

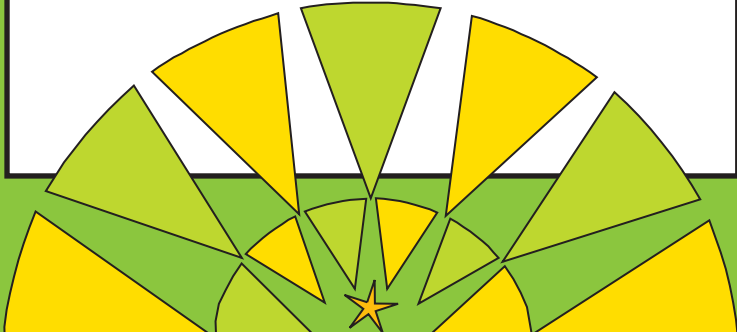


CHAPTER TWO

GETTING STARTED

Once you have found an Internet service provider (ISP) and have answered the questions listed in Chapter One (“Introduction”), you are ready to start developing your first CGI/Perl program. This chapter describes the steps needed to get started on programming CGI/Perl. By the time you reach the end of the chapter, you will have entered and run two small programs that will help you understand the CGI/Perl development process and the procedure for generating HTML documents from CGI/Perl.

Chapter Objectives

- ★ Set up an ISP account, if necessary
 - ★ Describe the steps needed to develop Perl programs using your ISP account
 - ★ Describe the steps needed to change Perl programs
 - ★ Learn how to create and run basic CGI/Perl programs
- 

🌀 Is Your Web Server Account Ready to Go?

Whether you are running your own Web server or using an ISP's Web server, you need to figure out certain things before you can start writing your first program. In particular, you may need to do some administrative account configuration or install the FTP or Telnet software on your PC. Whether you need to take these steps depends on how your Web server is configured and whether you already have the FTP and Telnet software you need.

These administrative tasks are described below:

- ★ *Getting the appropriate FTP or Telnet software.* If you are using FTP or Telnet to access your Web server, you may need to install such software on your PC.
- ★ *Connecting to the Web server.* If you are using FTP or Telnet to access your Web server, you need to know how to connect to the Web server.
- ★ *Setting up your directories.* Your Web server account may not be set up correctly, so it's a good idea to check out the directories there.
- ★ *Getting the location of the Perl interpreter.* You will need the location of Perl on your Web server to run CGI/Perl applications.

Each of these administrative steps is described in detail next.

★ **SHORTCUT** Working with Your Own Web Server

For the purposes of this chapter, the steps necessary to use Telnet and FTP to connect to a UNIX Web server are shown. If you are running a different Web server, some exceptions to these steps are noted. You need to consult your Web Server documentation for details on how to work with it.

★ **WARNING** SSH instead of Telnet

Some ISPs do not allow Telnet access to their sites, but instead permit you to use a special secure connection program called SSH to log in. I use an SSH Telnet program with my Web server called *Absolute Telnet*. Once configured properly, it operates like most every other Telnet program.

Do You Have the Necessary Telnet or FTP Software?

If you are running Windows on a PC (and your ISP permits Telnet access), you can use Microsoft's Telnet for Windows to access your Web server. You can also find many Telnet programs for purchase or for free on the Internet. (Try searching for *Telnet* on your favorite Internet search site.)

If you plan to use FTP, you may need to obtain and install an FTP program on your PC. Several FTP programs are also available for free and for purchase over the Internet. The examples in this book use an FTP program called *WS_FTP Pro*. A popular "lite" version of this program is also available for free over the Internet.

Getting Connected with Telnet and FTP

As indicated earlier, you may be using either Telnet or FTP to access your Web server. The following sections explain how to connect to a Web server, first using Telnet and then using FTP.

Connecting with Telnet

To connect to your Web server with Telnet, follow these steps:

1. *Connect to the Internet.* If you are not already connected, you need to connect to the Internet.
2. *Start Telnet.* On a Windows PC you can often start Microsoft Telnet for Windows by clicking **Start**, **Run**, and entering `telnet`.
3. *Connect to your Web server with Telnet.* If everything works correctly, you will see an initial Telnet screen. Use that screen to connect to your Web server. In Figure 2.1, Absolute Telnet is used to connect to a Web server called `condor.depaul.edu`.



Figure 2.1 Connecting to a Web Server via Telnet

4. *Log into the server.* If a connection is established, then you will see an initial login prompt asking for your user ID and password (see the top of Figure 2.2). Use that screen to enter the user ID and password that you obtained from your ISP. Be careful when entering these, as UNIX is case sensitive. For example, the user ID *dlash* is not the same as *Dlash*.

If you have correctly entered your user ID and password, the bottom of Figure 2.2 shows the output you will receive. If you wish to end your session enter the command `logout`.

Connecting with FTP

If you are using FTP to connect to your Web server, then follow these steps to establish an FTP connection:

1. *Connect to the Internet.* If you are not already connected, you need to access the Internet.

