CHAPTER 24

The Advertising Tracker

OVERVIEW

The advertising tracker script allows a Web site to collect statistics on how many people use a hypertext link to jump to an advertised company's site. Just as television and radio stations, as purveyors of information, have seen the profit in selling advertising to companies whose products are displayed along with information people wish to see, so the Web has exploded with sites that offer advertising space on their Web pages. These sites are visited by enough people that companies want to pay to have their logo displayed there.

As Web sites compete for advertising dollars, they must increasingly gather statistics about the people who use the Web site as a point from which they jump to an advertised company's URL. Two statistics come to mind. The first is how many people view the ad. This statistic is easy to collect, because counter programs are relatively common and can be placed alongside an ad to count the number of times the page is displayed. A second statistic that an advertiser usually wants collected is how many people clicked on the ad to jump to the advertised site. This statistic—harder to collect—will be explained in this chapter.

Generally, ads on the Web consist of a clickable image that is referenced to another URL on the Internet. The user clicks on the image and is taken to the Web site that the image advertises. Unfortunately, this arrangement leaves you without a record of where the user went except via a server log. Even then, you may not have access to the server log if you are on a shared server set up by an Internet service provider.

The alternative solution is to have the clickable image send users to an ad tracker CGI script before sending them to the site. This script records the "hit" to the ad image and then sends a redirection signal to the user's Web browser to tell it to automatically go to the advertiser's Web site.

Installation and Usage

The ad tracking files should install into a directory called **Ad_track**. The files and subdirectories associated with this application, along with their required permissions, are shown in Figure 24.1.

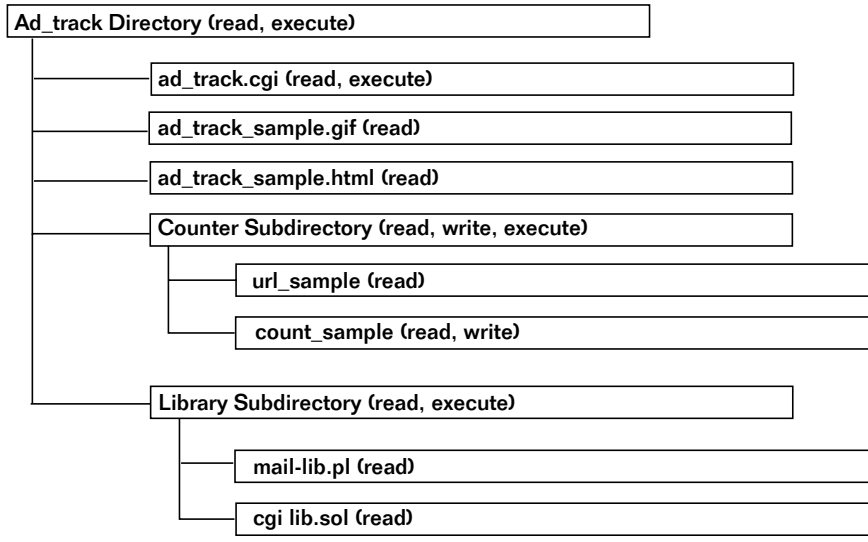

Figure 24.1 *Ad tracking script directory structure and permissions.*

Ad_track is the root application directory. It must be readable and executable by the Web server. In addition to the application files, the **Counter** and **Library** subdirectories are also located here.

ad_track.cgi is the CGI script that increments the counter information and redirects the user to a new URL. This file must be readable and executable.

ad_track_sample.html is a sample usage of the ad tracking script. If your server is configured so that HTML files cannot be viewed from a CGI directory, you need to move this file and update the reference to **ad_track.cgi** to address it from the new location. This file must be readable.

ad_track_sample.gif is an image file that is referenced by the **ad_track_sample.html** file. This file should be placed in the same directory as **ad_track_sample.html** and should be readable by the Web server.

The **Library** subdirectory stores all the external libraries the script needs to access. This subdirectory should be readable and executable. All the files in it should be readable. These files include **cgi-lib.pl** (used for processing HTML forms) and **cgi-lib.sol** (used for the counter and lock file routines).

The **Counter** subdirectory is used by the ad tracker to keep the URL setup and counter files. This directory must be readable, writable, and executable.

The **url_[page name]** file is located in the **Counter** subdirectory. This file contains the URL where the user is sent when **ad_track.cgi** is given the `[page_name]` as a variable on the URL calling the script. This file must be readable by the Web server.

The **count_[page_name]** file is located in the **Counter** subdirectory. This file contains the current count of clients who have been sent to `[page_name]` by **ad_track.cgi**. This file is created by the **ad_track.cgi** program and by default is readable and writable.

Server-Specific Setup and Options

Only one file needs to be set up for each ad you wish to track. You associate a word or "page" with the advertisement and then create a file in the **Counter** subdirectory called **url_[page_name]**, where **`[page_name]`** is your

code word for the ad. For example, if a car is being advertised on your site, you would create a file called **url_car** in the **Counter** subdirectory.

