19 COMMON GATEWAY INTERFACE (CGI)

KEY OBJECTIVES

After completing this chapter readers will be able to—

- understand the concept of server-side programming
- get an idea about the execution philosophy of CGI programs
- get an overview about languages used to write CGI programs
- understand CGI environment variables and their importance
- write basic CGI programs using C/C++, Perl, Python, etc.
- learn how to retrieve parameters from CGI programs
- get an overview about the shortcomings of CGI programs

19.1 INTERNET PROGRAMMING PARADIGM

Internet programming can be classified into two categories: *client-side programming* and *server-side programming*. In the client-side programming paradigm, programs/scripts are downloaded, interpreted, and executed by the browser. The author of the programs does not have any idea about the type and version of the browser used to execute them. So, if the browser is not compatible with the technology used, the content will not be presented properly. Let us take a specific example.

Applets are client-side programming technology, where special Java programs are embedded directly into web pages with the help of the `<applet>` tag. When a browser loads such a web page, the applet byte code is also downloaded and executed on the client side. If the browser uses an old Java Runtime Environment (JRE), applet byte code cannot be executed properly and the entire thing becomes garbled. Moreover, for large applets, download time becomes significant. Hence, this technology tends to be unacceptable. These issues have enforced businesses to use server-side programming.

19.2 SERVER-SIDE PROGRAMMING

Server-side programming solves the problem discussed in Section 19.1. The basic idea of this paradigm is that programs are executed in the web server. Hence, there is no issue of browser incompatibility or long download time. The web server sends web pages (containing simple code generated by those programs) that even an old browser can understand.

The Common Gateway Interface (CGI) is one of the important server-side programming techniques. The CGI connects a web server to an external application. Figure 19.1 shows the CGI architecture.

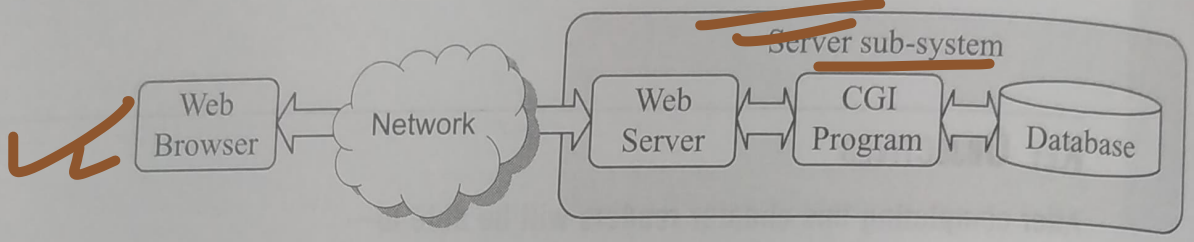

Figure 19.1 CGI Architecture

When a CGI-enabled web server receives a request for a CGI program, it does not send the file as it is. Instead, the web server executes [Figure 19.1] the program at the server end and sends the output back to the client, which is then displayed in the browser's window. This simple and elegant idea can be used to develop many powerful applications.

CGI has numerous advantages. Most of the web servers have built-in support for CGI. So, if you have your web server installed in your computer, you can start writing without any further effort.

Moreover, the CGI specification is independent of any programming language. It defines how information is transferred from a web server to an external application and from the external application to the web server. So, these external applications may be developed using a programming language that fits the application.

19.3 LANGUAGES FOR CGI

The most powerful feature of CGI technology is that virtually any programming language can be used to write CGI programs as long as it can read from a standard input and write to a standard output. Following are some popular CGI programming languages.

C/C++

C is one of the popular programming languages. It is well known for its extremely good performance. It is widely used on many different software platforms and is a primary language for CGI. There is virtually no computer architecture for which a C compiler does not exist.

C++ supports Object Oriented Programming (OOP) and is suitable for large complex applications. So, if your CGI program should handle a large problem, C++ is ideal.

Perl

This interpreted language provides powerful text processing and file manipulation facilities. Perl borrows features from other programming languages such as C, shell script (sh), AWK, and

sed. Due to its flexibility and adaptability, it is widely used for CGI programming. It is also used for network programming, applications that require database access, system administration, and graphics programming.