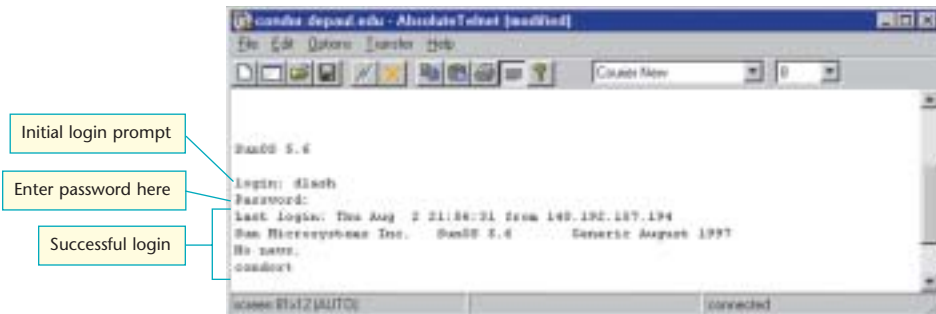


Figure 2.2 A Successful Telnet Login

2. **Start FTP.** Start your FTP software. This software may be installed in a variety of places. You can often find it on a Windows PC by clicking **Start**, **Find**, **Files** or **Folders**. Enter `ftp` in the box labeled **Named** and click **Find Now**.
3. **Connect to your Web server with FTP.** Use the initial FTP screen to connect to your Web server. Figure 2.3 shows an initial connection screen. On this screen I entered my host name (`www.aw.com`), my userid (`perlpgm`), and a password (asterisks).

Once you are connected via FTP to your Web server, you will see a screen similar to Figure 2.4. The files and directories on your PC are shown on the left side of this screen. The files and directories on your Web server are shown on the right side.



Figure 2.3 An Initial FTP Login Screen

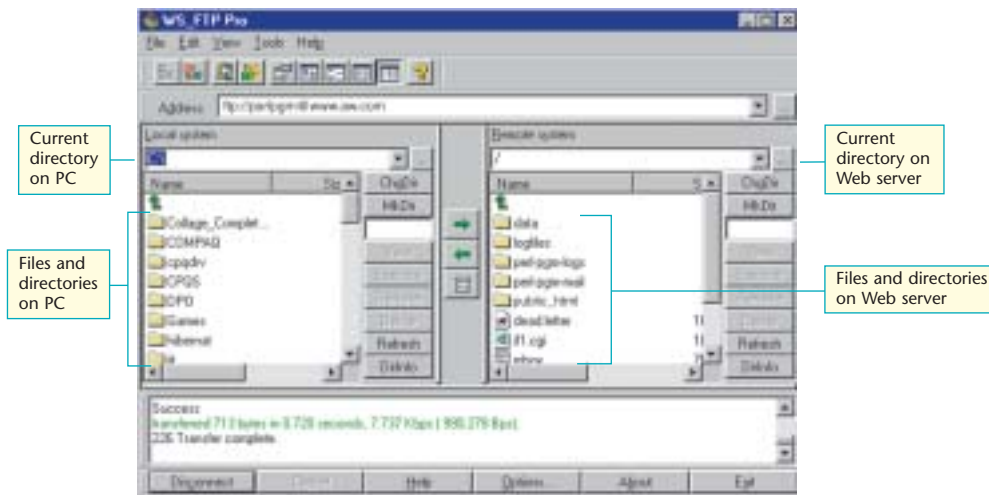


Figure 2.4 Initial FTP Screen after Logging In via FTP

Are the Required Directories Set Up Correctly?

After connecting to your Web server, you should verify that you have the directories required to store your Perl programs on the Web server. Your Web server probably has certain rules governing where your HTML pages and Perl programs must reside. For example, HTML pages are often kept in a directory called `public_html` and CGI/Perl programs in a directory called `cgi-bin`. However, many other directory name configurations are possible.

It makes good sense for you to log into your Web server and verify that the required directories exist. The remainder of this section first describes some elements of the UNIX file system and then describes some ways to navigate a UNIX file system with Telnet and FTP.

A Little about a UNIX Web Server's Directories and Files

Most UNIX Web server systems enable multiple users to store files on them. When your user account is initially created on the UNIX system, a **home** directory is also created for you in which you can store your files. All files and directories you store will be stored under your home directory. My home directory on my ISP's UNIX Web server is `/home/perlpgm`. (See Figure 2.5.) Also, in Figure 2.5, you can see that `jsmith` has a home directory at `/home/jsmith`. Other directories, such as `/bin`, `/usr`, and `/etc` are UNIX system directories that hold application programs (such as the Perl interpreter program) and system programs and files needed to run the UNIX

system and Web server software.

★ TIP Home Directories on a Web Server

Most UNIX Web servers are configured to set your initial directory to your home directory when you successfully connect with either Telnet or FTP.

