CHAPTER 1

Getting and Setting Up Your Scripts

OVERVIEW

Obtaining CGI scripts and configuring them to work under your Web server software are the first steps in getting a CGI application up and running. For many Web servers, these tasks are easy, but other Web servers make it difficult. For example, some servers have been configured to make it difficult to run scripts for security reasons. This chapter covers the basics of obtaining CGI scripts as well as the secrets of making them work in almost any Web server environment that allows them to run.

This book comes with a great many CGI scripts that we have written, and there are also a variety of scripts and programs on the Internet that you may want to use to complement the array of applications provided here. The guidelines in this chapter are useful for configuring these third-party scripts as well.

In addition, we will explain the major techniques used to unpack the scripts from the variety of formats you'll find on the Internet. Once unpacked, the scripts may need slight modifications to run on your Web server. We'll discuss techniques for doing this in reference to a variety of Web server platforms such as NCSA, Apache, and Netscape servers.

FINDING CGI APPLICATIONS ON THE WEB

Much of what is called "CGI development" does not involve the creation of new Web applications. Rather, Webmasters tend to use prewritten CGI scripts, customizing them for their own needs. Certainly, there will be unique projects that require new algorithms. But for most projects, a good CGI developer can draw from a huge arsenal of routines and predesigned applications that need only minor tweaking to use in a local Web server environment.

In the spirit of public domain, many authors have offered their work to the Web community so that developers need not continually reinvent the wheel and can move rapidly to bigger and better projects.

NOTE

Each archive or reference has its own unique form of intellectual property protection, so you should be careful about how you use the scripts you find. For example, although many scripts are in the public domain, others may be considered shareware. This usually means that you can try the script for free but should eventually pay for it if you like it. It is almost always appropriate, however, to notify the author that you are using one of his or her scripts even if it is in the public domain.

The following list outlines of some of the sites that we have found most when trying to find applications that have already been written to handle projects we've been asked to complete. When the site contains noteable features, we have attempted to note them in the description.

Selena Sol's Public Domain CGI Script Archive. This site contains all the scripts in this book in their most recent form. It also contains various other scripts and scripting information that didn't make it to press, but is worth your persisting.

http://www.eff.org/~erict/Scripts

Web Masters. The ultimate reference page for CGI developers.

http://kufacts.cc.ukans.edu/info/forms/forms-intro.html

Netamorphix CGI Resource Page. A fantastic site for basic information on CGI scripting as well as detailed and encompassing lists (Unlike most CGI resource lists, Netamorphix provides well-described list elements—short abstracts of each and every script on a monthly rotation) of shareware and freeware CGI scripts.

http://www.netamorphix.com/cgi.html

Pure CGI Scripts. A very good list of links to freeware scripts recommended by Pure Web.

http://www.netlink.co.uk/users/PureAmiga/pcgi/index.html

The Zone Coasters Scripts. A small, but well-written script archive containing an excellent survey script, a guestbook, a BBS, a chat script, and a homepage maker that were particularly interesting to us.

http://www.ocii.com/~bsowa/scripts.html

Shareware.com. You can grab almost any shareware around for NT, Mac, or UNIX Web servers, including Perl, gzip, and tar. Each of these is a good addition to any CGI support library, because nearly all the CGI scripts on the archives listed here are either tarred or gzipped.

http://www.shareware.com/

CGI Scripts To Go. A good collection of various CGIs in one place. The maintainer has taken a good deal of time to describe each script and to compare it to similar ones on the same topic.

http://www.virtualville.com/library/scripts.html

Rod Ellis's Script Archive. An excellent archive with examples of form processing, search engine, message board, HTML-CGI interfaces for building HTML pages on the fly, shopping carts, and authentication.

http://www.microlink.net/~gentle/Demo/

Storm's Public Domain CGI Archive. Storm is a great programmer who is highly respected. His scripts include Guestbook, Feedback, Random Image Gen, Addlink, Chat, Web Forum, and Random Link. He also continues to develop new programs.

http://www.he.net/~storm/html/scripts/index.html

Jon Weiderspan's CGI Applications Directory. An excellent launchpad! Jon has put together a fantastic list of links for almost any CGI application you want (both public domain listings and commercial listings).