Tcl

Tcl is a scripting language that was originated from "Tool Command Language", but is conventionally rendered as "Tcl". It is commonly used for CGI scripting as well as rapid prototyping, other scripted applications, GUIs, and testing.

Python

It is an interpreted, interactive, portable, Object Oriented Programming (OOP) language. Its significant power and clear syntax make Python an excellent instructional tool and ideal for Common Gateway Interface (CGI) programming. Language features include modules, classes, very high level dynamic data types, and dynamic typing.

Unix/Linux Shell

A shell is a command line interpreter that provides an interface to the users, to execute their commands. Unix/Linux shell is extremely powerful for manipulating files, pattern searching, and matching, and is ideal for CGI scripting in the Unix/Linux platform.

19.3.1 Preferred Language

The commonly used languages are Perl, C/C++, and shell script. However, your choice depends on what you want to do because different languages have different specialized features. For example, Perl is extremely powerful for string and file manipulation while C/C++ is better for complex and larger programs.

It depends on your taste as well as the facilities available on your system. If you prefer a *programming language* such as C/C++ or Fortran, you have to compile the program and generate an executable code before it runs. The original source code is no longer needed. This allows you to hide your sensitive program code from others who have access to the CGI directory. Moreover, the programs are already compiled and they take less time to start servicing than an interpreted one.

On the other hand, if you use any *scripting language*, such as Perl, Tcl, or Unix/Linux shell, you need to keep the script itself in the "cgi-bin" directory. Many programmers prefer CGI scripts instead of programs, as scripts are easier to modify, debug, and maintain than typical compiled programs.

19.4 APPLICATIONS

There are numerous applications of the CGI. It is usually used to build database applications in conjunction with HTML forms. For example, suppose we want to "hook up" our database to the World Wide Web (WWW), so that people from all over the world can query it, all we need to do is write a CGI program that can access this database. The web server will execute this program to store information to the database and receive results from the database, which are then sent back to the client.

19.5 SERVER ENVIRONMENT

Most the web servers standardize the CGI mechanism. Usually, the directory "cgi-bin" exists under the web server's installation directory. The files in this directory are treated differently. Any file requested from this special "cgi-bin" directory is not simply read and sent. Instead, it is executed in the computer where the web server is installed. The output of this program is actually sent to the browser that requested this file. The program is an executable file of typically a Perl or Python script.

Though the "cgi-bin" directory is usually considered as the container of CGI scripts, most web servers allow us to specify other directories as the CGI directory.

Suppose that you have entered the following URL in the address bar of your browser:

```
http://192.168.1.2/cgi-bin/hello.pl
```

The web server recognizes the file `hello.pl`, which is nothing but a Perl script in the `cgi-bin` directory. It then executes `hello.pl` and sends the output of this script to the browser, which then displays it on the screen.

Though CGI programs are typically written on a Unix platform, you can also write your CGI programs in a Windows environment. For this purpose, you need a web server. Perl is the typical language used for CGI programs, but you can use C/C++ or Python as well. In this section, we shall discuss how to install and configure Apache and Perl on your computer running Windows. Once Apache and Perl are successfully installed, you can test your CGI programs using the address `http://locahost` in your own computer. This is quite useful to test your CGI programs locally before installing them in the actual server.

Download the installer file for Apache from `http://httpd.apache.org/download.cgi`. Now, install it by double clicking on the installer file. We are assuming that you have installed Apache in the directory "D:\Apache Software Foundation".

Now, download and install ActivePerl from `http://www.activestate.com/Products/ActivePerl/`. We are assuming that you have successfully installed Perl in the "D:\Perl" directory.

19.5.1 Configuring Apache

To enable CGI programs, you need to modify the Apache configuration file `httpd.conf`, which can be found in the `conf` directory under the apache installation directory. Go to the `<Directory>...</Directory>` section. Uncomment or add the following line:

```
Options MultiViews Indexes SymLinksIfOwnerMatch Includes ExecCGI
```

The `Options` specifies what options are available in this directory. `ExecCGI` enables the CGI scripting. Go to the `AddHandler` section and add or uncomment the following line.

```
AddHandler cgi-script .cgi .pl
```

This causes files with the extensions `.cgi` and `.pl` to be treated as CGI programs. Finally, save the configuration file and restart Apache. Ensure that the server restarted successfully by checking `http://localhost/` in your browser.