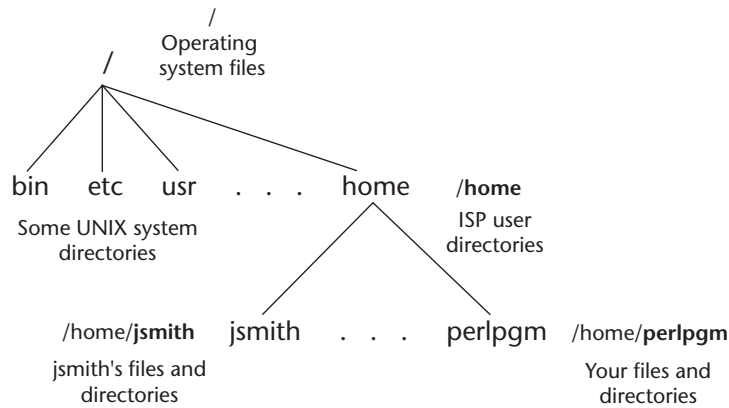


Figure 2.5 High-Level UNIX Server Directories

Within my home directory (that is, `/home/perlpgm`), my ISP's Web server has specific rules mandating where my HTML and Web application program files can reside. For example, all of my HTML files need to be placed in a directory called `public_html`. (See Figure 2.6.) My home page must be created inside `public_html` and must be saved in a file called `index.html`. All of my Web application programs must reside within the `cgi-bin` directory (that is, `/home/perlpgm/public_html/cgi-bin`). Note another aspect of Figure 2.6: It shows that I also created a directory called `/home/perlpgm/data` that is located on my Web server but is not accessible from the Internet (because it is not found under `/home/perlpgm/public_html`).

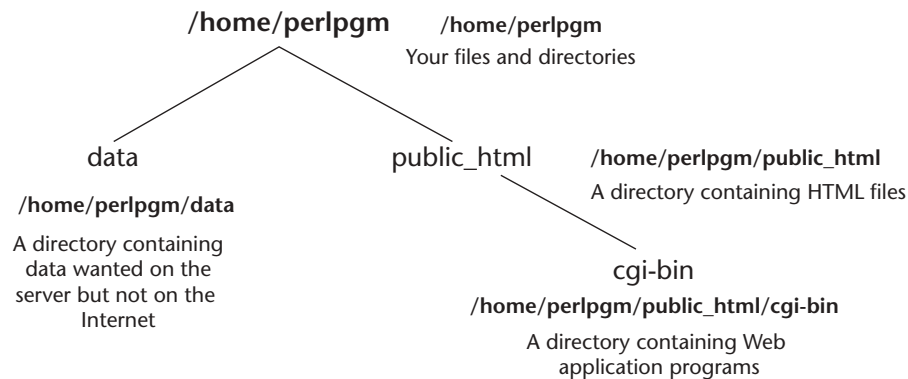



Figure 2.6 Directories on my ISP's Web Server Needed for my HTML Documents and Web Application Programs

Navigating UNIX Directories with FTP

You can also navigate your files and directories using FTP. Once connected, you can navigate directories and files by double-clicking the folder's icon. In addition, you can navigate up one level by double-clicking the up arrow icon. Figure 2.8 shows the results of using FTP to navigate to `/home/perlpgm/public_html` on my Web server and `C:\temp` on my PC.

To copy files from your PC to your Web server, you can change directories on your Web server (on the right side of the screen shown in Figure 2.8, labeled *Remote system*) to the directory in which you want to store your files. You can then copy your files by highlighting the file on your PC side (on the left side of Figure 2.8, labeled *Local system*) by single-clicking it, and then hitting the right arrow  in the middle of the FTP window.



Files and directories on your PC

Web server's files and directories

Highlight a file on the PC side and click the right arrow to copy files to your Web Server.

Figure 2.8 Using FTP to Navigate Directories on my Web Server (right) and PC (left)

Where Is Perl?

Another piece of information you need before you can begin to write Perl programs is the location of the **Perl interpreter**. The Perl interpreter is a computer program that translates Perl program commands into a set of commands that are more understandable to a computer. It will also run your Perl programs and generate any output. The Perl interpreter's command name is simply `perl`. The interpreter can be installed in any of several places on a Web server, so you need a way to find it.

If you use FTP access to the Web server, you will need to ask the ISP where the Perl interpreter is installed (or search its Web site to find this information). However, if your ISP allows Telnet access and operates a UNIX Web server, finding Perl yourself is not difficult. Simply connect to the Web server with Telnet, enter

2. *Change your program's access permissions.* Set the security permissions to make your program accessible over the Internet.
3. *Check your program's syntax.* Verify that all of your program statements are valid.
4. *Run your program.* Start your program to execute the commands in your Perl program and generate any desired results.

The following sections describe these steps in more detail.

Create Your Program File

Editors are computer applications that enable you to create, change, and save files. On Microsoft Windows, Notepad is a simple editor that works well for Perl development. (Several other Windows-based editors are available as well.) On UNIX systems, the Pico, Vi, and Emacs editors are popular choices. This section describes the use of **Pico** on a UNIX Web server, as it is the easiest of the UNIX editors to learn how to use.

★ TIP Using a Windows-Based Editor

If you plan to use a Windows-based editor (such as Notepad), you must save your programs on the PC and then use a program such as FTP to transfer the files to the Web server. That is, start Notepad, enter your program, save it on your PC, and then use FTP to copy it to your Web server.

To use Pico on a UNIX Web server, first start a Telnet session to the Web server and log in (as described earlier in this chapter). Once connected, enter the command to start the Pico editor:

```
pico
```

If everything is working correctly, you should see an initial Pico editor window similar to that shown in Figure 2.10.