http://www.comvista.com/net/www/cgi.html

Dale Bewley's Public Domain Tools Site. Contains a motley of excellent tools (especially graphics tools) for the Web administrator.

http://www.engr.iupui.edu/~dbewley/perl/

David Woolley's List of BBS Conferencing Systems. This is an excellent launch point for links to the various messaging systems on the Web.

http://freenet.msp.mn.us/people/drwool/webconf.html

New Breed's Software Page. Includes SSI, CGI libraries, index creator, **leave_link.cgi**, Guestbook, and Counter programs.

http://zippy.sonoma.edu/~kendrick/nbs/unix/www

Matt Wright's Script Archive. What can I say? Matt's Script Archive is one of the best out there and has been for a long time.

http://www.worldwidemart.com/scripts/

Nick Bicanic's Animation Page. This is an especially good site from which to learn animation. Nick is one of the original CGI animation pioneers and has continued to keep up with the latest methods and technologies.

http://bakmes.colorado.edu/~bicanic/altindex.html

Yahoo CGI Resources. No surprises here. Yahoo is one of the premier indexing sites on the Internet.

http://www.yahoo.com/Computers_and_Internet/Internet/

World_Wide_Web/CGI_Common_Gateway_Interface/

EIT Software Archive. Contains a bunch of cool Web administration tools.

http://www.eit.com/goodies/software/

The CGI Collection. A nice listing of various CGI resources, such as Mailform, W3OClock, Guestbook, PickMail, Logger, Access Counter, SWISH, and Register.

http://www.selah.net/cgi.html

Matt Johnson's CGI Page. This extensive CGI resource contains information about CGI scripts, Perl, and lots of sample code.

http://www.hamline.edu/personal/matjohns/webdev/cgi/

Web Developer's Virtual Library. A great many CGI links. It is an excellent resource for the C CGI programmer and even contains a couple of NT links.

http://www.stars.com/Vlib/Providers/CGI.html

Otis's CGI Library. A good collection of CGIs (multimailer, lottery, log analyzers, animation).

http://www.middlebury.edu/~otisg/Scripts/index.shtml

Chris Stephens's Shareware CGIs. A good collection, including the rare AppleScript CGIs and even VMS scripts.

http://www.cbil.vcu.edu:8080/cgi-bin/cgis.html

Grant Neufold's Random URL Shareware CGI. Random URL is a CGI application for WebSTAR or MacHTTP that randomly returns a URL to clients from a given text file. The program also works with any other Macintosh CGI sdoc–compatible Web server.

http://arpp1.carleton.ca/grant/mac/cgi/random.html

Christoffer "Toffe" Sundqvist's Web Developers' Bookmarks. A fairly good list of links. Not particularly well organized or presented, but it does list some resources not found elesewhere.

http://www.abo.fi/~csundqvi/cgi.htm

Mooncrow's CGI/PERL Source Page. A fantastic list of links that no CGIer should miss with topics ranging from magic cookies to supporting develper's applications to related programming resourdces (like sed resources).

http://www.seds.org/~smiley/cgiperl/cgi.htm

O'Reilly's WebSite Software Library. If you are an NT CGI programmer, this is a "must have" for your bookmarks. It is almost the best NT resource we have seen.

http://website.ora.com/software/

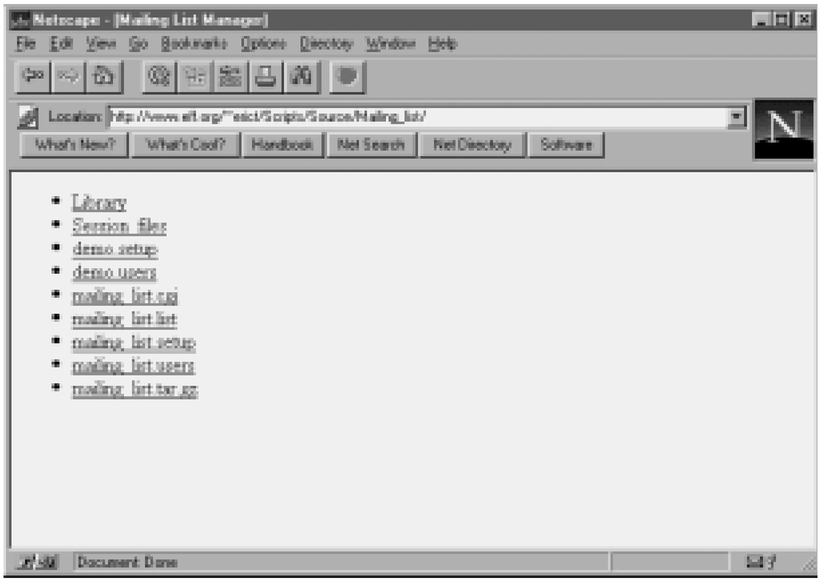
Dave Elis's CGI Page.