This file contains only one line. The line contains the URL that the user's Web browser will be directed to load after incrementing the count file. Here is an example of **url_car**:

```
http://www.foobarcar.com/year2000car.html
```

Running the Script

To use the **ad_track.cgi** program, you need to refer to the script along with one URL variable: `page`. `page` needs to be set to the suffix of the **url_** file in the **Counter** subdirectory that contains the URL the user will be directed to. Here is a sample using the car example if it is installed in an **Ad_track** subdirectory under a **cgi-bin** directory:

```
http://www.foorbar.com/cgi-bin/Ad_track/ad_track.cgi?page=car
```

When this URL is called for the first time, a counter file is created in the **Counter** directory called **count_car** with the number 1 in it. This number is incremented every time someone accesses the **ad_track.cgi** with the `page=car` tag.

In addition to the car example, the script comes with a sample HTML file that contains a reference to the **ad_track.cgi** file using the **url_sample** file that comes on the accompanying CD-ROM. Clicking on the reference to the **ad_track.cgi** script in this HTML document takes you to Selena Sol's script page and increments the counter in the **count_sample** file. Figure 24.2 shows an example of what the HTML output looks like for the **ad_track_sample.html** file. The HTML for **ad_track_sample.html** follows:

```
<HTML>
<HEAD>
<TITLE>
Sample For Hit Counter
</TITLE>
</HEAD>
<BODY>
```

```
<A HREF=ad_track.cgi?page=sample>
<IMG SRC=ad_track_sample.gif></A>
</BODY>
</HTML>
```

Figure 24.2 *ad_track_sample.html.*

DESIGN DISCUSSION

The purpose of **ad_track.cgi** is to track a user's clicks on an advertising banner displayed at a site. The script increments a counter for the advertised page being accessed and then sends the user to that page. Figure 24.3 shows a basic diagram of this logic.

First, the script starts Perl in the **/usr/local/bin** directory. The location of the supporting files is defined. The default is to store the libraries in the **Library** subdirectory under the current directory where the main script is located. **cgi-lib.pl** is loaded along with **cgi-lib.sol**. **cgi-lib.pl** is used to parse the form variables, and **cgi-lib.sol** supplies the lock file and counter routines.

```
#!/usr/local/bin/perl
$lib = "./Library";
require "$lib/cgi-lib.pl";
require "$lib/cgi-lib.sol";
```

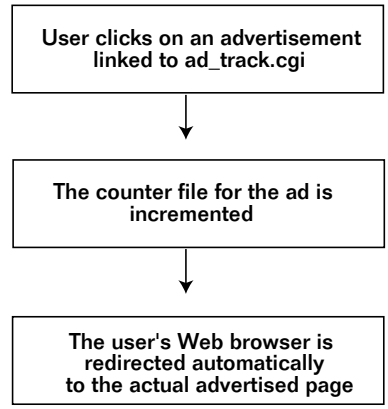

Figure 24.3 *Basic flowchart for advertisement tracking.*

Unlike almost all the other scripts in this book, the script does not print the standard "Content-type: text/html\n\n" HTTP header at the beginning. This is because the script does not output an HTML file. Instead, it outputs a special redirection signal telling the user's Web browser to go to a different URL to obtain an HTML document. The `ReadParse` function processes the incoming form variables into the `%in` associative array.

```
&ReadParse;
```

`$countfile` is set to the location and filename of the file that maintains the count of hits for the page being accessed by the user. The count file begins with `count_` and ends with the contents of whatever the `page` form variable is set to. The counter file is located in the **Counter** subdirectory underneath the current directory.

```
$countfile = qq!Counter/count_$in{"page"}!;
```

We use a Perl technique to delimit strings using the ! symbol instead of double quotes. This method lets us use double quotes within a string without having to escape them. The code looks more readable, because we need not escape the double quotes when referencing the form variables inside the associative array. This technique is documented in Appendix A.

`$urlfile` contains the location and filename of the file that contains the URL of the page the user is sent to after the counter file is updated. Like the counter file, the URL file is located underneath the **Counter** subdirectory. The name of the file is **url_** plus the value of the `page` form variable.

```
$urlfile = qq!Counter/url_$in{"page"}!;
```

First, the file containing the URL for the page is opened. Then a `while` loop is used to read the file. Any line that has a length greater than 1 becomes the new URL to which the user is redirected.

```
open (URLFILE, "$urlfile") ||
    &CgiDie("$urlfile won't open");
while (<URLFILE>)
     {
     chop;
     if (length($_) > 1)
          {
          $url = $_;
          }
     }
close URLFILE;
```

The `counter` routine in **cgi-lib.sol** is called to increase the counter for the page. The lock file routines in **cgi-lib.sol** are also used to protect the counter file so that no other instance of this script can change the counter file while the current instance is manipulating it. Remember, more than one person may be running the same script at the same time on the Web.

```
&GetFileLock("$countfile.lock");
&counter($countfile);
&ReleaseFileLock("$countfile.lock");
```

Finally, the special HTTP header "Location:" is printed to the Web server. This action causes a redirection signal to be sent to the user's Web browser. In this case, the `$url` is sent to the user's Web browser to tell it to move to the new site right away. The program then ends.

```
print "Location: $url\n\n";
```

Image maps work in the same basic way. When you click on an image map, a CGI program can be executed that accepts the coordinates as form variables and then translates them to a URL. This URL is sent to the user's Web browser, which redirects the user elsewhere.