SCRIPT_NAME=/path/script_name—The path and name of the script being accessed as it would be referenced in a URL. This can be used to construct URLs, for insertion in script-generated HTML documents that refer back to this same script. For example, the concatenated string

```
http://$SERVER_NAME:$SERVER_PORT$SCRIPT_NAME
```

(the *$NAME* here refers to the contents of the environment variable *NAME*) generates the URL of the script from information contained in the environment variables.

QUERY_STRING=query_string—This contains the query string that follows the question mark in a URL; in this case, ian+bradley. It is still encoded so that blanks are represented by plus signs and FORM name=value pairs will be separated by the ampersand character. The person writing the gateway program must write code to interpret the information in the query string. If this string results from an ISINDEX search request, then *QUERY_STRING* information is also passed to the program as command-line arguments. However, you can access the same information from the *QUERY_STRING* variable.

REMOTE_HOST=domain.name—This contains the Internet domain name of the host making the request. If the domain name is unavailable, this field is left blank. The numerical IP address is available in the *REMOTE_ADDR* variable. In this example, I accessed the server on leonardo from leonardo itself.

REMOTE_ADDR=xxx.xxx.xxx.xxx—This contains the numeric IP address of the remote host accessing the server.

AUTH_TYPE=type—This lists the authentication method required to authenticate a user who desires access, and is used only with scripts that are so protected. Many servers contain configuration files to indicate which gateway programs (or regular files) require authentication prior to access. If authentication is not required, this variable is empty.

FreeWheel, Ex. 1222, Part 2 of 2

REMOTE_USER=name—This is set only if authentication is required, and contains the authenticated name of the user gaining access. This can be used to record the name of the user and to control access, depending on the user's identity. This field is left blank if authentication is not required.

REMOTE_IDENT=name—This contains the remote username retrieved from the server using the *identd* identification daemon.

CONTENT_TYPE=MIME_type—If the client is POSTing or PUTting data to the server, this variable contains the MIME content-type of the data. If no data is being sent, this is left blank. The data itself is available to the gateway program by reading from standard input.

CONTENT_LENGTH=length—If the client is POSTing or PUTting data to the server, this variable contains the length of the data message. If no data is being sent, this is left blank. The gateway program does not have to read all the data before returning data to the client or before exiting.

In addition to these server-specific environment variables, every piece of information found in the HTTP request headers (the headers sent from the client to the server to access the script) is passed to the gateway program as an environment variable. These variable names are constructed by capitalizing the name in the request header field and adding the prefix *HTTP_*. Thus, the accept headers becomes the environment variable *HTTP_ACCEPT*. In the examples, the browser sent only two types of request headers; namely, the accept and User_Agent headers (see, for example, Figures 4.1 or 4.4), so there are only two *HTTP_* environment variables:

HTTP_ACCEPT =type/subtype, type/subtype ....—This contains a comma-separated list of all the MIME-types acceptable to the client, as indicated by the Accept headers sent from the client to the server.

HTTP_USER_AGENT=program/version—This contains the contents of the User_Agent field in the request headers. This information can be used to modify the behavior of a gateway program, depending on the capabilities of the client.

THE POST METHOD AND STANDARD INPUT

The next two examples look at the data passed to the program shown in Figure 4.16 from an HTML form. The examples access this script using the two HTML FORM examples discussed in Examples 3 and 4 (the FORM is shown in Figure 4.5, and shown rendered by a browser in Figure 4.6). The script prints out the relevant environment variables, and also reads data from the input (the `read var` command on the seventh line from the bottom) and prints the resulting input to standard output.

EXAMPLE 9: FORM-BASED GET REQUEST

This example accesses the program shown in Figure 4.16, using the FORM template shown in Figure 4.5—that is, a GET request. Figure 4.6 shows the browser display of this FORM prior to sending the query to the server, while Figure 4.7 shows the data that is sent to the server and shows how the FORM data are encoded for transmission and appended to the URL.

Figure 4.17 shows the results returned by the script in Figure 4.16. You will note that there are no command-line arguments. In parsing the URL, the server detected equal signs in the query string. This implies a non-ISINDEX query, so the server does not create command-line arguments. The remaining quantities are obvious. The *REQUEST_METHOD* environment variable is set to GET, and the query string is placed in the *QUERY_STRING* environment variable. The *CONTENT_TYPE* and *CONTENT_LENGTH* variables are empty, since there is no data sent in a GET method, while the *PATH_INFO* and *PATH_TRANSLATED* variables are also empty, since there was no extended path information in the query.

```
#!/bin/sh
echo Content-TYPE:  text/html
echo
```

```
# is a FORMs test script -- it prints the environment variable
# contents generated by a FORM access to this script.
echo "<HEAD>"
echo "<TITLE>FORMs Test Page </TITLE>"
echo "</HEAD>"
echo "<P> Number of Command-line Arguments = $#. They are:" for i in $*
do
   echo " <code> $i </code> "
done
echo "<h2> The Environment Variables </h2>"
echo "<pre>"
echo "SERVER_NAME = $SERVER_NAME"
echo "SERVER_PORT = $SERVER_PORT"
echo "REQUEST_METHOD = $REQUEST_METHOD"
echo "PATH_INFO = $PATH_INFO"
echo "PATH_TRANSLATED = $PATH_TRANSLATED"
echo "SCRIPT_NAME = $SCRIPT_NAME"
echo "QUERY_STRING = $QUERY_STRING"
echo "CONTENT_TYPE = $CONTENT_TYPE"
echo "CONTENT_LENGTH = $CONTENT_LENGTH"
echo
if [ -n "$CONTENT_LENGTH" ]; then # Read/print input data (if any).
  echo "<H2>data at Standard Input is:</h2>"
  echo "<PRE>"
  read "var"  # read data from standard input into "var"
  echo "$var" # print var to standard output
  echo "</PRE>"
else
  echo "<h2> No Data at standard input </h2>"
fi
echo "</BODY>
```