http://www.cyserv.com/pttong/cgi.html

Installing a Predesigned CGI Application

Once you locate a CGI application that meets your needs, you need to download it to your local CGI executable directory. Most of the time, scripts on the Internet are compacted into one file to make them easier and faster to download. Generally, all you need to do is to click on a hyperlink on the Web to download the file. You should also be aware that some sites do not archive applications in a neatly packed single file. You may need to download multiple files to get the entire CGI application. Figure 1.1 shows a listing for a mailing list manager program on Selena Sol's Archive (`http://www.eff.org/~erict/Scripts/`).

Notice the link for **mailing_list_tar.gz**. This is the type of file you will want to look for. The **.gz** extension lets us know that this file has been compressed by gnuzip. (Other extensions may include **.zip** [Windows] or **.sit** [Mac]). The **.tar** extension means that the file contains an archive of files archived using tar, a utility that compacts many files into one. So, in this example, all the files related to the mailing_list_.cgi CGI program are contained in the **mailing_list_tar.gz** file. To get the mailing list archive, you transfer one file to your local machine and then expand all the files locally into the predesignated directory structures.


Netscape - [Mailing List Manager]
File Edit View Go Bookmarks Options Directory Window Help
Location:
What's New? What's Cool? Handbook Net Search Net Directory Software
Document: Done

http://www.eff.org/~erict/Scripts/Source/Mailing_list/

- Library
- Session_files
- demo.setup
- demo.users
- mailing_list.cgi
- mailing_list.list
- mailing_list.setup
- mailing_list.users
- mailing_list.tar.gz

Figure 1.1 *Example of downloading an application.*

Archiving and Compressing Files

There is an important distinction between archive utilities and compression utilities. *Archiving* is a process that combines many files and directories into one file. *Compressing* makes a file (or files) smaller so that less time is needed to download it. Because tar, gzip, and compress are by far the most common archive and compression utilities used by CGI developers—and because they are generally the least understood—we will cover them here in greater detail.

NOTE

The following utilities are typically distributed with most flavors of Unix. If your Unix system does not have the following utilities or you are using another operating system such as Windows NT http://www.shareware.com is an excellent resource for downloading these utilities.

Gzip

According to the man pages for gzip, "Gzip reduces the size of the named file using Lempel-Ziv coding." Thus, gzip compresses a large file into a file that is much smaller. The benefit of compression is that when you're downloading the file to your local server, it will take less time because there is less data in the file to transfer. Once you download the **.gz** file to your local directory, the first thing you need to do is to decompress it. This process is easy and uses a sister program called gunzip.

Gunzip has many options and can be made to do complex things. If you are interested in complex management of gzip files, we recommend that you read the man pages on gunzip. However, for most CGI scripts that are downloaded in the gnuzip format, it will be enough to type the following on the command line:

```
gunzip filename.gz
```

In the case of our mailing list script, type the following:

```
gunzip mailing_list.tar.gz
```

Gunzip will decompress the program and remove the **.gz** extension. If you ran this step on the **mailing_list_tar.gz** file and pulled up a directory listing, you would see the file **mailing_list.tar** in its place. The library would be in the tar format, which we discuss next.