If everything goes well, your web server is ready to serve the CGI request. Save your CGI programs in the cgi-bin directory under the Apache installation directory.

Use the following line as the first line of your Perl program:

```
#!D:/perl/bin/perl.exe
```

This tells the web server where to find the Perl interpreter to interpret Perl programs.

19.6 ENVIRONMENT VARIABLES

In order to pass and retrieve parameters, web servers use several environment variables. The web server usually sets these environment variables before starting a CGI program. The CGI program can inspect those environment variables [Table 19.1] to retrieve information. Note that CGI environment variables are the primary source of information that server-side programs can use.

Table 19.1 CGI Environment Variables

Variable Name	Description
SERVER_NAME	The server's DNS name or IP Address
SERVER_SOFTWARE	The name and version of the web server software answering the request
GATEWAY_INTERFACE	The version of the CGI specification that the server complies with
SERVER_PROTOCOL	The HTTP version used by the server
SERVER_PORT	The port number used by the web server
CONTENT_TYPE	The type of the data of the content. It is used when the client is sending attached content to the server e.g., example file upload, etc.
CONTENT_LENGTH	The length of the query information in case of the POST method
HTTP_ACCEPT	The list of MIME types that the client can accept
HTTP_USER_AGENT	The browser name and version that the user is using to make this request
PATH	The value of PATH environment variable
QUERY_STRING	The URL-encoded information which follows the ? in the URL
REMOTE_ADDR	The IP address of the remote host that made the request
REMOTE_PORT	The port number from which the request was sent
REQUEST_METHOD	The HTTP method used in the request. The most common methods are GET, HEAD, and POST
SCRIPT_FILENAME	The full path of the CGI script being executed
SCRIPT_NAME	The relative path of the script being executed

The following Perl script prints available environment variables with their values.

```
#!D:/perl/bin/perl.exe
print "Content-type: text/html\n\n";
print "<table style=\"font-size:12; font-family:arial\" border=\"0\" >";
print "<caption>CGI variables</caption>";
foreach $var (sort(keys(%ENV))) {
   $val = $ENV{$var};
   print "<tr><td>${var}</td><td >${val}</td></tr>\n";
}
print "</table>";
```

Ignore the syntax of this Perl program. It simply iterates through the associative array variable ENV, which contains all the environment variables and prints their values in a tabular format. We shall discuss the syntax briefly in Section 19.9. The script results in the output shown in Figure 19.2.

