(strcmp(getenv("REQUEST_METHOD"),"GET")==0)
   {
     querystring = getenv("QUERY_STRING");
     if(querystring == NULL)
```

```
{
      printf("Content-type: text/html%c%c",10,10);
    printf("No query information to decode.\n");
    exit(1);
  }
for(x=0;querystring[0] != '\0';x++)
  {
    num_value=x;
        getword(entries[x].val,querystring,'&');
        plustospace(entries[x].val);
        unescape_url(entries[x].val);
        getword(entries[x].name,entries[x].val,'=');
    }
}
else
{
        printf("Content-type: text/html%c%c",10,10);
        printf("The form was submitted with an unknown method!\n");
}
/*获取客户端用户的一些状态信息和客户的留言*/
 navigator=getenv("HTTP_USER_AGENT");
 if((host=getenv("REMOTE_HOST"))==NULL) host=getenv("REMOTE_ADDR");
 name=getcgivalue("name");
 email=getcgivalue("email");
 URL=getcgivalue("URL");
 from=getcgivalue("from");
 comment=getcgivalue("URL_Comment");
/*对客户端的用户做一些基本的合法性检查*/
 if(name==NULL)
 {
      printf("Content-type: text/html%c%c",10,10);
      printf("<P>非常抱歉，您必须填写您的姓名!\n");
      exit(0);
 }
if(comment==NULL)
 {
      printf("Content-type: text/html%c%c",10,10);
      printf("<P>非常抱歉，您还没有提出宝贵的意见!\n");
      exit(0);
```

```
  }
if (URL!=NULL)
  {
    if (strstr(URL, ".") == NULL)
      {
        printf("Content-type: text/html%c%c",10,10);
        printf("<P>非常抱歉，您输入的 URL 不正确!\n");
        exit(0);
      }
if (strstr(URL, "http://") == NULL)
    {
     sprintf(tempstring, "http://%s", URL);
     strcpy(URL, tempstring);
    }
}
if(email!=NULL)
  {
     if((goodemailaddress(email))==0)
        {
         printf("Content-type: text/html%c%c",10,10);
         printf("<P>非常抱歉，您的 E-mail 地址不合法!\n");
         exit(0);
        }
    }
tn1 = tmpnam(NULL);  /*创建一个唯一的临时文件名*/
suggest=cgifopen(tn1,"w+t");
fprintf(suggest,"客户端主机为：%s\n",host);
fprintf(suggest,"发送时间为：%d 年%d 月%d 日%d 时%d 分\n",today.tm_year+1900,today.tm_mon+1,
today.tm_mday,today.tm_hour,today.tm_min);
fprintf(suggest,"客户端所用浏览器为：%s\n",navigator);
if(email!=NULL) fprintf(suggest,"客户端用户电子函件信箱为：%s\n",email);
fprintf(suggest,"客户端用户的姓名为: %s\n",  name);
if(URL!=NULL) fprintf(suggest,"客户端用户留下的的 URI 为: %s\n", URL);
if(from!=NULL) fprintf(suggest,"客户端用户来自: %s\n", from);
fprintf(suggest,"客户端用户的留言为：\n\n");
fprintf(suggest," %s\n\n",  comment);
fclose(suggest);
sprintf(command, "\usr\sbin\Mail -s \"关于主页的建议\"zhangys@hotmail.com < tn1" , %s);
system(command);   /*将客户端用户的留言发送给服务器管理员*/
```

```
  unlink(tn1);    /*删除临时文件*/

  tn2 = tmpnam(NULL);
  tempto=cgifopen(tn2,"w");/*打开一个临时文件*/

  tempfrom = cgifopen("\usr\people\tribo\public_html\guests.html","r");
/*打开留言簿*/
/*将 guests.html 中"<! ENDHEAD >"以前的字符串复制到 TEMP 文件中去 */
/*复制的字符串必须包含"<! ENDHEAD >"*/

do
    {
        fgets(tempstring, 10240, tempfrom);
        tempstring[strlen(tempstring)-1] = '\0';
        fputs(tempstring,tempto);
        fputc('\n',tempto);
    }while (!feof(tempfrom) && strcmp(tempstring, "<! ENDHEAD >") != 0);
/*将客户端用户在留言簿上的留言写入临时文件*/

    fprintf(tempto, "\n<hr>\n");
    fprintf(tempto, "\n<ul>\n");
    fprintf(tempto,"<li><b>留言时间</b>:  %d 年%d 月%d 日%d 时%d 分\n",today.tm_year+1900,
today.tm_mon+1,today.tm_mday,today.tm_hour,today.tm_min);
    fprintf(tempto, "<li><b>姓名:</b> %s\n", name);
    if (email!=NULL)
      fprintf(tempto, "<li><b>E-mail:</b> <a href=\"mailto:%s\">%s</a>\n", email,email);
    if (URL!=NULL)
      fprintf(tempto, "<li><b>我的主页:</b> <a href=\"%s\">%s</a>\n", URL, URL);
    if (from!=NULL) fprintf(tempto, "<li><b>我来自:</b>%s\n", from);
    fprintf(tempto, "<li><b>我的建议:</b>\n");
    fprintf(tempto, "%s\n", comment);
    fprintf(tempto, "</ul>\n\n");
/*将 guests.html 中 "<! ENDHEAD >" 以后的字符串复制到 TEMP 文件中去 */
    filecopy(tempfrom, tempto);
    fclose(tempto);
/*重新打开临时文件和 guests.html 文件*/
    tempfrom = cgifopen(tn2, "r");
    tempto = cgifopen("\usr\people\tribo\public_html\guests.html","w");
/*打开留言簿*/
    filecopy(tempfrom, tempto);   /*用 TEMP 文件的内容覆盖 guests.html 文件*/
    fclose(tempto);
    unlink(tn2);   /*删除临时文件*/
    /*程序最后的代码用于向用户显示一些欢迎信息，以示留言成功*/
```

```
printf("Content-type: text/html%c%c",10,10);
printf("<FONT SIZE=\"+2\">");
printf("<P>非常感谢您的留言");
printf("<P>欢迎您经常光顾本主页！ ");
printf("<\/FONT>");
return 0;
}
/*以下为本程序所使用的 NCSA 的 util.c 所提供的部分函数和本书所补充的函数*/
void getword(char *word, char *line, char stop)
{
    int x = 0,y;
    for(x=0;((line[x]) && (line[x] != stop));x++)
        word[x] = line[x];
    word[x] = '\0';
    if(line[x]) ++x;
    y=0;
    while(line[y++] = line[x++]);
}
char *makeword(char *line, char stop)
{
    int x = 0,y;
    char *word = (char *) malloc(sizeof(char) * (strlen(line) + 1));
    for(x=0;((line[x]) && (line[x] != stop));x++)
        word[x] = line[x];
    word[x] = '\0';
    if(line[x]) ++x;
    y=0;
    while(line[y++] = line[x++]);
    return word;
}
char *fmakeword(FILE *f, char stop, int *cl)
{
    int wsize;
    char *word;
    int ll;
    wsize = 102400;
    ll=0;
    word = (char *) malloc(sizeof(char) * (wsize + 1));
```

```
while(1)
 {
     word[ll] = (char)fgetc(f);
     if(ll==wsize)
       {
         word[ll+1] = '\0';
         wsize+=102400;
         word = (char *)realloc(word,sizeof(char)*(wsize+1));
       }
     --(*cl);
     if((word[ll] == stop) || (feof(f)) || (!(*cl)))
     {
         if(word[ll] != stop) ll++;
         word[ll] = '\0';
         return word;
     }
     ++ll;
   }
}
char x2c(char *what)
{
    register char digit;
    digit = (what[0] >= 'A' ? ((what[0] & 0xdf) - 'A')+10 : (what[0] - '0'));
    digit *= 16;
    digit += (what[1] >= 'A' ? ((what[1] & 0xdf) - 'A')+10 : (what[1] - '0'));
    return(digit);
}
void unescape_url(char *url)
{
    register int x,y;
    for(x=0,y=0;url[y];++x,++y)
    {
        if((url[x] = url[y]) == '%')
        {
            url[x] = x2c(&url[y+1]);
            y+=2;
        }
    }
    url[x] = '\0';
```

```
}
void plustospace(char *str)
{
    register int x;
    for(x=0;str[x];x++) if(str[x] == '+') str[x] = ' ';
}
/*以下的函数用于获取表单上某一元素所储存的数据*/
/*当表单元素不含数据时返回 NULL 指针*/
char* getcgivalue(char *s)
{
 int x,length, i=0;
 char *buf;
 for (x = 0; x <= num_value; x++)
 {
   if (strcmp(entries[x].name,s) == 0)
   {
       if((length=strlen(entries[x].val))==0)     return NULL;
       if(( buf=(char *)malloc(length))==NULL)
        {
          printf("Not enough memory to allocate buffer \n");
          exit(0);
        }
      while((entries[x].val[i])!='\0')
      {
       buf[i]=entries[x].val[i];
       i++ ;
       }
       buf[i]='\0';
   }
}
return buf;
}
/*以下函数用于检查用户输入的 E-mail 地址的合法性*/
int goodemailaddress(char    * addr)
{
  if (strchr(addr, '@') == NULL)     return 0;
  if (addr[strlen(addr - 1)] == '@' || addr[strlen(addr - 1)] == '.')
     return 0;
  if (strchr(strchr(addr, '@'), '.') == NULL) return 0;
```

```
    if (strchr(addr, ' ') != NULL)      return 0;
    if (addr[0] == '@' || addr[0] == '.')      return 0;
     return 1;
}
FILE  * cgifopen(char  * filename, char  * mode)
{
    FILE  * fp;
    fp = fopen(filename, mode);
    if (fp == NULL)
    {
        printf("Content-type: text/html%c%c",10,10);
        printf("<P>非常抱歉，系统无法打开暂时文件!\n");
        printf("<P>请您等候片刻以后再试或用别的方式将留言转给我们!\n");
        exit(1);
    }
    return(fp);
}
void filecopy(FILE  * from, FILE  * to)
{
     char c;
     while (1)
     {
          c = fgetc(from);
          if(feof(from))
               return;
          fputc(c,to);
      }
       fclose(from);
}
```

本程序还有改进的余地，可以让 CGI 程序根据用户定义的数量和顺序显示客户留言。实际上 Web 页面的贵宾留言簿的最佳实现方式是使用一个小型的数据库。本书的第九章将利用 VB 的数据库功能实现一个贵宾留言簿。

7.1.4　给需要身份认证的 Web 资源增加用户

一般情况下任何人都可以访问 Web 服务器上的 WWW 资源，但 Web 服务器管理员也可以让服务器上的某些资源只能被某些特定的组和用户访问，这些组和用户是由 Web 服务器管理员定义的。对于一些商业化的 Web 服务器产品，管理员可以通过浏览器在本地或远程调用服务器的管理模块很方便的对服务器上需要进行身份认证的资源增加、删除或修改用户。而对于一些共享的 Web 服务器软件（比如 NCSA），服务器管理员一般只能利用服

务器本身提供的工具在本地进行用户的增加、删除和修改。事实上也可以为这些共享的 Web 服务器软件编写一些 CGI 程序以实现这些功能。考虑到目前大部分共享的 Web 服务器软件与 NCSA HTTPd 兼容，本节将以 NCSA HTTPd 为平台编写一个给需要身份认证的 Web 资源增加用户的 CGI 程序。

Web 服务器 NCSA HTTPd 通常有四个设置文件：access.conf、httpd.conf、mime.types 和 srm.conf。access.conf 文件说明了客户端用户对所有 Web 服务器资源的访问权限。服务器管理员也可以再在 Web 服务器资源目录树下的某一目录内放置一个访问控制文件，该文件将说明本目录和本目录下的所有子目录的访问控制，文件的具体名称将由 srm.conf 设置。如果没有这个访问控制文件，Web 服务器对这一目录的管理将以 access.conf 为准。

例如假设 srm.conf 设置每个资源目录的访问控制文件名为“.htaccess”（注：第一个字符为‘.’的文件名所表示的文件在 UNIX 系统上为隐含文件，使用命令“ls -a”才能看到），在 Web 服务器的资源目录下的文件“.htaccess”有如下的内容：

```
AuthName Add_Password Test
AuthType Basic
AuthUserFile /usr/local/etc/httpd/htdocs/zhangys/.htpasswd
<Limit GET POST>
 require valid-user
</Limit>
```

文件的第一行说明了本目录下的资源名称为“Add_Password Test”，第二行说明了身份认证的方法是“Basic”，第三行说明了存放用户和用户口令的文件，最后三行说明了只有在文件“ /usr/local/etc/httpd/htdocs/zhangys/.htpasswd ”中有帐号的文件才能访问本目录的资源。

存放用户帐号的文件具有如下的形式：

```
zhangys:6vIGcoHfQJJNU
wangy:EQE4hKKdOcdH.
norton:fHn/ExLz3swEw
qqq:M8pJ9pWTXMhKU
```

文件中的字符“:”前是用户名，在它后面是用户的口令，口令是经过加密处理的。可以用 NCSA HTTPd 提供的工具 htpasswd 直接向文件中增加用户和经过加密处理的口令。

当客户端用户访问了具有以上“.htaccess”文件的目录中的资源时，浏览器将自动弹出一个如图 7-4 所示的要求进行身份认证的对话框。客户端用户只有在通过身份认证以后才能访问相应的资源。

本节的 CGI 程序可以给存放访问控制某一目录的用户名和口令的文件“.htpasswd”增加一个用户名和口令，Web 服务器管理员可以利用它完成部分非本地的用户管理任务。该程序所要处理的 Web 页面上的表单的 HTML 代码如下所示：

```
<html><head><title>贵宾留言簿</title></head>
<body>
<h3>贵宾留言簿测试</h3>
<form action="/cgi-bin/ap.exe" method="post">
```

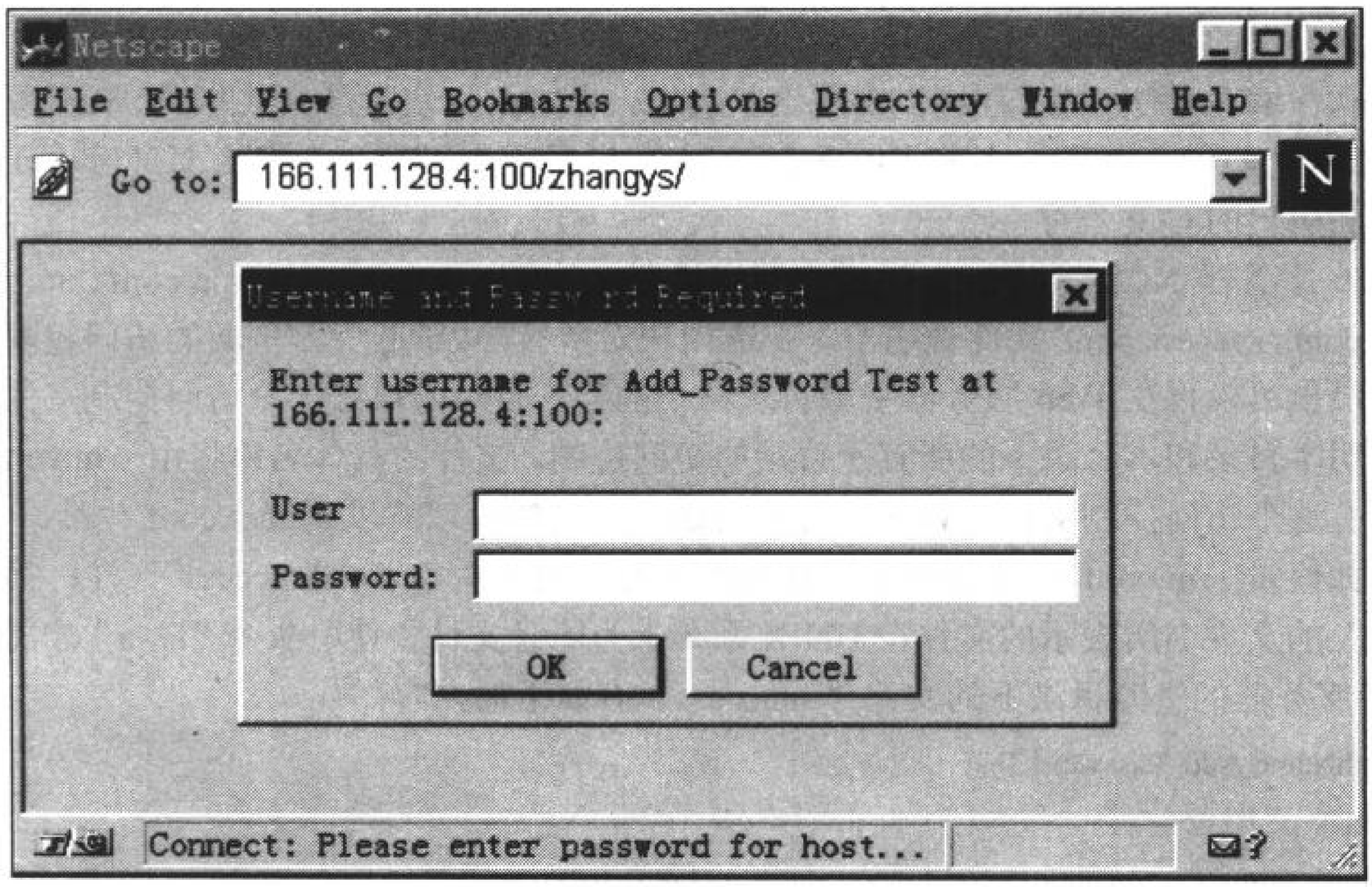

图 7-4　身份认证对话框

```
用户名: <input type="text" name="name"><br>
密码 1: <input type="password" name="passwd1"><br>
密码 2: <input type="password" name="passwd2"><br>
<input type="submit" value=" 执行 ">
</form>
</body></html>
```

该 CGI 程序的源代码如下：

```
/* addpasswd.c */
#include <stdio.h>
#include <string.h>
#include <sys/signal.h>
#include <stdlib.h>
#include <time.h>
#define USER_FILE "/usr/local/etc/httpd/htdocs/zhangys/.htpasswd"
#define MAX_ENTRIES 10000
#define BUFSIZE 1024
typedef struct
{
    char *name;
    char *val;
} entry;
int   num_value=0;
entry entries[MAX_ENTRIES];
```

```
char *makeword(char *line, char stop);
char *fmakeword(FILE *f, char stop, int *len);
char x2c(char *what);
char *getcgivalue(char *s);
char *crypt(char *pw, char *salt);
void unescape_url(char *url);
void plustospace(char *str);
void add_password(char *user, char *pw, FILE *f);
void filecopy(FILE  * from, FILE  * to);
static unsigned char itoa64[] =  "./0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmn
opqrstuvwxyz";            /* 0 ... 63 => ascii - 64 */
to64(s, v, n)
  register char *s;
  register long v;
  register int n;
{
    while (--n >= 0)
    {
        *s++ = itoa64[v&0x3f];
        v >>= 6;
    }
}

main( )
{
  FILE *fpass, *fptmp;
  int x,i, contentlength;
  char *username, *passwd1, *passwd2, *tn;
  char  buf[BUFSIZE];
/*以下代码用于 CGI 程序的初始化准备工作，只允许使用 POST 方法*/
  if(strcmp(getenv("REQUEST_METHOD"),"POST")==0)
  {
   if(strcmp(getenv("CONTENT_TYPE"),"application/x-www-form-urlencoded"))
     {
      printf("Content-type: text/html%c%c",10,10);
      printf("This script can only be used to decode form results. \n");
      exit(1);
      }
      contentlength = atoi(getenv("CONTENT_LENGTH"));
```

```
for(x=0;contentlength && (!feof(stdin));x++)
{
    num_value=x;
    entries[x].val = fmakeword(stdin,'&',&contentlength);
    plustospace(entries[x].val);
    unescape_url(entries[x].val);
    entries[x].name = makeword(entries[x].val,'=');
}
  }
 else
 {
      printf("Content-type: text/html%c%c",10,10);
      printf("非常抱歉，本 CGI 程序只能处理表单用 POST 方法传来的数据!\n");
 }
printf("Content-type: text/html%c%c",10,10);  /*输出 MIME 类型*/
/* 以下代码用于获取用户的输入，并对输入做一些基本的检查*/
username=getcgivalue("name");
passwd1=getcgivalue("passwd1");
passwd2=getcgivalue("passwd2");
if(username==NULL)
{
      printf("<P>非常抱歉，您还没有输入用户名\n");
      exit(1);
 }
if(strlen(passwd1)<6)
{
    printf("<P>非常抱歉，密码必须至少有六个字符的长度\n");
    exit(1);
 }
if(strcmp(passwd1,passwd2))
{
    printf("<P>两次输入的密码不相符合,请您再试一遍！ ");
    exit(1);
 }
tn = tmpnam(NULL);  /* 获取一个唯一的临时文件名 */
if(!(fptmp = fopen(tn,"w")))  /*打开一个临时文件 */
{
    fprintf(stderr,"Could not open temp file.\n");
    exit(1);
```

```
    }
  if(!(fpass = fopen(USER_FILE,"r")))      /*打开存放用户名称和口令的文件*/
  {
      fprintf(stderr, "Could not open passwd file for reading.\n",USER_FILE);
      exit(1);
    }
/*以下 for 循环用于检查客户端用户输入的用户名是否和已经存在的用户名重名。
若不重名，将 passwd 文件的内容拷入临时文件*/
  for(i = 0; ; i++)
    {
     register int len;
     len = strlen(username);
     fgets(buf, BUFSIZE, fpass);
     if(feof(fpass))
       break;
     if((strncmp(buf, username, len) == 0)&& (buf[len] == ':'))
      {
        printf("<P>用户名 %s 已经存在，请重新取一个用户名!\n",username);
        exit(0);
       }
     else
       fputs(buf, fptmp);
     }
/*对用户的口令进行加密处理并将用户名和加密处理后的口令存入临时文件*/
    add_password(username,passwd1,fptmp);
    fclose(fpass);
    fclose(fptmp);
    if(!(fptmp = fopen(tn,"r")))   /*重新打开临时文件*/
    {
        fprintf(stderr,"Could not open temp file.\n");
        exit(1);
    }
    if(!(fpass = fopen(USER_FILE,"w")))   /*重新打开 passwd 文件*/
    {
        fprintf(stderr, "Could not open passwd file for writing.\n",USER_FILE);
        exit(1);
    }
    filecopy(fptmp, fpass);   /*用临时文件覆盖 passwd 文件*/
    unlink(tn);   /*删除临时文件*/
```

```
    printf("<P>新用户已经增加到数据库中!\n");
    exit(0);
  }
/*以下为本程序所使用的 NCSA 的 util.c 所提供的部分函数和本书所补充的函数*/
char *makeword(char *line, char stop)
{
    int x = 0,y;
    char *word = (char *) malloc(sizeof(char) * (strlen(line) + 1));
    for(x=0;((line[x]) && (line[x] != stop));x++)
        word[x] = line[x];
    word[x] = '\0';
    if(line[x]) ++x;
    y=0;
    while(line[y++] = line[x++]);
    return word;
}
char *fmakeword(FILE *f, char stop, int *cl)
{
    int wsize;
    char *word;
    int ll;
    wsize = 102400;
    ll=0;
    word = (char *) malloc(sizeof(char) * (wsize + 1));
    while(1)
    {
        word[ll] = (char)fgetc(f);
        if(ll==wsize)
        {
            word[ll+1] = '\0';
            wsize+=102400;
            word = (char *)realloc(word,sizeof(char)*(wsize+1));
        }
        --(*cl);
        if((word[ll] == stop) || (feof(f)) || (!(*cl)))
        {
            if(word[ll] != stop) ll++;
            word[ll] = '\0';
            return word;
```

```
            }
            ++ll;
        }
    }
    char x2c(char *what)
    {
        register char digit;
        digit = (what[0] >= 'A' ? ((what[0] & 0xdf) - 'A')+10 : (what[0] - '0'));
        digit *= 16;
        digit += (what[1] >= 'A' ? ((what[1] & 0xdf) - 'A')+10 : (what[1] - '0'));
        return(digit);
    }
    void unescape_url(char *url)
    {
        register int x,y;
        for(x=0,y=0;url[y];++x,++y)
        {
            if((url[x] = url[y]) == '%')
            {
                url[x] = x2c(&url[y+1]);
                y+=2;
            }
        }
        url[x] = '\0';
    }
    void plustospace(char *str)
    {
        register int x;
        for(x=0;str[x];x++) if(str[x] == '+') str[x] = ' ';
    }
    /*以下的函数用于获取表单上某一元素所储存的数据*/
    /*当表单元素不含数据时返回 NULL 指针*/
    char* getcgivalue(char *s)
    {
       int x,length;
       int i=0;
       char *buf;
       for (x = 0; x <= num_value; x++)
       {
```

```
        if (strcmp(entries[x].name,s) == 0)
        {
            if((length=strlen(entries[x].val))==0)     return NULL;
            if(( buf=(char *)malloc(length))==NULL)
             {
                printf("Not enough memory to allocate buffer \n");
                exit(0);
             }
                while((entries[x].val[i])!='\0')
                {
                   buf[i]=entries[x].val[i];
                   i++ ;
                }
                buf[i]='\0';
            }
        }
        return buf;
}
/*以下程序可以对用户口令进行加密，并将用户名和经过处理后的口令存入文件*/
void add_password(char *user, char *pw, FILE *f)
{
        char *cpw, salt[3];
        (void)srand((int)time((time_t *)NULL));
        to64(&salt[0],rand( ),2);
        cpw = crypt(pw,salt);
        free(pw);
        fprintf(f,"%s:%s\n",user,cpw);
}
/*以下程序用于文件拷贝*/
void filecopy(FILE   * from, FILE   * to)
{
        char c;
        while (1)
        {
            c = fgetc(from);
            if(feof(from))
                return;
            fputc(c,to);
        }
```