Tar

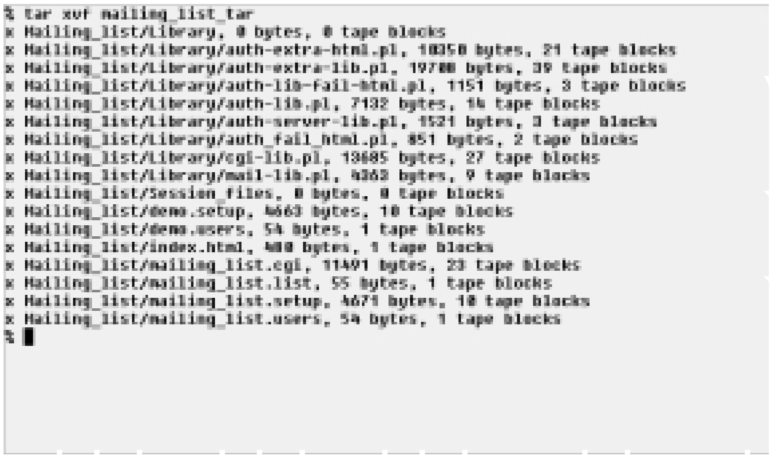
Tar is a UNIX command that allows you to create a single archive file containing many files. Such archiving allows you to maintain directory relationships and facilitates the transfer of complex programs that have many separate but integrated parts whose relationships must be preserved. Like gzip and gunzip, tar has a motley of options that allow you to do archiving and unarchiving in many ways. But for the purpose of untarring CGI files, the commands are fairly simple. At the command line, simply type

```
tar xvf filename.tar
```

Or, in the case of our mailing list file, type

```
tar xvf mailing_list.tar
```

Tar will go through the archive file and separate each individual directory and file, expanding them into their appropriate places underneath the current directory. Figure 1.2 shows the output of the above command. The “xvf” letters in the tar command are parameters that tell the program to extract the files and directories out ot the archive. Specifically, “x” tells tar extract; “w” tells tar to output information about the status of the extraction, and “f’ informs tar to use the “.tar” filename as the source of the files to be extracted. Tar is actually short for [T}ape {AR}chive and by default archives to a tape drive.



```
% tar xvf mailing_list_tar
x Mailing_list/Library, 0 bytes, 0 tape blocks
x Mailing_list/Library/auth-extra-html.pl, 10350 bytes, 21 tape blocks
x Mailing_list/Library/auth-extra-lib.pl, 19700 bytes, 39 tape blocks
x Mailing_list/Library/auth-lib-fail-html.pl, 1151 bytes, 3 tape blocks
x Mailing_list/Library/auth-lib.pl, 7132 bytes, 14 tape blocks
x Mailing_list/Library/auth-server-lib.pl, 1521 bytes, 3 tape blocks
x Mailing_list/Library/auth_fail_html.pl, 851 bytes, 2 tape blocks
x Mailing_list/Library/cgi-lib.pl, 13605 bytes, 27 tape blocks
x Mailing_list/Library/mail-lib.pl, 4362 bytes, 9 tape blocks
x Mailing_list/Session_files, 0 bytes, 0 tape blocks
x Mailing_list/demo.setup, 4663 bytes, 10 tape blocks
x Mailing_list/demo.users, 54 bytes, 1 tape blocks
x Mailing_list/index.html, 400 bytes, 1 tape blocks
x Mailing_list/mailing_list.cgi, 11491 bytes, 23 tape blocks
x Mailing_list/mailing_list.list, 55 bytes, 1 tape blocks
x Mailing_list/mailing_list.setup, 4671 bytes, 10 tape blocks
x Mailing_list/mailing_list.users, 54 bytes, 1 tape blocks
% █
```

Figure 1.2 *Untarring* ***mailing_list.tar***.

WARNING

Sometimes programmers forget to untar an archive starting at the directory above where the files are located. This is a problem, because, when the tar file expands, all the individual files will end up in the current subdirectory. If you are in a subdirectory where you have other files, they can make your work area messy. It is always best to create a subdirectory and then untar the files in that subdirectory. Later, you can move the previously archived files when you know the directory and file structures.

Compress

On occasion, some files on the Internet will be compressed using the UNIX `compress` command. These files can be recognized easily by the **.Z** extension. To decompress these files, merely use the `uncompress` command:

```
uncompress compressedfile.Z
```

The file will now be uncompressed and ready for further processing.