Figure 2.10 The Initial Pico Editor Screen

★ **SHORTCUT** Some Pico Editor Commands

Within Pico, press **Ctrl-X** (hold the **Ctrl** key down and press **X**) to exit the editor, **Ctrl-O** to write the edit session to a file, and **Ctrl-R** to read in a file.

Creating Your First Program

After you start the editor, you are ready to create a program. For your first program, let's keep things simple. Using your editor, enter the follow-

ing code as shown in Figure 2.11 (the line numbers here are for reference only; do not use them in your program!):

1. `#!/usr/bin/perl`
2. `# This program prints out a simple message`
3. `print "Steady Plodding Brings Prosperity\n";`

★ Line 1 gives the full path name to the Perl interpreter. It must consist of a pound sign (#), followed by an exclamation point (!), followed by the Perl interpreter path name. You must enter this information without any spaces between the pound sign, exclamation point, and path name to Perl, as shown in line 1.

★ Line 2 is a comment line. Comments, which always start with a pound sign (#), describe what the program or particular statement does. Comments do not affect the execution of a program, but they can be invaluable in a more complex program when you are trying to understand how it works.

★ **TIP** Comment Usage

Comments can also be used after the semicolon in a Perl statement. Thus line 3 could also be written as follows:

```
print "Steady plodding brings prosperity\n"; # This line prints a message
```

★ Line 3 uses the Perl `print` command. This command outputs the text within the quotation marks when your program runs. Be careful to enter the line exactly as shown above, making sure that it ends with a semicolon (;).



Figure 2.11 Using Pico on my Web Server to Enter a Program

When you finish entering this code, save your program. Some ISPs require that you save your file with a special suffix or extension such as `.cgi` or `.pl`. For example, you might need to use a `.cgi` extension for CGI programs on your Web server, as in `simple1.cgi`. If you are using Telnet to connect to your Web server, you might be able to save your file directly in the required directory or folder: On my Web server I need to save the file into the following directory path:

```
/home/perlpgm/public_html/cgi-bin/simple1.cgi
```

Here, the name of the program is `simple1.cgi`. It is being stored in the `cgi-bin` directory, which is located within the `public_html` directory in my home directory.

★ **TIP** Saving Your Programs into Place

There are many ways to save your program into the right place. For example, if you develop your program on a Windows-based system, you might use FTP to connect to the Web server, navigate to the proper directory, and then copy your program into the desired location. If you use Telnet to connect to a UNIX Web server, you could use the three UNIX commands described earlier to navigate to the proper directory and then edit and save your program directly in the necessary directory. The important thing is to get a copy of your CGI/Perl program into the appropriate directory as required by your ISP.

Change the Program's Permissions

If you are using a UNIX Web server, you will likely need to set certain **access permissions** to your files after saving them on the Web server or copying them to the Web server with FTP. UNIX access permissions are used to define the access rights of your files. You can set the read permissions (that is, define whether the file can be read), the write permissions (that is, define whether the file can be changed), or the execute permissions (that is, define whether the file can be executed as a program.) You can set these access permissions for your user ID, for your user ID's group (that is, a set of user IDs that want to share data), and for everyone else.

The following shows how to change file access permissions on a UNIX Web server using Telnet and FTP.

Setting Permissions with Telnet on a UNIX Web Server

On a UNIX server with Telnet access, you can change your program's permissions by using the UNIX `chmod` command. In Figure 2.12, the `chmod` command is used to change the access permissions of a file to enable it to execute over the Internet.

As you can see in Figure 2.12, you specify two things for the `chmod` command: a three-digit number and a name of a file.

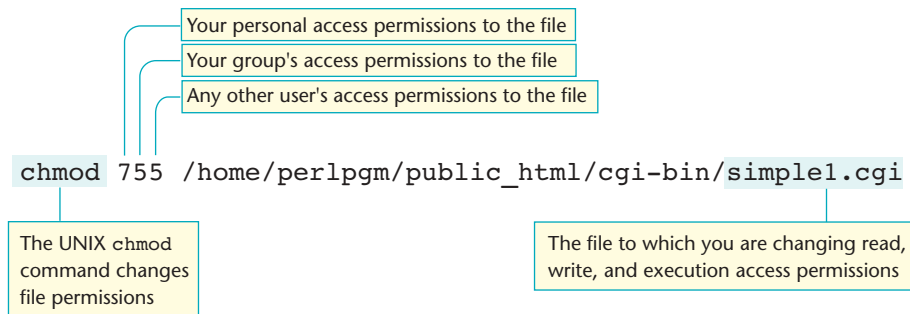


Figure 2.12 General Format of the UNIX `chmod` Command

The three-digit number really consists of three separate numbers. The first digit on the left sets your personal access permissions to the file—for example, read, write, or execute access permission. The second digit sets access permissions for your work group (usually your group includes only you, unless you define a group of users with whom you want to share your files). The final digit specifies the permissions for any other users on the system. Some common settings for each of these digits are shown below:

Digit	Permissions Meaning
7	Gives read, write, or execute access to the file
6	Gives read or write access to the file, but not execute access
5	Gives read or execute access to the file, but not write access
4	Gives read access to the file, but not write or execute access

The following examples illustrate access permission settings for the UNIX `chmod` command:

- ★ **`chmod 777 simple1.cgi`** enables the end user, members of his or her work group, and anyone else to read, write, or execute the file `simple1.cgi`. This setting is usually not a good idea, because it allows anyone to change the file.
- ★ **`chmod 755 simple1.cgi`** means that you can read, write, or execute the file `simple1.cgi`, but everyone else can only read or execute it.
- ★ **`chmod 644 simple1.cgi`** means that you can read or write the file `simple1.cgi`, but everyone else can only read it. This setting is useful for data input files and log files. (Data input files and log files are covered in Chapter 7.)