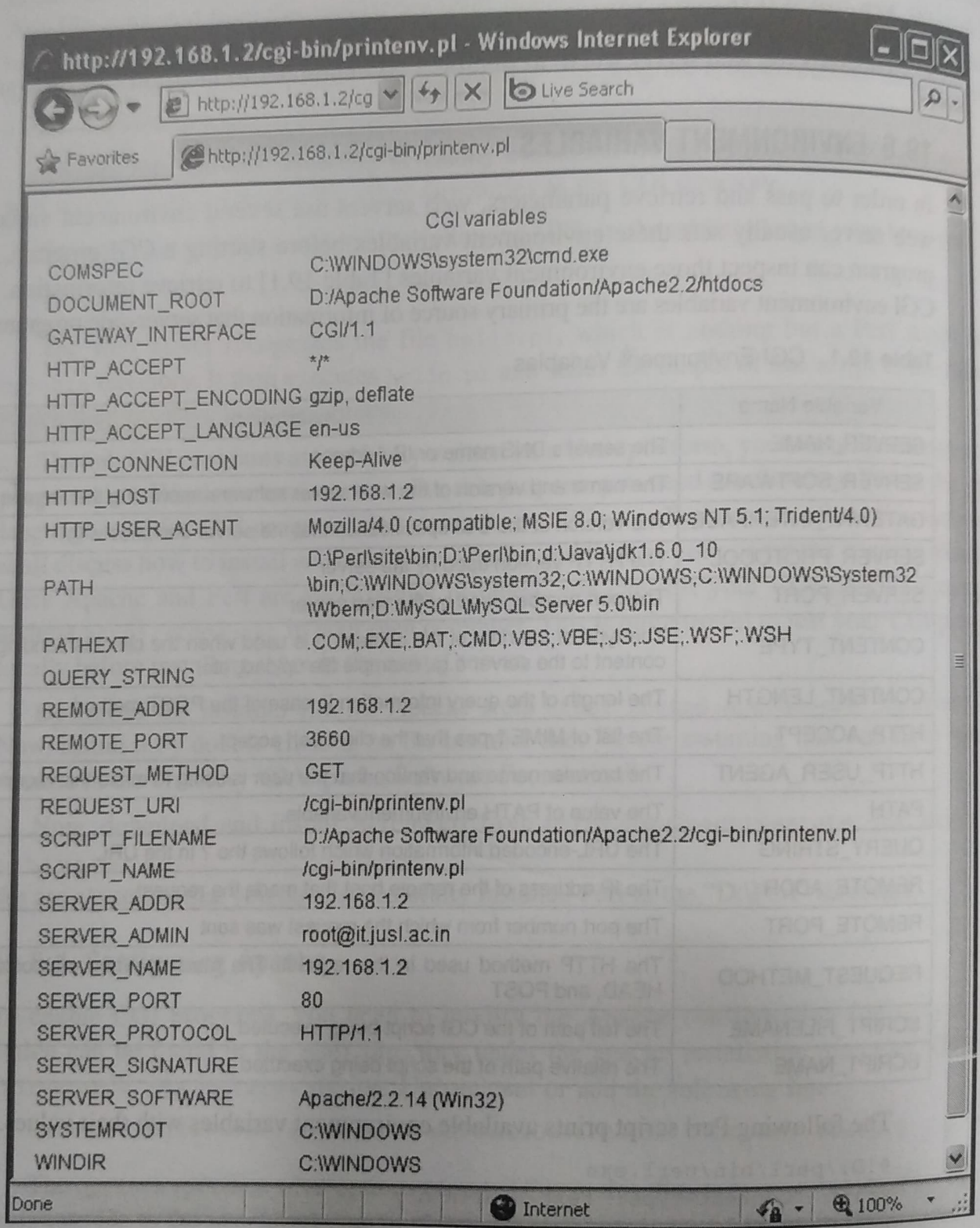

Figure 19.2 CGI environment variables

19.7 CGI BUILDING BLOCKS

Any CGI script basically consists of three steps:

- Read parameters passed to this script
- Process these parameters
- Write the HTML response to the standard output

19.8 CGI SCRIPTING USING C, SHELL SCRIPT

Writing CGI programs using C/C++ language is somehow different from writing them in shell script. Here, shell refers to the Unix/Linux shell. The Windows shell is not so powerful and is hardly used. Note that shell scripts are interpreted by the underlying shell. You simply need to mention which shell should be used to interpret your script in the first line of your script as follows:

```
#!/bin/bash
```

C/C++ programs are, on the other hand, compiled programs. Writing CGI programs using C/C++ basically consists of two steps:

- Compile the program to generate the executable file
- Put this executable file in the `cgi` directory.

Compiling a C/C++ program is different for different compilers. Ask your system administrator to know the compilation procedure of C/C++ programs. Moreover, C/C++ programs are not platform-independent. So, you must compile your program in the computer where the web server runs, using a suitable compiler available on that platform.