```
    fclose(from);
}
```

读者还可以编写修改用户口令、增加组以及删除用户和组的 CGI 程序。当然，这些 CGI 程序都应该放在一个只有 Web 服务器管理员才有权限存取的地方。本程序仅适用于 UNIX 系统下和 NCSA HTTPd 兼容的 Web 服务器。

7.1.5　主页计数器

在第二章本书曾经介绍了一个文本主页计数器。该计数器在主页被访问时被 SSI 激活。如果 Web 服务器的 SSI 功能没有被打开，就不能使用这类计数器。这时一般都要使用图形计数器，而这类图形计数器一般都是由 CGI 程序实现的。CGI 程序根据计数文件中的数字动态生成一个 GIF 或 JPG 等浏览器支持的图像文件，在产生 Web 页面的 HTML 文件中加入如下的语句：<img src="/cgi-bin/count.cgi >，就可以向 Web 页面的访问者显示曾经访问过该 Web 页面的人数了。

在 WWW 上可以找到很多免费共享的主页图形计数器。其实这类图形计数器的原理非常简单，有时候编写这类计数器程序所用的时间并不比在万维网上寻找这类计数器并最终完成其编译和安装所花费的时间多多少。您完全可以亲自动手为自己的主页编写一个这样的计数器以体验一下 CGI 编程的乐趣。本节的例子将向您说明这类计数器的基本原理。

本节的图形计数器是在第二章的文本计数器的基础上实现的。该 CGI 程序将从计数文件中读取访问人数，并根据访问人数的每一位具体数字调用不同的小 GIF 图像，最终将这些小 GIF 图像拼成一幅大的 GIF 图像并向 STDOUT 输出。因此在编写 CGI 程序之前您还得准备十个有关阿拉伯数字的小 GIF 图像，WWW 上可以找到很多有关阿拉伯数字的 GIF 图像，您也可以利用一般的图像编辑处理软件自己制作这些 GIF 图像。当然这些 GIF 图像应该有一致的尺寸大小和背景。

为了能够在 CGI 程序中方便的处理 GIF 图像，本程序使用了 GD 图形处理库，您可以在 WWW 上的站点 http://www.boutell.com/gd/找到这个图形库的源代码。当然，如果十分了解 GIF 图像的文件格式，您也可以不用 GD 库而自己编写这方面的代码。CGI 程序的源代码如下所示，图 7-5 就是该 CGI 程序所生成的一个图像。

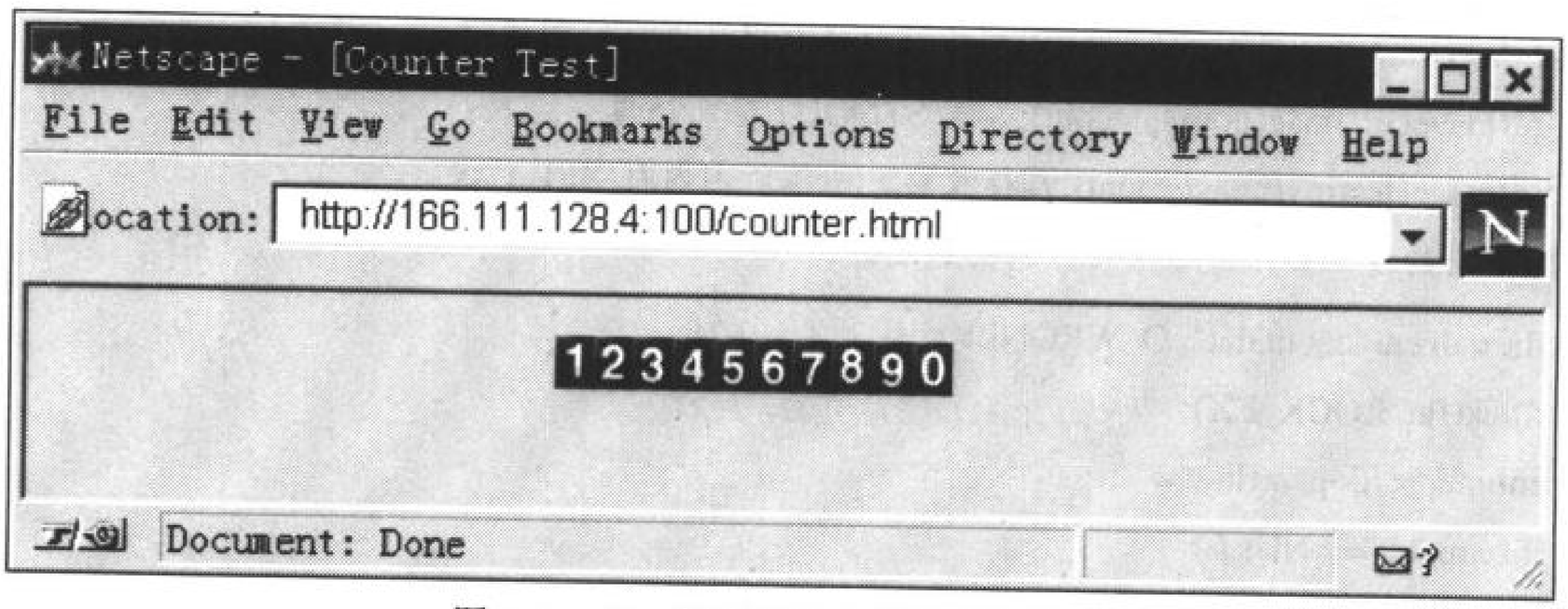

图 7-5　CGI 程序所生成的图像计数器

```
/* count.c */
#include "gd.h"
```

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <sys/file.h>
gdImagePtr creatimage(char *count);
char *gif[]={"0.gif", "1.gif", "2.gif", "3.gif", "4.gif", "5.gif", "6.gif", "7.gif", "8.gif", "9.gif"};
int main( )
{
  FILE  * counter;
  char count[100];
  int fln, ncount;
  gdImagePtr imagecount;  /*声明一个 GIF 图像 */

  /*以只读方式打开计数文件并将其锁定*/
  fln = open("count.dat", O_RDONLY);
  flock(fln, LOCK_EX);
  counter = fdopen(fln, "r");
  if (counter == NULL)
    {
      printf("无法打开计数器文件\"count.dat\"\n");
      exit(0);
    }

  printf("Content-type: image/gif%c%c",10,10); /*说明 CGI 程序的 MIME 类型*/
  fgets(count,100, counter);  /*以字符串方式读取计数文件中的数据*/
  fclose(counter);
  imagecount=creatimage(count);/*根据读取的数据生成一幅图像*/
  gdImageGif(imagecount, stdout); /*向 STDOUT 输出图像*/
  gdImageDestroy(imagecount); /*释放 GIF 图像所占内存空间*/
  /*以写的方式打开计数文件并将其锁定*/
  fln = open("count.dat", O_WRONLY);
  flock(fln, LOCK_EX);
  counter = fdopen(fln, "w");
  if (counter == NULL)
    {
      printf("无法打开计数器文件 \"count.dat\" ");
      exit(0);
    }
```

```
    ncount=atoi(count)+1;
    fprintf(counter, "%d\n", ncount);
    fclose(counter);
    exit(0);
}

/*以下函数用于根据访问人数（字符串的形式）生成 GIF 图像*/
gdImagePtr creatimage(char *count)
{
    gdImagePtr im,im_in;
    FILE *in;
    char sn[2];
    int black, number, len, i, ntemp;
    int imagewidth, width, height;
    /*调入一个数字 GIF 图像以获取图像的尺寸大小*/
    in = fopen(gif[0], "rb");
    im_in = gdImageCreateFromGif(in);
    fclose(in);
    width= gdImageSX(im_in);
    height=gdImageSY(im_in);
    gdImageDestroy(im_in); /*释放 GIF 图像所占内存空间*/
    len=strlen(count)-1;
    /* 确定访问人数的位数，由于本书编程的 UNIX 系统平台在文件结束时自动在文件中加一回车符，
在计算访问人数的位数时应将其扣除，具体的 UNIX 系统平台可能不一样，应根据具体情况决定是否对 len
减--。*/
    imagewidth=len   *width;
    /*设置图像的尺寸大小*/
    im = gdImageCreate(imagewidth, height);
    /*设置图像的背景颜色为黑色*/
    black = gdImageColorAllocate(im, 0, 0, 0);
    /*以下的 for 循环根据访问人数每一位数字的不同分别把不同的 GIF 数字图像
    拷贝到 GIF 图像 im 上*/
    for(i=0;i<len;i++)
    {
        sn[0]=count[i];
        sn[1]='\0';
        ntemp=atoi(sn);
        in = fopen(gif[ntemp], "rb");
        im_in = gdImageCreateFromGif(in);
```

```
        fclose(in);
        gdImageCopy(im, im_in, i*width, 0, 0, 0, width, height);
        gdImageDestroy(im_in);
    }
    return im;
}
```

本 CGI 程序只是用来说明图像计数器的原理，还有许多可以改进的地方。比如 CGI 程序的用户应该可以定义计数文件和阿拉伯数字的 GIF 图像文件的存放位置（当前的程序是将所有的文件放在一个目录下）。CGI 程序也可以不对某一台主机或某几个域内的主机的访问计数。实现这些功能并不困难，读者可以在这几方面对本 CGI 程序进行修改。

如果要在 Windows 95 和 NT 平台上运行本程序，需要对程序代码进行一些修改。有关锁定文件的代码应该改成 Windows 95 或 NT 系统所支持的代码，在将所生成的图像向 STDOUT 输出时必须调用函数 setmode 将输出方式设置成二进制方式。本书的第六章提供了一个 Windows 95 下使用 GD 库进行 CGI 编程的例子

由于使用了 GD 库，本程序的编译和普通程序有所不同，在 UNIX 系统上使用 Make 命令，计数器程序源码 count.c 应该放在 GD 库所在的目录。Makefile 文件的内容如下所示：

```
CC=cc
AR=ar
CFLAGS=-O
LIBS=-L./ -lgd -lm
all: libgd.a ncount
count: count.o libgd.a gd.h
$(CC) count.o -o count $(LIBS)
libgd.a: gd.o      gd.h
rm -f libgd.a
$(AR) rc libgd.a gd.o
```

在 DOS 或 Windows 95 平台下编译时则比较简单。程序源码仍然应和 GD 库放在一起，可以利用 BC 或 VC 这样的集成编译环境的工程项目功能对程序进行编译。也可以使用行编译命令。比如若使用的是 BC 编译器，行编译命令则为："bcc32　count.c　gd.c"。

7.2　Visual Basic 编程实例

7.2.1　用 VB 编写的 CGI 程序处理中文信息

用 VB 编译生成的 CGI 程序一般不能正确处理中文信息。这主要表现在两个方面，一是 CGI 程序向 STDOUT 输出的中文在 Web 页面上无法正确显示；一是 CGI 程序无法对 Web 页面上表单传来的中文信息正确解码。

对于第一个问题，在向 STDOUT 输出中文字符串时可以在该字符串后跟着输出一些空格。当使用 HTML 标识符<P>、</P>对 Web 页面进行排版时，浏览器在显示该 Web 页

面时会“吃掉”多余的空格而只保留一个。在这种情况下这些空格对 Web 页面的外观基本没有影响。如果使用 HTML 标识符<PRE>、</PRE>对 Web 页面进行排版，由于空格不能被浏览器吃掉，Web 页面的外观将会受到很大的影响。不过这时可以用 HTML 的表格<table>、<P>代替<PRE>对 Web 页面进行排版。

第二个问题则由于汉字是双字节的原因。对于一个特殊的单字节字符比如“/”，浏览器通常将它译码成十六进制数的形式比如%2F，“%”表示它后面跟的是两位十六进制数。当 VB 程序对其进行处理时调用 Chr$函数就可以将其恢复为原来的面貌。一个汉字则被浏览器译码成四位十六进制数，比如%D5%C5，如果 CGI 程序还像以前一样分别调用 Chr(D5)和 Chr(C5)，由于 D5、C5 都不是正常的单字节十六进制数码，Chr 函数返回空，汉字将无法正确还原。正确的作法应该是将有关汉字的四位十六进制数一起传给函数 Chr，如 Chr(D5C5)，此时汉字才能被正确还原（用 C 语言编写的 CGI 程序则不存在这个问题）。

因此可以让 CGI 程序对四位连续的十六进制数一起进行译码以使汉字被正确还原。但在这种情况下，当客户端用户输入了两个连续的 Web 系统保留字符时 CGI 程序又可能把它们当成汉字来处理。这时可以让 CGI 程序在需要对四位连续的十六进制数进行译码时首先检查前面两位是否为 Web 系统的保留字符，如果是则仍然按照单字节的字符处理。不过如果客户端用户在表单内填写了很多汉字，CGI 程序的负担将会大大加重。事实上大多数情况下客户端用户很少会使用两个连续的 Web 系统的保留字符，可以只让 CGI 程序对最容易出现的情形如“://”（当客户端用户在表单中输入某一 URL 时会出现这种情况）进行检查。

即使是用中文版 VB 编译的 CGI 程序也存在以上两个中文处理问题。本章所列举的 VB-CGI 实用程序都将利用本小节所提供的方法对汉字进行译码和输出。

7.2.2 主页意见箱

本小节将用 VB-CGI 程序实现 7.1.2 节所提供的主页意见箱。一般，将客户端用户在主页意见箱中填写的信息传送给 Web 服务器管理员的最佳方法就是电子函件。在 UNIX 平台下通过 system 函数调用系统 Mail 或 sendmail 命令可以很方便的实现这一点。但在 VB 编程环境下则没有这么方便的函数可以使用。

为了能够在 VB-CGI 程序中通过 Internet 电子函件将用户填写的信息传给 Web 服务器管理员，本书编写了一个动态库，该动态库提供一个类似 UNIX 系统命令 Mail 的函数 sendmail。假设本节 VB-CGI 程序所要处理的表单和 7.1.2 节所要处理的一样。该 CGI 程序的源码如下所示：

```
'comment.bas
'声明程序中要用到的三个 WINDOWS API 函数，一个用于处理标准输入输出文件句柄，另两个用于写文件
Declare Function GetStdHandle Lib "kernel32"    (ByVal nStdHandle As Long) As Long
Declare Function ReadFile Lib "kernel32"   (ByVal hFile As Long,
    lpBuffer As Any,   ByVal nNumberOfBytesToRead As Long,
    lpNumberOfBytesRead As Long, lpOverlapped As Any) As Long
Declare Function WriteFile Lib "kernel32"   (ByVal hFile As Long,
```

```
    ByVal lpBuffer As String, ByVal nNumberOfBytesToWrite As Long,
    lpNumberOfBytesWritten As Long, lpOverlapped As Any) As Long
'声明程序要使用的用于发送电子函件的函数
Declare Function sendmessage Lib "E:\TEMP\DLL\sendmail.DLL" (ByVal what As String,
        ByVal subject As String, ByVal mailbox As String) As Integer
Public Const STD_INPUT_HANDLE = -10&
Public Const STD_OUTPUT_HANDLE = -11&
Public Const FILE_BEGIN = 0&
    Public hStdIn   As Long    ' 标准输入文件句柄
    Public hStdOut   As Long    ' 标准输出文件句柄
Public sFormData   As String   ' 用于存储没有经过 URI 译码的用户输入数据
Public lContentLength As Long
Public CGI_RequestMethod
Sub Main( )
Dim CGI_ContentLength   As String,   CGI_QueryString As String
Dim sBuff   As String
Dim lBytesRead As Long ,   rc As Long , I As Long
Dim chinesetail As String, email As String, system As String, url As String
Dim comment As String, mailmessage As String
Dim navigator As String, host As String
Dim datenow As String, timenow As String
Dim statein As Integer
  '获取标准输入输出文件句柄
hStdIn = GetStdHandle(STD_INPUT_HANDLE)
hStdOut = GetStdHandle(STD_OUTPUT_HANDLE)
  '获取有关的环境变量
CGI_RequestMethod = Environ("REQUEST_METHOD")
CGI_QueryString = Environ("QUERY_STRING")
CGI_ContentLength = Environ("CONTENT_LENGTH")
lContentLength = Val(CGI_ContentLength)
sBuff = String(lContentLength, Chr$(0))
    OutPut "Content-type: text/html" & vbCrLf      '输出 MIME 信息
OutPut "<FONT SIZE=""+2""">"
 'CGI 程序的初始化工作
If CGI_RequestMethod = "POST" Then
    sBuff = String(lContentLength, Chr$(0))
    rc = ReadFile(hStdIn, ByVal sBuff, lContentLength, lBytesRead, ByVal 0&)
    sFormData = Left$(sBuff, lBytesRead)
ElseIf CGI_RequestMethod = "GET" Then
```

```
    sFormData = CGI_QueryString
Else
    OutPut "Unknow Form Method !"
End If
 '为了在页面上正确显示中文生成一个空格串
     chinesetail = String(400, " ")
 '以下代码用于获取相关的环境变量和用户输入的信息
navigator = Environ("HTTP_USER_AGENT")
host = Environ("REMOTE_HOST")
If Len(host) = 0 Then
   host = Environ("REMOTE_ADDR")
End If
email = GetCgiValue("email")
system = GetCgiValue("SYSTEM")
subject = GetCgiValue("URL_Subject")
url = GetCgiValue("URL")
comment = GetCgiValue("URL_Comment")
If Len(comment) = 0 Then
    OutPut "<P>非常抱歉！您还没有提出建议！" & chinesetail
    Exit Sub
End If
'根据有关信息存入字符串 mailmessage 以便发送
mailmessage = "客户端用户所用的浏览器：" & navigator & vbCrLf
If system <> "浏览器所使用的系统平台:" Then
 mailmessage = mailmessage & "浏览器所使用的系统平台:" & system & vbCrLf
End If
mailmessage = mailmessage & "客户端用户的主机：" & host & vbCrLf
If Len(email) <> 0 Then
 mailmessage = mailmessage & "客户端用户的电子函件信箱：" & email & vbCrLf
End If
mailmessage = mailmessage & "发送建议的时间：" & Time$ & Date$ & vbCrLf
If Len(subject) <> 0 Then
   mailmessage = mailmessage & "客户端用户的建议主题：" & subject & vbCrLf
End If
If Len(url) <> 0 Then
   mailmessage = mailmessage & "客户端用户的建议所指的主页：" & url & vbCrLf & vbCrLf
End If
mailmessage = mailmessage & "客户端用户的具体建议内容：" & vbCrLf
mailmessage = mailmessage & comment & vbCrLf
```

```
OutPut "<P>请稍候，系统正在传递您的信息....." & chinesetail
'调用函数 sendmessage 发送用户的建议
statein = sendmessage(mailmessage, "Comment About Homepage" , "zhangys@hotmail.com")
If statein <> 0 Then
   OutPut "<P>非常抱歉，系统无法传递您的意见!" & chinesetail
   OutPut "<P>请您等候片刻以后再试或用别的方式将意见转给我们!" & chinesetail
   Exit Sub
End If
OutPut "<P>您的建议已经被成功传送给 Web 服务器管理员！" & Name & chinesetail
OutPut "<P>非常感谢您对本主页的宝贵意见！" & chinesetail
OutPut "<P>欢迎您经常光顾本主页！" & chinesetail
OutPut "</FONT>"
End Sub

Sub OutPut(s As String)      ' 本子程序用于向标准输出写信息
Dim lBytesWritten As Long
s = s & vbCrLf
WriteFile hStdOut, s, Len(s), lBytesWritten, ByVal 0&
End Sub

Function   GetCgiValue(cgiName As String) As String
' 本子程序可以获取表单上某一元素的数据
Dim delim2 As Long ' position of "="
Dim delim1 As Long ' position of "&"
Dim n As Integer, pointer1 As Long, pointer2 As Long
Dim length As Long, length1 As Long
Dim tmpstring1 As String, tmpstring2 As String
pointer1 = 1
pointer2 = 1
delim2 = InStr(pointer2, sFormData, "=")
pointer2 = delim2 + 1
Do
length = delim2 - pointer1
tmpstring1 = Mid(sFormData, pointer1, length)
delim1 = InStr(pointer1, sFormData, "&")
pointer1 = delim1 + 1
length1 = delim1 - pointer2
If delim1 = 0 Then length1 = lContentLength - pointer2
If tmpstring1 = cgiName Then
```

```
    tmpstring2 = Mid$(sFormData, pointer2, length1)
    GetCgiValue = UrlDecode(tmpstring2)
    Exit Do
  End If
  If delim1 = 0 Then
    Exit Do
  End If
  delim2 = InStr(pointer2, sFormData, "=")
  pointer2 = delim2 + 1
  Loop
  End Function

' 本函数可以对用户输入的数据进行 URI 解码
Public Function UrlDecode(ByVal sEncoded As String) As String
Dim pointer   As Long          ' sEncoded position pointer
Dim pos   As Long          ' position of InStr target
Dim temp As String

If sEncoded = "" Then Exit Function
pointer = 1
Do  ' 本循环体用于将“+”转换成空格
    pos = InStr(pointer, sEncoded, "+")
    If pos = 0 Then Exit Do
    Mid$(sEncoded, pos, 1) = " "
    pointer = pos + 1
Loop
pointer = 1
On Error GoTo errorUrlDecode
Do  '本循环体用于将%XX 转换成字符。对于两个连续的%XX，如果第一个%XX 不是某些特殊的
Web 系统保留字符，将把它们转换成汉字
    pos = InStr(pointer, sEncoded, "%")
    If pos = 0 Then Exit Do
    temp = Chr$("&H" & (Mid$(sEncoded, pos + 1, 2)))
    If Mid(sEncoded, pos + 3, 1) = "%" And (temp <> ":") And (temp <> "/")
      And (temp <> "(") And (temp <> ")") And (temp <> ".") And (temp <> ",")
      And (temp <> ";") And (temp <> "%") Then
      Mid$(sEncoded, pos, 2) = Chr$("&H" & (Mid$(sEncoded, pos + 1, 2)) _
      & (Mid$(sEncoded, pos + 4, 2)))
      sEncoded = Left$(sEncoded, pos)   &   Mid$(sEncoded, pos + 6)
```

```
        pointer = pos + 1
      Else
        Mid$(sEncoded, pos, 1) = temp
        sEncoded = Left$(sEncoded, pos)   &   Mid$(sEncoded, pos + 3)
      pointer = pos + 1
      End If
    Loop
    On Error GoTo 0          'reset error handling
    UrlDecode = sEncoded
    Exit Function
    errorUrlDecode:
    If Err.Number = 13 Then          ' Type Mismatch error
      Err.Clear
      Err.Raise 65001, , "Invalid data passed to UrlDecode() function."
    Else
      Err.Raise Err.Number
    End If
    Resume Next
    End Function
```

本 CGI 程序所使用的动态库 zys.dll 的源码如下所示，它为 VB-CGI 程序提供了一个发送电子函件的方法。如果要让 Web 页面成为一个发送电子函件的接口（E-mail Gateway），该动态库还应该提供更多的接口参数，比如 Replyto 等等。读者可以根据自己的需要对该动态库进行改进。由于该动态库的编程涉及 WINSOCKET 接口规范，具体的编程细节可以参考其他书籍，本书这里就不做详细的介绍了。

```
// sendmail.cpp
#include   "string.h"
#include "stdio.h"
#include "winsock.h"
#define WINSOCK_VERSION 0x0101  / * Program requires Winsock version 1.1* /
#define DEFAULT_PROTOCOL 0       / * No protocol specified, use default * /
#define NO_FLAGS 0                    / * No special flags specified * /
char sScratchBuffer[100];             / * General purpose buffer for messages * /
int SendMail(SOCKET hSocket,LPSTR message, LPSTR subject , LPSTR mailbox)
{
    BYTE   sReceiveBuffer[4096];          / * Buffer for mail server replies * /
    Int   iLength;                      / * Length of mail server reply * /
    Int   iMsg = 0;                     / * Index into SMTP command array * /
    Int   iEnd;                         / * Pointer to the end of the reply buffer * /
    char   MailMessage[7][10240];
```

```
    char *ptr;
ptr=strchr(mailbox,'@');
/*  设置发给 Mail 服务器的 SMTP 命令 */
      sprintf(MailMessage[0],"HELO %s\r\n",(ptr+1));
      sprintf(MailMessage[1],"MAIL FROM:<nobody@WebServer>\r\n");
      sprintf(MailMessage[2],"RCPT TO:<%s>\r\n",mailbox);
      sprintf(MailMessage[3],"DATA\r\n");
      sprintf(MailMessage[4],"Subject: %s\r\n\r\n",subject);
      sprintf(MailMessage[5],"%s\r\n\r\n.\r\n", message);
      sprintf(MailMessage[6],"QUIT\r\n");
      iEnd = 0;
     do
       {
       if (send(hSocket, (LPSTR)MailMessage[iMsg],strlen(MailMessage[iMsg]), NO_FLAGS) ==
SOCKET_ERROR)
            return(-1);
            iLength = recv(hSocket, (LPSTR)sReceiveBuffer+iEnd,
            sizeof(sReceiveBuffer)-iEnd, NO_FLAGS);
            if (iLength == 0 || iLength == SOCKET_ERROR)        return(-2);
            iMsg++;
            }
      while (iMsg < 7);
      return (0);
}

SOCKET ConnectMailServerSocket(LPSTR HOST_NAME)
 {
   WSADATA wsaData;                          /* Winsock implementation details */
   LPHOSTENT lpHostEnt;                      /* Internet host information structure */
   SOCKADDR_IN sockAddr;                     /* Socket address structure */
   LPSERVENT lpServEnt;                    /* Service information structure */
   short nProtocolPort;                      /* Protocol port */
   int nConnect;                             /* Socket connection results */

   SOCKET hServerSocket = INVALID_SOCKET;      /* Default socket number */
   if (WSAStartup(WINSOCK_VERSION, &wsaData))      return (NULL);
   else                          /* Resolve the host name */
      {
        lpHostEnt = gethostbyname(HOST_NAME);
```

```
        if (!lpHostEnt)     return (NULL);
    else        /* Create the socket
      {
        hServerSocket = socket(PF_INET, SOCK_STREAM, DEFAULT_PROTOCOL);
         if (hServerSocket == INVALID_SOCKET)
           return (INVALID_SOCKET);
        else      /* Configure the socket */
           {       /* Get the time service information */
          lpServEnt = getservbyname("mail", DEFAULT_PROTOCOL);
         if (lpServEnt == NULL)
           nProtocolPort = htons(IPPORT_SMTP);
          else
           nProtocolPort = lpServEnt->s_port;
           /* Define the socket address */
           sockAddr.sin_family = AF_INET;
           sockAddr.sin_port = nProtocolPort;
           sockAddr.sin_addr = *((LPIN_ADDR)*lpHostEnt->h_addr_list);
           /* Connect the socket */
           nConnect = connect(hServerSocket, (PSOCKADDR)&sockAddr,
                              sizeof(sockAddr));
                 if( nConnect)
                      {
                        hServerSocket = INVALID_SOCKET;
                      }
                   }
                }
          }
          return(hServerSocket);
      }

extern "C" int _export FAR PASCAL sendmessage(LPSTR what, LPSTR subject, LPSTR mailbox)
{
    int temp;
    SOCKET hSocket;            /* Socket handle for the SMTP connection */
    char *ptr;
    ptr=strchr(mailbox,'@');
     hSocket = ConnectMailServerSocket((ptr+1) );
     if (hSocket != INVALID_SOCKET)
      {
```

```
        temp = SendMail(hSocket, what,subject,mailbox);
        closesocket(hSocket);
    }
        else return (-1);
        return(temp);
}
int _export FAR PASCAL LibMain(HANDLE hInstance, WORD wDataSeg, WORD wHeapSize, LPSTR
lpszCmdParam)
 {
      if (wHeapSize > 0)      UnlockData (0);
      return(1);
 }

int _export FAR PASCAL WEP (int nParam)
{
      return(1);
}
```