Figure 4.16 Test script ***form1*** accessed by the HTML **FORM** in Figure 4.5. This script returns an HTML document listing the script command-line arguments (if there are any), the contents of all the environment variables, and any data read from standard input (if any exists).

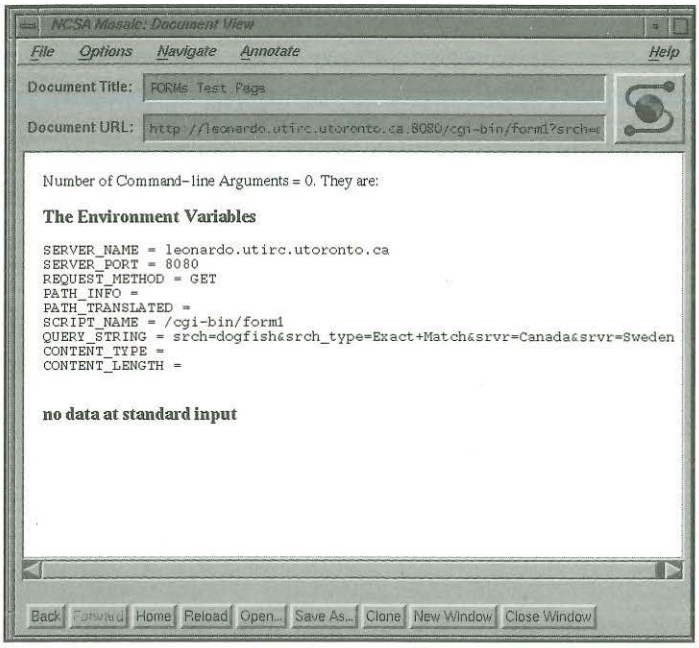


Figure 4.17 Data returned from the script shown in Figure 4.16 when accessed by the **FORM** shown in Figure 4.7 using the GET HTTP method.

Further processing requires more sophisticated programming tools to parse the *QUERY_STRING* and break it into its component parts. This is not difficult, recalling that: the ampersand character divided the different segments; the equal sign is used to relate FORM variable names to the assigned values; and spaces in the query strings are translated into plus signs. Finally, you must unencode all the special characters that may have been encoded, using the URL encoding scheme discussed in Chapter 3. Chapter 5 lists some useful CGI libraries that can help you do this processing.

EXAMPLE 10: FORM-BASED POST REQUEST

This example again accesses the program shown in Figure 4.16 using the FORM shown in Figure 4.5, but, this time, using the POST method.

Figure 4.8 shows the data actually sent to the server. In this case, the data is sent to the server as an encoded message following the headers. There are two extra headers: the content-length header, which tells the server the length of the following message; and the content-type header, which tells the server that this is an `application/x-www-form-urlencoded` MIME-type. This is a special MIME-type used to indicate FORM data that has been encoded using the URL encoding scheme.

Figure 4.18 shows the results returned by the script in Figure 4.16, and displays the data that arrived at the script. There are no command-line arguments—this time, because there is no query string. Most of the environment variables are the same as with the GET request shown in Figure 4.17. Obvious differences are the *REQUEST_METHOD* variable that is now POST instead of GET, and the null *QUERY_STRING*. In addition, the *CONTENT_TYPE* and *CONTENT_LENGTH* are not empty but contain the length of the message and the content-type, as indicated in the fields sent by the client.

Where is the query data? With the POST method, this is sent to the gateway program as an input stream, which the program reads from standard input. The script in Figure 4.16 reads this standard input and prints the results back to standard output. This is printed at the bottom of Figure 4.18 and is just the query data sent by the client (and shown at the bottom of Figure 4.8). This is encoded using the URL encoding mechanisms. To further process this data, you must parse it to separate the fields. Again, there are CGI libraries, discussed in Chapter 5, that can help in the processing of these data.

Example 10 may seem similar to Example 9 but is, in fact, different in important ways. First, many systems have a finite data space for environment variables so that large messages passed via GET URLs can be truncated. In addition, the POST method allows for complicated MIME messages to be sent from client to server, something that is difficult if not impossible with the GET method.