Setting Permissions

Copying a series of scripts into directories is only one part of the equation of installing or programming CGI scripts and making them run. Frequently, the Web server needs to be given special permission to run your scripts and have the scripts perform their job with the appropriate “rights.”

The cardinal rule of setting up Web server software is that the server should be given only minimal capabilities. This rules out the Web server running as the ROOT user (Super user on UNIX). More often than not, it means that the Web server is running as a user—the user “nobody”—and has no rights to do anything significant. By default, “nobody” usually has no permission to read any files in directories that you create. However, when you download scripts, you need to make it so that the scripts can be read and executed by the Web server software. In other words, “nobody” must be able to get to the files.

In UNIX, the magic command for performing this task is `chmod`. The directory structure of a message board CGI script is a good example of how you should use the `chmod` command. For this example, we will assume that you are sharing a server on an Internet service provider, which is typically the most restrictive situation in terms of your security options. For this example, all your CGI scripts are located in a directory called **Bbs**. The messages themselves need to be located in a directory called **Messages** under the **Bbs** directory. The example assumes we are in the directory above the one where the CGI scripts reside when we execute the `chmod` command.

Four different sets of permissions need to be granted for the user "nobody" to run this CGI script. First, the **Bbs** directory must be both readable and executable by the world. We need to read the entries in the directory, and it must be executable so that we can go into the directory and open the files in it:

```
chmod 755 Bbs
```

The 755 tells `chmod` to make the **Bbs** directory readable, writable, and executable by the owner of the file (you) while making it only readable and executable by the world and the group you are in.

How did we come up with the number? Files in UNIX have three types of permissions: user (the owner of the file), group (the security group you are in), and other (for the world to see). Each digit in the number corresponds to one of these categories. The first digit is user, the second digit is group, and the final digit is other. Thus, in the example, 7=USER, 5=GROUP, and 5=OTHER.

The value of the digit determines the permissions granted to that area. Permissions consist of three numbers: 4 for read, 2 for write, and 1 for execute access. By adding these numbers together, you form the permissions that make up one digit. For example, 4 + 2 + 1 = 7, which grants read, write, and execute permissions. 4 + 1 = 5, which grants only read and execute permissions. Thus, 755 grants 7 (read, write, execute) to the owner of the file, and 5 (read and execute) to the group the file is in, and 5 to the world.

The files within the **Bbs** directory need to be readable and executable, because the Web server will be running the CGI scripts in it.

```
chmod 755 Bbs/*
```

This command operates the same as the previous one except that it changes all the files (* pattern matches everything) inside the **Bbs** directory to be readable and executable by everyone. These scripts should not be writable! Nor should the **Bbs** directory be writable, because that would allow other users on the system to place scripts there. The **Messages** directory is another issue. Because it contains messages generated by the users, the **Messages** directory must be made writable so that the CGI script can write to the directory. In addition, the files generated in that directory should also be writable in case the **Bbs** script needs to access them. To change the directory so that it is also writable, use the following command:

```
chmod 777 Bbs/Messages
```

This command will change the **Messages** directory under the **Bbs** directory to be readable, writable, and executable by the world.

WARNING

You may be tempted to simply use `chmod 777` on all the files and directories, because it ensures that the Web server can do anything with the files. However, it is strongly advised that you do not leave the files in this state. It is considered a significant security risk to leave your scripts open to changes by the Web server instead of being read-only. Anyone on the server could use a rogue CGI script to write over your scripts and make them do something different. There is still a risk involved in making the Messages directory writable, but if someone messes with your area, the intruder will destroy only a bit of data and not your main programs. It is OK to set the scripts to 777 if you are trouble-shooting a problem and want to rule out permissions entirely, but do not leave the scripts like this.

On another security note, if you are concerned with the security of your data such as running a shopping cart, please do not use a shared server where other people can write CGI scripts using the same Web server configuration. It is much better to use your own server software or purchase space on a *virtual server*, which can be shared but is set up so that each user's scripts are shielded from those of other users.

Server-Based CGI Execution Issues

Assuming your Web server is set up to run CGI scripts, you may find that there are various ways the server can be set up to recognize that a script is an executable program. There are two common ways that Web servers typically handle the recognition of CGI scripts; each has its benefits and drawbacks.