该动态库的资源定义文件内容如下：

```
LIBRARY    sendmail
EXETYPE    WINDOWS
CODE       PRELOAD MOVEABLE DISCARDABLE
DATA PRELOAD MOVEABLE SINGLE
HEAPSIZE   1024
EXPORTS sendmessage
```

本动态库在 Windows 95 下编译通过。CGI 程序在微软的 Personal Web Server 下能正常运行，但在网景公司的 Web 服务器 FastTrack 2.0 (Windows 95 Beta 版)下无法和 Mail Server 建立连接。作者相信这是 Beta 版软件的一个 bug。由于其他的原因，本书没有在其他 Web 服务器上测试本 CGI 程序。

7.2.3 贵宾留言簿

最后，本书将用 VB 实现 7.1.3 节中用 C 语言实现的贵宾留言簿。留言簿文件的插入位置定位仍然和 7.1.3 节中所用的方法相同。程序中调用了 WIN32 API 函数以获得一个唯一的临时文件名。

对于本程序所生成的留言簿，最后的留言出现在文件的最前面。如果要想使最后留言的人的留言出现在留言簿的最后面，只需将留言簿文件中的定位字符串“<! ENDHEAD >”移到留言簿文件中客户留言部分和 HTML 文件结尾部分之间的位置就行了。在留言簿文件的结尾部分，您可以给留言簿加上一些链接以指回您的主页或其他地方。假设本程序所要处理的 HTML 表单和 7.1.3 节的相同。本程序的源代码如下所示：

```
'guestbook.bas
```

```
'声明程序中要用到的四个 Windows API 函数，一个用于处理标准输入输出文件句柄，两个用于读写
文件，一个用于获取唯一的临时文件名
Declare Function GetStdHandle Lib "kernel32" (ByVal nStdHandle As Long) As Long
Declare Function ReadFile Lib "kernel32"  (ByVal hFile As Long,  lpBuffer As Any,
    ByVal nNumberOfBytesToRead As Long, lpNumberOfBytesRead As Long,
    lpOverlapped As Any) As Long
Declare Function WriteFile Lib "kernel32"  (ByVal hFile As Long,
    ByVal lpBuffer As String, ByVal nNumberOfBytesToWrite As Long,
    lpNumberOfBytesWritten As Long, lpOverlapped As Any) As Long
Declare Function GetTempFileName Lib "kernel32"  Alias "GetTempFileNameA"
   (ByVal lpszPath As String,  ByVal lpPrefixString As String,
    ByVal wUnique As Long,  ByVal lpTempFileName As String) As Long
Public Const STD_INPUT_HANDLE = -10&
Public Const STD_OUTPUT_HANDLE = -11&
Public Const FILE_BEGIN = 0&
Public hStdIn  As Long   ' 标准输入文件句柄
Public hStdOut  As Long   ' 标准输出文件句柄
Public sFormData  As String  ' 用于存储没有经过 URI 译码的用户输入数据
Public lContentLength As Long
Public CGI_RequestMethod  As String
Public chinesetail As String

Sub Main( )
Dim CGI_ContentLength  As String,   CGI_QueryString As String
Dim sBuff  As String
Dim lBytesRead As Long,  rc As  Long
Dim I As Long
Dim sEmail As String, sName As String, sURL As String, sfrom As String
Dim sComment As String, tempFileName As String
Dim guestbook As String  ' 留言簿文件名
Dim tempstring As String
'CGI 程序的初始化工作
hStdIn = GetStdHandle(STD_INPUT_HANDLE)
hStdOut = GetStdHandle(STD_OUTPUT_HANDLE)
CGI_RequestMethod = Environ("REQUEST_METHOD")
CGI_QueryString = Environ("QUERY_STRING")
CGI_ContentLength = Environ("CONTENT_LENGTH")
lContentLength = Val(CGI_ContentLength)
sBuff = String(lContentLength, Chr$(0))
```

```
OutPut "Content-type: text/html" & vbCrLf  ' 输出 MIME 类型
OutPut "<FONT SIZE=""+2""">"
If CGI_RequestMethod = "POST" Then
    sBuff = String(lContentLength, Chr$(0))
    rc = ReadFile(hStdIn, ByVal sBuff, lContentLength, lBytesRead, ByVal 0&)
    sFormData = Left$(sBuff, lBytesRead)
ElseIf CGI_RequestMethod = "GET" Then
    sFormData = CGI_QueryString
Else
  OutPut "Unknow Form Method !"
End If
'为了在页面上正确显示中文而生成一个空格串
chinesetail = String(400, " ")
'获取客户端用户的输入
sName = GetCgiValue("name")
sEmail = GetCgiValue("email")
sURL = GetCgiValue("URL")
sfrom = GetCgiValue("from")
sComment = GetCgiValue("URL_Comment")
'对客户端用户的输入进行检查
If Len(sName) = 0 Then
    OutPut "<P>非常抱歉！您还没有填写姓名！" & chinesetail
    Exit Sub
End If
If Len(sComment) = 0 Then
    OutPut "<P>非常抱歉！您还没有提出建议！" & chinesetail
    Exit Sub
End If
If Len(sEmail) <> 0 And IsGoodMail(sEmail) = -1 Then
    OutPut "<P> 非常抱歉，您输入的 E-mail 地址不合法！" & chinesetail
    Exit Sub
End If
If InStr(sURL, "http://") = 0 And InStr(sURL, ".") <> 0 Then
   sURL = "http://" & sURL
End If
'获取唯一的临时文件名和留言簿文件并打开它们
tempFileName = TempFile("c:\windows\temp", "gbk")
guestbook = "e:\netscape\server\docs\guests.html"
Open tempFileName For Output As #1
```

```
Open guestbook For Input As #2
Do      '本循环体用于将留言簿中字符串"<! ENDHEAD >"前面的内容写入临时文件
Line Input #2,  tempstring
Print #1,  tempstring
Loop While tempstring <> "<! ENDHEAD >" And   Not   EOF(2)
'向临时文件中插入客户端用户的留言
Print #1, "<hr>" & vbCrLf
Print #1, "<ul>" & vbCrLf
Print #1, "<li><b>留言时间</b>:  " & Date$ & " " & Time$ & vbCrLf
Print #1, "<li><b>姓名: </b>" & sName & vbCrLf
If Len(sEmail) <> 0 Then
   Print #1, "<li><b>E-mail: </b><a href=""mailto:" & sEmail & """ >" & sEmail & "</a>" & vbCrLf
End If
If Len(sURL) <> 0 Then
   Print #1, "<li><b>我的主页: </b> <a href=""" & sURL & """ >" & sURL & "</a>" & vbCrLf
End If
If Len(sfrom) <> 0 Then
   Print #1, "<li><b>我来自: </b>" & sfrom & vbCrLf
End If
Print #1, "<li><b>我的建议: </b>" & vbCrLf
Print #1, sComment & vbCrLf
Print #1, "</ul>" & vbCrLf
  Do      '本循环体用于将留言簿剩余的东西写入留言簿
  Line Input #2, tempstring
  Print #1, tempstring
  Loop While Not EOF(2)
Close #1
Close #2
Kill guestbook     '删除旧的留言簿
Name tempFileName As guestbook   '将临时文件改成新的留言簿
OutPut "<P>非常感谢您的留言!" & chinesetail
OutPut "<P>欢迎您经常光顾本主页！ " & chinesetail
OutPut "</FONT>"
End Sub

Sub OutPut(s As String)   '本子程序用于向标准输出写信息
Dim lBytesWritten As Long
s = s & vbCrLf
WriteFile hStdOut, s, Len(s), lBytesWritten, ByVal 0&
```

```
End Sub

Public Function GetCgiValue(cgiName As String) As String
'-----------------------------------------------
' 本子程序可以获取表单上某一元素的数据
'-----------------------------------------------
Dim delim2   As Long          ' position of "="
Dim delim1   As Long          ' position of "&"
Dim n As Integer
Dim pointer1 As Long
Dim pointer2 As Long
Dim length As Long
Dim length1 As Long
Dim tmpstring1 As String
Dim tmpstring2 As String
pointer1 = 1
pointer2 = 1
delim2 = InStr(pointer2, sFormData, "=")
pointer2 = delim2 + 1
Do
  length = delim2 - pointer1
  tmpstring1 = Mid(sFormData, pointer1, length)
  delim1 = InStr(pointer1, sFormData, "&")
  pointer1 = delim1 + 1
  length1 = delim1 - pointer2
  If delim1 = 0 Then length1 = lContentLength + 1 - pointer2
  If tmpstring1 = cgiName Then
    tmpstring2 = Mid$(sFormData, pointer2, length1)
    GetCgiValue = UrlDecode(tmpstring2)
    Exit Do
  End If
  If delim1 = 0 Then
    Exit Do
  End If
 delim2 = InStr(pointer2, sFormData, "=")
 pointer2 = delim2 + 1
 Loop
End Function
```

```
Public Function UrlDecode(ByVal sEncoded As String) As String
'-----------------------------------------------------
' 本函数可以对用户输入的数据进行 URI 解码
'-----------------------------------------------------
Dim pointer   As Long        ' sEncoded position pointer
Dim pos   As Long        ' position of InStr target
Dim temp As String
If sEncoded = "" Then Exit Function
 '本循环体用于将“+”转换成空格
pointer = 1
Do
    pos = InStr(pointer, sEncoded, "+")
    If pos = 0 Then Exit Do
    Mid$(sEncoded, pos, 1) = " "
    pointer = pos + 1
Loop
pointer = 1
On Error GoTo errorUrlDecode
Do   '本循环体用于将%XX 转换成字符。对于两个连续的%XX，如果第一个%XX 不是某些特指的
Web 系统保留字符，将把它们转换成汉字
    pos = InStr(pointer, sEncoded, "%")
    If pos = 0 Then Exit Do
    temp = Chr$("&H" & (Mid$(sEncoded, pos + 1, 2)))
    If Mid(sEncoded, pos + 3, 1) = "%" And (temp <> ":") And (temp <> "/")
        And (temp <> "(") And (temp <> ")") And (temp <> ".") And (temp <> ",")
        And (temp <> ";") And (temp <> "%") Then
        Mid$(sEncoded, pos, 2) = Chr$("&H" & (Mid$(sEncoded, pos + 1, 2)) _
        & (Mid$(sEncoded, pos + 4, 2)))
        sEncoded = Left$(sEncoded, pos)   &   Mid$(sEncoded, pos + 6)
        pointer = pos + 1
    Else
       Mid$(sEncoded, pos, 1) = temp
          sEncoded = Left$(sEncoded, pos)   & Mid$(sEncoded, pos + 3)
        pointer = pos + 1
        End If
    Loop
    On Error GoTo 0        'reset error handling
    UrlDecode = sEncoded
    Exit Function
```

```
errorUrlDecode:
If Err.Number = 13 Then          'Type Mismatch error
    Err.Clear
    Err.Raise 65001, , "Invalid data passed to UrlDecode() function."
Else
    Err.Raise Err.Number
End If
Resume Next
End Function

Public Function TempFile(sPath As String, sPrefix As String) As String
'-----------------------------------------------
'本函数可以获得一个唯一的临时文件名
'-----------------------------------------------
Dim x    As Long
Dim rc As Long
TempFile = String(127, Chr$(0))
rc = GetTempFileName(sPath, sPrefix, ByVal 0&, TempFile)
x = InStr(TempFile, Chr$(0))
If x > 0 Then TempFile = Left$(TempFile, x - 1)
End Function

Public Function IsGoodMail(email As String) As Integer
'-----------------------------------------------
'本函数用于检查 E-mail 地址的合法性
'-----------------------------------------------
If InStr(email, "@") = 0 Or InStr(email, ".") = 0 _
   Or Left$(email, 1) = "@" Or Left$(email, 1) = "." _
   Or Right$(email, 1) = "@" Or Right$(email, 1) = "." Then
        IsGoodMail = -1
        Exit Function
End If
IsGoodMail = 0
End Function
```

第八章　客户端用户状态信息的保存

在第一章我们就已经了解到，CGI 程序是工作在无状态的环境中的。Web 服务器每次激活一个 CGI 程序时都为它准备一个全新的运行环境。Web 服务器不为 CGI 程序保存其上一次或前几次的运行环境（这里所指的运行环境均是针对同一个客户端用户）。而在某些情况下 CGI 程序必须了解它以前的运行环境，比如客户端用户通过 CGI 程序执行一个数据库查询，数据库服务器一共返回了 1000 条记录，但每次 Web 页面只显示 20 条记录。这时 CGI 程序必须为客户端用户保存前几次的状态信息。否则 Web 页面每显示一次查询结果，都要对数据库执行一次查询，这将大大加重 Web 服务器和数据库的负担。

由于 HTTP 协议的无状态特性，CGI 程序在为客户端用户保存状态信息时通常要将这些信息存在客户端的主机或 Web 服务器上，并在 Web 页面上设置一些标志以存取这些信息。一般来说，保存客户端用户状态信息的方法有以下几种：

- 利用环境变量 PATH_INFO 和 QUERY_STRING；
- 利用表单的隐含文本框；
- 利用 HTTP Cookie。

8.1　利用环境变量 PATH_INFO 和 QUERY_STRING

为客户端用户保存状态信息的一个方法就是使用环境变量 PATH_INFO 和 QUERY_STRING。我们在 HTML 文件中一般使用如下的 URL 来请求相应的 CGI 程序：

```
http: // www.youhost.com/cgi-bin/yourcgi.exe
```

事实上在这个 URL 后面还可以附加一些可以传递给 Web 服务器的信息。例如，以下的 URL：

```
http: // www.youhost.com/cgi-bin/yourcgi.exe/filename?write
```

不仅向 Web 服务器请求一个 CGI 程序的执行，而且还将字符串“filename”和“write”传递给服务器。服务器将把“filename”放在环境变量 PATH_INFO 中，而把“write”放在环境变量 QUERY_STRING 中。通常 Web 服务器将 URL 字符串中“?”后的信息放入环境变量 QUERY_STRING 中，“/”和“?”之间的信息放入环境变量 PATH_INFO 中。

CGI 程序可以将客户端用户的一些状态信息附加在 URL 的后面传给客户端。例如：

```
printf( "<a href=\"Vcgi-binVyourcgi.exeV%s?%s \" >Next page<Va>\n " , state1, state2);
```

当客户端用户第二次请求 CGI 程序时，CGI 程序就将这些信息从环境变量中取出以作相应的处理。

不过由于 URL 字符串的长度不能超过 1024 个字节，当所存储的状态信息超过了 1024 字节时就有可能造成信息的丢失。这时可以将客户端用户的状态信息保存在服务器上的一个临时文件中，而在 URL 字符串中只保存该临时文件的路径和名称。

当不希望在 Web 页面上出现 SUBMIT 按钮而又想保存客户端用户的状态信息时，URL 字符串是除 Cookie 外的唯一选择。

8.2　利用表单的隐含文本框

表单的隐含文本框同样也可以用来为客户端用户保存状态信息。并且，它还没有使用URL 时字节数的限制。CGI 程序可以将客户端用户的状态信息保存在隐含文本框的 VALUE 属性中：

```
printf("<INPUT TYPE=hidden NAME=state VALUE=\"%s\">", stateinformation);
```

虽然隐含文本框不在 Web 页面上显示出来，但客户端用户可以用浏览器的“View Source”命令看到其内容，隐含文本框和 URL 一样不具有安全性。

目前大多数浏览器都支持隐含文本框，可能会有一些版本较低的浏览器不支持隐含文本框，这时 CGI 程序生成的 Web 页面会非常难看。在 CGI 程序中可以将隐含文本框的 SIZE 属性设置为“0”，以保证所有的浏览器都不在 Web 页面上显示隐含文本框。

8.3　使用隐含文本框的 CGI 程序实例

本节将提供一个利用隐含文本框保存客户端状态信息的 CGI 程序实例。

在 7.2 节中，本书曾经提供了一个主页意见箱。本节将对该意见箱做一些改进。当客户端用户在主页意见箱中输入了信息并按下了 SUBMIT 按钮以后，CGI 程序并不是立刻将用户的建议发送给 Web 服务器管理员，而是将所有的用户输入信息存入 Web 服务器上的一个临时文件中并在客户端的 Web 页面上显示出来以希望用户确认一下。只有在用户确认以后临时文件中的内容才会被 CGI 程序发送给 Web 服务器管理员。本 CGI 程序将利用隐含文本框保存临时文件的名称，并利用 SUBMIT 按钮不同的 VALUE 值来判断是哪一个 Web 页面调用了 CGI 程序。CGI 程序所要处理的表单和 7.2 节的表单相同。该 CGI 程序的源代码如下所示。

```
/* newcomment.c */
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#define MAX_ENTRIES 10000
typedef struct
 {
     char *name;
     char *val;
 } entry;
int   num_value=0;
entry entries[MAX_ENTRIES];
char *makeword(char *line, char stop);
char *fmakeword(FILE *f, char stop, int *len);
char x2c(char *what);
void getword(char *word, char *line, char stop);
```

```
void unescape_url(char *url);
void plustospace(char *str);
void getcomment(char *tn1 );
void verify(char *tn1 );
char *getcgivalue(char *s);
char *host,*submittime,*navigator,*systems;
char *subject,*email,*comment,*URL;
main( )
{
    FILE *suggest;
    int x, contentlength;
    char *querystring,*tn1;
    char command[200];
char *actiontype, *filename;
/*以下代码用于 CGI 程序的初始化工作报告*/
    if(strcmp(getenv("REQUEST_METHOD"),"POST")==0)
     {
      if(strcmp(getenv("CONTENT_TYPE"),"application/x-www-form-urlencoded"))
        {
        printf("Content-type: text/html%c%c",10,10);
        printf("This script can only be used to decode form results. \n");
        exit(1);
        }
        contentlength = atoi(getenv("CONTENT_LENGTH"));
        for(x=0;contentlength && (!feof(stdin));x++)
        {
         num_value=x;
         entries[x].val = fmakeword(stdin,'&',&contentlength);
         plustospace(entries[x].val);
         unescape_url(entries[x].val);
         entries[x].name = makeword(entries[x].val,'=');
        }
      }
    else if(strcmp(getenv("REQUEST_METHOD"),"GET")==0)
      {
       querystring = getenv("QUERY_STRING");
       if(querystring == NULL)
        {
          printf("Content-type: text/html%c%c",10,10);
```

```
    printf("No query information to decode.\n");
    exit(1);
  }
  for(x=0;querystring[0] != '\0';x++)
  {
  num_value=x;
  getword(entries[x].val,querystring,'&');
  plustospace(entries[x].val);
        unescape_url(entries[x].val);
        getword(entries[x].name,entries[x].val,'=');
       }
    }
   else{
        printf("Content-type: text/html%c%c",10,10);
        printf("The form was submitted with an unknown method!\n");
    }
/*获取 SUBMIT 按钮的 VALUE 值以判断 CGI 程序被哪一个 Web 页面调用*/
 actiontype=getcgivalue("Submit Comments");
 if(strcmp(actiontype," 提　交 ")==0)        /*如果是初次调用*/
    {
      tn1 = tmpnam(NULL);            /*获取一个唯一的临时文件名*/
      getcomment(tn1);            /*将用户的建议存入临时文件*/
      verify(tn1 );                  /*向客户端显示用户的输入，并希望用户确认*/
    }
  else
  {   /*如果是确认 Web 页面调用了 CGI 程序，CGI 程序将从隐含文本框中获取临时文件名，将临时文件的内容发送给 Web 服务器管理员   */
    filename =getcgivalue("fname");        /*获取临时文件名*/
    printf("Content-type: text/html%c%c",10,10);
    printf("<FONT SIZE=\"+2\">");
    printf("<P>请稍候，系统正在传递您的信息.....");
    sprintf(command,"/usr/sbin/Mail -s \"关于主页的建议\" zhangys@hotmail.com\n", filename);
   if(system(command)==-1) /*向服务器管理员发送用户建议*/
    {
      printf("<P>非常抱歉，系统无法无法传递您的意见!\n");
      printf("<P>请您等候片刻以后再试或用别的方式将意见转给我们!\n");
    }
  unlink(filename);   /*删除临时文件*/
  printf("<P>非常感谢您对本主页的宝贵意见");
```

```
    printf("<P>欢迎您经常光顾本主页！ ");
    printf("<\/FONT>");
    }
    return 0;
}
/*以下为本程序所使用的 NCSA 的 util.c 所提供的部分函数和本书所补充的函数*/
void getword(char *word, char *line, char stop)
    {
      int x = 0,y;
      for(x=0;((line[x]) && (line[x] != stop));x++)
            word[x] = line[x];
      word[x] = '\0';
      if(line[x]) ++x;
      y=0;
      while(line[y++] = line[x++]);
}
char *makeword(char *line, char stop)
    {
      int x = 0,y;
      char *word = (char *) malloc(sizeof(char) * (strlen(line) + 1));
      for(x=0;((line[x]) && (line[x] != stop));x++)
            word[x] = line[x];
      word[x] = '\0';
      if(line[x]) ++x;
      y=0;
      while(line[y++] = line[x++]);
      return word;
    }
char *fmakeword(FILE *f, char stop, int *cl)
    {
      int wsize;
      char *word;
      int ll;
      wsize = 102400;
      ll=0;
      word = (char *) malloc(sizeof(char) * (wsize + 1));
      while(1)
        {
          word[ll] = (char)fgetc(f);
```

```
if(ll==wsize)
  {
    word[ll+1] = '\0';
    wsize+=102400;
    word = (char *)realloc(word,sizeof(char)*(wsize+1));
  }
--(*cl);
if((word[ll] == stop) || (feof(f)) || (!(*cl)))
        {
          if(word[ll] != stop) ll++;
          word[ll] = '\0';
          return word;
        }
        ++ll;
    }
}
char x2c(char *what)
  {
    register char digit;
    digit = (what[0] >= 'A' ? ((what[0] & 0xdf) - 'A')+10 : (what[0] - '0'));
    digit *= 16;
    digit += (what[1] >= 'A' ? ((what[1] & 0xdf) - 'A')+10 : (what[1] - '0'));
    return(digit);
  }
void unescape_url(char *url)
  {
    register int x,y;
    for(x=0,y=0;url[y];++x,++y)
      {
        if((url[x] = url[y]) == '%')
          {
            url[x] = x2c(&url[y+1]);
            y+=2;
            }
      }
    url[x] = '\0';
  }
void plustospace(char *str)
  {
```

```
    register int x;
    for(x=0;str[x];x++) if(str[x] == '+') str[x] = ' ';
  }
/*以下的函数用于获取表单上某一元素所储存的数据，当表单元素不含数据时返回 NULL 指针*/
char *getcgivalue(char *s)
 {
  int x,length;
  int i=0;
  char *buf;
  for (x = 0; x <= num_value; x++)
   {
    if (strcmp(entries[x].name,s) == 0)
      {
       if((length=strlen(entries[x].val))==0)     return NULL;
       if(( buf=(char *)malloc(length))==NULL)
        {
           printf("Not enough memory to allocate buffer \n");
           exit(0);
        }
       while((entries[x].val[i])!='\0')
          {
           buf[i]=entries[x].val[i];
           i++ ;
           }
           buf[i]='\0';
      }
   }
     return buf;
}
/*以下函数用于获取客户端用户的一些状态信息和填写的建议并保存在临时文件中*/
void getcomment(char *tn1 )
{
  FILE *suggest;
  struct tm   today;
  time_t   now;
  time(&now);   /*获取用户填写建议的时间*/
  today = *localtime(&now);
  mktime(&today);
  /*以下代码用于获取用户的一些状态信息和填写的建议*/
```

```
navigator=getenv("HTTP_USER_AGENT");
if((host=getenv("REMOTE_HOST"))==NULL) host=getenv("REMOTE_ADDR");
submittime=getenv("DATE_LOCAL");
systems=getcgivalue("SYSTEM");
email=getcgivalue("email");
subject=getcgivalue("URL_Subject");
URL=getcgivalue("URL");
comment=getcgivalue("URL_Comment");
if(comment==NULL)
  {
    printf("Content-type: text/html%c%c",10,10);
    printf("<P>非常抱歉，您还没有提出宝贵的意见!\n");
    exit(0);
  }
if((suggest=fopen(tn1,"w+t"))==NULL)
 {
  printf("Content-type: text/html%c%c",10,10);
  printf("<P>非常抱歉，系统无法打开暂时文件!\n");
  printf("<P>请您等候片刻以后再试或用别的方式将意见转给我们!\n");
  exit(1);
  }
fprintf(suggest,"客户端主机为: %s\n",host);
fprintf(suggest,"发送时间为:  %d 年%d 月%d 日%d 时%d 分\n",today.tm_year+1900,
      today.tm_mon+1,today.tm_mday,today.tm_hour,today.tm_min);
fprintf(suggest,"客户端所用浏览器为: %s\n",navigator);
if((strcmp(systems,"浏览器所使用的系统平台:"))!=0)
 fprintf(suggest,"浏览器所使用的系统平台为: %s\n",systems);
if(email!=NULL) fprintf(suggest,"客户端用户电子函件信箱为: %s\n",email);
if(subject!=NULL) fprintf(suggest,"建议的主题是: %s\n",  subject);
if(URL!=NULL) fprintf(suggest,"建议所指的主页是: %s\n", URL);
fprintf(suggest,"客户端所提的建议为: \n\n");
fprintf(suggest," %s\n\n",  comment);
   fclose(suggest);
}
/*以下函数用于向客户端用户输出一个确认 Web 页面*/
void verify(char *tn1 )
{
   fprintf(stdout,"Content-type: text/html%c%c",10,10);
   fprintf(stdout,"<HTML><HEAD><TITLE>意见箱</TITLE></HEAD>");
```

```
fprintf(stdout,"<BODY BGCOLOR=\"#FFFFFF\">");
fprintf(stdout,"<P>您输入了以下信息: \n");
fprintf(stdout,"<ul>\n");
if((strcmp(systems,"浏览器所使用的系统平台:"))!=0)
 fprintf(stdout,"<li><B>系统平台为: <\/B>%s\n",systems);
if(email!=NULL) fprintf(stdout,"<li><B>电子函件信箱为: <\/B>%s\n",email);
if(subject!=NULL) fprintf(stdout,"<li><B>建议的主题是: <\/B>%s\n",  subject);
if(URL!=NULL) fprintf(stdout,"<li><B>建议所指的主页是: <\/B>%s\n", URL);
fprintf(stdout,"<li><B>建议为: <\/B>\n");
fprintf(stdout,"%s\n",  comment);
fprintf(stdout, "<\/ul>\n");
fprintf(stdout,"<P><P>如果您按下面的上交按钮, 这些信息和您的主机的 IP 地址和上交时 间都
将被\n");
fprintf(stdout,"<BR>传递给 Web 服务器管理员! \n");
fprintf(stdout,"<P>如果您觉得以上建议需要修改,可以使用浏览器的 BACK 按钮退回到以前的\n");
fprintf(stdout,"<BR>Web 页面继续修改! \n");
fprintf(stdout,"<P><FORM  ACTION=\"\/cgi-bin\/comment.exe\" METHOD=POST>\n");
 /*用隐含文本框保存临时文件名*/
fprintf(stdout,"<INPUT TYPE=\"HIDDEN\" NAME=\"fname\" VALUE=\"%s\">\n",tn1);
fprintf(stdout,"<INPUT TYPE=\"SUBMIT\" NAME=\"Submit Comments\" VALUE=\"  上交 \">");
fprintf(stdout,"<\/FORM>\n");
fprintf(stdout,"<\/BODY><\/HTML>");
}
```