★ TIP Alternative Ways to Execute `chmod`

You could also execute the `chmod` command by first changing to the directory containing the file and then executing the `chmod` command. For example, I could use the following commands to first change into my `cgi-bin` directory (from my home directory) and then run the `chmod` command on the file, as shown here:

```
cd public_html/cgi-bin
chmod 755 simple1.cgi
```

Setting Permissions with FTP on a UNIX Web Server

Setting permissions with FTP is somewhat easier than using Telnet. As with Telnet, you can use FTP to set read, write, and execute access permissions for you, your group, and everyone else. To set permissions with FTP, follow these steps:

1. Log into the Web server using the FTP command.
2. Navigate to the appropriate directories on the Web server.
3. Select the file you want to change on your Web server, then right-click it. A drop-down menu will appear. Select **FTP commands**, and then **chmod**. Figure 2.13 shows the window that then appears, enabling you to change the access permissions.
4. Select the desired read, write, and execute access permissions for your user ID, your group, and anyone else.

★ TIP More on Permissions

You may want to review the “Setting Permissions with Telnet on a UNIX Web Server” section for more information on the various permission settings.

Check and Correct Your Program's Syntax

Computer languages have a very specific set of rules or grammar that define valid statements. The process of **syntax checking** verifies that program statements are grammatically correct as specified by the program language grammar. For example, Perl grammar requires each line of code to end with a semicolon. If a line of a Perl program was missing this semicolon, it would have a **syntax error**. It is good practice to check the syntax of your programs before attempting to run them. Checking this syntax is akin to checking the spelling and grammar of a document before asking someone else to read it! If you skip this step, it may create confusion.

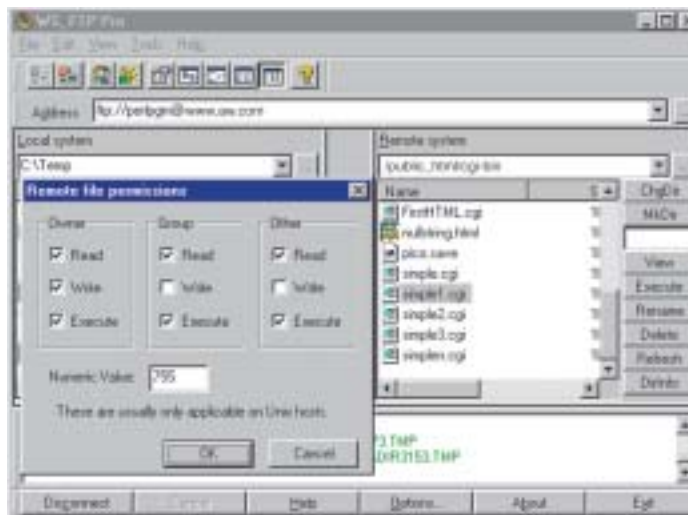


Figure 2.13 Setting Permissions Using the FTP Command

To check your program's syntax on a Web server with Telnet access, establish a Telnet session, navigate to the directory that contains the file, and then enter `perl -c filename`, where *filename* is the program file whose syntax you want to check. As an example, the following code changes the directory (from my home directory) and checks the syntax of a program file called `simple1.cgi`:

```
cd public_html/cgi-bin
perl -c simple1.cgi
```

You could also specify the full path to the file as follows:

```
perl -c /home/perlpgm/public_html/cgi-bin/simple1.cgi
```

If your program contains no syntax errors, the command will let you know by returning `syntax OK`. That is, it will respond with the following message:

```
simple1.cgi syntax OK
```

★WARNING Checking Syntax without Telnet Access

If you cannot Telnet into your Web server, you may not have a method for checking the syntax of your Perl applications before running them. In this case, it may make sense to download a copy of Perl, install it directly on your PC, and then check the syntax of your programs there.

If your program does contain syntax errors, don't panic! Instead, realize that when you enter commands with incorrect syntax, the syntax checker can merely guess at the true nature of the problem. Sometimes, the errors identified by the syntax

checker are only clues as to what is wrong. For example, if you accidentally include only one quotation mark in a `print` statement (and forget the other quotation mark), it may confuse the syntax checker, especially when you have many `print`

statements. When your program contains a syntax error, you need to revise it to correct the error, save the program again, and then recheck it. For large programs, you might have to repeat this process several times to remove all of the syntax errors.