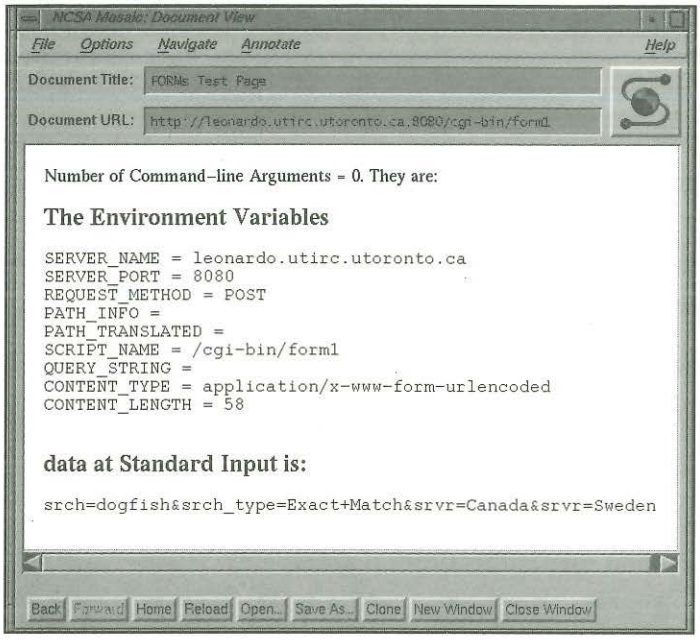


Figure 4.18 Data returned from the script shown in Figure 4.16 when accessed by the **FORM** shown in Figure 4.7 and modified to use the POST HTTP method.

SECURITY CONSIDERATIONS

There is always a security risk associated with running a gateway program on a server, since a rogue program can easily corrupt the data files being managed by the server. Most HTTP servers can restrict executable programs to special URLs (typically, those pointing to the directories */cgi-bin* or */htbin*). The server administrator can maintain strict control over the installation of programs in these areas, and can verify that installed gateway programs will not be dangerous to the integrity of the server database.

The details of these security features depend strongly on the server that you are using. In general, most servers allow significant customization of these

features. Chapter 7 gives a brief review of different servers and their capabilities in this regard.

REFERENCES

The HTTP specifications are currently under development. Some useful reference documents are:

Working Draft on the HTTP protocol:

```
http://info.cern.ch/hypertext/WWW/Protocols/HTTP/HTTP2.html
```

CERN data encryption and user-authentication notes:

```
http://info.cern.ch/hypertext/WWW/AccessAuthorization/Overview.html
```

NCSA data encryption and user-authentication notes:

```
http://hoohoo.ncsa.uiuc.edu/docs/PEMPGP.html RFC931 References
```

NCSA Mosaic authentication tool:

```
http://wintermute.ncsa.uiuc.edu:8080/auth-tutorial/tutorial.html
```

CERN proxy HTTP server:

```
http://info.cern.ch/hypertext/WWW/Daemon/User/Proxies/Proxies.html
```

Using proxy gateways (Mosaic for X-Windows):

```
http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/proxy-gateways.html
```

CGI specifications:

```
http://hoohoo.ncsa.uiuc.edu/cgi/overview.html
```

Demonstration CGI documents:

```
http://hoohoo.ncsa.uiuc.edu/cgi/ex mples.html
```

CHAPTER

5

HTML AND CGI TOOLS

This chapter discusses programs or tools that are useful in developing or presenting HTML documents on the World Wide Web. The first half covers tools useful in developing HTML documents, such as image processing tools, the active image map facility, and programs for automatically generating a Table of Contents from a collection of HTML documents. The second half of the chapter looks at useful CGI programs. This includes programs for sending mail messages, for accessing databases, and for processing query data. The chapter concludes with some references to CGI gateway program archive sites and to sites containing up-to-date information on World Wide Web database techniques.

IMAGES IN HTML DOCUMENTS

Inlined images are one of the most useful features of HTML. The image tag `<IMG SRC="image_url">` allows you to include images within your document and have them displayed with the text, provided the browser is capable of doing so. But, don't forget about clients using text-only browsers. Adding an `ALT="description"` attribute to an image element is only a little extra work and is well worth the effort.

Although some browsers support multiple image formats, there are only three formats that are always viewable. These are the GIF format images (GIF87 and GIF89), X-Bitmaps, and X-Pixelmaps. The different file types can usually be inferred from the filename suffix: GIF image files usually have the suffix *.gif*; X-Bitmaps, the suffix *.xbm;* and X-Pixelmaps, the suffix *.xpm*.

X-BITMAP IMAGES

X-Bitmaps are a common format on UNIX workstations, and you will often find them in image and icon libraries. These simple black-and-white bitmaps are useful, as most browsers interpret the white portion of the image as *transparent*. This means that the black part of the image is displayed in black, while the white part is treated as transparent and takes on the color of the underlying window. This makes for much more attractive graphics, since the square box surrounding the black image is invisible to the user. However, X-Bitmaps are an inefficient way to store an image and are uncommon outside the UNIX environment.

X-PIXELMAPS

X-Pixelmaps are similar to X-Bitmaps but assign 8 bits to each pixel. This allows for images containing 256 distinct colors, and X-Pixelmaps are often used for X-Windows icons. The main drawback of X-Pixelmaps (and X-Bitmaps) is size: X-Pixelmaps are an inefficient way of storing image

information so that the image files tend to be much larger than corresponding GIF images.