8.4　HTTP Cookie

环境变量 PATH_INFO、QUERY_STRING 和表单的隐含文本框保存客户端用户的状态信息的功能是非常有限的，至少当浏览器关闭以后这些被保存的状态信息就会消失。另一个用于保存客户端用户状态信息的方法 HTTP Cookie 则比较实用和灵活。HTTP Cookie 可以将客户端用户的状态信息储存在客户端主机上，同时 CGI 程序可以定义状态信息的有效时间，哪些主机和 Web 页面可以读取这些状态信息。HTTP Cookie 的使用将使您的 Web 服务器具有更强大的交互式能力。

8.4.1　什么是 HTTP Cookie

当 Web 服务器向客户端用户传递被请求的 HTTP 资源时，也可能把有关这一次 HTTP 连接的某些状态信息传送给客户端用户。客户端用户的浏览器会自动保存这些状态信息。这些状态信息还定义了自身存在的有效期，哪些域中的主机和 Web 页面可以使用它们。当以后客户端用户调用了这些状态信息中定义的 HTTP 资源时，浏览器就会自动将这些信息返回给 Web 服务器。通常我们把这些存储在客户端主机的硬盘和内存中的状态信息称为 Cookie。

如果您想让客户端用户能够根据自己的需要定义 Web 页面的外观，或者您想使 Web 页面依据具体的情况而变化，或者想使 Web 和数据库的连接更有效，Cookie 可以说是目前唯一的选择。

Cookie 仅仅是一些文本信息，所有的 Cookie 都将被浏览器保存在一个文本文件中。因此 Cookie 不会以病毒的形式对计算机造成危害。Cookie 向 Web 服务器提供的都是客户端用户本身所愿意提供的东西，也不会泄露客户端用户的任何个人隐私。事实上通过更改设置，您可以让浏览器在保存一个 Cookie 时发出一个警告信息或拒绝接受任何 Cookie。

8.4.2　HTTP Cookie 工作原理

Web 服务器和浏览器之间通过 HTTP 头互相传送 Cookie。当 Web 服务器向浏览器发送一个 Cookie 时，Web 服务器产生的响应头（Response Header）将包括如下一行：

Set-Cookie: NAME=*VALUE*; expires=*DATE*; path=*PATH*; domain=*DOMAIN_NAME*; secure

如果客户端的浏览器支持 Cookie，响应头中所定义的 Cookie 将被浏览器接受。浏览器则通过请求头（Request Header）以“NAME=VALUE”的形式将所有符合要求的 Cookie 回传给 Web 服务器，此时浏览器发出的请求头将包括如下的一行：

Cookie: NAME1=OPAQUE_STRING1; NAME2=OPAQUE_STRING2

Web 服务器将根据浏览器的 Cookie 请求头设置相关的环境变量。例如可以通过以下的响应头设置了一个 HTTP Cookie “filename”：

Set-Cookie: filename=zys.doc; path=/zhangys; expires=Wednesday, 09-Nov-99 23:12:40 GMT

当客户端用户请求的 WWW 资源位于路径“/zhangys”下时，浏览器的请求头将包含下列信息：

Cookie: filename=zys.doc;

一般，Set-Cookie 响应头各部分的含义如下：

- NAME=*VALUE*

这是 Set-Cookie 响应头唯一一个必须拥有的部分。NAME 是您所要设置的 Cookie 的名称，VALUE 是该 Cookie 的值。在字符串“NAME=*VALUE*”中，不能出现“；”、“，”和空格等字符。如果必须使用它们，可以用这些特殊字符的十六进制代码“%XX”代替它们。

- expires=*DATE*

Set-Cookie 响应头的 expires 属性可以设置 Cookie 的生存期，当 expires 属性所定义的时间到期以后，浏览器将不再保存和传送相应的 Cookie。设置 expires 的 DATE 应具有以下的形式：

- Wdy, DD-Mon-YYYY HH:MM:SS GMT

字符串中的 GMT 表示这是一个格林尼治时间。

这一项有关 Cookie 的设置是可选的。如果没有设置，当浏览器和 Web 服务器的一次连接结束后，Cookie 的生命也就结束了。

- domain=DOMAIN_NAME

Set-Cookie 响应头的 domain 属性定义了哪些 Internet 域中的 Web 服务器可以读取浏览器所存取的 Cookie。浏览器在向 Web 服务器发送请求头时，将首先检查用户输入的 URL

中的主机是否在 Set-Cookie 响应头所定义的 Internet 域中。如果不在，浏览器就不会传送相应的 Cookie，否则继续进行相关项的检查。

一台 Internet 域中的 Web 服务器只能为本域和本域中的子域设置 Cookie。这一项设置是可选的。缺省时 domain 属性的值就是该 Web 服务器的域名。

- path=*PATH*

Set-Cookie 响应头的 path 属性定义了一台 Web 服务器上哪些路径下的 WWW 资源可以获取服务器所设置的 Cookie。当用户输入的 URL 通过了域名检查后，浏览器将继续检查 URL 中的路径部分，只有在通过了路径检查后，相应的 Cookie 才会被传给 Web 服务器。一般如果用户输入的 URL 中的路径部分从第一个字符开始包含 path 属性所定义的字符串，浏览器就认为通过了检查。比如若 path 属性的值为"/foo"，则路径"/foobar"和"/foo/bar.html"都将满足要求。如果 path 属性的值为"/"，Web 服务器上所有的 WWW 资源都将可以读取所设置的 Cookie。

这一项的设置也是可选的。缺省时 path 属性的值为 Web 服务器传给浏览器的资源的路径名。

- secure

如果一个 Cookie 被 Set-Cookie 响应头标记为"secure"，只有当浏览器和 Web 服务器之间的通信协议为加密认证协议时，浏览器才向服务器提交相应的 Cookie。当前这种协议只有一种，即 HTTPS。

8.4.3 HTTP Cookie 使用时的一些限制

在使用 Cookie 时，必须注意 HTTP Cookie 的一些限制。

一般在任何时候客户端的浏览器所存储的 Cookie 的总数不能超过 300 个，每个 Cookie 的字节数（包括 Set-Cookie 响应头的所有字符）不能超过 4K。每个 Web 服务器或 Internet 域所设置的 Cookie 总数不能超过 20 个（这时服务器和 Internet 域是单独计数的，即使 Web 服务器处于 Internet 域中）。

当 Cookie 的数目超过了 300 或 20 的限制时，最近使用次数最少的 Cookie 将被丢弃。同样，当 Cookie 的字节数超过了 4K 的限制时，超过 4K 部分的信息也将被丢弃。

8.4.4 用 CGI 程序设置和读取 HTTP Cookie

用 CGI 程序设置 HTTP Cookie 非常简单。由于 Web 服务器是通过响应头来设置 Cookie 的，可以让 CGI 程序通过向 STDOUT 输出 Set-Cookie 响应头来设置 Cookie。例如，以下的 CGI 程序在客户端上设置了两个名为"Myname"和"Myage"的 Cookie。

```
#include <stdio.h>
main( )
{
   printf("Content-type: text/html\n");
   printf("Set-cookie: Myname=Zhang%20Yishan;
            expires=Thursday, 01-Jan-98 12:00:00 GMT\n");
   printf("Set-cookie: Myage=30;
```

```
        expires=Thursday, 01-Jan-98 12:00:00 GMT\n\n");
/*和其他 CGI 程序一样，在程序输出的响应头和 Web 页面之间必须有一空行将两者分开*/
  printf("<H1>两个 COOKIE 已经被设置</H1>");
  return;
}
```

当客户端的用户关闭浏览器而 Cookie 的生命还没有结束时，浏览器在退出之前将把内存中的 Cookie 保存在一个文本文件中，对于 Windows 95 或 Windows 3.1 系统下的 Netscape Navigator，这个文件的名称是 cookies.txt，它一般位于浏览器所在的目录。例如当运行了本节设置 Cookie 的 CGI 程序时，本人所使用的 Netscape Navigator 的 cookies.txt 的内容如下：

```
# Netscape HTTP Cookie File
# http://www.netscape.com/newsref/std/cookie_spec.html
# This is a generated file!  Do not edit.
166.111.129.13   FALSE /cgi-bin  FALSE    83656000  Myname    ZhangYishan
166.111.129.13   FALSE /cgi-bin  FALSE    883656000 Myage     30
Webcrawler.com FALSE /   FALSE   1030185237 AnonTrack  9710901B3400EF15
netscape.com   TRUE /   FALSE   946684799   NETSCAPE_ID  c65ffb1e,c40e94ae
yahoo.com   TRUE   /   FALSE 915145200   Y   v=1&n=b33tm5nc0quci
```

文件 cookies.txt 的每一行表示了某台 Web 服务器所设置的 Cookie。其中每一行的每一部分从左到右分别表示如下的含义：

- 可以读取 Cookie 的 Web 服务器或 Internet 域；
- 是否一个 Internet 域中的所有主机都可以读取 Cookie（TRUE 或 FALSE）；
- 可以读取 Cookie 的 Web 服务器或 Internet 域上的路径名；
- 是否必须使用安全协议传送 Cookie（TRUE 或 FALSE）；
- Cookie 的生命截止日期，它具有 UNIX 系统时间的形式；
- Cookie 的名称；
- Cookie 的值。

当用户运行 Netscape 时，浏览器将把文件中的内容加载到内存。因此客户端的用户可以通过删除文件 cookies.txt 或修改其内容来删除或更改 Cookie。当然也可以通过 CGI 程序删除或更改 Cookie。可以设置一个具有相同名称和相同路径的 Cookie，如要删除以前设置的 Cookie，这时只要将 Cookie 的 expires 属性设置成过去的某一时间就行；如要修改 Cookie，只需修改 Cookie 的 Value 值就行了，但程序必须记住以前所设置的日期期限。

如果客户端浏览器发送来的请求头包含 Cookie，Web 服务器在为被激活的 CGI 程序准备运行环境时将把所有的 Cookie 放在环境变量 HTTP_COOKIE 中。以下的 CGI 程序程序可以显示环境变量“HTTP_COOKIE”所存储的 Cookie。

```
#include <stdio.h>
#include <stdlib.h>
main( )
{
```

```
    char *cookie;
    cookie=getenv("HTTP_COOKIE");
    printf("Content-type: text/html\n\n");
    printf("<H2>所设置的 Cookie 是: %s</H2>\n",cookie);
    return;
}
```

当执行了本节设置 Cookie 的 CGI 程序后，再运行以上显示 Cookie 的 CGI 程序，可以发现环境变量 HTTP_COOKIE 的值为“Myname=ZhangYishan; Myage=30”，不同的 Cookie 被“;”分开。要想使用 Cookie 还必须编程将各个 Cookie 的名称和值从环境变量中分别取出，这是一件很容易实现的事情。

8.5　使用 HTTP Cookie 的计数器

7.1.5 节的计数器是对所有访问者的访问计数，而本节的计数器将只对某一个访问者单独计数。

当 CGI 程序被激活时，首先检查环境变量 HTTP_COOKIE 的值，如果为空，或者不为空但却没有相应的 Cookie，CGI 程序将为客户端的浏览器设置一个用来记载用户访问次数的 Cookie。如果相应的 Cookie 已经被设置了，CGI 程序将该 Cookie 的值读出加一，并用新值重新设置这个 Cookie。最后 CGI 程序将根据所获得的 Cookie 值生成一幅数字 GIF 图像。

客户端用户对 Cookie 的改动可能会影响 CGI 程序的正确执行。不过由于 Cookie 特有的性质，这个影响也只局限于改动 Cookie 的某一客户端用户的 Web 页面。

该 CGI 程序的源代码如下所示：

```
/*cookiecount.c*/
#include <gd.h>
#include <stdio.h>
#include <stdlib.h>
#include <fcntl.h>
#include <time.h>
gdImagePtr creatimage(char *count);
char *wday[ ]={"Sunday","Monday","Tuesday","Wednesday","Thursday",
                "Friday","Saturday","Unknow"};
char *mon[ ]={"Jan","Feb","Mar","Apr","May","Jun","Jul","Aug","Sep","Oct","Nov","Dec"};
char *gif[ ]={"0.gif", "1.gif", "2.gif", "3.gif", "4.gif", "5.gif", "6.gif", "7.gif", "8.gif", "9.gif"};
int main( )
{
   struct tm   today;
   time_t   now;
   char * month, *weekday,tmpstring[100];
```

```
char *cookie, *expires,cookiecount[100]="1";
char *ptr1, *ptr2;
int ghour, gmin, gsec, gyear, gday, countdata;
int x, result;
long cookietime;
gdImagePtr imagecount; /*声明一个 GIF 图像 */
/*setmode 函数将标准输出设置为二进制模式，在 UNIX 系统下不必使用该函数*/
setmode(fileno(stdout),O_BINARY);
/*获取当前的格林尼治时间*/
cookietime=time(&now);
today = *gmtime(&cookietime);
month=mon[today.tm_mon];
weekday=wday[today.tm_wday];
gday=today.tm_mday;
gyear=today.tm_year+1900+10;
gsec=today.tm_sec;
gmin=today.tm_min;
ghour=today.tm_hour;
/*生成一个符合 Set-cookie 响应头的截止时间字符串*/
sprintf(expires, "%s, %2d-%s-%4d %d:%d:%d GMT", weekday,gday,month,gyear,ghour,gmin,gsec);
cookie = getenv("HTTP_COOKIE"); /*获取浏览器传送来的 Cookie*/
if(cookie == NULL)
{
 /*若浏览器没有传来 Cookie,则设置一个 Cookie 用来记载用户的访问次数*/
 countdata=2;
 printf("Content-type: image/gif\n");
 printf("Set-cookie: Hits=%d; expires=%s\n\n",countdata,expires);
 }
else {
 /*读取相应的 Cookie,如果没有，则设置一个 Cookie 用来记载用户的访问次数；如果有，将读出
的 Cookie 值加一，按新值重新设置 Cookie*/
    ptr1=strstr(cookie,"Hits");
  if(ptr1==NULL)
    {
    countdata=2;
    printf("Content-type: image/gif\n"); /*说明 CGI 程序的 MIME 类型*/
    printf("Set-cookie: Hits=%d; expires=%s\n\n",countdata,expires);
    }
  else{
```

```
/*读取相应的 Cookie 值*/
 ptr2=strstr(ptr1,"=");
 for(x=0;ptr2[x+1] != '\0'&&ptr2[x+1] != ';';x++)
        cookiecount[x]=ptr2[x+1];
cookiecount[x]='\0';
countdata=atoi(cookiecount)+1;
printf("Content-type: image/gif\n");
        printf("Set-cookie: Hits=%d; expires=%s\n\n",countdata,expires);
        }
    }
  imagecount=creatimage(cookiecount);/*根据读取的数据生成一幅图像*/

  gdImageGif(imagecount, stdout); /*向 STDOUT 输出图像*/
  gdImageDestroy(imagecount); /*释放 GIF 图像所占内存空间*/
  exit(0);

}
/*函数可以根据数字字符串生成 GIF 图像*/
gdImagePtr creatimage(char *count)
{
  gdImagePtr im,im_in;
  FILE *in;
  char sn[2];
  int black, number, len,i,ntemp;
  int imagewidth,width,height;
  /*调入一个数字 GIF 图像以获取图像的尺寸大小*/
    in = fopen(gif[0], "rb");
    im_in = gdImageCreateFromGif(in);
    fclose(in);
    width= gdImageSX(im_in);
    height=gdImageSY(im_in);
    gdImageDestroy(im_in); /*释放 GIF 图像所占内存空间*/
   len=strlen(count);
  /*确定访问人数的位数*/
   imagewidth=len*width;
  /*设置图像的尺寸大小*/
   im = gdImageCreate(imagewidth, height);
  /*设置图像的背景颜色为黑*/
   black = gdImageColorAllocate(im, 0, 0, 0);
```

```
/*以下的 for 循环根据访问人数每一位数字的不同分别把不同的 GIF 数字图像拷贝到 GIF 图像 im 上
*/
    for(i=0;i<len;i++)
    {
      sn[0]=count[i];
      sn[1]='\0';
      ntemp=atoi(sn);
      in = fopen(gif[ntemp], "rb");
      im_in = gdImageCreateFromGif(in);
      fclose(in);
    gdImageCopy(im, im_in, i*width, 0, 0, 0, width, height);
    gdImageDestroy(im_in);
    }
     return im;
  }
```

本程序在 Windows 95 和 UNIX 平台下调试通过，在 UNIX 系统下编译本程序时必须将程序中的语句“#include <fcntl.h> ”和“ setmode(fileno(stdout),O_BINARY)”删除。

由于使用了 GD 库，本程序的编译和普通程序有所不同，具体的编译方法可以参考 7.1.5 节。CGI 程序动态生成 GIF 图像的原理也可以参考 7.1.5 节。

第九章　动态网页的制作技术

利用目前的 HTML 语言，您可以将文本、声音、图像、视频、超文本链接和其他对象放在一起形成一个丰富多彩的 Web 网页。一个制作好的 Web 页面将被放在 Web 服务器上，当接受到客户端用户的请求时这个 Web 网页就会被 Web 服务器传给客户端的用户。如果作者不去修改，这样的 Web 网页的内容是永远不会改变的。我们把这样的网页称为静态网页。

通常在实践中还需要制作另一种类型的网页，比如网页应该能够对客户端用户的输入或产生的事件作出反应，网页可以不用作者的干预而自动反应 Web 服务器每天、每月被访问的状况，网页可以实时地向用户通报股市行情，网页可以自动对不同类型的客户端用户做出不同的反应。和静态网页相对应，这类网页一般被称为动态网页。

在这一章，本书将向读者介绍一些动态网页的制作技术。随着 Internet 和 WWW 的发展，制作动态网页的技术也越来越多，比如现在读者可以选择 Java、VBscript、Javascript、ActiveX、ASP 和 CGI 等技术制作动态网页，一些商用 Web 服务器还提供了专门的开发工具。事实上从广义上讲本书前面各章节的 CGI 程序所生成的 Web 网页都可以被称为动态网页，因为它们都是根据用户的输入由 CGI 程序实时产生的，在 Web 服务器上您将找不到这些网页的 HTML 文件。本章这里所讨论的动态网页是指存在于 Web 服务器上的 HTML 文件，由于采用了一些特殊的技术而成为了动态网页。考虑到本书的内容是 CGI 编程，本章在这里将主要介绍和 CGI 编程有关的动态网页制作技术：

- 普通 CGI 程序；
- Client Pull；
- Server Push；
- 外部程序。

9.1　普通 CGI 程序

一个具有图形计数器的 Web 网页是一个典型的动态网页，因为 Web 页面上表示数字的 GIF 图像每次存取时都不一样。本书 4.4 节的 CGI 程序 chuel.c 可以根据浏览器的不同类型而向浏览器提供不同的 URL，拥有指向该 CGI 程序的超级链接的 Web 网页也是一个动态网页。

利用 CGI 程序制作动态网页，主要是要利用 CGI 程序设置响应头的功能。churl.c 就是通过设置 Location 响应头来向客户端提供不同的 URL。以下的 CGI 程序同样也利用了 Location 响应头，它将向客户端用户随机的提供一个 URL。

```
/* randurl.c */

#include <stdio.h>

#include <stdlib.h>

#include <time.h>
```

```
main( )
 {
  char *where[ ]={"http://www.edu.cn", "http://www.tsinghua.edu.cn",
                  "http://www.pim.tsinghua.edu.cn", "http://www.sgi.com",
                   "http://www.microsoft.com"};
  int i;
  time_t now;
  srand((unsigned) time(&now));  /*初始化随机数发生器*/
  i=rand( ) % 5;          /*产生一个小于 5 的随机非负整数*/
    /*根据随机数向客户端输出一个 URL*/
  fprintf(stdout, "Location: %s%c%c",where[i],10,10);
  return 0;
  }
```

在 HTML 文件中加入以下语句：<A HREF ="/cgi-bin/randurl.exe" >... </A>，就可以在 Web 页面上实现一个随机链接。这项功能可以用来在几个 Web 服务器之间平均分配访问负担。比如您有一文件让万维网上的用户下载。如果这个文件非常有用，前来下载的客户非常多，您可以将该文件放在几个 Web 服务器上，而用 CGI 程序 randurl.c 在这几个服务器之间平均分配访问客户。

以下的 CGI 程序同样也是通过设置 Location 响应头向客户端的浏览器随机地提供一幅图像。

```
/*randimage.c*/
 #include <stdio.h>
 #include <stdlib.h>
 #include <time.h>
 main( )
  {
   char *image[ ]={"/image/a151.jpg", "/image/a152.jpg", /image/a153.jpg",
        "/image/a154.jpg",   "/image/a155.jpg"};
   int i;
   time_t now;
   srand((unsigned) time(&now));
   i=rand( ) % 5;
   fprintf(stdout, "Location: %s%c%c",image[i],10,10);
   return 0;
   }
```

在 HTML 文件中加入如下语句：<IMG　SRC="/cgi-bin/randimage.exe">，每次当客户端用户调用您的 Web 页面时，它将会在同样的位置显示不同的图片。

9.2 Client Pull

Client Pull 是 Netscape 公司对 HTML 的一个扩展，不过现在大多数浏览器都支持 Client Pull。

通常当客户端用户向 Web 服务器发出了请求之后，Web 服务器就将某一 Web 网页传给浏览器。除非客户端用户执行了 Reload 命令或发出新的请求，该 Web 网页将一直停留在浏览器的窗体中。而在 Web 网页中使用了 Client Pull 后，浏览器将会自动在某一时间间隔后更新 Web 网页或跳向别的 Web 网页。

Client Pull 是通过 HTML 文件中的 META HTTP-EQUIV 标识符发生作用的。该标识符必须位于 HTML 文件头内，即在<head>和</head>之间。当浏览器看到了 HTML 文件中的 META 标识符后，它会将 META 标识符之间的字符串解释成一个 HTTP 头（HTTP Header）。而通常 HTTP 头都是由 Web 服务器或 CGI 程序产生的。META 标识符的使用使得 HTML 文件本身就可以向浏览器发送一些 HTTP 头。我们知道目前的 HTTP 头已经支持网页的自动更新和 URL 重新定向，Client Pull 的工作原理就是在 HTML 文件内实现这些 HTTP 头。

Client Pull 的第一个用途是让一 Web 网页自动在某一时间间隔后更新。例如如下的 HTML 文件将在 5 秒钟后自动更新。

```
<HTML><HEAD>
  <META HTTP-EQUIV="Refresh" CONTENT=5>
  <TITLE>Client Pull Test</TITLE>
</HEAD>
<BODY BGCOLOR="#ffffff" TEXT="#000000" LINK="#0000ff" VLINK="#01424a">
<P ALIGN="center">
<IMG SRC="/cgi-bin/randimage.exe" >
</P>
</BODY></HTML>
```

请注意文件中的 HTML 语句：<META HTTP-EQUIV="Refresh" CONTENT=5> ，正是它告诉了浏览器在 5 秒钟后更新 Web 网页。该 HTML 语句和以下的 HTTP 响应头相对应：

```
Refresh: 5
```

Client Pull 应该和一些后台程序联合起来使用才更有意义。比如一个反映股市行情的 Web 网页使用了 Client Pull，它将每 15 秒钟自动更新一次。而在 Web 服务器上应该有一个后台程序，它每隔 15 秒钟自动对相应的 HTML 文件更新一次。

Client Pull 的另一个用途是让浏览器在一定的时间间隔后转向另一 Web 网页。例如如下的 HTML 文件将使浏览器在 5 秒钟后自动转向“www.edu.cn”。

```
<HTML>
<HEAD>
  <META HTTP-EQUIV="Refresh" CONTENT="5; URL=http://www.edu.cn">
  <TITLE>Client Pull Test</TITLE>
</HEAD>
<BODY BGCOLOR="#ffffff" TEXT="#000000" LINK="#0000ff" VLINK="#01424a">
```

```
<P ALIGN="center">
<IMG SRC="/cgi-bin/randimage.exe" >
</P>
</BODY></HTML>
```

文件中的语句<META HTTP-EQUIV="Refresh" CONTENT="5; URL=http://www.edu.cn">和以下的 HTTP 响应头相对应：

```
Refresh: 5; URL=http://www.edu.cn
```

通过使用 Client Pull 的这项功能，您可以为您的 Web 站点建立一缺省的浏览顺序。如果客户端用户在某一时间间隔内没有做任何选择，Web 服务器将按照预先设定的路线自动显示 Web 网页。