19.9 WRITING CGI PROGRAMS

In this section, we shall develop simple programs to display "Hello World!" using different languages such as Perl, C, and Python. Following is the program (`hello.pl`) written in Perl.

```
#!D:/perl/bin/perl.exe
print "Content-type: text/html\n\n";
print "<html>\n";
print "  <head><title>First Perl script</title></head>\n";
print "  <body>\n";
print "     <h2>Hello, World!</h2>\n";
print "  </body>\n";
print "</html>\n";
```

The script `hello.pl` is a simple script that prints a simple HTML document on the standard output, i.e., screen. The first line of any Perl script should look like this:

```
#!D:/perl/bin/perl.exe
```

The `#!` indicates that this is a script. The path `D:/perl/bin/perl.exe` refers to the full path name of the Perl interpreter to be used by the web server to execute the Perl script. However, this could be different in different systems. For a Unix or Linux OS, this is typically `/usr/bin/perl` or `/usr/local/bin/perl`. If you are not sure about the path, type "whereis perl" or "which perl" at the command prompt, it shows the path where the Perl interpreter is stored. Alternatively, ask the webmaster about the location of the Perl interpreter.

The remaining part consists of actual Perl statements. Before printing anything else, you should use the following line:

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

This prints `Content-type: text/html` followed by a blank line. This is sent as a HTTP response header, which specifies the type of the content to be displayed in the browser's screen. In our case, we shall send an HTML document and that is why the `Content-type` header is specified as `text/html`. This Perl program, except `Content-type` header, basically generates the following:

```
<html>
  <head><title>First Perl script</title></head>
  <body>
    <h2>Hello, World!</h2>
  </body>
</html>
```

If you enter the following URL, you will see the result shown in Figure 19.3.

```
http://192.168.1.2/cgi-bin/hello.pl
```

Here, 192.168.1.2 is the IP address of the computer where the web server is running. If you are testing your program locally, you can also use the following URL:

```
http://localhost/cgi-bin/hello.pl
```

Alternatively, you can use

```
http://127.0.0.1/cgi-bin/hello.pl
```

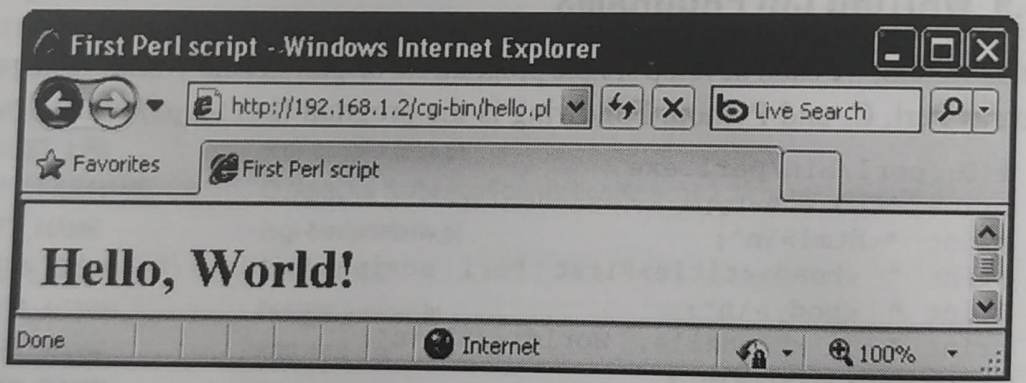

Figure 19.3 Hello world CGI program using perl

Let us now write a CGI program (`hello.c`) using C language for the same purpose.

```
//hello.c
int main() {
   printf("Content-type: text/html\n\n");
   printf("<html>\n");
   printf(" <head><title>First Perl script</title></head>\n");
   printf(" <body>\n");
   printf("    <h2>Hello, World!</h2>\n");
   printf(" </body>\n");
   printf("</html>\n");
  return 0;
}
```

Compile the program using a suitable compiler such as Microsoft's Visual C++. Microsoft's Visual C++ compiler will generate the executable file `hello.exe`. Rename this file to hello and put it in the CGI directory. If you are using the Unix/Linux platform, use the following command to compile the C program and generate an executable file:

```
gcc -o hello hello.c
```

It generates an executable file `hello`. Put this file in the CGI directory. Now, type the following URL, you will see the output shown in Figure 19.4.

```
http://192.168.1.2/cgi-bin/hello
```