The top screen in Figure 2.14 shows a Pico editor window of program containing a syntax error. Notice that the last line is missing a quotation mark before the semicolon. The correct line should read as follows:

```
print "Steady Plodding Brings Prosperity\n";
```

The bottom screen in Figure 2.14 shows the output after checking the syntax of the file with the `perl -c` command. Note that the output points to line 4 and indicates that a quotation mark is missing.

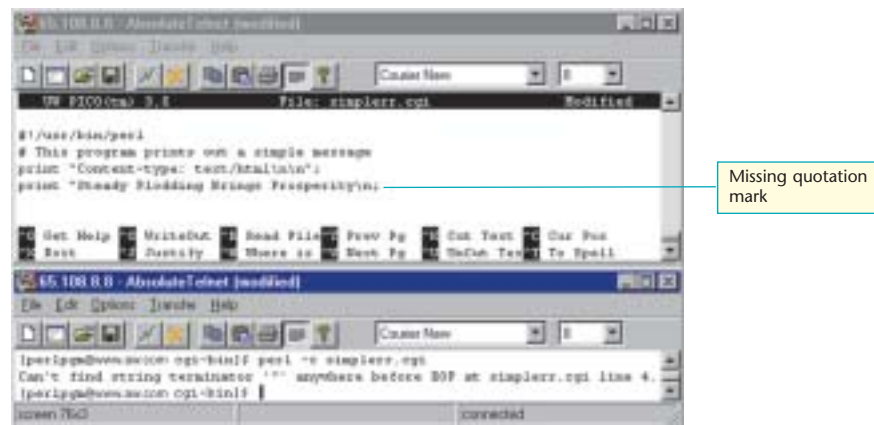


Figure 2.14 A Program with Illegal Syntax

★ TIP Dealing with Multiple Syntax Errors

When the `perl -c` command identifies multiple syntax errors, try to fix the first error reported by the syntax checker. Don't try to correct too many errors at once. It is usually better to fix only the first or second error and then recheck the program's syntax.

Running Your Program

There are at least two different ways to run your Perl programs:

- ★ *Directly on a Web server or PC without a browser:* You can start your programs either on the Web server if you have Telnet access or on your PC if you have Perl installed.
- ★ *Using your browser over the Internet.* You can start your programs by using a Web browser over the Internet and have your results display in the browser window.

Running Your Program Directly on the Web Server or PC without a Browser

Perl programs can run perfectly well without the use of a browser and Internet connection. Many Perl programs perform all sorts of tasks that don't require results to be output to the Internet. For example, a system administrator may run a Perl program that checks how well the Web server is performing. Another Perl program might be used to clean old files off a computer disk.

For our purposes, it is useful to know that you can run programs either on your PC (if you have Perl installed) or directly on a Web server via a Telnet session. Running Perl on your PC or Web server directly can be helpful when you are trying to get your programs to work. For example, when you try to start a program from your browser, you might receive an error message. If you can execute your program on the Web server (without using the browser), you can sometimes narrow down the possibilities as to the source of the problem.

If you have a UNIX Web server and can Telnet into it, you can start the program given in the previous section by entering the following command:

```
/home/perlpgm/public_html/cgi-bin/simple1.cgi
```

You can also run the program directly from the `cgi-bin` directory by changing the directory via the UNIX `cd` command (described earlier in this chapter). The following changes the directory into my `cgi-bin` on my Web server (from my home directory) and executes the program `simple1.cgi`:

```
cd public_html/cgi-bin
./simple1.cgi
```

Figure 2.15 shows the result of executing `simple1.cgi` in this way.



Figure 2.15 The Output of Your First Program Executed Directly on the Web Server

If Perl is installed on your PC, you can run your script there as well. Follow these two steps to run your program on a Windows-based PC:

1. *Open an MS-DOS prompt window.* Click **Start**, **Run** and then enter `command`.
2. *Run the program.* At the MS-DOS prompt, enter the location of Perl, followed by the location of your program:

```
C:\Perl\bin\Perl C:\temp\simple1.pl
```

Often you can omit the full path for Perl and simply enter the following:

```
Perl C:\temp\simple1.pl
```

★ **SHORTCUT** Using `cd` on a Windows-Based Machine

On a Windows-based PC, you can use the `cd` command while running in MS-DOS mode to change to the directory containing your program file. For example:

```
cd C:\temp
Perl simple1.pl
```

Getting Ready to Run Your Program over the Internet

Another way to start a Perl program is to use a browser over the Internet. Because this book focuses on Web programming, we will typically use this method to run our programs. To run your programs using a browser over the Internet, you need to add the following line to the example program:

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

Adding this line is critical if you want to use a browser to execute your Perl program. This line, which is sometimes called the **MIME content-type** line, tells the browser to expect output of type `text` or `HTML` from the CGI/Perl program. Be careful to enter this line *exactly* as shown above (including the double `\n` characters). If you misspell a word, forget a colon, or leave off a `\n`, your browser will not be able to start your program. This line should also be the first `print` statement in your program, appearing immediately after the code identifying the location of Perl (omit the line numbers when inputting this code):

```
1. #!/usr/bin/perl
2. print "Content-type: text/html\n\n";
3. # This program prints out a simple message
4. print "Steady Plodding Brings Prosperity\n";
```

To change your program, you must change your existing program file. You should follow the same basic steps every time you change your program file:

1. *Edit the program.* Start the editor (such as Pico, Vi, Emacs, or Notepad) and open your file.
2. *Change the program.* In our example, you are adding the MIME content-type as the second line.
3. *Save the file.* If you are developing the program on a Web server, then save the file to the proper directory on the server. If you are developing the program on your PC, then save the file there and use FTP to copy it back to the server.
4. *Check the program's syntax.* Run the `perl -c filename` command.
5. *Run the program.* You can run your program with or without your browser to test it (even with the MIME content-type line).