GIF IMAGE FILES

GIF is the most common image format in World Wide Web applications. This format can store black-and-white or color images, although it is limited to a maximum of 256 colors per image. The GIF format codes the image information using a *color indexing* scheme. When you create a GIF image, the software takes the raw image data and tries to find the best set of 256 (or fewer) colors for representing the image, and creates a *color table* containing these colors. For example, if the picture were of a red sunset, this table might contain mostly reds; whereas if it were a picture of a forest, it might mostly be greens. The software then goes through the image and assigns to each pixel the number from 0 to 255 corresponding to the color in the color table that comes closest to the actual color of the pixel. The result is a GIF image file consisting of an array of color indices, plus a *color map* table that maps each of these indices onto a particular color. The combination can yield a breathtakingly successful rendering of the original image, even though there are only 256 colors present.

There are two versions of the GIF format: GIF87 and GIF89. The GIF89 format is more sophisticated than the GIF87, and particularly allows you to select one of the colors in the image as *transparent*. The use and creation of transparent GIFs are described later on.

REDUCING IMAGE FILE SIZE: THE COLOR MAP

Color images are much bigger than black-and-white images, since they contain a lot more information: 8 or more bits of information per pixel, as opposed to only 1 bit per pixel for black and white. Also, files of images that contain more colors are often bigger than those that contain only a few colors, even if they are physically the same size. This is because the GIF format is clever enough to know, if an image has only eight colors

instead of 256, that it needs fewer bits per pixel to map each color properly. Thus, if you want to make your image files small (which is always a good thing on the World Wide Web), you should use as few colors as possible. Often, you can use image-processing programs, such as **Adobe Photoshop**, **Paintshop Pro** (a shareware program for Windows PCs), **Graphic Converter** (a shareware program for the Macintosh), or the **pbmplus** package (a collection of shareware programs for UNIX workstations), to process images and reduce the number of colors they contain.

There is another reason to reduce the size of the color map in an image, due to the limitations of the color display capabilities of many computers. Most computers are equipped with graphics cards capable of displaying at most 256 colors. If your WWW browser tries to display a 256-color GIF image, things will look fine. However, suppose your browser is displaying two GIF images at the same time, each containing 256 colors, and each image with its own color map. Clearly, this is a problem—both color maps cannot be used, since this requires 512 colors, not 256. Most browsers try to *dither* the color—that is, they try to find an average color map that is acceptable to both images. This works sometimes, and sometimes fails miserably, resulting in poor-quality color images.

You can alleviate this problem in two ways: by ensuring that the images you have created for simultaneous display all use the same color map, or by limiting each image to at most 40 or 50 colors. The former solution is useful, but difficult to implement. The second option can be accomplished using a number of image-processing programs, including the ones previously listed. Almost all of them allow you to reduce the size of the color table and then dither the remaining colors to make the image look as good as possible. This allows you to display four or five images together before you *run out* of space in the color map. It also has the side effect of reducing the size of the image files, which is a nice bonus.

REDUCING IMAGE SIZE: RESCALING IMAGES

There are other image-processing tricks to consider. For example, you may want to reduce the size of an image either because the original is just too big, or because you want to create a small icon of the image and link it to the bigger version. You can create reduced image sizes with a variety of programs, including the ones listed previously. When you *shrink* an image you may also want to *smooth* it at the same time. Smoothing reduces edge sharpness created by the size reduction process. Some programs do this automatically, while others let you turn on and off this feature. Note that you want to smooth while doing the size reduction—not after—since, once the image is shrunk, the smooth function simply blurs what's left.

TRANSPARENT GIF IMAGES

Unlike X-Bitmaps, there is no implicit transparent color for a GIF image. The background of a GIF image is simply another color which is displayed as such when the image is viewed by a WWW browser. This is inconvenient if the image is simply a black-and-white logo, a color *bullet* on a plain background, or an equation to be presented inline with the text. Fortunately, the GIF89 format allows you to declare one of the color indexes as transparent, giving the same transparency features as the black-and-white X-Bitmaps. To enable transparency, you find the color index value that corresponds to the part of the image you want to make transparent (usually, the background color) and then use a special program to modify the image file to make this color index transparent. The procedure is outlined in the following section.

TRANSPARENCY (MACINTOSH)

Aaron Giles of Cornell University Medical College has developed an elegant and useful program called **transparency**. This simple, graphically oriented program allows you to easily edit GIF images and make one of the colors transparent.

You operate **transparency** by either dragging a file to the transparency icon or by double-clicking on the transparency icon and opening the desired GIF image file. To select the transparency color you simply place the mouse pointer inside the image and hold down the mouse button. You are then presented with the color palette for the image, and can select the color you wish to make transparent by putting the mouse pointer over the desired color and releasing the button. You can choose to have no transparent color by selecting the NONE bar at the top of the palette. Upon releasing the mouse button, there is a short pause, after which the image is redrawn with the selected color rendered transparently. You now select the Save as GIF89... menu from the pull-down File menu to save the newly transparent version.

Transparency is available at a number of anonymous FTP sites, including:

```
ftp://ftp.med.cornell.edu/pub/aarong/transparency/
```

The program is found in a file named *transparency*sit.hqx*, where the star represents the version number. Additional information about **transparency** can be found at:

```
http://www.med.cornell.edu/~giles/projects.html
```

GIFTRANS (UNIX AND PC-DOS)

Giftrans is a simple C-language program, written by Andreas Ley, that can convert any GIF files into the GIF89 format. While making this conversion, **giftrans** also allows you to make one of the colors transparent. It is currently available in source code (for compilation on any platform, but most specifically for UNIX workstations) and as an executable program for PCs running DOS. **Giftrans** is a command-line program, so it does not need Windows or a GUI to run. If you have the correct information, you can run **giftrans** from a simple text terminal.