Client Pull 的时间间隔可以设置为 0，这将使浏览器立刻转向另一 Web 网页。

用 CGI 程序输出和 Client Pull 相对应的响应头，同样可以在 CGI 程序生成的 Web 网页上实现同样的功能，例如：

```
printf( "Refresh: 30\n");
printf( "Content-Type: text/html\n\n");
```

和

```
printf( "Refresh: 5;URL=http://www.edu.cn \n");
printf("Content-Type: text/html\n\n");
```

9.3　Server Push

Server Push 是另一项制作动态网页的技术。和 Client Pull 不同，Server Push 将由 Web 服务器而不是浏览器承担产生动态网页的任务。

一个 Server Push 始终维持 Web 服务器和客户端浏览器的 Socket 连接，并且可以连续向浏览器发出几帧或无数帧的数据，每帧的数据可以为各种 MIME 类型。当浏览器接收到数据以后就立刻显示它们，并在 Web 页面上用新收到的一帧的数据代替以前的数据。

为了让浏览器知道它所要接收的数据是一个 Server Push 类型的数据，CGI 程序首先必须声明它所要输出数据的类型为：multipart/x-mixed-replace，multipart/x-mixed-replace 是 MIME 类型 multipart/mixed 的变体。MIME 类型 multipart/mixed 可以用来向浏览器一次发送具有多种 MIME 类型（比如文本和图形）的数据。multipart/x-mixed-replace 中的“x”表示目前本 MIME 类型还处于试验中，“replace”则表示客户端收到数据后必须用刚接收到的数据块代替以前收到的数据块。各种 MIME 类型的数据块可以用预先定义的字符串分开。用于分离不同数据块的字符串前必须加上符号“--”，而表示整个发送数据结束的字符串后也必须加上“--”符号。由于每个单独的数据块的 MIME 类型可能不一样，各个数据块都必须说明自己的 MIME 类型，说明 MIME 类型的字符串和数据块的内容之间必须用空行分开。例如如下的 C 语言语句可以向浏览器发送一个 multipart/x-mixed-replace 类型的数据：

```
printf("Content-type: multipart/x-mixed-replace; boundary=BoundaryString\n\n");
printf("--BoundaryString\n");
```

```
printf (Content-type: text/html\n\n");
printf("<h1>Server Push Test1</h1>\n");
printf("--BoundaryString\n");
printf (Content-type: text/html\n\n");
printf("<h1>Server Push Test2</h1>\n");
printf("--BoundaryString--\n\n");
```

对于一个 Server Push 应用程序所生成的数据，在没有接收到表明数据结束的字符串（例如“--BoundaryString--”）之前浏览器不会关闭和 Web 服务器的连接。因此除非客户端主动中断连接，一个 Server Push 应用程序向浏览器发送数据的过程可以持续到永远。利用 Server Push 的这个特性可以在 Web 上制作一些需要 Web 服务器参与的动画，例如在 Web 页面上显示当前时间等等（利用 Server Push 和 GD 库，可以很容易编写出这类程序）。这时每个 Server Push 应用程序发送的数据块都是一些二进制的图像数据。

在某些情况下我们希望 Server Push 应用程序能以最快的速度向客户端传送数据，例如程序向客户端传送动画帧的时候。这时 Server Push 应用程序应该将输出缓冲关掉。因为当程序用 printf 语句向标准输出写信息时，首先是向缓冲区写，缓冲区写满了以后系统才将缓冲区内的信息写向标准输出。等待缓冲区写满的时间可能会影响程序的执行效率。C 语言中关掉标准输出缓冲的语句是：

```
setbuf(stdout,NULL);  或  setvbuf(stdout,NULL,_IONBF,0);
```

也可以用以下语句强迫系统将缓冲区的内容移向标准输出：

```
fflush(stdout);
```

以下的 CGI 程序说明了 Server Push 的工作原理：

```
/*pushtest.c*/
#include <windows.h>
#include <stdio.h>
void main( )
{
 int x;
 char *pushes[10] = {
                "Server Push Test Sample", "Server Push Test <i>Sample</i>!",
                "<font size=+1>This is a Server Push Test Sample</font>",
                "<font size=+2>This is a Server Push Test Sample</font>",
                "<font size=+3>This is a Server Push Test Sample</font>",
                "<font size=+4>This is a Server Push Test Sample</font>",
                "<font size=+5>This is a Server Push Test Sample</font>",
                "<font size=+4>This is a Server Push Test Sample</font>",
                "<font size=+3>This is a Server Push Test Sample</font>",
                "<font size=+2>This is a Server Push Test Sample</font>",
                  };
setbuf(stdout,NULL); /*关闭标准输出缓冲*/
```

```
/*输出 Server Push 的 MIME 类型*/
printf("Content-type: multipart/mixed;boundary=BoundaryString\n\n");
/*输出各数据块间的分界字符串*/
printf("--BoundaryString\n Content-type: text/html\n\n");
printf("<h1>Server Push Test</h1>\n");
for (x = 0; x < 10; x++)
{
 printf("\n--BoundaryString\n"); /*输出各数据块的 MIME 类型*/
   printf("Content-type: text/html\n\n"); /*输出各数据块间的分界字符串*/
   Sleep(1500);
   printf("<P>Server Push demonstration. Push %i:<br>%s\n" , x+1, pushes[x]);
  }
 printf("\n--BoundaryString--\n\n");   /*表示所有数据均已输出完毕*/
}
```

9.4　外部程序

还有一些 Web 网页变化不是很经常但却具有周期性。比如对 Web 服务器的访问者和被访问对象进行统计和分析的网页。这些网页一般需要每天或每月更新一次。对于这类网页用 CGI 程序处理有些不值得也没有必要。

这时可以用普通的外部程序对这些网页进行处理，比如外部程序可以对 Web 服务器的日志文件进行分析和处理并更新有关 Web 服务器的访问者和被访问对象的网页。而利用系统的一些命令和程序可以让这些外部程序按照一定的时间间隔运行，比如可以在每天晚上运行一次。

在 UNIX 系统上这个命令就是 cron，而在 Windows NT 上则可以使用命令 AT。WWW 上很多对 Web 服务器的被访问情况进行分析的共享程序均采用这种运行方式。

第十章　Web 网页和数据库的连接

Web 网页和数据库的连接，可以使各类系统平台上的用户方便的利用现有的数据库资源。而对于一些需要快速查询和检索的信息，使用数据库也比使用普通的文本更加有效和方便。基本上 WWW 中的任何一个 Web 站点都有和数据库相连接的愿望。目前很多公司正在或准备构筑企业内部的 Intranet，一个很重要的因素就是通过 Web 系统可以很方便的在企业内部实现各种数据库资源的共享，而通过浏览器也可以向各类系统平台的用户提供一个一致的访问界面。

Web 服务器和数据库服务器的连接有两种类型，如图 10-1 所示。一类是 Web 服务器和数据库服务器位于同一台主机上（这种情况也包括 Web 服务器和一些桌面型数据库比如 Access 等的连接），此时 Web 服务器和数据库服务器直接打交道。另一类是 Web 服务器和数据库服务器分别位于不同的主机上，但这时 Web 服务器的主机必须安装相关的数据库服务器的前端工作软件，Web 服务器将通过它和数据库服务器打交道。当然，一个 Web 服务器可以同时和不同主机上的几个数据库服务器打交道。

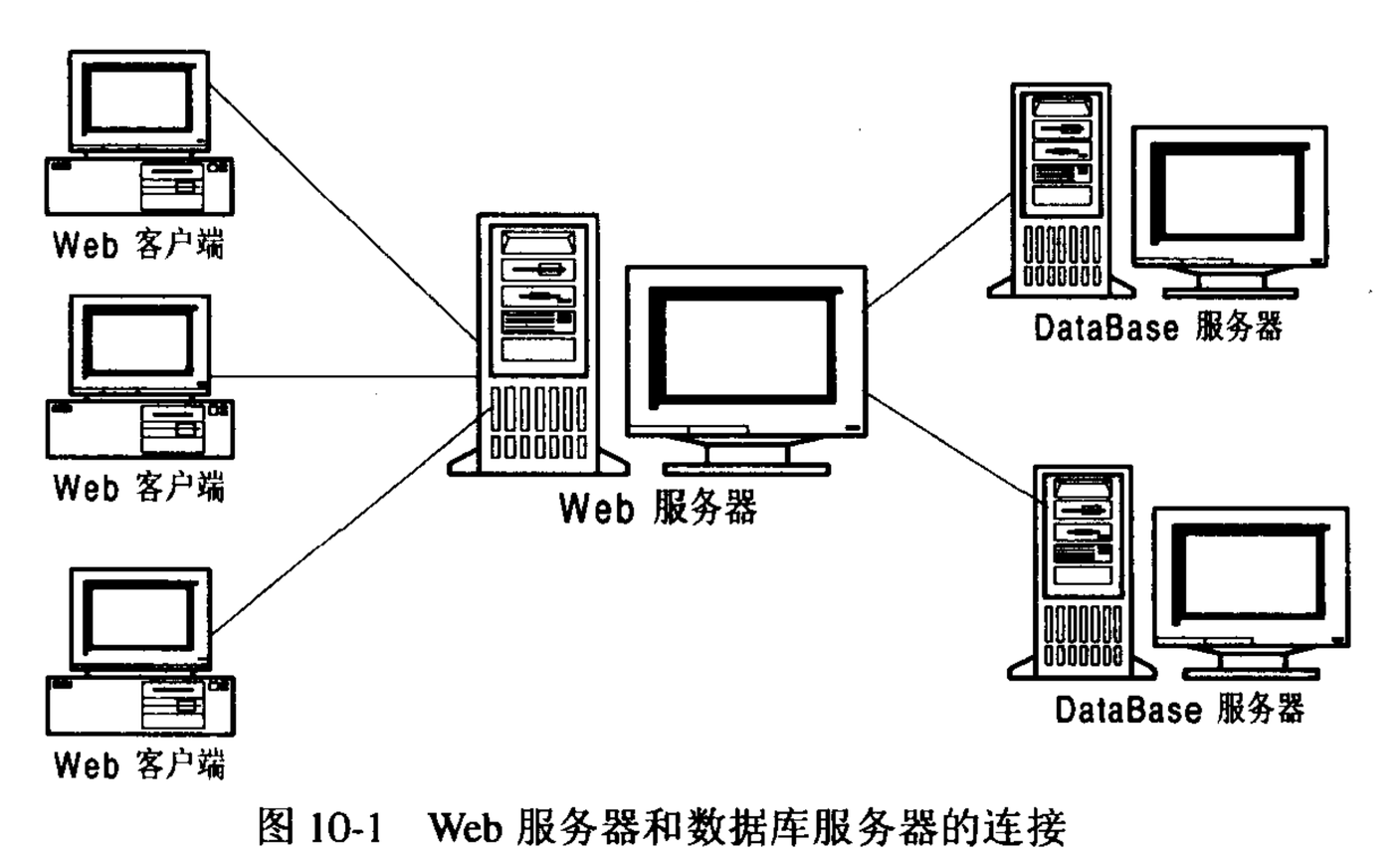

图 10-1　Web 服务器和数据库服务器的连接

Web 服务器不管是和数据库服务器的前端工作软件还是和数据库服务器本身的连接都必须借助于一个接口才能进行，可以用各种工具比如 CGI 开发这个接口。而目前很多数据库产品本身也附带了和 Web 系统的接口。这个接口的工作原理非常简单，它负责将 Web 网页上的相关信息转换成相应的数据库执行命令并将这些命令传给数据库服务器，最后再将数据库服务器的输出结果转换成 HTML 文件传给 WWW 上的客户端用户。但编写一个好的处理 Web 服务器和数据库服务器相连的 CGI 程序并不容易。首先您必须对目前流行的关系数据库模型和客户/服务器工作方式有所了解。您还必须对相关 CGI 程序中的数据库操作命令进行仔细的规划以使它们能以最高的效率安全地运行，这些都属于数据库编程的范畴。

考虑到 Visual Basic 的强大的数据库处理能力，本章将以 VB 为编程语言介绍如何编写处理数据库的 CGI 程序。由于现在已经有很多这方面成熟的产品可供使用，本章同时还

将介绍微软公司的 IDC（Internet Database Connector）、Active Server Pages 和 Netscape 公司的 LiveWire。本章的主要内容有：

- 用 VB 编写处理数据库的 CGI 程序；
- 用 IDC 实现 Web 服务器和数据库的连接；
- 用 Active Server Pages 实现服务器和数据库的连接；
- 用 LiveWire 实现 Web 服务器和数据库的连接。

10.1　VB 数据库 CGI 程序

10.1.1　VB 的数据库编程能力

Visual Basic 提供了基于 Microsoft Jet 数据库引擎（即驱动 Microsoft Access 的数据库引擎）的数据访问能力。该 Jet 引擎可以存储、检索、更新数据的结构，并提供了功能强大的、面向对象的数据访问对象（DAO）编程接口。数据库引擎存在于程序和物理数据库文件之间。这使用户与正在访问的特定数据库无关。不管这个数据库是本地的 Visual Basic 数据库，还是它所支持的其他任何数据库格式，所使用的数据访问对象和编程技术都是相同的。

Jet 数据库引擎被包含在一组动态链接库 (DLL) 文件中，在运行时，这些文件被链接到 Visual Basic 程序中。它把应用程序的请求翻译成对.mdb 文件或其他数据仓库的物理操作。它真正读取、写入和修改数据库，并处理所有内部事务，如索引、锁定、安全性和引用完整性。它还包含一个查询处理器，接收并执行结构化查询语言 (SQL) 的查询，来实现所需的数据库操作。另外还包含一个结果处理器，用来管理查询所返回的结果。

借助于 DAO，编程人员可以使用 Microsoft Jet 数据库引擎和 ODBC Direct 两种方法访问支持 ODBC 的数据库。Microsoft Jet 提供了很多功能。它可以处理建立和管理连接、把查询翻译成服务器可接受的形式和管理返回数据等许多数据库编程问题。而通过使用 ODBC Direct，DAO 可以绕过 Microsoft Jet 数据库引擎直接和 ODBC 驱动程序管理器打交道。此时程序将具有更高的效率，但编程则没有 Microsoft Jet 数据库引擎灵活方便。

对于客户/服务器模式的数据库，Visual Basic 也提供了多种访问方法。比如可以使用 Data 控件或 DAO，或者使用 RemoteData 控件与远程数据对象 (RDO)，为了提高程序的性能也可以直接调用 VBSQL 或 ODBC API 之类的数据访问应用程序接口 (API)。

10.1.2　用 VB 编写处理数据库的 CGI 程序

用 VB 编写处理数据库的 CGI 程序，和编写普通的数据库程序基本上没有什么区别。编程的主要工作也集中在数据库处理部分。和普通数据库程序的不同之处在于，处理数据库的 CGI 程序必须根据从 Web 网页上获得的信息形成数据库的操作命令，而程序最后的处理结果也将被传送给客户端的 Web 网页。

如果 CGI 程序只是和本地数据库相联系，可以使用数据访问对象 (DAO)；如果要访问远程的数据库，可以使用远程数据对象 (RDO)或 ODBC API。CGI 程序也可以使用第三方厂家提供的应用程序访问接口（API）直接访问远程的数据库。在 CGI 程序中一般不能使用数据访问控件，不过只有使用 VB 的专业或企业版才能编写使用数据访问对象或远程

数据对象的 CGI 程序。

本节在这里将通过一个小型的 ACCESS 数据库实现第七章曾经实现的贵宾留言簿。用数据库实现贵宾留言簿将有以下好处：可以很容易的按照用户定义的次序或数目显示贵宾留言簿的记录，当贵宾留言簿的内容较大时使用数据库比使用 HTML 文件的效率要高。本程序将使用 DAO 通过 Microsoft Jet 数据库引擎访问 ACCESS 数据库。对于数据库中用户可以不输入的字段，应该允许零字节长度，这主要是便于程序的处理。该程序将通过表单被激活的方法判断客户端用户是要留言还是要浏览留言簿。用于留言的 Web 页面的 HTML 文件和 7.1.3 节的 HTML 文件相同，用于显示留言簿的表单的 HTML 代码如下：

```
<html><head><title>贵宾留言簿</title></head>
<body bgcolor="#FFFFFF">
<form action="/cgi-bin/guest.exe" method="get">
<p>使用降序浏览留言簿(时间）<input type="radio" name="order" value="down">
<p>使用升序浏览留言簿(时间）<input type="radio" name="order" value="">
<p>希望查看的最大记录数:
<br><input type="text" size="30" name="num" cols="40" align="left"></p>
<p><input type="submit" name="SUBMIT" value=" 浏览 "  align="left">n="left"> </form>
</body></html>
```

程序的源代码如下所示：

```
'guestbook.bas
'声明程序中要用到的三个 Windows API 函数，一个用于处理标准输入输出文件句柄，另两个用于写
文件
    Declare Function GetStdHandle Lib "kernel32" (ByVal nStdHandle As Long) As Long
    Declare Function ReadFile Lib "kernel32" (ByVal hFile As Long, lpBuffer As Any,
        ByVal nNumberOfBytesToRead As Long, lpNumberOfBytesRead As Long,
        lpOverlapped As Any) As Long
    Declare Function WriteFile Lib "kernel32"  (ByVal hFile As Long,  ByVal lpBuffer As
        String, ByVal nNumberOfBytesToWrite As Long,
        lpNumberOfBytesWritten As Long, lpOverlapped As Any) As Long
    Public Const STD_INPUT_HANDLE = -10&
    Public Const STD_OUTPUT_HANDLE = -11&
    Public Const FILE_BEGIN = 0&
    Public hStdIn  As Long   ' 标准输入句柄
    Public hStdOut  As Long   ' 标准输出句柄
     Public sFormData  As String  ' 用于存储没有经过 URI 译码的用户输入数据
    Public lContentLength  As Long
    Public CGI_RequestMethod  As String
    Sub Main( )
    Dim CGI_ContentLength  As String
    Dim CGI_QueryString  As String
```

```
Dim sBuff   As String       ' buffer to receive POST method data
Dim lBytesRead   As Long          ' actual bytes read by ReadFile( )
Dim rc   As Long          ' return code
Dim chinesetail As String
Dim sEmail As String, sName As String, sURL As String, sfrom As String
Dim sComment As String, SubmitValue As String
Dim sHost As String, submittime As String, sbrowser As String
Dim db As Database                  '声明一个数据库对象
'CGI 程序的初始化工作或取标准输入输出文件句柄和相关的环境变量
hStdIn = GetStdHandle(STD_INPUT_HANDLE)
hStdOut = GetStdHandle(STD_OUTPUT_HANDLE)
CGI_RequestMethod = Environ("REQUEST_METHOD")
CGI_QueryString = Environ("QUERY_STRING")
CGI_ContentLength = Environ("CONTENT_LENGTH")
lContentLength = Val(CGI_ContentLength)
sBuff = String(lContentLength, Chr$(0))
If CGI_RequestMethod = "POST" Then
    sBuff = String(lContentLength, Chr$(0))
    rc = ReadFile(hStdIn, ByVal sBuff, lContentLength, lBytesRead, ByVal 0&)
    sFormData = Left$(sBuff, lBytesRead)
ElseIf CGI_RequestMethod = "GET" Then
    sFormData = CGI_QueryString
Else
 OutPut "Unknow Form Method !"
End If
OutPut "Content-type: text/html" & vbCrLf     '输出 MIME 信息
chinesetail = String(400, " ")  '为了在页面上正确显示中文生成一个空格串
If CGI_RequestMethod = "POST" Then   '如果用户是留言，则向数据库中插入信息
   OutPut "<FONT SIZE=""+2""> "
   sHost = Environ("REMOTE_ADDR")
   If Len(sHost) = 0 Then
      sHost = Environ("REMOTE_HOST")
   End If
   sbrowser = Environ("HTTP_USER_AGENT")
   '获取用户的输入
   sName = GetCgiValue("name")
   sEmail = GetCgiValue("email")
   sURL = GetCgiValue("URL")
   sfrom = GetCgiValue("from")
```

```
sComment = GetCgiValue("URL_Comment")
'对用户的输入进行检查
If Len(sName) = 0 Then
    OutPut "<P>非常抱歉！ 您还没有填写姓名！ " & chinesetail
    Exit Sub
End If
If Len(sComment) = 0 Then
 OutPut "<P>非常抱歉！ 您还没有提出建议！ " & chinesetail
 Exit Sub
End If
If Len(sEmail) <> 0 And IsGoodMail(sEmail) = -1 Then
    OutPut "<P> 非常抱歉，您输入的 E-mail 地址不合法！ " & chinesetail
    Exit Sub
End If
If InStr(sURL, "http://") = 0 And InStr(sURL, ".") <> 0 Then
   sURL = "http://" & sURL
End If
sName = SqlString(sName)
sEmail = SqlString(sEmail)
sURL = SqlString(sURL)
sComment = SqlString(sComment)
sfrom = SqlString(sfrom)
sbrowser = SqlString(sbrowser)
sHost = SqlString(sHost)
submittime = "'" & Format(Now, "mm-dd-yyyy hh:mm:ss") & "'"
'根据有关信息生成相应的 SQL 语句
SQL = "INSERT INTO guestbook1 (姓名, E-mail, 主页地址, 来自, 留言, 留言时间,_
HostIP, Browser) VALUES ("    & sName & ", " & sEmail & ", " & sURL & ", " & _
sfrom & ", " & sComment & ", " & submittime & ", " & sHost & ", " & sbrowser & " );"
'打开数据库
Set db = DBEngine.Workspaces(0).OpenDatabase ("d:\zys\doc\book\vb\guestbook.mdb")
db.Execute SQL    '执行 SQL 语句
db.Close          '关闭数据库
OutPut "<P>非常感谢您的留言!" & chinesetail
OutPut "<P>欢迎您经常光顾本主页！ " & chinesetail
OutPut "</FONT>"
OutPut "</FONT>"
Exit Sub
End If
```

```
If CGI_RequestMethod = "GET" Then  '如果用户只是浏览，则查询数据库
  Dim showorder As String, sNumber As String
  Dim Number As Integer
  Dim intNumReturned As Integer, intNumColumns As Integer
  Dim intRow As Integer, intColumn As Integer
  Dim guestbookshow As Recordset
  Dim varRecords As Variant
  Number = 100  '缺省的输出记录数
  showorder = GetCgiValue("order")  '记录的输出顺序
  sNumber = GetCgiValue("num")      '用户定义的输出记录数
  If Len(sNumber) <> 0 Then
    Number = Val(sNumber)
  End If
  If showorder = "down" Then
    SQL = "SELECT 姓名, E-mail, 主页地址, 来自, 留言, 留言时间" _
     & " FROM guestbook1 ORDER BY 留言时间 DESC;"
  Else
    SQL = "SELECT 姓名, E-mail, 主页地址, 来自, 留言, 留言时间" _
     & " FROM guestbook1 ORDER BY 留言时间;"
  End If
  '打开数据库
  Set db = DBEngine.Workspaces(0).OpenDatabase ("d:\zys\doc\book\vb\guestbook.mdb")
  Set guestbookshow = db.OpenRecordset(SQL)  '根据用户的输入生成查询集
  varRecords = guestbookshow.GetRows(Number)  '将生成的查询集存入一个二维数组中
  intNumReturned = UBound(varRecords, 2) + 1  '确定数组各维的上下标
  intNumColumns = UBound(varRecords, 1) + 1
   OutPut "<html><head><title>"
   OutPut "贵宾留言簿" & chinesetail
   OutPut "</title></head>"
   OutPut "<body bgcolor=""#FFFFFF"">"
   OutPut "<h2>" & "贵宾留言簿" & chinesetail
   OutPut "</h2>"
  '将查询集向 Web 页面输出
  For intRow = 0 To intNumReturned - 1
     OutPut "<hr> " & "<ul>"
    For intColumn = 0 To intNumColumns - 1
      If Len(varRecords(intColumn, intRow)) <> 0 Then
        OutPut "<li><b>"
        OutPut guestbookshow.Fields(intColumn).Name & chinesetail
```

```
            OutPut "</b>:  " & chinesetail
            OutPut varRecords(intColumn, intRow) & chinesetail
          End If
        Next intColumn
        OutPut "</ul>"
      Next intRow
    db.Close
    End If
    End Sub

    Sub OutPut(s As String)  ' 本子程序用于向标准输出写信息
    Dim lBytesWritten As Long
    s = s & vbCrLf
    WriteFile hStdOut, s, Len(s), lBytesWritten, ByVal 0&
    End Sub

    Public Function GetCgiValue(cgiName As String) As String
    '-----------------------------------------------------------
    ' 本子程序可以获取表单上某一元素的数据
    '-----------------------------------------------------------
      Dim delim2   As Long        ' position of "="
      Dim delim1   As Long        ' position of "&"
      Dim n As Integer
      Dim pointer1 As Long
      Dim pointer2 As Long
      Dim length As Long
      Dim length1 As Long
      Dim tmpstring1 As String
      Dim tmpstring2 As String
      pointer1 = 1
      pointer2 = 1
      delim2 = InStr(pointer2, sFormData, "=")
      pointer2 = delim2 + 1
      Do
        length = delim2 - pointer1
        tmpstring1 = Mid(sFormData, pointer1, length)
        delim1 = InStr(pointer1, sFormData, "&")
        pointer1 = delim1 + 1
        length1 = delim1 - pointer2
```

```
If delim1 = 0 Then length1 = lContentLength + 1 - pointer2
If tmpstring1 = cgiName Then
   tmpstring2 = Mid$(sFormData, pointer2, length1)
   GetCgiValue = UrlDecode(tmpstring2)
   Exit Do
End If
If delim1 = 0 Then
    Exit Do
 End If
delim2 = InStr(pointer2, sFormData, "=")
pointer2 = delim2 + 1
Loop
End Function

Public Function UrlDecode(ByVal sEncoded As String) As String
'-----------------------------------------------------------
' 本函数可以对用户输入的数据进行 URI 解码
'-----------------------------------------------------------
Dim pointer   As Long          ' sEncoded position pointer
Dim pos   As Long          ' position of InStr target
Dim temp As String
If sEncoded = "" Then Exit Function
 '本循环体用于将“+”转换成空格
pointer = 1
Do
    pos = InStr(pointer, sEncoded, "+")
    If pos = 0 Then Exit Do
    Mid$(sEncoded, pos, 1) = " "
    pointer = pos + 1
Loop
pointer = 1
    On Error GoTo errorUrlDecode
    Do    '本循环体用于将%XX 转换成字符。对于两个连续的%XX，如果第一个%XX 不是某些特指的
Web 系统保留字符，将把它们转换成汉字
        pos = InStr(pointer, sEncoded, "%")
        If pos = 0 Then Exit Do
        temp = Chr$("&H" & (Mid$(sEncoded, pos + 1, 2)))
       If Mid(sEncoded, pos + 3, 1) = "%" And (temp <> ":") And (temp <> "/") _
          And (temp <> "(") And (temp <> ")") And (temp <> ".") And (temp <> ",") _
```

```
And (temp <> ";") And (temp <> "%") Then
Mid$(sEncoded, pos, 2) = Chr$("&H" & (Mid$(sEncoded, pos + 1, 2))
    & (Mid$(sEncoded, pos + 4, 2)))
sEncoded = Left$(sEncoded, pos) & Mid$(sEncoded, pos + 6)
pointer = pos + 1
Else
  Mid$(sEncoded, pos, 1) = temp
    sEncoded = Left$(sEncoded, pos) & Mid$(sEncoded, pos + 3)
  pointer = pos + 1
  End If
Loop
On Error GoTo 0        'reset error handling
UrlDecode = sEncoded
Exit Function
errorUrlDecode:
If Err.Number = 13 Then        'Type Mismatch error
  Err.Clear
  Err.Raise 65001,"Invalid data passed to UrlDecode() function."
Else
  Err.Raise Err.Number
End If
Resume Next
End Function

Public Function IsGoodMail(email As String) As Integer
'本函数用于检查用户输入的 E-mail 地址的合法性
If InStr(email, "@") = 0 Or InStr(email, ".") = 0 Or Left$(email, 1) = "@" Or Left$(email, 1) = "." Or
Right$(email, 1) = "@" Or Right$(email, 1) = "." Then
        IsGoodMail = -1
        Exit Function
End If
IsGoodMail = 0
End Function

Public Function SqlString(s As String) As String
'本函数可以给字符串加上单引号，用于生成 SQL 语句
SqlString = "'" & s & "'"
End Function
```