Running Your Program over the Internet

Once you have added the MIME Content-type line to your program and saved the revised program on the Web server, you are ready to test it over the Internet. To run your program over the Internet, follow these steps:

1. *Connect to the Internet.* If you dial in remotely, you will need to connect to the Internet.
2. *Start your browser.* You can use almost any browser program, including Netscape or Internet Explorer.
3. *Enter the URL or Web address to your file.* If you store your files in a `cgi-bin` directory under your `public_html` directory, then the address to your program (named `simple1.cgi`) might be `http://yourwebsite.com/cgi-bin/simple1.cgi`.

You might need to check with your ISP to figure out the correct Web site address. For my Web server, I saved the program shown above in a file called `simple2.cgi` in my `cgi-bin` directory on the Web server, so it can be executed by accessing the following Web address:

```
http://www.aw.com/~perlpgm/cgi-bin/simple2.cgi
```

Figure 2.16 shows this program's output when started from a browser.



Figure 2.16 Output of Running the Example Program from a Browser Window

Dealing with Problems

Unfortunately, when you run your programs with a browser, you may not always receive the most informative error messages. Many Web servers redirect the errors from CGI programs into a separate error log located on the server. As a result, you receive a generic, cryptic message when you run a program with an error from a Web browser. For this reason, you may need to run your program without the browser when possible to help eliminate these errors.

Two common messages that you might see are **Internal Server Error** (Figure 2.17) and **500 Server Error**. When you receive either of these messages, here are some things to check:

- ★ *Verify that the program syntax is correct.* The Perl interpreter should indicate that your program is free of syntax errors. (Use the `perl -c` command described earlier.)

- ★ *Verify your program's access permission.* Make sure that the permissions set for the program and directories are correct.
- ★ *Verify that you saved your file with the proper extension.* As mentioned earlier, your files may need to be saved with a `.cgi` or some other extension.
- ★ *Verify that your program is stored in the correct directory.* As noted previously, you may need to store your programs in a special directory (such as `cgi-bin` inside the `public_html` directory).
- ★ *Verify that you have the correct Web address to your program.* Double-check the accuracy of the Web address you entered in your browser.
- ★ *Confirm that the first line of your program gives the correct location of the Perl interpreter.* Verify that you entered the location of the Perl interpreter correctly.
- ★ *Confirm the accuracy of your MIME Content-type line.* Double-check the MIME Content-type line, making sure that it is the first `print` statement in your program, that it has the proper upper- and lower-case characters, and that you have not omitted any words.

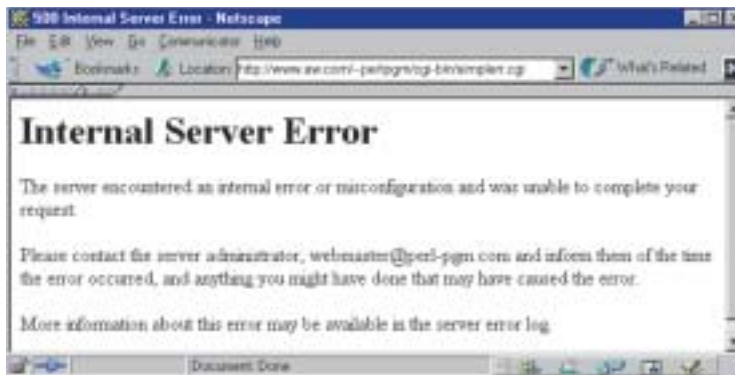


Figure 2.17 An Internal Server Error

Generating HTML Statements from Perl Programs

Now that you have seen how to create and change a simple program, let's create an HTML document from a CGI/Perl program. This process is possible because of the way Web servers, CGI applications, and Web browsers work together. Using the CGI standard, the HTML document output from your CGI/Perl program is provided as input to your browser. The browser displays this HTML document just like any other HTML file.