Typically, **giftrans** is used as follows:

```
giftrans -t xx image.gif > transparent_image.gif
```

which translates the file *image.gif* into *transparent_image.gif* and makes the color labeled by xx transparent. There are several ways of labeling the colors, the most common being to:

- Specify the absolute RGB value for the color (as a 24-bit RGB value). For example, the command

  ```
  giftrans -t #ffffff image.gif > transparent_image.gif
  ```

  makes the color white transparent (`ffffff` is the RGB code for white).

- Specify the color index value. For example, the command

  ```
  giftrans -t 21 image.gif > transparent_image.gif
  ```

  makes the color found in color index 21 transparent.

How do you find the color index or RGB value of the color you want transparent? To find this information, you need a graphics program that can tell you the color indices or RGB values for a given pixel. Several shareware graphics programs can do this: **xv** is a common UNIX program suitable for this task. With **xv**, you load the image file, place the mouse over the desired color, and depress the left mouse button. This gives the mouse coordinates with respect to the upper left-hand corner of the image, plus color information for the pixel. This is presented in the following manner:

```
132, 34  5 = 203, 203, 203 (0, 1, 79 HSV)
```

The first two numbers are the x and y coordinates of the pixel. The number just before the equal sign is the color index, while the next three numbers are the RGB values for this color (in decimal numbers). The three numbers in brackets are an alternative color coding scheme.

To make this color transparent, you turn to **giftrans**. You have two ways to make the preceding color transparent, using the color index:

```
giftrans -t 5 image.gif > transparent_image.gif
```

or the RGB value:

```
giftrans -t #cbcbcb image.gif > transparent_image.gif
```

where cb is the hexadecimal code for the decimal number 203. If you run **giftrans** without any arguments, the program prints out a list of possible arguments and their meanings.

Giftrans is available from a number of anonymous FTP sites. Its original home is:

```
ftp://ftp.rz.uni-karlsruhe.de/pub/net/www/tools/
```

which contains the files:

giftrans.exe	DOS executable version of **giftrans**
giftrans.c	Source code for **giftrans** (if you are compiling on a PC, you also need the file ***getopt.c***)
giftrans.1	UNIX-style manual page for **giftrans**; useful also with DOS

The program **xv** is available from many anonymous FTP sites. You will need both a C compiler and the standard X11 libraries to compile **xv** on your machine. **Xv** is available at:

```
ftp://ftp.cis.upenn.edu/pub/xv/xv-3.00a.tar.Z
ftp://ftp.ira.uka.de/pub/x11/xv-3.00.tar.Z
ftp://sunb.ocs.mq.edu.au/X11/Graphics/xv-3.00.tar.gz
```

ACTIVE IMAGES

Active images, clickable images—whatever they are called, you have certainly seen them and you would love to have them. Having an active image means that you can click your mouse on the image and have different things happen, depending on where you clicked. For example, the active image could be a city map, where clicking on different locations inside the image returns information about particular buildings, transportation routes, or historic monuments.

Active images require special features on both the client and the HTTP server. First, the client must be able to measure the coordinates of your

mouse pointer when you click on the active image, and must be configured to send this information to an HTTP server gateway program for interpretation and action. Second, there must be a gateway program on the server capable of interpreting this image data. Third, there must be a database on the server relating, for this particular image, the click coordinates to the appropriate action. This means you have to sit down with your image file, mark out regions of the image, link them to particular functions, and store this information in a file that can be read by the gateway program.

Allowed Formats for Active Images

You must use GIF format for active images. Several browsers cannot measure image coordinates in X-Bitmaps, so active images will not work with this format. If you have an X-Bitmap or X-Pixelmap image that you want to use as an active image, you must convert it into the GIF format. The various image-processing programs mentioned earlier in this chapter can all do this conversion.

Things to Think about Before Starting

Before you get carried away with active images, stop and think about your audience. First, you must recall that not everyone will be able to view your active images, either because their browser does not support that capability (rare now but still possible), or because they are using a text-only browser, such as **lynx** (quite common). Also, people geographically far away or linked by slow Internet connections often disable image loading because of the inherent slowness in accessing large image files. Consequently, you should make the images as small as possible and should try to provide a text-only way of accessing the same data. For example, your active image document can have a line of text explaining what the image does and offering a hypertext link to a document providing a text-only approach to the same information. This will involve a little extra work, but the customer satisfaction is well worth the effort.

Making the Image Active

In HTML, you include an image within a document using the IMG element. For example:

```
<IMG SRC="image.gif">
```

This image is passive and just sits there. As discussed in Chapters 1 and 2, you can turn this image into a hypertext link by putting it inside an anchor element; for example:

```
<A HREF="http://some.site.edu/linked_doc.html">
   <IMG SRC="image.gif">
</A>
```

Now, when you click on the image, you are linked to the indicated document. In a sense, this is an active image but the action is restricted to a single function.