10.2　IDC

10.2.1　IDC 简介

微软的 IDC（Internet Database Connector）是 Internet Information Server 的一个部件，它可以接收 Web 页面上客户端用户的输入并把用户的输入转换为相应的 SQL 语句，最后将 SQL 语句传送给相应的 ODBC 数据源并将查询结果按照事先定义的样式回送给客户端用户。IDC 只能存取支持 ODBC 的数据库，如果您想绕过 ODBC 直接存取数据库，就不能使用 IDC。

IDC 是一个 ISAPI 应用程序（httpodbc.dll），实际上它在微软为各类视窗平台提供的各种 Web 服务器下都能正常工作。如果您在微软的 Personal Web Server 的放置 CGI 程序目录下能找到文件 httpodbc.dll，您就可以使用 IDC。

图 10-2 说明了 IDC 的工作原理。当 Web 服务器发现浏览器的请求是一个 IDC 文件时，Web 服务器的 IDC 部件 httpodbc.dll 将打开相应的 IDC 文件，并将相应的 ODBC 数据源名称和 SQL 语句传给 ODBC 管理器。ODBC 管理器的相关驱动器在对 SQL 语句进行相关的语法检查后，将 SQL 语句再传送给支持 ODBC 的数据库。最终的执行结果经 ODBC 管理器被传送给 IDC。IDC 将根据模板文件和接收到的执行结果生成一个 HTML 文件。

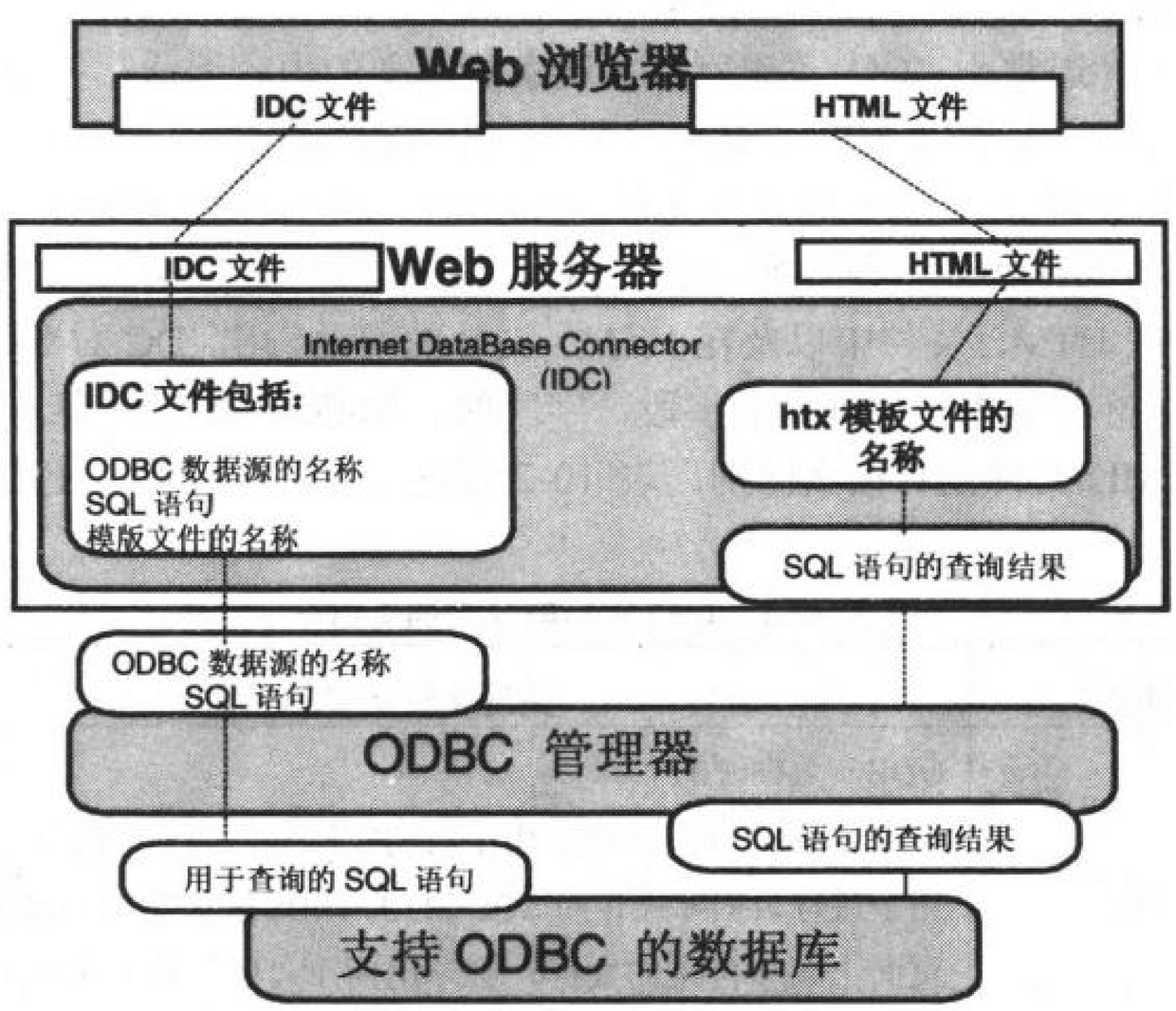

图 10-2　IDC 的工作原理

10.2.2　IDC 文件和 HTX 文件

IDC 文件是具有后缀名.idc 的文本文件。Internet Database Connector 通过 IDC 文件获取 Web 页面上表单的输入、所要连接的 ODBC 数据源名称和所要执行的 SQL 语句。IDC 文件利用关键字来定义文件所需的各部分内容。在 HTML 文件中可以通过超文本链接和表单的 ACTION 属性来激活 IDC 文件，就像使用 CGI 程序一样。IDC 文件必须放在有执行

权限的目录中。以下就是一个用于数据库查询的 IDC 文件 paper.idc 的内容：

```
Datasource: zys
Template: paper.htx
DefaultParameters:IDa=1,IDb=5000,TIME1=93/01/01,TIME2=99/12/31,
                  TITLE=%%, NAME1=%%, NAME2=%%,MCXJZ=%%
MaxRecords: 5000
SQLStatement:
+SELECT *
+FROM paper
+WHERE 发表时间>=#%TIME1%#
+AND 发表时间<=#%TIME2%#
+AND 序号>=%IDa%
+AND 序号<=%IDb%
+AND 论文题目 LIKE '%%%TITLE%%%'
+AND 第一作者 LIKE '%NAME1%'
+AND 第二作者 LIKE '%NAME2%'
+AND 期刊名称 LIKE '%%%MCXJZ%%%'
```

一个 IDC 文件通常由 IDC 关键字和关键字所定义的内容组成，关键字和所定义的内容之间用“:”分开。表 10-1 说明了 IDC 文件所使用的关键字。

HTX 文件是具有后缀名.htx 的文本文件，Internet Database Connector 根据 HTX 文件向浏览器输出相关的查询结果。和 IDC 文件一样，HTX 文件也拥有自己的关键字。这些关键字定义了如何在 HTML 文件中以及在 HTML 文件的何处输出 IDC 对数据库的操作结果。HTX 文件可以直接引用相关数据库的字段，字段的名称必须用“%”括起来。除此之外，HTX 文件和 HTML 文件没有什么区别。表 10-2 说明了 HTX 文件所使用的关键字及其用法。

表 10-1　IDC 文件所使用的关键字

IDC 关键字	关键字所定义的内容
Datasource *	ODBC 数据源的名称
Template *	HTX 模板文件的名称，用于向浏览器输出数据库操作结果
SQLStatement *	用于进行数据库操作的 SQL 语句。如果在 SQL 语句中要使用 Web 页面上表单传来的数据，必须用两个“%”将表单元素的 NAME 属性字符串括起来。比如本节 IDC 文件例子中的%NAME1%，NAME1 是表单上一文本输入框的 NAME 属性值，而两个连续的“%”则表示一个通配符。SQL 语句可以占用 IDC 文件的很多行，但每行必须以“+”开始。SQL 语句的语法必须符合 ODBC 标准。本关键字所定义的内容也可以是数据库中一个现有的查询或表，不过这时查询或表的名称必须用“'”括起来，比如'guestbook1'
DefaultParameters	当客户端用户没有输入时变量所应具有的缺省值

IDC 关键字	关键字所定义的内容
Expires	服务器的内存缓冲的更新时间间隔。当客户端用户执行相同的查询时，IDC 并不直接和 ODBC 数据源打交道而是直接参与使用内存缓冲中的数据。Expires 可以强迫 IDC 在一定的时间间隔后直接查询数据库。不过只有在 IDC 文件中使用了 Expires 关键字后 IDC 才具有内存缓冲的功能
MaxFieldSize	每条记录所能使用的缓冲内存的最大值，缺省值是 8192 个字节
MaxRecords	一次查询所能返回的最大记录数。缺省时 IDC 将返回所有满足查询条件的记录。可以利用 MaxRecords 限制 IDC 返回的记录数
Username	存取 SQL Server 数据库所需的用户名
Password	存取 ODBC 数据源所需的密码
RequiredParameters	浏览器所必须传送的变量值

注：*表示的关键字为 IDC 文件所必须的，其他均为可选项。

表 10-2　HTX 文件的关键字及其用法

HTX 文件关键字	关键字的用法
<%begindetail%> <%enddetail%>	<%begindetail%>和<%enddetail%>定义了 IDC 返回的数据在 HTX 文件中的开始位置和结束位置。在这两个关键字之间可以用数据库的字段名称代替实际的返回数据。实践中只需定义一个记录的输出方式，所有的返回记录将按这种方式输出。如果没有任何记录返回，IDC 在向浏览器输出 Web 页面时将忽略<%begindetail%>和<%enddetail%>所定义的内容。以下是一个使用<%begindetail%>和<%enddetail%>的例子： **<%begindetail%>** **期刊名称：《<%期刊名称%>》，<%卷期年页号%> ** **<%enddetail%>** 其中“期刊名称”和“卷期年页号”是数据库表中的两个字段名称
<%if%><%else%> <%endif%>	可以用这些关键字对 IDC 返回的数据进行一些条件逻辑运算。并根据逻辑运算的不同结果分别在 Web 页面上显示不同的信息。比如可以利用这些关键字在 IDC 没有找到相关的数据时返回一些信息： **<%If CurrentRecord EQ 0%>** **<P><I>对不起，没有找到您所要求的文章！ </I> </P>** **<%endif%>** 除了 EQ，这些关键字间可以使用的关系运算符还有：LT（小于）、GT（大于）和 CONTAINS（包含）。用于进行关系运算的两个操作数可以是正整数、字符串、数据库的字段值或 IDC 的内置变量
CurrentRecord	该关键字表示一个 IDC 的内置变量，它表示了一次 HTTP 会话连接关键字<%begindetail%>和<%enddetail%>被 IDC 使用的次数。当<%begindetail%>和<%enddetail%>第一次被使用时，该变量的值为 0

HTX 文件关键字	关键字的用法
MaxRecords	IDC 内置变量，它记录了相应 IDC 文件关键字 MaxRecords 的值
Parameters	该关键字可以存取任何 IDC 文件传递给 IDC 的变量值。但在变量名前须加上前缀“.idc”，比如在 HTX 文件中可以用<%idc.NAME1%>存取相应 IDC 中的变量 NAME1 的值
HTTP variables	IDC 所内置的一些 HTTP 变量和 CGI 环境变量。参见微软的相关手册

以下是一个和本节 paper.idc 相对应的 HTX 文件 paper.htx。

```
<html>
<head><title>查询结果</title></head>
<body bgcolor="#FFFFFF">
<%begindetail%>
<hr>
<p><font size="2">序号：<%序号%>，类别：<%类别%> <br>
文章题目：<%论文题目%> <br>
文章作者：<%第一作者%> <%第二作者%> <%第三作者%> <br>
期刊名称：《<%期刊名称%>》，<%卷期年页号%> <br>
发表时间：<%发表时间%> </font></p>
<%enddetail%>
<hr>
<%If CurrentRecord EQ 0%>
<p><i>对不起，没有找到您所要求的文章！ </i> </p>
<%endif%>
<p><a href="/index.htm"><i>开始一个新查询 </i></a></p>
</body></html>
```

10.2.3 用 IDC 实现贵宾留言簿

本节将用 IDC 实现用 VB 曾经实现的贵宾留言簿。使用了两个 IDC 文件，一个用于向数据库中加入客户端用户的留言，一个用于向客户端用户显示客户的留言。和两个 IDC 文件相对应，两个 HTX 文件用于显示 IDC 的处理结果。IDC 文件和 HTX 文件都应该放在一个具有执行权限的目录中。

在使用贵宾留言簿之前必须将有关留言簿的 ACCESS 数据库加到 ODBC 管理器中。Windows 95 下的 ODBC 管理器在控制面板中。打开 ODBC 管理器后应首先选择 ACCESS 驱动器，然后选择“系统 DSN”按钮，在弹出的菜单中选择“添加（Add）”按钮，在“Data Source Name”对话框中填写“guestbook”，然后选择“Select”按钮，加入相应的 ACCESS 数据库。

虽然 HTX 文件可以直接存取服务器设置的环境变量，但 IDC 文件却缺少这种功能，因此使用 IDC 将无法向数据库中加入诸如当前时间、客户端主机的 IP 地址等信息。

用于向留言簿中加入客户留言的表单的 HTML 代码如下所列：

```
<html><head><title>贵宾留言簿</title></head>
<body>
<h3>贵宾留言簿测试</h3>
<form action="/query/guestbook.idc" method="post">
您的姓名: <input type="text" name="name"><br>
您的 E-mail 信箱: <input type="text" name="email"><br>
您的主页的 URL: <input type="text" name="URL"><br>
您的建议:<br> <textarea name="URL_Comment" rows=4 cols=30></textarea><br>
您来自: <input type="text" name="from"><br><input type="submit" value=" 留言 ">
</form>
</body></html>
```

和它相对应的 IDC 文件 guestbook.idc 的内容为：

```
Datasource: guestbook
Template: guestbook1.htx
DefaultParameters:
SQLStatement:
+INSERT INTO guestbook1 (姓名, E-mail, 主页地址, 来自, 留言)
+ VALUES ('%name%', '%email%', '%URL%', '%from%', '%URL_Comment%');
```

guestbook.idc 的模板文件 guestbook1.htx 主要是向用户返回一些感谢信息，内容为：

```
<html><head><title>贵宾留言簿</title></head>
<body bgcolor="FFFFFF">
<p> <font size="+2">
<P>非常感激您的留言!
<p> 欢迎您经常光临本主页!
</font>
</body></html>
```

读者可以发现它和普通的 HTML 文件没有任何区别。

用于浏览留言簿的 HTML 文件如下所列：

```
<html><head><title>贵宾留言簿</title></head>
<body bgcolor="#FFFFFF">
<p><form action="/query/showguestbook.idc" method="get">
<p>使用降序浏览留言簿(时间 ) <input type="radio" name="order" value="DESC">
<p>使用升序浏览留言簿(时间 ) <input type="radio" name="order" value="">
<p>希望查看的最大记录数: <BR><input type="text" size="30" name="num" cols="40" >
<p><input type="submit" name="SUBMIT" value=" 浏览 ">
</form>
</body></html>
```

处理该 HTML 文件的 IDC 文件 showguestbook.idc 的内容如下：

```
Datasource: guestbook
Template: guestbook2.htx
DefaultParameters:
SQLStatement:
+SELECT 姓名,E-mail,主页地址,来自, 留言, 留言时间
+FROM guestbook1 ORDER BY 留言时间 %order%
```

由于要显示查询结果，该 IDC 的模板文件 guestbook2.htx 要比 guestbook1.htx 复杂的多，它的内容如下：

```
<html><head><title>贵宾留言簿</title></head>
<body bgcolor="FFFFFF">
<p> <font size="2">
<%BeginDetail%>
<hr><ul>
<li><b>姓名: </b><%姓名%>
<%if E-mail GT "" %>
<li><b>E-mail: </b><%E-mail%>
<%endif%>
<%if 主页地址 GT "" %>
<li><b>主页地址: </b><%主页地址%>
<%endif%>
<li><b>来自: </b><%ComeFrom%>
<li><b>留言: </b><%留言%>
<li><b>留言时间: </b><%留言时间%>
</ul>
<%EndDetail%>
</font>
</body></html>
```

虽然使用 IDC 可以不用进行数据库编程，但 IDC 的使用显然没有 CGI 编程灵活。例如在 CGI 程序中可以存取服务器设置的环境变量或系统的相关资源，但 IDC 就不行了。

10.3　ASP

ASP（Active Server Pages）是微软的 Internet Information Server 3.0 的一个部件。利用 ASP，用户可以在服务器端执行用 VBscript 或 Javascript 编写的嵌入 HTML 文件中的小程序，而在以前，这些小程序都是在客户端的浏览器上执行的。由于是在服务器上执行，程序可以不必考虑与各类浏览器的兼容问题。虽然在服务器上执行会增加服务器的负担，但由于此时程序可以存取服务器的有关资源，这些用解释性语言编写的程序将具有强大的交互式能力。实际上微软的基于 Windows 95 和 NT Workstation 平台的 Web 服务器在安装了相关部件后同样也支持 Active Server Pages 技术。通常使用 Active Server Pages 技术的 Web 网页具有后缀名.asp。当客户端用户的请求是一个 ASP 文件时，服务器将在传送该文件之

前执行文件中的程序脚本，并把程序的输出放在页面中相应的位置。ASP 文件中所有位于“<%”和“%>”之间的字符串，都将被服务器解释为程序代码。例如以下的 ASP 文件 hello.asp 将在浏览器的窗体上显示如图 10-3 所示的 Web 页面：

```
<HTML>
<BODY>
<% For i = 3 To 7 %>
<FONT SIZE=<% = i %>>
Hello World!<BR>
<% Next %>
</BODY> </HTML>
```

ASP 本身提供了很多 ActiveX Server 部件以扩展 ASP 的编程能力，这些部件对调用者隐藏了具体的编程细节。通过对这些部件的调用，编程人员用很少的代码就可以完成许多复杂的任务。用户也可以用其他编程语言开发自己的 ActiveX Server 部件。

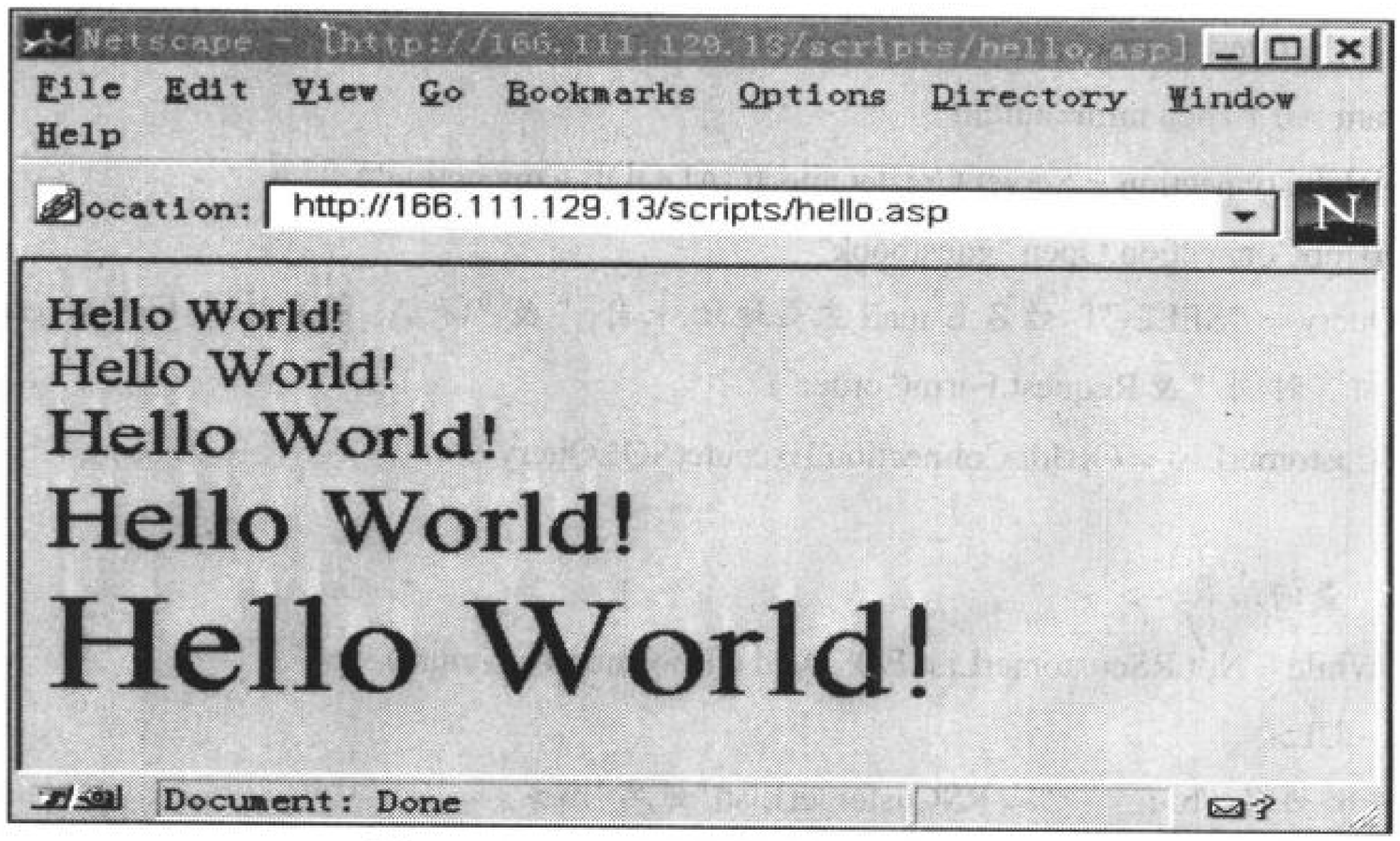

图 10-3　ASP 文件所产生的 Web 页面

数据库存取部件（Database Access Component）是 ASP 提供的一个对支持 ODBC 的数据库进行操作的部件。借助于 ADO 模型（ActiveX Data Objects），数据库存取部件向编程人员提供了非常容易和方便的编程界面。ADO 模型为数据库编程提供了一系列的对象，例如 Connection 对象可以在程序和数据库之间建立连接，而 Command 和 Recordset 对象则可以对已建立的 Connection 连接对象进行某些操作。和它们的名称相符合，Command 对象可以用来执行 SQL 语句，Recordset 对象可以存储程序对数据库的操作结果。在 ASP 文件中使用这些对象是非常简单的，例如如下的 VBscript 语句将生成一个 Recordset 对象：

```
<% Set RS = Server.CreateObject("ADODB.Recordset") %>
```

语句中的 Server 是 ASP 的一个内置的对象，它可以用来控制 ASP 文件的运行环境。除了 Server 对象，ASP 还有 Request、Response、Session 和 Application 等内置对象。这些对象可以用来完成获取客户端用户信息，向用户发送信息和存储、共享客户端用户信息的任务。

以下用 VBscript 编写的 ASP 文件 showguestbook.asp 可以将本章 10.1、10.2 节所建立的贵宾留言簿数据库中客户端用户的留言显示给客户：

```
<%@ LANGUAGE = VBScript %>
<!-- showguestbook.asp -->
<HTML>
<HEAD><TITLE>贵宾留言簿</TITLE></HEAD>
<BODY bgcolor="FFFFFF">
<!-- 和 ODBC 数据源相连 -->
<%
  Dim num
  Dim number
  num=100
  number=0
  numm = Request.Form("num")
  if(numm > 0 ) Then num=numm
  Set OBJdbConnection = Server.CreateObject("ADODB.Connection")
     OBJdbConnection.Open "guestbook"
  SQLQuery = "SELECT 姓名,E-mail,主页地址,来自, " & "留言, 留言时间 FROM guestbook1" & "
ORDER BY 留言时间 " & Request.Form("order")
  Set RSCustomerList = OBJdbConnection.Execute(SQLQuery)
 %>
<!-- 显示查询结果 -->
<% Do While   Not RScustomerList.EOF And CBool(number - num) %>
  <HR><UL>
   <li><b>姓名</b>:     <%= RSCustomerList("姓名")%>
   <% If (RScustomerList("E-mail") > "") Then %>
    <li><b>Email</b>:    <%= RScustomerList("Email") %>
   <% End If %>
   <% If (RScustomerList("主页地址") > "") Then %>
    <li><b>主页地址</b>:    <%= RScustomerList("主页地址") %>
   <% End If %>
   <% If (RScustomerList("来自") > "") Then %>
    <li><b>来自</b>:    <%= RScustomerList("来自") %>
  <% End If%>
  <li><b>留言</b>:    <%= RScustomerList("留言") %>
  <li><b>留言时间</b>:    <%= RScustomerList("留言时间") %>
  </UL>
 <% number = number+1 %>
 <% RScustomerList.MoveNext
```

```
Loop %>
</BODY> </HTML>
```