First, a specific directory or directories are earmarked as containing only executable programs. Any file in such a directory will be run as a program no matter what its extension. The second way a server can be set up to recognize CGI scripts is by file extension. In this case, you indicate CGI scripts by adding a **.cgi** extension to the end of the filename. Thus, if you called a script, **bbs_forum.pl**, you would rename it to **bbs_forum.cgi** to have it recognized as a runnable CGI script by the Web server.

Typically, the first method is an option only if you have your own Web server. Internet service providers find it time-consuming to set up a specific CGI binary executable (**cgi-bin**) directory for each user on the system. The advantage of this method is that it is relatively straightforward. HTML files belong in a documents directory area, and scripts belong in a scripts directory. Keep in mind, though, that the Web server cannot read HTML files inside a script directory. If a CGI script relies on an HTML file for something such as form input, you'll need to place that HTML file in a document directory.

When you belong to an Internet service provider, the likelihood is high that you are sharing the Web server with other people whom you do not know. ISPs try to be very careful about the sorts of permissions they allow a Web server to have on a machine. On the other hand, restricting the access of the Web server greatly reduces your file options: how to name files, where to put them, how to run them, and what they can do. The most common way around this problem is for an ISP to require users of a shared server to use the **.cgi** extension for all CGI scripts. One advantage of this technique is that HTML files that belong to the CGI application can reside in the same directory with the CGI files. This arrangement makes it easier to maintain the application, because you do not have to go to two different directories to maintain the files.

All the scripts in this book end in **.cgi** by default. This is the most flexible way to initially name your scripts, because if they are placed on a server that recognizes the **.cgi** suffix, they will run. As a bonus, if they are placed in a directory using the CGI-specified directory method, they will also run.

Some ISPs may not give their users access to run CGI scripts using the **.cgi** extension by default. For example, they may provide a directory such as **public_html** inside your home directory to hold HTML files; if you put a **.cgi** file there, it will not run. If this happens, there are a couple of tricks you may be able to pull to make your ISP's Web server run the scripts. If your ISP is using the NCSA or Apache Web server, there is a good possibility that it has been configured to allow a user to "override" Web server options by placing an **.htaccess** file in the directory where you wish to run CGI scripts. The "." before **htaccess** is not a typo. It is important to include. In the directory you wish to run CGI scripts in, create an **.htaccess** file containing the following line:

```
Options ExecCGI
```

This single line is all you need! Hopefully, it will get you up and running with CGI scripts. However, keep in mind that **.htaccess** is the default filename for overriding Web server configuration options. Your Web server may be set up differently.

You can spend a great deal of time going through these steps, but the easiest way to find out how your server is configured is to ask your Internet service provider.

Finding System Files

On UNIX, CGI scripts written in Perl usually contain a magic line that expects to find Perl in a particular directory. In addition, some CGI applications may expect external programs such as sendmail to be located in a certain location on the server. Most of the time, the references to these locations will be correct, because most servers are set up in a standard way. However, you may run across a situation in which the programs used by the scripts are not where they are supposed to be. One of the last steps in setting up scripts is to figure out the location of these files so that the scripts can be changed to reflect the new file locations.

The classic example of a reference to an absolute path in a CGI script is the first line of the Perl code:

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

This line instructs the server to run the ensuing script through the Perl interpreter and indicates where to find the Perl interpreter. The Perl interpreter is a program that reads your script and translates it into a form that your server can run. In the preceding example, the server will know that it can find the Perl interpreter in the directory **/usr/local/bin**. Although many servers may contain Perl in **/usr/local/bin**, others may have installed it in other areas, such as **/usr/bin** or **/opt/bin**.

If Perl is not in **/usr/local/bin**, your first bit of customizing is to find out where your local Perl is and change this line to reference the correct location. There are several ways of finding files on your system. Not all of them work on every server, so be prepared to experiment with the following techniques.