To make an image fully active, you must do two things. First, you need to add the **ISMAP** attribute to the **IMG** element, which tells the browser that this is an active image. Second, you must surround the **IMG** element with a hypertext anchor that points to the gateway program that can process the image and coordinate data. Thus, the HTML markup for an active image is:

```
<A HREF="http://some.site.edu/cgi-bin/imagemap/my_database">
        <IMG SRC="image.gif" ISMAP>
</A>
```

This tells the browser that this is an active image and that, when a user clicks inside the image, the coordinates of the click should be sent to the gateway program ***imagemap*** at the given URL. The coordinate information is sent to this URL using the HTTP GET method. The HTTP request header sent to the server looks like:

```
GET /cgi-bin/imagemap/my_database?x,y HTTP/1.0
```

where `x` and `y` are the *integer pixel coordinates* of the mouse pointer measured from the *upper left-hand corner* of the image.

Note that the path *my_database* is appended to the end of this URL. As discussed in Chapter 4, URLs that point to gateway programs are treated in a special way, and any directory-like information appended to the URL after the program name (like *my_database*) is treated as *extra path* information and is passed as a parameter to the gateway program. In this example, *my_database* is additional path information used by the *imagemap* program to find the image map database for this particular image. This allows the imagemap program to be used for any number of active images, each image having its own personalized database.

Imagemap is an actual CGI program for handling active image data, and is distributed with the current NCSA HTTPD server. An older version of this program is also available and has similar, if somewhat more limited, functionality. The newer and more flexible version is available at:

```
http://hoohoo.ncsa.uiuc.edu/docs/setup/admin/imagemap.txt
```

Imagemap is a CGI-compliant C-language program and can be used on any HTTP server that supports the CGI specifications. The remainder of this discussion centers on using the newer **imagemap** program and on creating the associated databases. Additional online documentation, including interactive examples, can be found at:

```
http://hoohoo.ncsa.uiuc.edu/docs/setup/admin/NewImagemap.html
```

INSTALLING IMAGEMAP

If you are installing the **imagemap** program, or are replacing your old version, you simply take the file:

```
http://hoohoo.ncsa.uiuc.edu/docs/setup/admin/imagemap.txt
```

rename it *imagemap.c*, and compile with your C compiler. You should then install the resulting *imagemap* program in your server CGI executable directory. On NCSA servers, this is usually the *cgi-bin* directory located in the directory containing the server executable, server support directories, and configuration files. (Note that this is *not* the directory that contains the

HTML and other documents you make available via the server.) Once installed, the program is accessible through URLs, such as:

```
http://some.site.edu/cgi-bin/imagemap
```

where the *cgi-bin* directory is a *virtual* name used by the server to reference the directory containing CGI programs. Check your server configuration files to verify that this name is correct.

CREATING THE IMAGE DATABASE

This section describes the format of the image map database and explains how it is constructed. It also describes some software tools useful in building these image map databases.

The image map database file relates a region of the image to a URL to be accessed when you click inside that region. You can specify these regions as circles, rectangles, polygons, or points. You can also include comments in these map files by placing a hash character (#) as the first character in the line. For example, `here` is a simple map file named ***blobby.map*** (Figure 5.1):

```
# Imagemap file for blobby.gif
circle    /dir1/blob2/his_head.html        50,20 50,30
rect      /dir1/blob2/his_left_foot.html   25,78 40,85
poly      /dir1/blob2/his_body.html        45,38 35,50 40,72 50,75 60,72 65,50 55,38
default   /cgi-bin/nph-no_op.sh
```

Figure 5.1 The ***blobby.map*** image map database.

This file declares the circle centered at coordinates 50,20 (and with an edge at 50,30) to be linked to the designated URL, and makes similar declarations for a rectangle (indicated by `rect`) and a polygon (indicated by `poly`). These coordinates are measured in pixels from the upper left-hand corner of the image. The default method indicates the URL to access if the user clicks in places not falling inside the mapped regions.

The general form for map file entry is:

```
method  URL  x1,y1 x2,y2 ... xn,yn
```

where `method` specifies the manner in which the region is being specified (`circle`, `rect`, `poly`, `point`, `default`); `URL` is the URL to be accessed if the click occurs inside this region; and `xn,yn` are the integer coordinates of a point measured from the upper left-hand corner of the image. These coordinates are measured in pixels, so you will need some way of measuring the pixel coordinates in an image. Tools for doing this are presented later.

Note that mapped regions can overlap. The **imagemap** program accesses the map file from the top so that if a click occurs at a point lying within two mapped regions, the program takes the one nearest the top of the file.

The URLs specified in a map file can be complete URLs or partial URLs of the form:

```
/path/stuff/file
```

This references a file or gateway program relative to the HTTP server *document directory* (see Figure 1.11). With the NCSA server, you can also use the form:

```
/~user/stuff/file
```

where `user` is the name of a user on the system. This references a file or gateway program relative to the user's personal HTML area.

The following is a list of the different methods for declaring active regions and the parameters required for these methods:

`circle  URL  center edgepoint`—This maps the region inside the indicated circle to the given URL. Two coordinate pairs `x,y` are required: one for the circle `center`, and the other for an `edgepoint` lying on the edge of the circle.