激活该 ASP 文件的 Web 表单的 HTML 文件的内容如下：

```
<html><head><title>贵宾留言簿</title></head>
<body bgcolor="#FFFFFF">
<p><form action="/query/showguestbook.asp" method="post">
<p>使用降序浏览留言簿(时间 ) <input type="radio" name="order" value="DESC">
<p>使用升序浏览留言簿(时间 ) <input type="radio" name="order" value="">
<p>希望查看的最大记录数: <BR><input type="text" size="30" name="num" cols="40" >
<p><input type="submit" name="SUBMIT" value=" 浏览 ">
</form>
</body></html>
```

为了让 ASP 文件正常运行，ASP 文件必须放在有执行权限的目录下，ASP 文件所需操作的数据库必须设置成 ODBC 管理器的系统数据库。整个系统还必须安装 ASP 的相关服务器部件和 ADO 部件。

虽然 ASP 只能存取支持 ODBC 的数据库，只能使用微软的基于视窗平台的 Web 服务器产品，但由于它使用的 VBscript 编程语言是 Visual Basic 的一个子集，熟悉 VB 的编程人员可以很快掌握 ASP 的使用。

10.4　Livewire

10.4.1　Livewire 简介

Livewire 是 Netscape 公司提供的用于管理 Web 服务器和开发客户/服务器模式应用程序的一个开发环境。它有三个部件：Web 站点管理器、Livewire 应用编译器和 Livewire 服务器扩展部分。Livewire 必须和 Netscape 公司的 Web 服务器一起使用，它可以工作在各种 UNIX 系统平台、视窗平台和 Macintosh 平台上。

Livewire 使得以前在客户端的浏览器上运行的用 Javascript 编写的嵌在 HTML 文件中的小程序现在可以在服务器端运行。这和微软的 ASP 技术很类似。一个 Livewire 应用的 Javascript 代码既可以嵌入在 HTML 文件中，也可以作为一个单独的文件而存在（此时它是作为一个库被 Livewire 应用程序调用）。Livewire 应用程序也可以调用 Java、C 语言或 C++ 程序库以及 CGI 程序。Livewire 应用程序必须编译成独立于平台的中间代码（bytecodes）以后才能使用。当然，在 Livewire 应用程序中也可以使用客户端执行 Javascript 代码。HTML 文件中客户端执行的 Javascript 代码应该放在 SCRIPT 标识符中，服务器端执行的 Javascript 代码则应放在 SERVER 标识符中。

Livewire 应用程序一般拥有 request、client、project 和 server 等四个内置对象。每当接收到客户端用户的一个请求时，服务器就生成一个 request 对象，该对象保存了当前请求的有关信息。当不同的客户端用户请求服务器的资源时，服务器会生成一个 client 对象，该对象保存了有关客户端用户的状态信息。当一个 Livewire 应用程序被激活时，服务器会生成一个 project 对象，该对象保存了有关整个 Livewire 应用程序的信息。借助于 project 对

象，可以在各个客户端用户之间共享信息。server 对象则包含了整个服务器的状态信息，服务器一启动就会生成一个 server 对象。借助于 project 对象，可以在所有的客户端用户和 Livewire 应用程序之间共享信息。

借助于 Livewire，Web 网页可以存取 Informix、Oracle、Sybase 数据库和其他一些支持 ODBC 的数据库。这些数据库既可以和 Web 服务器在同一台主机上，也可以位于不同的主机上。

有关 Livewire 的使用、设置和 Javascript 的编程读者可以参考有关书籍和 Netscape 公司相关产品的联机手册。本书在这里只是说明一种解决问题的方法。

10.4.2 用 Livewire 实现贵宾留言簿

Livewire 应用程序是通过对 database 对象的使用和各类数据库发生联系的。database 对象没有属性但拥有各种操作数据库的方法，例如 connect、cursor、execute 等。对这些方法的说明和使用可以参考相关产品的联机手册。当一个 Livewire 应用程序准备和一个数据库服务器相连接时就生成一个 database 对象。一个应用程序只能拥有一个 database 对象。例如如下的 Javascript 代码将和名为“servername”的服务器的 Informix 数据库“dbname”建立连接，所使用的用户名和口令分别为“ADMIN”和“123ZJY”，如果连接不成功，则向客户端输出相关信息。

```
database.connect("INFORMIX", "servername", "ADMIN", "123ZJY", "dbname")
if(!database.connected( ))
{write("<HR>，非常抱歉，与数据库的连接没有成功！ ")}
```

本节将用 Livewire 实现本章 10.2 节用 IDC 曾经实现的贵宾留言簿。用于向留言簿数据库中插入客户留言的 Livewire 程序 insert.html 如下所示：

```
<HTML>
<HEAD><TITLE>贵宾留言簿</TITLE>
</HEAD>
<BODY bgcolor="#FFFFFF">
<!-- 和 ODBC 数据源 guestbook 相连 -->
<SERVER>
database.connect("ODBC","guestbook","","","")
</SERVER>
<!-- 生成一个 Date 对象已获取当前时间 -->
<SERVER>today = new Date( )</SERVER>
<!-- 利用 SQL 语句向数据库中插入用户留言 -->
<SERVER>
database.execute("INSERT INTO guestbook1 (姓名, E-mail, 主页地址, 来自, 留言, HostIP, Browser)
VALUES (" + request.name + ", " + request.email + ", " + request.URL + ", " + request.from + ", " +
request.URL_Comment + ", " + request.ip + ", " + request.agent + ")")
</SERVER>
<!-- 切断和数据库的连接 -->
```

```
<SERVER>database.disconnect()</SERVER>
<P> 非常感谢您的留言，欢迎您经常光顾本主页！
</body></html>
```

用于向客户端用户显示留言的 Livewire 程序 list.html 的内容如下所示：

```
<HTML><HEAD>
<TITLE>贵宾留言簿</TITLE>
</HEAD>
<BODY bgcolor="#FFFFFF">
<!-- 和 ODBC 数据源 guestbook 建立连接 -->
<SERVER>
database.connect("ODBC","guestbook","","","")
</SERVER>
<!-- 利用 cursor 方法生成一个相关 SQL SELECT 语句的结果集 -->
<SERVER>
custs = database.cursor("SELECT 姓名, E-mail, 主页地址, 来自, 留言, 留言时间 FROM guestbook1
ORDER BY 留言时间 ")
</SERVER>
<!-- 缺省的向用户显示的记录数 -->
<SERVER>maxnumber=200</SERVER>
<!-- 获取用户希望看到的记录数 -->
<SERVER>
if(parseInt(request.num)!=0){maxnumber=parseInt(request.num)}
</SERVER>
<!-- 向用户输出查询结果 -->
<SERVER>
number=0
while (custs.next( ) && number<maxnumber)
 {
        write("<HR>")
        write("姓名： "+custs[0]+"<BR>" )
        if(custs[1]>"") {write("E-mail:  " + custs[1] + "<BR>")}
        if(custs[2]>"") {write("主页地址： " + custs[2] + "<BR>")}
        if(custs[3]>"") {write("来自： " + custs[3] + "<BR>")}
        write("留言： " + custs[4] + "<BR>")
        write("留言时间： " + custs[5] + "<BR>")
        number=number+1
}
</SERVER>
<SERVER>custs.close()</SERVER>
```

```
<SERVER>database.disconnect()</SERVER>
</body></html>
```

在使用 insert.html 和 list.html 之前必须用 Livewire 对它们进行编译，可以用 Livewire 的调试器查看程序的错误信息。通常一个Livewire应用的所有文件例如insert.html和list.html应该放在一个目录下。在使用编译好的 Livewire 应用之前应该先用 Livewire 应用管理器将它激活，并确保 Web 服务器允许执行 Livewire 应用。调用 insert.html 和 list.html 的 Web 页面和 10.2 节所使用的 Web 页面一样（addguest.html 和 showguest.html）。

本节的有关贵宾留言簿的 Livewire 应用的名称为 guestbook，则在 HTML 文件中调用本 Livewire 应用的相对路径则为“/guestbook”，和文件所在的具体目录名无关。因此 addguest.html 和 showguest.html 中的表单激活相应的 Livewire 应用的 URL 应改为如下的形式：

```
<form action="/guestbook/list.html" method="post">
<form action="/guestbook/insert.html" method="post">
```

Livewire 的使用和设置较 ASP 要复杂一些，但考虑到它可以运行在各种操作系统平台上，并且除了 ODBC 还支持其他的客户/服务器模式的数据库，花些精力掌握它还是值得的。

第十一章　Image Map 的使用

一般 Web 网页的超文本链接均是建立在文本或图片之上，一段文本或一个图片指向一个超文本链接。如果使用了 Image Map，则可以在一个图片中建立多个超文本链接，也许您已经在很多 Web 网页中看见了 Image Map 的应用。例如很多学校的 Web 主页都有一个校园浏览图，当客户端用户用鼠标单击图片的某一区域时，Web 浏览器将根据鼠标在图片上的单击部位把您带向校园的某一具体风景点。本章将向读者介绍各类 Image Map 的原理和应用。

- Web 服务器内置的 Image Map；
- 客户端浏览器内置的 Image Map；
- 用 CGI 程序实现 Image Map。

11.1　什么是 Image Map

Web 网页上的 Image Map 实际上就是一个图片，这个图片可以是 GIF 或 JPEG 等 Web 浏览器所支持的任何图像文件格式。不过在一个 Image Map 图片上您可以定义很多区域，这些区域可以是一个圆、一个矩形或任意多边形，甚至可以是一个点。Image Map 上的每个区域可以指向万维网 WWW 上的一个 URL。当客户端用户用鼠标单击了 Image Map 图片上已经定义的某一区域时，Web 浏览器将把鼠标在图片上的单击部位的坐标位置通过环境变量 QUERY_STRING 传送给 Web 服务器或相应的 CGI 程序，Web 服务器或 CGI 程序将根据相应的 Image Map 配置文件确定单击部位属于图片上哪个区域，并将相对应的 URL 发送给客户端的 Web 浏览器。

有些类型的 Image Map 要求为整个 Image Map 定义一个缺省的 URL，当客户端用户的鼠标单击了 Image Map 上没有定义的部位时，Web 服务器将向客户端传送一个缺省的 URL。

11.2　Web 服务器内置的 Image Map

很多 Web 服务器例如 NCSA HTTPd 1.5、Apache Server 1.0 和 Netscape 的各类 Web 服务器等都自带 Image Map 功能，这就意味着您不用 CGI 程序就可以处理 Web 网页上的 Image Map 图片。服务器内置的 Image Map 将比外部的 CGI 程序具有更快的响应速度，并且可以降低服务器的负担。本节将以 NCSA HTTPd 1.5 为例说明这类 Image Map 的使用。

在使用服务器内置的 Image Map 功能时必须为 Web 网页上相应的 Image Map 设置一个配置文件，该文件名的后缀为 .map。同时在服务器的配置文件 srm.conf 中应该加入以下命令将服务器内置的 Image Map 功能打开。

```
AddType  text/x-imagemap  .map
```

Image Map 的配置文件由注释行和设置行组成。注释行由字符“#”打头，设置行语句则具有下列的形式：

method　URL　coord1　coord2 ...　coordn

其中 method 说明了 Image Map 上区域的类型，URL 说明了区域所指向的 URL，coord1 等坐标点则表示了区域的范围。坐标点由两个用逗号分开的整数组成，例如“123，145”，坐标点间用空格分开。

method 的取值有以下几种类型：

- default：定义 Image Map 缺省的 URL，它不需要坐标点。
- circle：定义了 Image Map 上的一个圆形区域。它需要两个坐标点。一个坐标点为圆形区域的圆心，另一个坐标点为圆周上任意一点。
- poly：定义了 Image Map 上的一个多边形区域。它需要多边形的顶点作为它的坐标点，一个多边形最多可以拥有 100 个顶点。
- rect：定义了 Image Map 上的一个矩形区域。它需要矩形的左上点和右下点作为它的坐标点。
- point：定义了 Image Map 上的一点。

以下就是一个 Image Map 配置文件：

```
# sample.map
default   /examples/none.html
circle    /examples/ow.html   313,28 313,44
poly      /examples/rail.html   175,320 227,383 347,268 345,166
rect      /examples/post.html   397,89 436,265
```

Web 服务器将按照 Image Map 配置文件所设置的顺序检查客户端用户的鼠标单击部位属于 Image Map 的哪一区域。如果配置文件所定义的区域有重合，应该将优先考虑的区域放置在配置文件的前面。可以用 Internet 上的很多共享的软件工具生成一个 Image Map 的配置文件。各类支持 Image Map 功能的 Web 服务器的有关 Image Map 的配置文件基本相同。

Image Map 的配置文件设置好以后，就可以在 Web 网页上加上一个 Image Map 图片了。例如如下的 HTML 语句将为 Web 网页加上一个 Image Map 图片：

```
<A HREF="http://www.bb.aa/zys/sample.map">
<IMG SRC="/zhangys/gifs/sample.gif"  ISMAP></A>
```

在 HTML 标识符<IMG>中必须使用标识符 ISMAP，否则浏览器不会将客户端用户的鼠标在图片上单击部位的坐标传送给 Web 服务器，服务器的 Image Map 功能将无法正确执行。

11.3　客户端浏览器内置的 Image Map

在 Web 网页内使用客户端浏览器内置的 Image Map，当客户端用户用鼠标单击了 Web 网页上的 Image Map 图片时，浏览器将根据客户端用户所单击的部位生成一个 HTTP 请求头并将该请求头（而不是单击部位的坐标）发送给 Web 服务器。显然以前部分由 Web 服务器完成的工作现在则由浏览器完成，这可以减轻 Web 服务器的负担。虽然如此，当客户端的浏览器没有内置的这项功能时，Web 网页上的 Image Map 图片将仅仅是一个图片。

为了让浏览器知道向万维网上的何处资源发出请求，在 Web 网页内部必须包含 Image

Map 的设置信息。浏览器将根据网页内部所包含的有关设置信息和客户端用户揿击部位的坐标决定向万维网上的何处资源发出请求。

为了在 Web 网页上插入一个客户端浏览器内置的 Image Map 图片，可以在 HTML 标识符 <image> 中使用参数 USEMAP：

```
<IMG SRC="Location of the image"  USEMAP="Location of the map">
```

参数 USEMAP 的值表示了相关 Image Map 的配置信息在 Web 网页中的位置。在 Web 网页中可以用 <MAP> 标识符表示 Image Map 的配置信息。<MAP> 标识符的使用如下所示：

```
<MAP NAME="...">  Image Map 的配置信息  </MAP>
```

其中 NAME 属性的值应和 USEMAP 的值相等。例如在一 Web 网页中有一客户端浏览器内置的 Image Map：

```
<IMG SRC= "logo.gif"  USEMAP= "#logo">
```

则在同一 Web 页面内应用以下的 HTML 语句表示 Image Map 的配置信息：

```
<MAP NAME="logo">  ......  </MAP>
```

如果 Web 站点上的很多网页都拥有一个相同的 Image Map，这时可以将该 Image Map 的配置信息放在一个单独的文件中，而在每个拥有 Image Map 图片的网页上可以用以下的 HTML 语句调用这个文件中的 Map 配置信息：

```
<IMG  SRC="logo.gif"  USEMAP="logo.html#logo">
```

文件 logo.html 的内容由标识符<MAP NAME="logo">、</MAP>和位于标识符间的配置信息组成。

在<MAP>间可以用以下的 HTML 语句定义 Image Map 的配置信息：

```
<AREA  SHAPE="..."  COORDS="..."  HREF="..."  ALT="...">
```

SHAPE 参数说明了 Image Map 图片上所定义区域的类型，目前它可能的取值有 RECT、CIRCLE 和 POLYGON，分别表示矩形、圆和多边形。如果 SHAPE 缺省，则所定义的区域为矩形。

参数 COORDS 的值则由一系列用逗号分开的正整数组成，它向浏览器表明了所定义区域的边界。对于矩形区域来说，它由矩形的左上顶点和右下顶点的坐标组成。对于圆形区域来说，COORDS 的值由圆心的坐标和半径组成。对于多边形区域来说，COORDS 的值由多边形的顶点坐标组成（注意 COORDS 的值不能超过 1024 个字节）。

参数 HREF 定义了当客户端用户单击了 Image Map 图片相应的区域时浏览器应该向 WWW 上的何处资源发出请求。

参数 ALT 定义了 Image Map 图片上相关区域的文字说明。当客户端用户使用的是文本方式的浏览器时，浏览器将向用户显示一系列由 ALT 所定义的文本说明，当用户用鼠标选择了其中一项时，浏览器将向 HREF 所定义的万维网资源发出请求。

和 Web 服务器内置的 Image Map 一样，当所定义的 Image Map 图片上的两个区域有重合时，最先定义的区域将覆盖后定义的区域。

以下是一个包含客户端浏览器内置的 Image Map 的 Web 网页：

```
<html> <head><title>Client Side Image Map Test</title></head>
<body>
<img src="/zhangys/gifs/sample.gif "  USEMAP="#sample">
```

```
<map name="sample">
<area shape="circle"   href="/examples/ow.html " coords="313,28,16" alt="The ow">
<area href="/examples/post.html" coords="397,89 436,265" alt="The Post">
<area shape="polygon" href="/examples/rail.html " coords="175,320 227,383 347,
        268 345,166" alt="The Rail">
</map>
</body> </html>
```

11.4　用 CGI 程序实现 Image Map

虽然用 Web 服务器和浏览器的内置功能可以很容易的在 Web 网页上实现 Image Map，但并不是每种 Web 服务器和浏览器都支持 Image Map 功能。而用 CGI 程序实现的 Image Map 则可以在任何种类的浏览器和支持 CGI 的 Web 服务器上运行，正是由于这个原因，WWW 上的很多 Image Map 都是由 CGI 程序实现的。

用 CGI 程序实现 Image Map 时，CGI 程序首先要获得客户端用户单击部位的坐标。根据相关的配置文件和所获得的坐标，CGI 程序可以判断单击部位属于图片上的哪一区域。最后，CGI 程序将表示 WWW 资源的 URL 传送给客户端浏览器，客户端浏览器将根据受到的响应头向新的 URL 发出请求。

可以用两种方法将客户端用户在 Image Map 图片上单击部位的坐标传送给 CGI 程序。一种方法是 11.2 节所介绍的<IMG>标识符：

```
<A HREF="http://www.bb.aa/cgi-bin/imagemap/sample.map">
<IMG SRC="/zhangys/gifs/sample.gif"   ISMAP></A>
```

此时单击部位的坐标将通过环境变量 QUERY_STRING 的值传送给 CGI 程序，组成坐标的两位整数被逗号分开。

另一种方法就是使用<INPUT>标识符：

```
<INPUT TYPE=IMAGE   SRC="..."   NAME="..."   ALIGN="...">
```

其中 SRC 表示 Image Map 图片的位置，NAME 属性储存了客户端用户单击部位的坐标。ALIGN 属性的含义此时类似标识符<IMG>的 ALIGN 属性的含义。

当客户端用户用鼠标单击了 Image Map 图片的任何位置时都相当于用户单击了表单的 Submit 按钮。

不过很遗憾，不是所有的浏览器都支持用<INPUT>标识符表示的 Image Map 图片。读者应该尽量使用第一种方法。

万维网上有很多关于 Image Map 的 CGI 程序。其中使用最普遍的是 NCSA 的程序 imagemap.c，在 NCSA HTTPd 的软件包中都有这个程序。Web 网页可以将配置文件的位置通过环境变量 PATH_TRANSLATED 传送给 CGI 程序。配置文件的结构和 NCSA HTTPd 内置 Image Map 所需的配置文件的结构一样。以下是 imagemap.c 的源代码：

```
/ * mapper 1.2 7/26/93 Kevin Hughes, kevinh@pulua.hcc.hawaii.edu

"macmartinized" polygon code copyright 1992 by Eric Haines, erich@eye.com

All suggestions, help, etc. gratefully accepted!
```

```
1.1 : Better formatting, added better polygon code.
1.2 : Changed isname( ), added config file specification.
11/13/93: Rob McCool, robm@ncsa.uiuc.edu
1.3 : Rewrote configuration stuff for NCSA /htbin script
12/05/93: Rob McCool, robm@ncsa.uiuc.edu 1.4 : Made CGI/1.0 compliant.
06/27/94: Chris Hyams, cgh@rice.edu
Based on an idea by Rick Troth (troth@rice.edu)
1.5 : Imagemap configuration file in PATH_INFO.   Backwards compatible.
 Old-style lookup in imagemap table:
   <a href="http://foo.edu/cgi-bin/imagemap/oldmap">
 New-style specification of mapfile relative to DocumentRoot:
   <a href="http://foo.edu/cgi-bin/imagemap/path/for/new.map">
 New-style specification of mapfile in user's public HTML directory:
   <a href="http://foo.edu/cgi-bin/imagemap/~username/path/for/new.map">
07/11/94: Craig Milo Rogers, Rogers@ISI.Edu
1.6 : Added "point" datatype: the nearest point wins.   Overrides "default".
08/28/94: Carlos Varela, cvarela@ncsa.uiuc.edu
1.7 : Fixed bug:   virtual URLs are now understood.
Better error reporting when not able to open configuration file.
03/07/95: Carlos Varela, cvarela@ncsa.uiuc.edu
1.8 : Fixed bug (strcat->sprintf) when reporting error.
Included getline() function from util.c in NCSA httpd distribution. * /
#include <stdio.h>
#include <string.h>
#ifndef pyr
#include <stdlib.h>
#else
#include <ctype.h>
#endif
#include <sys/types.h>
#include <sys/stat.h>
#define CONF_FILE "/usr/local/etc/httpd/conf/imagemap.conf"
#define MAXLINE 500
#define MAXVERTS 100
#define X 0
#define Y 1
#define LF 10
#define CR 13
int isname(char);
```

```
int main(int argc, char * argv)
{
    char input[MAXLINE], *mapname, def[MAXLINE], conf[MAXLINE], errstr[MAXLINE];
    double testpoint[2], pointarray[MAXVERTS][2];
    int i, j, k;
    FILE *fp;
    char *t;
    double dist, mindist;
    int sawpoint = 0;
    if (argc != 2)
        servererr("Wrong number of arguments, client may not support ISMAP.");
    mapname=getenv("PATH_INFO");
    if((!mapname) || (!mapname[0]))
        servererr("No map name given. Please read the <A HREF=\"http://hoohoo.ncsa.uiuc.edu/docs/
setup/admin/Imagemap.html\">instructions</A>.<P>");
    mapname++;
    if(!(t = strchr(argv[1],',')))
        servererr("Your client doesn't support image mapping properly.");
    *t++ = '\0';
    testpoint[X] = (double) atoi(argv[1]);
    testpoint[Y] = (double) atoi(t);
    /* if the mapname contains a '/', it represents a unix path we get the translated path, and skip reading the
configuration file. */
    if (strchr(mapname,'/'))
      {
      strcpy(conf,getenv("PATH_TRANSLATED"));
      goto openconf;
      }
    if ((fp = fopen(CONF_FILE, "r")) == NULL)
        {
        sprintf(errstr, "Couldn't open configuration file: %s", CONF_FILE);
        servererr(errstr);
        }
    while(!(getline(input,MAXLINE,fp)))
      {
        char confname[MAXLINE];
        if((input[0] == '#') || (!input[0]))
            continue;
        for(i=0;isname(input[i]) && (input[i] != ':');i++)
```

```
            confname[i] = input[i];
        confname[i] = '\0';
        if(!strcmp(confname,mapname))
            goto found;
    }
/* if mapname was not found in the configuration file, it still might represent a file in the server root
directory we get the translated path, and check to see if a file of that name exists, jumping to the opening of the
map file if it does. */
    if(feof(fp))
     {
      struct stat sbuf;
      strcpy(conf,getenv("PATH_TRANSLATED"));
      if (!stat(conf,&sbuf) && ((sbuf.st_mode & S_IFMT) == S_IFREG))
      goto openconf;
      else
      servererr("Map not found in configuration file.");
      }
    found:
    fclose(fp);
    while(isspace(input[i]) || input[i] == ':') ++i;
    for(j=0;input[i] && isname(input[i]);++i,++j)
        conf[j] = input[i];
    conf[j] = '\0';
    openconf:
    if(!(fp=fopen(conf,"r")))
    {
     sprintf(errstr, "Couldn't open configuration file: %s", conf);
      servererr(errstr);
    }
    while(!(getline(input,MAXLINE,fp)))
        {
         char type[MAXLINE];
         char url[MAXLINE];
         char num[10];
         if((input[0] == '#') || (!input[0]))
             continue;
         type[0] = '\0';url[0] = '\0';
         for(i=0;isname(input[i]) && (input[i]);i++)
             type[i] = input[i];
```

```
type[i] = '\0';
while(isspace(input[i])) ++i;
for(j=0;input[i] && isname(input[i]);++i,++j)
      url[j] = input[i];
url[j] = '\0';
if(!strcmp(type,"default") && !sawpoint)
   {
      strcpy(def,url);
      continue;
   }
k=0;
while (input[i])
    {
      while (isspace(input[i]) || input[i] == ',')
            i++;
      j = 0;
      while (isdigit(input[i]))
            num[j++] = input[i++];
      num[j] = '\0';
      if (num[0] != '\0')
            pointarray[k][X] = (double) atoi(num);
      else
            break;
      while (isspace(input[i]) || input[i] == ',')
            i++;
      j = 0;
      while (isdigit(input[i]))
            num[j++] = input[i++];
      num[j] = '\0';
      if (num[0] != '\0')
            pointarray[k++][Y] = (double) atoi(num);
      else
        {
            fclose(fp);
            servererr("Missing y value.");
        }
}
pointarray[k][X] = -1;
if(!strcmp(type,"poly"))
```

```
        if(pointinpoly(testpoint,pointarray))
            sendmesg(url);
    if(!strcmp(type,"circle"))
        if(pointincircle(testpoint,pointarray))
            sendmesg(url);
    if(!strcmp(type,"rect"))
        if(pointinrect(testpoint,pointarray))
            sendmesg(url);
    if(!strcmp(type,"point"))
    {
    /* Don't need to take square root. */
    dist = (testpoint[X] - pointarray[0][X]) * (testpoint[X] - pointarray[0][X]) + (testpoint[Y] - poin
tarray[0][Y]) * (testpoint[Y] - pointarray[0][Y]);
    /* If this is the first point, or the nearest, set the default. */
    if ((! sawpoint) || (dist < mindist))
    {
      mindist = dist;
        strcpy(def,url);
      }
        sawpoint++;
        }
    }
    if(def[0])
        sendmesg(def);
    servererr("No default specified.");
}
sendmesg(char *url)
{
  if (strchr(url, ':'))     /* It is a full URL */
    printf("Location: ");
  else                      /* It is a virtual URL */
    printf("Location: http://%s:%s", getenv("SERVER_NAME"), getenv("SERVER_PORT"));
    printf("%s%c%c",url,10,10);
    printf("This document has moved <A HREF=\"%s\">here</A>%c",url,10);
    exit(1);
}
int pointinrect(double point[2], double coords[MAXVERTS][2])
{
    return ((point[X] >= coords[0][X] && point[X] <= coords[1][X]) && (point[Y] >= coords[0][Y] &&
```

```
point[Y] <= coords[1][Y]));
    }
    int pointincircle(double point[2], double coords[MAXVERTS][2])
    {
        int radius1, radius2;
        radius1 = ((coords[0][Y] - coords[1][Y]) * (coords[0][Y] -
                coords[1][Y])) + ((coords[0][X] - coords[1][X]) * (coords[0][X] -
                coords[1][X]));
        radius2 = ((coords[0][Y] - point[Y]) * (coords[0][Y] - point[Y])) +
                ((coords[0][X] - point[X]) * (coords[0][X] - point[X]));
        return (radius2 <= radius1);
    }
    int pointinpoly(double point[2], double pgon[MAXVERTS][2])
    {
        int i, numverts, inside_flag, xflag0;
        int crossings;
        double *p, *stop;
        double tx, ty, y;
        for (i = 0; pgon[i][X] != -1 && i < MAXVERTS; i++)
            ;
        numverts = i;
        crossings = 0;
        tx = point[X];
        ty = point[Y];
        y = pgon[numverts - 1][Y];
        p = (double *) pgon + 1;
        if ((y >= ty) != (*p >= ty))
            {
            if ((xflag0 = (pgon[numverts - 1][X] >= tx)) == (*(double *) pgon >= tx))
                {
                if (xflag0)
                  crossings++;
                }
                else {
                        crossings += (pgon[numverts - 1][X] - (y - ty) * (*(double *) pgon - pgon[numverts - 1][X]) /
(*p - y)) >= tx;
                        }
                }
                stop = pgon[numverts];
```

```
for (y = *p, p += 2; p < stop; y = *p, p += 2)
    {
      if (y >= ty)
      {
        while ((p < stop) && (*p >= ty))
            p += 2;
        if (p >= stop)
          break;
        if ((xflag0 = (*(p - 3) >= tx)) == (*(p - 1) >= tx))
        {
          if (xflag0)
            crossings++;
        }
        else {
          crossings += (*(p - 3) - (*(p - 2) - ty) * (*(p - 1) - *(p - 3)) / (*p - *(p - 2))) >= tx;
                    }
      }
      else {
          while ((p < stop) && (*p < ty))
              p += 2;
          if (p >= stop)
            break;
          if ((xflag0 = (*(p - 3) >= tx)) == (*(p - 1) >= tx))
          {
              if (xflag0)
                  crossings++;
              }
            else {
                  crossings += (*(p - 3) - (*(p - 2) - ty) * (*(p - 1) - *(p - 3)) / (*p - *(p - 2)))
>= tx;
                  }
        }
        }
        inside_flag = crossings & 0x01;
        return (inside_flag);
    }
servererr(char *msg)
    {
        printf("Content-type: text/html%c%c",10,10);
```

```
        printf("<title>Mapping Server Error</title>");
        printf("<h1>Mapping Server Error</h1>");
        printf("This server encountered an error:<p>");
        printf("%s", msg);
        exit(-1);
    }
int isname(char c)
    {
            return (!isspace);
    }
int getline(char *s, int n, FILE *f)
    {
        register int i=0;
        while(1)
         {
            s[i] = (char)fgetc;
            if(s[i] == CR)
               s[i] = fgetc;
            if((s[i] == 0x4) || (s[i] == LF) || (i == (n-1)))
             {
                s[i] = '\0';
                return (feof ? 1 : 0);
            }
            ++i;
        }
    }
```