As an example, consider the following CGI/Perl program. It uses `print` statements to output the HTML tags needed to create an HTML document. Figure 2.18 shows the output of this program if you saved it on your Web server, set the permissions properly, and used your browser to start the program. (Note that the line

numbers are included only for reference purposes. Do not use them in your program!)

```
1. #!/usr/bin/perl
2. print "Content-type: text/html\n\n";
3. print "<HTML> <HEAD> <TITLE> Example </TITLE>
   </HEAD>";
4. print "<BODY>";
5. print "<B><Font Size=5>This is a Test </FONT></B>";
6. print "A very Interesting test";
7. print "</BODY></HTML>";
```



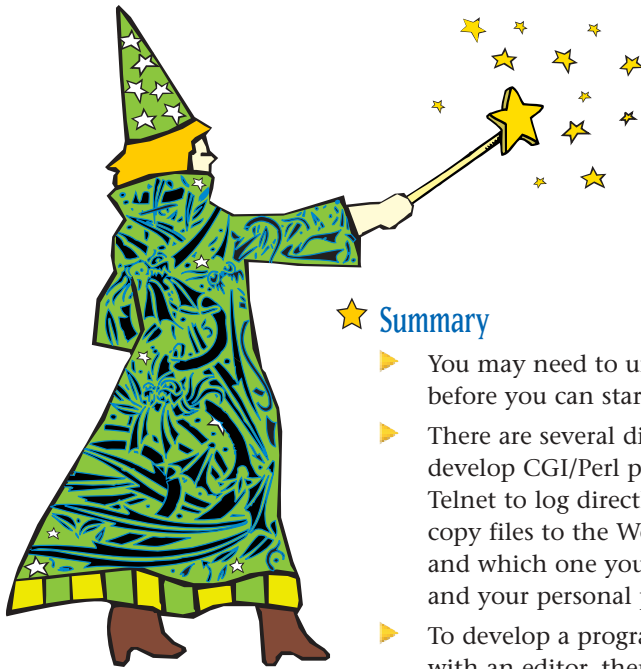
Figure 2.18 Generating a Simple HTML Document from a Perl Program

- ★ Line 1: Defines the location of the Perl interpreter
- ★ Line 2: The Mime Content-type line that notifies the Browser to expect text or HTML output
- ★ Lines 3-7: Generates a simple HTML document

★ **TIP** Using `\n`

The characters `\n`, when used within a `print` statement, cause Perl to output a new line. While HTML documents ignore this instruction (you must use the `
` tag to insert a line break in HTML), this code is useful for creating HTML document sources that are more *readable*. The source for your HTML documents will then look as if you typed them in an editor such as Notepad, rather than appearing on only one line.

Using Perl to generate HTML directly to your browser and creating HTML from your programs is easy. Although this example is very simple, you will see in the next chapter how this facility helps you to create dynamic Web pages.



★ Summary

- ▶ You may need to undertake some configuration steps before you can start developing your programs.
 - ▶ There are several different configurations you can use to develop CGI/Perl programs. Two common ones are using Telnet to log directly onto a Web server or using FTP to copy files to the Web server. Either method works well, and which one you choose depends on your ISP's policies and your personal preference.
 - ▶ To develop a program, you first create the program file with an editor, then enter your program, set the permissions, check the program's syntax, and finally run the program.
- ▶ Two statements are required in your CGI/Perl programs. The first line should identify the location of the Perl interpreter. The second line should specify the MIME Content-type.
 - ▶ You can generate text output and HTML output from your Perl programs by using `print` statements.

★ Online References

History of Telnet and FTP

<http://www.zakon.org/robert/internet/timeline/>

HTML Primer and FAQs about CGI

<http://www.htmlhelp.com/>

Home Page for the World Wide Web Consortium (Information on the HTML Standard and HTML Tutorials)

<http://www.w3.org/MarkUp/>

Primers on HTML

<http://www.htmlgoodies.com/html4-ref/>

Links to Tutorials on Perl and HTML; Downloadable Files and FAQs for FTP and Telnet Software.

<http://www.help.com>

Perl Information and Downloadable Versions

<http://www.perl.com>

3. The following program includes some syntax errors. Input this program into a file, check its syntax, and fix the errors. When you are finished, it should correctly display the document over the Internet.

```
1. #!/usr/bin/perl
2. Print "Content-type: text/html\n\n";
3. Print <HTML><HEAD><TITLE> Example </TITLE></HEAD>;
4. Print "This is an example of a program with
    syntax";
5. Print "errors. If you correct the errors, you can"
6. Print "get your program to work.</BODY></HTML>;"
```

4. Create a CGI/Perl program that outputs the following HTML document. Save your program on your Web server and view it over the Internet.

```
<HTML> <HEAD> <TITLE> More On Perl </TITLE></HEAD>
<BODY BACKGROUND="BLUE">
<H1> More On Perl </H1>
```

```
Many people who work with the <I>UNIX system</I>
appreciate the various features of the Perl
language. <BR><BR> Perl has features that combines
many of the UNIX utility programs such as:
```

```
<UL>
<LI> sed
<LI> grep
<LI> awk
</UL></BODY></HTML>
```

5. Run the program created in Exercise 4 directly on your Web server or your PC.
6. Create a CGI/Perl program that outputs the following HTML document. Save your program on your Web server and view it over the Internet.

```
<HTML> <HEAD> <TITLE> The Perl Language
</TITLE></HEAD>
<BODY>
<B><Font Size=5>T</FONT></B>he Perl language was first
    developed in 1987. Since then its use has grown
    rapidly. One use of Perl is to develop <I>Web
    application programs</I>.
</BODY></HTML>
```

7. Create a Perl program that outputs an HTML document that looks like the following. Give it a yellow background and a title of *My Preferences*.

Here are **three things I like**:

1. Baseball
2. Hot dogs
3. Apple pie