The first command to try is **which**. At the command prompt of your UNIX server, you type **which perl** and receive the following reply:

```
$ which perl
/bin/perl
```

In other words, Perl is located in **/bin** on this system. Thus, you will need to change the first line of your script to

```
#!/bin/perl
```

If that does not work, try the **whereis** command. This command could give you the following output:

```
$ whereis perl
perl: /usr/bin/perl /usr/local/bin/perl /usr/local/bin/perl4.036
/usr/local/bin/
perl5.002
```

This output tells us a little bit more information than we get with the **which** command. It gives us information about all the Perl interpreters contained in the system. In other words, there are several versions of Perl installed on this server, including 4.036 and 5.002! Because there are several versions, you could choose whichever one you wish to reference in the CGI script.

If those two commands fail, the next step is to try **whence**. This command is more specific to the Korn shell, so you should first change to the Korn shell if you can: Type **ksh** at the command line of your UNIX system. Now you should be free to use **whence** and get output that looks similar to the following:

```
$ ksh
$ whence perl
/usr/local/bin/perl
```

If all else fails, you might try the **find** command, but that is really pulling out all the stops when a simple E-mail to your system administrator might suffice. The syntax for **find** is as follows:

```
find / -name perl
```

In a few other cases you may need to find other programs on your system. Some CGI applications make calls to external system programs other than Perl. For example, many mail-enabled CGI scripts rely on the UNIX sendmail program to send E-mail. Usually, this application is located in **/usr/lib**. On some UNIX systems it may be located elsewhere. In such instances, you may have to define the directory paths for those programs in addition to Perl. All the commands we've discussed will help you find the location of other programs.

You may also run into a problem that is specific to some "virtual" Web servers on shared systems: What you see as your root directory when you log on may not be what the Web server sees. For example, your documents may be located in **/home/smithj/www/cgi-bin**, but a virtual Web server may see documents as being located in **/www/cgi-bin**. Some ISPs configure it that way, because it enhances security by making sure that the Web server you are using cannot touch the files created by any other Web server instance on a shared machine. CGI scripts on Virtual Web servers are more time-consuming to set up and are specific to each customer; if you are using such a server, make sure you are given instructions about how to access external programs such as Perl or sendmail.

CROSS-PLATFORM CONSIDERATIONS

A few years ago, UNIX was synonymous with the Internet. Almost anyone who wanted to get on the Internet did it by using a UNIX server. More recently, however, other types of operating systems have expanded their list of features to include Internet-based services. Shareware as well as commercial Web server products are available for Apple Macintosh, Windows 95, Windows NT, and a variety of other operating systems.

Unfortunately, Perl takes on slightly different forms depending on the operating system in use. Perl for UNIX has a number of features that are difficult or impossible to implement on other operating system architectures. This incompatibility can be a major problem, because Perl has become the de facto standard language for writing CGI scripts on the Internet.

Most of the scripts in this book have been written with cross-platform compatibility in mind. However, you may still need to make some changes to get the scripts to run. In addition, not all the scripts you download from the Internet are written to be so cross-platform–friendly. The guidelines that follow will help you to set up the scripts in this book and other Perl scripts downloaded from the Internet on your non-UNIX server. Many of these guidelines are programming-related and are meant to help you modify a Perl program if you find that it has not been written in a cross-platform–friendly manner.

We will focus on Windows NT versus UNIX for developing CGI scripts. Windows NT appears to be an up-and-coming contender for heavy network computing. In addition, it is also sufficiently different from UNIX that most of the guidelines will help you write scripts that should also work on other platforms such as Macintosh and Windows 95.

Batch File Encapsulation

Some Windows NT Web servers cannot recognize the **.cgi** or **.pl** file extensions. Nor do they look at the first line of the file (as UNIX does) to see which program should run the script using the magic `#!/usr/local/bin/perl` designation. The easiest way to get around this problem is to encapsulate the script in a batch file. Simply create an MS-DOS **.bat** file with roughly the same CGI script name and call the original CGI script with Perl. For example, the BBS script discussed later in this book has been designed to be compatible with NT Perl 5, but the script name is **bbs_forum.cgi**. To get this script to run on a server that cannot recognize **.cgi**, create a **bbs_forum.bat** file in the same directory as **bbs_forum.cgi** and place in it the following lines:

```
@echo off
perl bbs_forum.cgi
```

That's it! Now, whenever you refer to the script, use **bbs_forum.bat** file instead of **bbs_forum.cgi**. All that remains is to make sure that any references inside the program to **bbs_forum.cgi** are changed to **bbs_forum.bat**. For the BBS script, there is a variable in the **bbs.setup** file that allows you to specify the name to refer to the script by. In other programs, however, you may need to be careful to edit the scripts to reference the new batch filename.