第十二章　Windows CGI 规范

最初的 Web 服务器首先出现在 UNIX 系统平台上，作为外部扩展程序和 Web 服务器交互的接口—— 通用网关接口（CGI）的标准规范也是根据 UNIX 系统制定的。这种借助于环境变量和标准输入输出在 CGI 程序和 Web 服务器间实现通信的方式在 UNIX 系统下和 DOS 环境或非图形方式下可以工作得很好，但在 Windows 环境下就不行了。为了解决这个问题，WIN32 API 提供了一个处理标准输入输出文件句柄的函数。这就使得开发人员可以用一些 Windows 环境下常用的编程工具（如 VB 和 Delphi 等）开发标准的 CGI 程序，当然，这些 CGI 程序必须是 32 位的（本书前面所列举的 VB CGI 程序均为 32 位）。但在 16 位的 Windows 3.1 环境下还是不行。

为了解决这个问题，很多 Web 服务器（如 WebSite）利用 INI 文件和 CGI 程序通信，这也就是人们通常所说的 Windows CGI 规范。当使用 Windows CGI 规范时，Web 服务器不是向标准输出而是向一个 Windows INI 文件写信息，而 CGI 程序将从相应的 INI 文件而不是标准输入获取信息。

采用 Windows CGI 规范编写的 CGI 程序只能在特定的 Web 服务器下运行，并且由于 Windows CGI 规范本身的问题，这些 CGI 程序的执行效率没有标准 CGI 程序高。如果您是在 WIN32 环境下开发 CGI 程序，本书建议您利用 WIN32 API 函数编写标准的 CGI 程序。考虑到 Windows 3.1 下开发 CGI 程序的需要，本章将对 Windows CGI 规范进行介绍：

- Windows CGI 的工作原理；
- CGI 数据文件；
- Win-CGI 的输出。

12.1　Windows CGI 的工作原理

当支持 Windows CGI 规范的 Web 服务器所收到的客户端用户的请求是一个 Win-CGI 程序时，Web 服务器将把服务器和客户端的状态信息、客户端用户的输入信息写入一个 INI 文件中（CGI 数据文件）。如果客户端用户传送的信息比较多，Web 服务器会将这些信息放在几个文件中，这些文件的名称将在 CGI 数据文件中说明。Web 服务器在激活相应的 Win-CGI 程序时将把 CGI 数据文件的名称作为命令行参数传递给 Win-CGI 程序。通过调用相关的 WIN32 API 函数，Win-CGI 程序可以从 CGI 数据文件中获取运行环境的有关信息和客户端用户的输入。

在传递给 Win-CGI 程序的 CGI 数据文件中，服务器将通知 Win-CGI 程序写入程序输出结果的文件的路径和名称。在 Win-CGI 程序执行完毕以后，服务器将从该文件获取 Win-CGI 程序的运行结果并将结果传递给客户端的浏览器。图 12-1 说明了 Windows CGI 的工作原理。

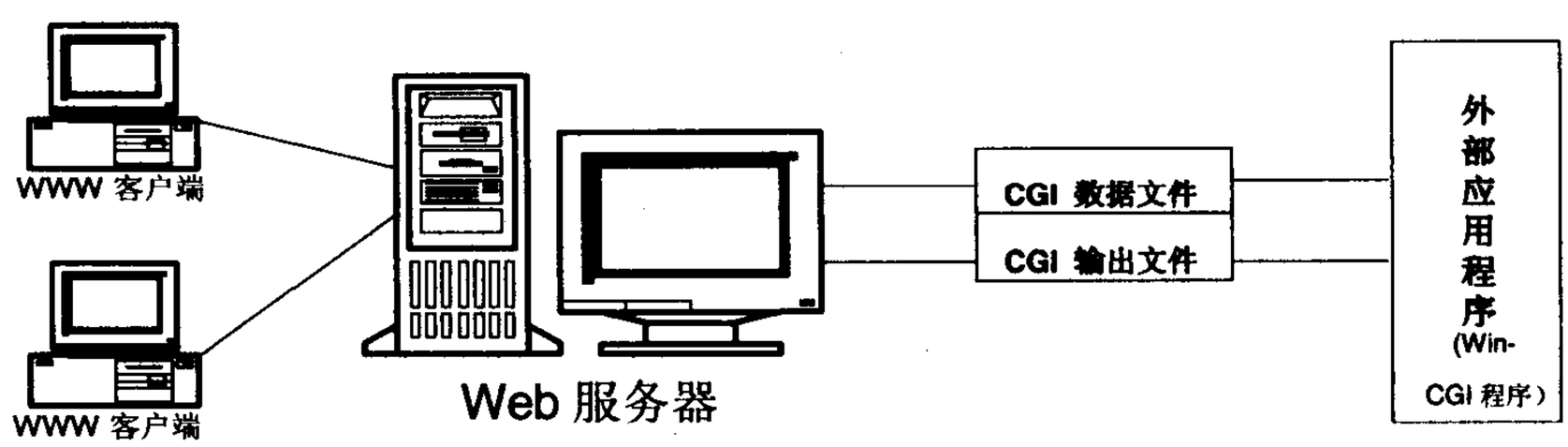

图 12-1　Windows CGI 的工作原理

由于 Win-CGI 程序是通过硬盘上的文件和 Web 服务器交换信息，而标准 CGI 程序则是通过内存缓冲，因此完成同样的任务，Win-CGI 程序没有标准 CGI 程序快。同时，由于是通过硬盘上的文件交换信息，Web 服务器只有在 Win-CGI 程序将所有的输出结果向文件写完后才能向客户端的浏览器传送程序的有关信息。这有时会带来一些问题。例如，在进行数据库查询时，Win-CGI 程序只有在所有的查询结果都写入文件后才能向客户端浏览器输出，这将使客户端的用户有一个等待时间，标准 CGI 程序则没有这个问题。Win-CGI 程序也不能利用本书第九章介绍的 Server Push 技术。

考虑到现在大多数 Web 服务器都工作在 32 位的操作系统下，您在编写 CGI 程序时应尽量利用 WIN32 API 函数处理标准输入输出而不是采用 Windows CGI 规范。

12.2　CGI 数据文件

支持 Windows CGI 规范的 Web 服务器通过 Windows INI 文件向 Win-CGI 程序传送信息。Win-CGI 程序可以通过调用 Windows API 函数很方便地从 CGI 数据文件中获取信息。以下就是一个 Win-CGI 程序执行时 Web 服务器所生成的 CGI 数据文件：

```
[CGI]
Request Protocol=HTTP/1.0
Request Method=POST
Executable Path=/win-cgi/test.exe
Document Root=e:/netscape/server/win-cgi
Referer=http://166.111.129.13/guest.html
User Agent=Mozilla/3.01Gold (Windows 95; I)
Content Type=application/x-www-form-urlencoded
Content Length=163
Content File=C:\ Windows \TEMP\1\1.con
Server Software=Netscape-FastTrack/2.0a
Server Name=albert.pim.tsinghua.edu.cn
Server Port=80
CGI Version=CGI/1.3a (Win)
Remote Address=166.111.129.13
HTTPS=off
```

```
[Accept]
image/gif=yes
image/x-xbitmap=yes
image/jpeg=yes
image/pjpeg=yes
*/*=yes
[System]
GMT Offset=28800
Debug Mode=Yes
Output File=C:\ Windows \TEMP\1\1.out
Content File=C:\ Windows \TEMP\1\1.con
[Extra Headers]
host=166.111.129.13
referer=http://166.111.129.13/guest.html
user-agent=Mozilla/3.01Gold (Windows 95; I)
content-length=163
content-type=application/x-www-form-urlencoded
connection=Keep-Alive
[Form Literal]
email=zhangys@hotmail.com
URL=166.111.129.13
[Form External]
name=C:\ Windows \TEMP\1\2.ext 6
from=C:\ Windows \TEMP\1\4.ext 4
SUBMIT=C:\ Windows \TEMP\1\5.ext 6
URL_Comment=C:\ Windows \TEMP\1\3.ext 14
```

通常，CGI 数据文件由以下八个部分组成：[CGI]、[Accept]、[System]、[Extra Headers]、[Form Literal]、[Form External]、[Form Huge]、[Form File]。各部分的内容如下：

- [CGI]

这部分以关键字字符串的形式提供了大多数 CGI 环境变量的值，如果某一关键字的值为空，该关键字将不在 CGI 数据文件中出现。这部分所拥有的关键字有：

（1）Request Protocol：本次连接所遵循的协议的名称和版本号，例如："HTTP/1.0"。

（2）Request Method：客户端请求所使用的方法，例如"POST"、"GET"等。

（3）Executable Path：Win-CGI 程序相对于 Web 服务器目录树根目录的路径和文件名。

（4）Document Root：Web 服务器目录树根目录"/"的物理路径名称，通过物理路径和相对路径，可以确定客户端用户所请求的 WWW 资源的准确位置。

（5）Logical Path：客户端用户所请求的 WWW 资源的逻辑路径名称，即 URL 字符串中的路径部分。

（6）Physical Path：客户端用户所请求的 WWW 资源的实际路径名称。

（7）Query String：URL 字符串中“?”后的内容。表单用“GET”方法传送的数据也放在 Query String 中。

（8）Request Range：客户端浏览器发出的 HTTP 请求头中表明请求头范围的字节数。

（9）Referer：调用该 Win-CGI 程序的 Web 网页的 URL。

（10）From：客户端用户的电子函件信箱。

（11）User Agent：客户端用户所使用的浏览器的类型和版本号。

（12）Content Type：浏览器所发送的数据的 MIME 类型。

（13）Content Length：浏览器所发送的数据的字节数。

（14）Server Software：Web 服务器软件的类型和版本号。

（15）Server Name：Web 服务器的 Internet 域名。

（16）Server Port：Web 服务器所使用的 TCP/IP 端口号。

（17）Server Admin：Web 服务器管理员（Webmaster）的电子函件信箱。

（18）CGI Version：Web 服务器所使用的通用网关接口标准的名称和版本号，例如：“CGI/1.3a (Win)”。

（19）Remote Host：客户端用户的主机的 Internet 域名。

（20）Remote Address：客户端用户的主机的 IP 地址。

（21）Authentication Method：对客户端用户进行身份认证的方法。

（22）Authenticated Username：客户端用户进行身份认证时所使用的用户名。

（23）Authenticated Password：客户端用户进行身份认证时所使用的口令。

从以上的说明可以看出，CGI 数据文件[CGI]部分所包含的关键字和标准 CGI 的环境变量的内容一样。

- [Accept]

本部分说明了客户端用户的浏览器所能接受的数据的 MIME 类型。

- [System]

CGI 数据文件的[System]部分说明了 Windows CGI 规范相对于标准 CGI 规范所特有的关键字。这些关键字有：

（1）GMT Offset：当地时间和 GMT 时间之间所相差的秒数，可以用来将当地时间转换为 GMT 时间。

（2）Debug Mode：如果关键字 Debug Mode 的值为 YES，Web 服务器在 Win-CGI 程序运行结束后将不删除程序运行时所生成的 CGI 数据文件和程序输出文件，这项功能主要用于调试程序。

（3）Output File：Win-CGI 程序所需要的输出文件名，Web 服务器将从该文件中获取程序的输出。

（4）Content File：如果浏览器向 Web 服务器发送了数据，服务器将把所接收到的没有经过 URI 解码的数据存入一个文件中。CGI 数据文件的关键字 Content File 说明了这个文件的路径和名称。

- [Extra Headers]

该部分列出了客户端浏览器发来的请求头（Request Header）中的一些附加信息。服务器在将这些信息写入 CGI 数据文件前将对它们进行 URI 解码。

● [Form Literal]

如果一个表单用 POST 方法向 Web 服务器请求一个 Win-CGI 程序，Web 服务器将对表单传送来的数据进行 URI 解码，并将分解开的表单数据以“Key = Value”的形式放在 CGI 数据文件的[Form Literal]部分。

● [Form External]

如果表单元素传送的数据超过 254 个字节，或者所传送的数据包含有 Web 系统保留字符和控制字符（汉字属于此类），Web 服务器将把经 URI 解码后的数据放入外部文件中，并在 CGI 数据文件的[Form External]部分说明文件的名称和长度。

● [Form Huge]

如果表单元素传送的数据超过 65,535 个字节，Web 服务器将不对该数据进行 URI 解码。不过服务器将以以下的形式向 Win-CGI 程序说明该表单元素所包含的数据在服务器所生成的 Content File 文件中的位置和长度：

key = offset　length

其中 key 为表单元素的 NAME 属性值，offset 是表单元素所传送的数据在关键字 Content File 所表示的文件中的起始位置，length 表示它的字节数。

● [Form File]

Win-CGI 程序的表单用来向 Web 服务器上载文件，CGI 数据文件的 [Form File] 部分将存储客户端上载的文件的有关信息。

12.3　Win-CGI 程序的输出

除了要借助于硬盘上的文件进行 I/O 缓冲，Win-CGI 程序的输出和标准 CGI 程序的输出基本上没有区别。

Win-CGI 程序的输出包括两部分：HTTP 响应头和程序的运行结果。在这两部分之间应该有一空行。Web 服务器从相关文件获取 Win-CGI 程序的输出后，将对 HTTP 响应头部分进行检查以添加必要的信息，对于程序的运行结果则不作任何的处理而直接传送给客户端的浏览器。

如果 Win-CGI 程序所生成的 HTTP 响应头的第一行（也就是 Win-CGI 程序输出文件的第一行）为字符串 "HTTP/1.0"，Web 服务器将认为程序本身所生成的 HTTP 响应头是完整的，服务器将不对程序生成的响应头进行检查，而是直接把它传送给客户端的浏览器。我们知道标准 CGI 规范是通过给程序加上前缀名“nph-”来区分这种情况的。

12.4　Win-CGI 编程

Win-CGI 程序首先要从命令行参数上获取 CGI 数据文件的名称。然后调用 WIN32 API 函数读取 CGI 数据文件的内容，在对相关数据做了处理之后再将输出结果存入 CGI 数据文件所规定的输出文件中。

Windows CGI 规范首先是由 WebSite 服务器的作者提出的。该服务器的作者同时还发布了一个 VB 共享模块 cgi32.bas。借助于这个共享模块，编程人员可以不考虑 Win-CGI 程序输入输出的细节而把精力集中在 Win-CGI 程序对用户的输入信息的处理上。

附录 A　MIME 类型

Internet 文档定义了许多 MIME，其中很多类型被 IANA（Internet Assigned Numbers Authority）采纳。所有的 MIME 类型可以分为以下七类：

- Text：用于描述文本数据的 MIME 类型，包括 plain text 和 HTML。
- Multipart：当所传送的数据由多种 MIME 类型组成时，即属此类。
- Message：用于说明各种消息的类型。
- Application：用于说明各种应用程序的文件类型。
- Image：用于说明图像文件的类型。
- Audio：用于说明声音文件的类型。
- Video：用于说明视频文件的类型。

每类又可分为多种子类型。类名称和子类型名称间用“/”连接起来。表 A-1 说明了 HTTP 所支持的 MIME 类型。

表 A-1　HTTP 所支持的 MIME 类型

MIME 类型	NCSA 文件扩展名	CERN 文件扩展名
application/acad		dwg, DWG
application/clariscad		CCAD
application/drafting		DRW
application/dxf		dxf, DXF
application/i-deas		unv, UNV
application/iges		igs, iges
application/octet-stream	bin	bin
application/oda	oda	oda
application/pdf	pdf	pdf
application/postscript		ai, PS, ps, eps
application/pro_eng		prt, PRT
application/rtf	rtf	rtf
application/set		set, SET
application/sla		stl, STL
application/solids		SOL
application/STEP		stp, STP, step, STEP
application/vda		vda, VDA
application/x-mif	mif	
application/x-csh	csh	csh
application/x-dvi TeX dvi dvi	dvi	dvi
dvi		

MIME 类型	NCSA 文件扩展名	CERN 文件扩展名
application/x-hdf	hdf	hdf
application/x-latex	latex	latex
application/x-netcdf	nc, cdf	nc, cdf
application/x-sh	sh	sh
application/x-tcl	tcl	tcl
application/x-tex	tex	tex
application/x-texinfo	texinfo, texi	texinfo, texi
application/x-troff	t, tr, roff	t, tr, roff
application/x-troff-man	man	man
application/x-troff-me	me	me
application/x-troff-ms	ms	ms
application/x-wais-source	src	src
application/zip	zip	
application/x-bcpio	bcpio	bcpio
application/x-cpio	cpio	cpio
application/x-gtar	gtar	gtar
application/x-shar	shar	shar
application/x-sv4cpio	sv4cpio	sv4cpio
application/x-sv4crc	sv4crc	sv4crc
application/x-tar	tar	tar
application/x-ustar	ustar	ustar
audio/basic	au, snd	au, snd
audio/x-aiff	aif, aiff, aifc	aif, aiff, aifc
audio/x-wav	wav	wav
image/gif	gif	gif
image/ief	ief	ief
image/jpeg	jpeg, jpg，jpe	jpg, JPG, JPE, jpe,JPEG, jpeg
image/tiff	tiff, tif	tiff, tif
image/x-cmu-raster	ras	ras
image/x-portable-anymap	pnm	pnm
image/x-portable-bitmap	pbm	pbm
image/x-portable-graymap	pgm	pgm
image/x-portable-pixmap	ppm	ppm
image/x-rgb	rgb	rgb
image/x-xbitmap	xbm	xbm
image/x-xpixmap	xpm	xpm

MIME 类型	NCSA 文件扩展名	CERN 文件扩展名
image/x-xwindowdump	xwd	xwd
multipart/x-zip		zip
multipart/x-gzip		gzip
text/html	html	html, htm
text/plain	txt	txt, g, h, C, cc, hh, m, f90
text/richtext	rtx	rtx
text/tab-separated-values	tsv	tsv
text/x-setext	etx	etx
video/mpeg	mpeg, mpg, mpe	MPG, mpg, MPE, mpe, MPEG, mpeg
video/quicktime	qt, mov	qt, mov
video/x-msvideo	avi	avi
video/x-sgi-movie	movie	movie

附录 B　服务器的状态码

当 Web 服务器收到了客户端用户发送来的请求以后，要回送一个 HTTP 响应头给客户端的游览器。在这个 HTTP 响应头中通常包含一些服务器的状态代码，以表明客户端的请求是合法还是非法，Web 服务器的运行是否正常等等。这些状态代码对于开发 CGI 程序和相关的应用软件非常有用。服务器的状态代码通常由三位整数组成。第一位整数表示状态代码的类型，其余两位表示状态代码的具体含义。

本附录的表 B-1 列出了服务器的状态代码的类型，表 B-2 说明了各种状态代码的含义。

表 B-1 服务器的状态代码的类型

状态代码	代码含义	代码说明
1xx	Informational	保留在将来使用
2xx	Success	客户端的请求被服务器成功的接收、理解和执行
3xx	Redirection	必须采取另一些操作以满足客户端的请求
4xx	Client Error	由于客户端的原因，服务器无法满足客户端的请求
5xx	Server Error	虽然客户端的请求合法，但由于服务器的原因无法满足客户端的请求

表 B-2 各种服务器的状态代码的类型

状态代码	代码含义	代码说明
201	Created	客户端的请求被服务器接收并被成功执行。在服务器发送给客户端的响应头中，包含一个新的 URL 地址，客户端的游览器在收到该响应头后将转向新的 URL
202	Accepted	服务器已经接收到客户端的请求并且正在处理，但还没有完成
203	Non-Authoritative Information	服务器返回的信息是从第三方获得的，不是服务器本身所具有的
204	No Content	客户端的请求被服务器完全满足，但服务器没有更新的信息返回给客户端
300	Multiple Choices	客户端所请求的资源可以在多个位置找到
301	Moved Permanently	客户端所请求的资源已被移向另一个位置，客户端可以根据服务器的要求返回一个或几个 URL 获得的资源
302	Moved Temporarily	客户端所请求的资源被暂时移向另一位置
303	See Other	客户端所请求的资源在另一位置，并且必须用 GET 方法获得

状态代码	代码含义	代码说明
304	Not Modified	如果客户端所请求的资源的最后更改日期在客户端用户所要求的日期之前，服务器将向客户端发送状态代码“304”，并不传送客户端所请求的资源
400	Bad Request	服务器无法理解客户端的请求
401	Unauthorized	客户端所请求的资源需要进行身份认证，服务器所回送的响应头必须包含进行身份认证所需的信息
402	Payment Required	本代码保留在将来使用
403	Forbidden	服务器理解客户端的请求，但由于某种原因服务器不能满足客户端的请求
404	Not Found	服务器没有找到客户端所请求的资源
405	Method Not Allowed	客户端请求头所说明的方法不适用于所请求的资源
406	None Acceptable	服务器发现了客户端所请求的资源，但客户端不支持该资源的 MIME 类型
407	Proxy Authentication Required	本代码保留在将来使用
408	Request Timeout	客户端没有在规定的时间内发出请求
409	Conflict	由于和资源的当前状态相冲突，服务器无法满足客户端的请求
410	Gone	客户端所请求的资源已不在服务器上
411	Authorization Refused	客户端用户的身份认证失败
500	Internal Server Error	服务器遇到了一个内部错误，无法满足客户端的合法请求
501	Not Implemented	服务器不支持用于满足客户端请求的方法
502	Bad Gateway	从网关返回的响应不合法
503	Service Unavailable	由于服务器暂时的过载或维护，服务器无法处理客户端的响应
504	Gateway Timeout	服务器没有在规定的时间内收到网关的响应

[General Information]
书名=CGI程序设计指南
作者=BEXP
页数=220
下载位置=http://book6.ssreader.com/diskjsj/js117/07/!00001.pdg