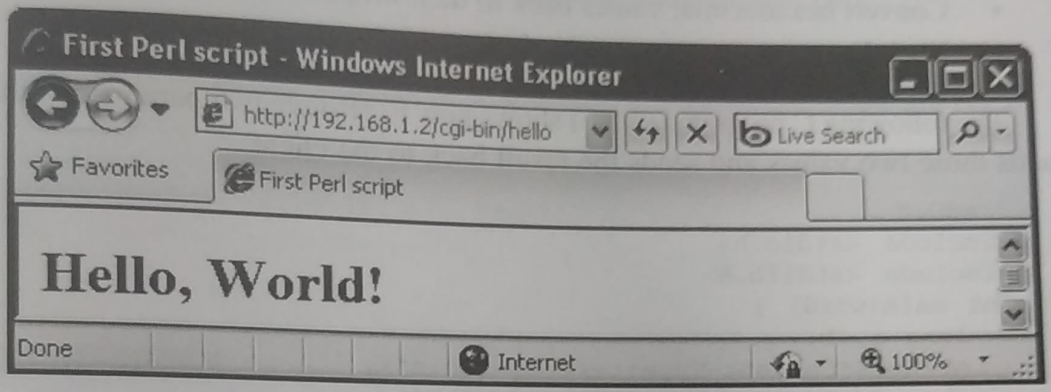

Figure 19.4 Hello world CGI program using C

The following program is written in Python (`hello.cgi`) and generates the same output.

```
#!D:\Python31\python.exe
print("Content-type: text/html\n\n");
print("<html>");
print(" <head><title>Python demo<title></title></head>");
print(" <body>");
print("    Hello World!");
print(" </body>");
print("</html>");
```

The first line indicates the location of the Python interpreter. We are assuming that you have installed the Python in the "D:\Python31" directory. Put this file in the CGI directory and use the following URL to view the result.

```
http://192.168.1.2/cgi-bin/hello.cgi
```

19.9.1 Getting Arguments

In this section, we shall discuss how CGI programs can retrieve URL-encoded parameters passed to them. The parameters passed have the following characteristics:

- They are sent in the form of name–value pairs.
- Each name–value pair starts with an & sign.
- The name and value are separated by the equal (=) sign.

URL encoding changes some special characters to placeholders and replaces them with hexadecimal values. To retrieve the information, the CGI program should implement the following basic steps:

- Get the information from the proper environment variable, depending upon the method type. For example, we can retrieve information from the `QUERY_STRING` environment variable for the GET method.
- Change all placeholders to their correct values.
- Split each name–value pair.
- Convert hexadecimal values back to their original characters.
- Find the respective name and value.

The following C program (`add.c`) shows how to retrieve two parameter values. The program adds those two values and sends the result back to the client.

```
//add.c
#include <stdio.h>
#include <stdlib.h>
int main(void) {
   long a, b;
   printf("Content-Type:text/html\n\n");
   char *data = getenv("QUERY_STRING");
   sscanf(data,"a=%d&b=%d", &a, &b);
   printf("%d + %d = %ld", a, b, a + b);
   return 0;
}
```

Here, we first obtain the entire parameter information using the `getenv()` function. The individual parameters are then obtained using the `sscanf()` function. Compile the program to have an executable file add using the following command:

```
gcc -o add add.c
```

Now, use the following URL to test the program.

```
http://192.168.1.2/cgi-bin/add?a=3&b=4
```

The result is shown in Figure 19.5.