WARNING

Encapsulating your scripts inside batch files can have security implications. To get a full appreciation of the problems that may arise, you should read the CGI programming security FAQ located at the following URL: `http://www-genome.wi.mit.edu/WWW/faqs/www-security-faq.html`

Shell Commands

Don't use shell commands. This rule is easier said than done, though, and, in some cases, using shell commands may be unavoidable. There may be a UNIX command, such as the **sendmail** command, that has no equivalent in the other operating system. On the other hand, it is easy to fall into the trap of using easy UNIX commands to substitute for programming the same thing in Perl.

For example, the UNIX **date** command is called frequently from some CGI scripts. This practice is unnecessary. Perl has a good facility for returning the current date and time. Simply use the **localtime** command:

```
($sec, $min, $hour, $mday, $mon,
     $year, $wday, $yday, $isdst)
         = localtime(time);
$mon++;
$date = "$mon/$mday/$year";
```

instead of using the UNIX-specific command:

```
$date = `date`;
```

The UNIX-specific command is short and easy, but if your script is already 100 or more lines long, a few more lines to get today's date won't hurt you—and the script will be cross-platform–friendly. In addition, it's bad practice to call external commands; spawning extra programs and processes takes computing time on busy Web servers.

UNIX Function Calls

Some functions listed in the Perl manual pass their parameters to a low-level UNIX system call. Frequently, these commands do not translate well to other systems, because UNIX calls are not implemented in every operating system. Two especially problematic functions tend to be used often in CGI programs: `crypt` and `flock`.

`Crypt` takes a password and encrypts it the same way passwords are encrypted in the UNIX **/etc/passwd** or **/etc/shadow** file. Windows NT encrypts passwords using a proprietary algorithm. Thus, Windows NT does not support the `crypt` function call; as of this writing, Perl for NT does not support it, either. For scripts that use the `crypt` function, such as the authentication library discussed later in this book, it is considered a good practice to have a flag variable that can be set so that encryption is turned off for systems that do not support `crypt`.

`Flock` is a file-locking function call. Although Windows NT Perl has recently started supporting `flock`, it is important to minimize your use of this function, because not even all UNIX systems support it. The best thing to do, if you wish to use file locking in your CGI scripts, is to implement a simple file-locking mechanism of your own. **CGI-LIB.SOL**, discussed later in this book, implements such a routine.

NOTE

It is better to use `flock` than a hand-rolled file-locking routine, especially if you have heavy server activity. But be aware of the trade-off of incompatibility with other servers and platforms.

Directory Referencing

UNIX separates directory names with a forward slash (/), such as **/usr/local/bin**. Windows NT separates directory names with a backslash (\), such as **\winnt351\system**. Because Perl for NT translates the UNIX convention into the Windows NT convention automatically, you should code your scripts on Windows NT to reference directories using a / slash instead of a \ slash to be compatible with UNIX.

E-Mail on NT

Windows NT before version 4.0 does not have the built-in capability to send Internet mail, which many Web applications use. For example, when someone submits an order to the shopping cart, frequently the order is E-mailed to the administrator of the shopping area to complete the order. The chapter on **MAIL-LIB.PL** includes a discussion of a Perl script that sends E-mail on Windows NT.

CONCLUSION

After reading this chapter, you should have a good idea of the general guidelines for downloading and running CGI scripts in a variety of environments. Merely downloading scripts is easy. Configuring them to run on a Web server can be more of a challenge, but by following the steps of dearchiving the applications, setting the appropriate permissions, and making sure the files are in a CGI executable directory or end with **.cgi**, you should be able to get the scripts up and running in no time. Similarly, you need to resolve cross-platform issues, such as running scripts for UNIX on a Windows NT Web server. By going through the steps outlined here, you should also be able to get a script running on a non-UNIX server.