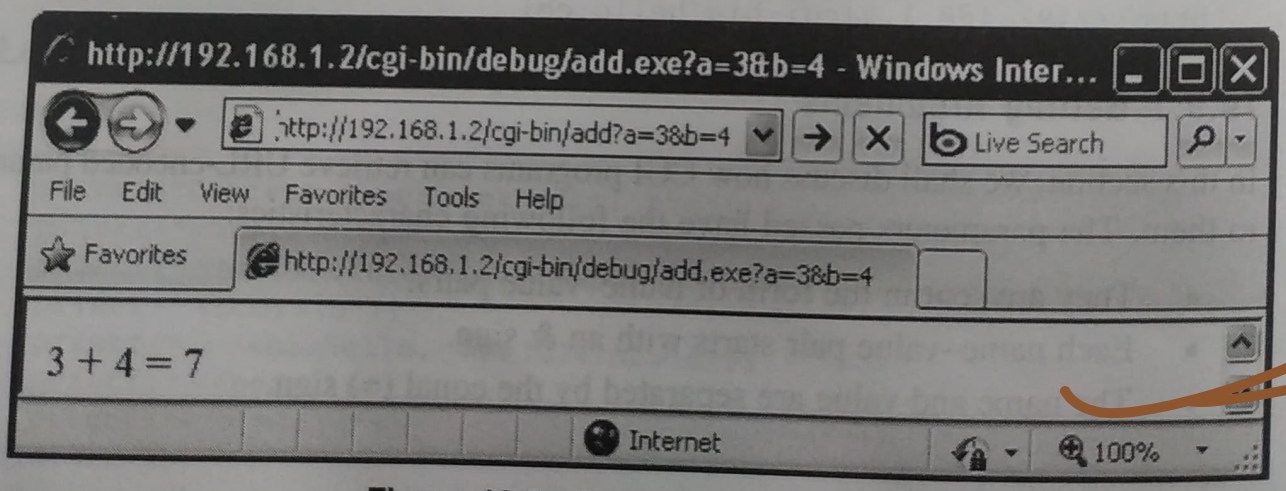

Figure 19.5 Retrieving parameters

Following is the equivalent Perl program.

```
#!D:/perl/bin/perl.exe
use CGI qw(:standard);
print "Content-type: text/html\n\n";
$a = param('a');
$b = param('b');
$c = $a + $b;
print "$a + $b = $c";
```

The following Perl script retrieves all parameters appended to the URL.

```
#!D:/perl/bin/perl.exe
print "Content-type:text/html\r\n\r\n";
$buffer = $ENV{'QUERY_STRING'};
@pairs = split(/&/, $buffer);
foreach $pair (@pairs)    {
   ($name, $value) = split(/=/, $pair);
   print "$name=$value<br>";
}
```

19.10 CGI SECURITY

The CGI technology allows users to run programs in a system remotely using their URLs. This may become vulnerable. Therefore, some security precautions need to be taken when CGI technology is used. Note that CGI programs are kept in a special directory, so that the web server can execute the program rather than just send them to the browser. This is the most important concept that CGI writers follow: The external users may try to use this concept to perform malicious events. The CGI directory is usually controlled by the webmaster, who should prohibit the average user from creating and running CGI programs. There are many ways to allow access to CGI scripts, but the webmaster decides how to set these up for you.

Although the CGI protocol is inherently secure, CGI scripts may become a major source of security holes. CGI scripts should be written with utmost care just as the server. The web administrator should not also trust the script writers and should not install arbitrary CGI scripts.

If you are a system administrator, a webmaster, or are otherwise involved with the administration of a network, you should create a security policy for your website. The following points should be taken into consideration while writing the security policy:

- Users who can access the system
- When they are allowed to use the system
- What operations they are allowed to perform
- A way to grant access to the system
- Acceptable permission to use the file system
- Procedure for monitoring the system
- Actions to be taken against suspected security breaches

19.11 ALTERNATIVES AND ENHANCEMENTS TO CGI

Each time the web server receives a CGI request, it starts a new process that serves the request. Starting a new process may take significant amount of time and memory than the actual task of generating the output. If users send CGI requests very frequently, they may quickly overwhelm the web server.

If the CGI program is an interpreted program such as Perl, Python, or shell script, it needs more time. We can use compiled programs such as C/C++, instead of scripting languages, to avoid overhead involved in interpretation.

The overhead in process creation may be reduced by using technology such as FastCGI or by using special extension modules provided by some web server.

FastCGI uses a single persistent process that handles many requests during its lifetime. This way, overhead involved in new process creation for every request may be avoided. Multiple simultaneous requests can be handled either by using multiple processes or using a single process with internal multiplexing. FastCGI allows web servers to perform simple operations, such as reading a file before the request is handed over to the FastCGI program. Some of the web servers that implement FastCGI are Apache HTTP Server, Microsoft IIS, Resin Application Server, Sun Java System Web Server, etc.

Another alternative to CGI is Simple Common Gateway Interface (SCGI), which is similar to FastCGI but easier to implement. In this technology, clients send requests to the SCGI server using an SCGI request message. The server sends the response back and closes the connection. Web servers that implement SCGI are Apache HTTP Server, Lighttpd, Cherokee, etc.

Another viable and effective solution to CGI is Java servlets. Servlets can avoid overhead involved in new process creation. Web servers load the servlet class when it starts up. The servlet can then serve many requests using a separate thread. Since threads are more lightweight than processes, servlets are more efficient than CGI programs.

Java Server Page (JSP) is an extension of Java servlet and is a potential solution to CGI technology. In the next two chapters, we shall discuss Java servlet and JSP, separately.

KEYWORDS

CGI alternatives: The competent technologies such as servlets, JSP, ASP, and PHP.

CGI directory: CGI programs are kept in some predefined special directory called CGI directory so that the web server knows that those programs must be executed, instead of being sent directly to the clients.

CGI Environment variables: The variables used to pass and retrieve information in CGI programs.

CGI Languages: The programming languages that can be used to write CGI programs.

Client-side programming: A programming paradigm where programs are executed in the client machine.

Common Gateway Interface: A standard that interfaces the HTTP server software with the CGI programs that run on the server.

Compiled CGI program: A program that is compiled to generate executable code, which acts as a CGI program.

Gateway program: A program that the web server contacts to process some request sent by the client.

Interpreted CGI program: A program whose statements are interpreted using an interpreter program.

SCGI: An extension to CGI called Simple Common Gateway Interface.

Server-side programming: A programming paradigm where programs are executed in the server machine.

SUMMARY

The Common Gateway Interface (CGI) is one of the popular server-side technologies. CGI has numerous advantages. Most of the web servers have built-in support for CGI. Moreover, the CGI specification is independent of any programming language; it defines how information is transferred from the web server to an external application and from the external application to the web server. So, these external applications may be developed using a programming language that fits the application.

When a CGI-enabled web server receives a request for a CGI program, it does not send the file as it is. Instead, the web server executes the program and sends the output back to the client, which is then displayed in the browser's window.

The most powerful feature of CGI technology is that virtually any programming language can be used to write the CGI program as long as it can read from a standard input and write to a standard output. Some examples are C/C++, Perl, Python, Tcl, shell, Ruby, etc. Compiled programs such as C/C++ run faster than interpreted programs. Interpreted programs, on the other hand, are easier to modify, debug, and maintain.

Most web servers standardize the CGI mechanism. Usually, the directory "cgi-bin" exists under the web server's installation directory. The files in this directory are treated differently. Any file requested from this special "cgi-bin" directory is not simply read and sent. Instead, it is executed in the computer where the web server is installed. The output of this program is actually sent to the browser that requested this file. The program usually is pure executable file of typically a Perl or Python script.

To pass and retrieve parameters, web servers use several environment variables. The web server usually sets these environment variables before starting a CGI program. The CGI program can inspect those environment variables to retrieve the information.

Since the CGI program can be written in almost all languages and in all platforms, the exact procedure for executing the CGI program varies from one platform/web server to another.

The CGI technology allows users to run programs in a system remotely using their URLs. This may become vulnerable. Therefore, some security precautions need to be taken when CGI technology is used.

WEB RESOURCES

`http://www.ietf.org/rfc/rfc3875.txt`
The Common Gateway Interface (CGI) Version 1.1

`http://hoohoo.ncsa.illinois.edu/cgi/`
The Common Gateway Interface

`http://www.w3.org/CGI/`
CGI: Common Gateway Interface

`http://www.fastcgi.com/devkit/doc/fcgi-spec.html`
FastCGI Specification

EXERCISES

Multiple Choice Questions

1. Server-side scripting is about "programming" the behavior of the
   (a) Server (b) Browser
   (c) HTML (d) All of the above

2. What is the full form of CGI?
   (a) Common Graphical Interface
   (b) Cascading Gateway Interface
   (c) Class Generator Instance
   (d) Common Gateway Interface

3. Which of the following defines how a web server communicates with external applications?
   (a) HTTP (b) CGI
   (c) SMTP (d) TCP/IP