`point  URL  x,y`—This declares a specific *point* in an image as active, and is really useful only if you have more than one active point. When you click on the image, and the click is not inside a circle, rectangle, or polygon,

the **imagemap** program locates the point closest to the coordinates of the click and accesses the indicated URL.

`poly URL x1,y1, x2,y2, ..... xn,yn`—This maps the region inside the indicated multisided polygon to the given URL. Each coordinate pair represents a vertex of the polygon. You should make sure that the line segments do not cross one another. The polygon is automatically closed by linking the last point `xn,yn` to the first point `x1,y1`. The current NCSA program limits you to, at most, 100 corners in a given polygon.

`rect URL upper_left_corner lower_right_corner`—This maps the region inside a rectangle to the given URL. The `x` and `y` coordinates of the upper left and lower right corners of the rectangle are required, in that order.

`default URL`— This URL is accessed if the click did not lie inside any other region. This is never accessed if you define a point, since a clicked location will always be *closest* to that point.

Referencing the Database

Now that you have a database—for example, the ***blobby.map*** database of Figure 5.1—where should you put it and how do you reference it? This database can go anywhere you can put regular HTML documents or data files; you indicate its location to the **imagemap** program via the URL. For example, let us suppose that the HTTP server document directory is ***/u/We***, and that the collection of documents and images related to blobby is found in the directory ***/u/Web/weird/blobby***. A reasonable choice is to put the map file in the same directory as the image so that the absolute path for the map file is ***/u/Web/weird/blobby/blobby.map***. You can access this map file from the ***imagemap*** program by writing in your HTML document:

```
<A HREF="http://some.site.edu/cgi-bin/imagemap/weird/blobby/blobby.map">
   <IMG SRC="blobby.gif" ISMAP>
</A>
```

where ***some.site.edu*** is the site containing all these documents. Note that the path to the ***blobby.map*** file, relative to the document directory of the

HTTP server (*/u/Web*), has been added to the imagemap URL. The string `/weird/blobby/blobby.map` is passed to the **imagemap** program, and is used by **imagemap** to locate the database *blobby.map*.

With the NCSA server, this method is not restricted to the regular document directory. If a user has created a personal HTML directory, he or she can place map files in this directory and access the map using URLs, such as:

```
<A HREF="http://some.site.edu/cgi-bin/imagemap/~user/path/blobby.map">
<IMG SRC="blobby.gif" ISMAP>
</A>
```

where user is the username of the person with personal HTML documents, and *path/blobby.map* is the path leading to the map file from the root of his or her personal document directory.

Getting a Click to Do Nothing

Sometimes, you want a click to do nothing: For example, your image could be a map of buildings, but you don't want anything to happen when the user clicks on nonmapped objects, such as roads or trees. The ideal solution would be for the client to retain the current document on the browser window screen. You can ask the client to do this by linking the `default` method in your imagemap file to a server script that sends a message to the client telling it to do exactly this: Do nothing and keep displaying the same page. This is the intent of the default method in Figure 5.1. This default method references the script *nph-no_op.sh,* which contains the five simple lines:

```
#!/bin/sh
echo "HTTP/1.0 204 No response -- server CGI-script output"
echo "Content-type: text/plain"
echo "Server: $SERVER_SOFTWARE"
echo
```

Output from a script with the *nph-* prefix is sent directly to the client without being parsed by the server (see Chapter 4 for more information about *nonparsed header* gateway programs). This script sends the HTTP status code 204, which tells the client that there is no server message and that the client should retain the current document.

Tools for Generating Imagemap Files

Generating a map file is not difficult and requires only some concentration, a piece of paper, and an image viewing program that gives you the pixel coordinates of the mouse pointer. The generic UNIX program for this purpose is **xv** (found at many anonymous FTP sites by searching for the string `xv-3.00`). You load the image into **xv**, locate the coordinates, and type the numbers into a map file. A similar procedure can be followed with typical Macintosh or PC image editing programs.

Of course, this process would be much easier if you had a graphical tool for editing images and automatically generating the map files. Thomas Boutell has satisfied this need with his program **mapedit**, which allows you to read the GIF image into a resizable window; use the mouse to draw circles, rectangles, and polygons on top; and specify a URL for each of the marked regions. You can also insert comments as you go, which is important if you want to understand the contents of the map file at a later date.

Mapedit is available for both UNIX systems and PCs running Windows. Both version are found at:

```
ftp://sunsite.unc.edu/pub/packages/infosystems/WWW/tools/mapedit
```

The PC version comes as a *zip* archive containing the executable program, support files, and documentation; while the UNIX version comes as a compressed tar file (the current version is in the file *mapedit1.1.2.tar.Z*). The UNIX package must be compiled for your system. You will need an ANSI C compiler, The X-Windows libraries at level X11R5 or higher, the X11 imake utility (usually a part of the X distribution), and the X11 Athena

widget set. Compilation instructions come with the installation kit. Additional information about **mapedit** can be found at:

```
http://sunsite.unc.edu/boutell/mapedit/mapedit.html
```

ICON ARCHIVE SITES

There are several sites on the World Wide Web that maintain extensive archives of image icons. These are often ideal places to find icons for use in your own document development. The following is a list of some of the more popular sites, along with pointers to more extensive lists of icon and image archives:

Icon Archives

```
http://www-ns.rutgers.edu/doc-images/
http://www.dsv.su.se/~matti-hu/archive.html
http://www.di.unipi.it/iconbrowser/icons.html
http://alice.cli.di.unipi.it/iconbrowser/icons.html
http://www.cit.gu.edu.au/~anthony/icons/index.html
```

Other Lists and Image Archives

```
http://white.nosc.mil/images.html
http://akebono.stanford.edu/yahoo/Computers/World_Wide_Web/Programming/Icons/
http://oneworld.wa.com/htmldev/devpage/dev-page3.html#doc-i
http://www.cs.yale.edu/HTML/YALE/CS/HyPlans/loosemore-sandra/clipart.html
```

CLIENT-SIDE EXECUTABLE PROGRAMS

The emphasis up to this point has been on programs that run on the server using data passed from the client. Sometimes, however, you want to execute a program on the *client*. For example, you might want the user to be able to start up a USENET newsreading program on the client by pressing a FORM button or a hypertext link. Or, you might want the user to be able to start up an interactive teleconferencing session in the same manner.

Both these situations require that the server send information to the client, requesting the execution of a program, or actually send a program to the client for execution. For example, the newsreader could be launched by sending the client a simple startup script, while the teleconferencing example would require a more complicated script that launches the teleconferencing program, and also provides the information needed to contact the other party.

Although clearly a powerful tool, client-side executable programs incur many security risks, because the program arriving from the server is an unknown quantity—the client has no way of knowing what it will actually do. For example, a vicious server administrator could create a script that deletes files on the client computer. These real possibilities warrant efforts on your part to protect your system.

The following discussion outlines how client-side program execution is allowed, and includes hints to ensuring the security of this procedure. It also discusses the packages **vsafecsh** and **w3launch,** which are useful tools for facilitating safe execution of client-side programs.

SENDING THE SCRIPT TO THE CLIENT

The first stage is to send the executable script or program from the server to the client. This is done from a CGI gateway program, using a special MIME content-type in the response headers (see Chapter 4) that tells the client that the data being sent by the gateway program is a program or script that the client should execute. As an example, you could use the header:

```
Content-type:  application/x-www-local-exec
```

Note that the type *must* be application, while the subtype name must begin with `x-`. The `x-` prefix is used for all local, experimental, or otherwise unregistered MIME-types. This example uses `x-www-local-exec`, but you can choose whatever you like.

Here is an example of a server-side CGI program that sends the preceding content-type header, followed by the executable script, to the client. The example is a simple UNIX Bourne-shell script:

```
#!/bin/sh
echo Content-type:  application/x-www-local-exec
echo
echo "#!/bin/sh"
#
#    Bourne shell scripts to run on the client. For example, this
#    could start up some client program, and use server environment
#    variables to control the configuration of the program
#
#    This UNIX example simply launches a local xterm window running the
#    tin Usenet newsreader program
#
echo
echo "xterm -display $DISPLAY -e  /usr/local/bin/tin"
echo
```

When a client browser accesses this CGI program, the browser receives the following message from the HTTP server:

```
HTTP/1.0 200 OK
Date: Monday, 03-Oct-94 22:19:31 GMT
Server: NCSA/1.3
MIME-version: 1.0
Content-type:  application/x-www-local-exec
Content-Length: 57

#!/bin/sh

xterm -display $DISPLAY -e  /usr/local/bin/tin
```

which includes the content-type header that was generated by the CGI program. The next step is to configure the client to know what to do with this special content-type, and with the message body.

CONFIGURING THE CLIENT

The browser must be configured to know what to do with messages of the `application/x-www-local-exec` MIME-type. To do this, you add information about this content-type to your browser's configuration files. For **Mosaic for X-Windows**, this means editing the ***mailcap*** file, which is the **Mosaic for X-Windows** file that matches MIME-types to helper programs. For this simple example, you could add the line:

```
application/x-www-local-exec;   /bin/sh %s
```

which tells **Mosaic for X-Windows** to pass any document of type `application/x-www-local-exec` to the program **/bin/sh** (the Bourne shell) for execution. The `%s` is the symbolic representation of the document sent to the client. Similar configurations are possible on Macintosh or PC browsers, in which case, you can often add the new MIME-types using a pull-down menu for helper applications.

The preceding example of a client-side program requires a UNIX client. This is true because of the UNIX Bourne-shell script being passed as an executable program, and because of the use of the UNIX **tin** newsreader as the client-side program. This illustrates how strongly client-side program execution can be tied to the client architecture. One way to resolve these differences is to have the server-side script use the *HTTP_SERVER_SOFTWARE* environment variable to determine the type of the host, and to then send the program or script file relevant to the client hardware and software.

SECURITY ISSUES

The preceding example program is harmless; however, scripts that contain the following lines pose greater concern:

```
#!/bin/sh

xterm -display my.home.display:0      # pop xterm somewhere else
```

This script would open an xterm window on the remote machine `my.home.display` but under the client user's account, thus giving the

person at `my.home.display` access to the user's entire directory. Obviously, you do not want to allow scripts to do this type of thing! Fortunately, you can add client-side security features that significantly reduce or eliminate the risk of such nightmarish events.

There are essentially two approaches to improved security. The first approach is to execute all programs on a client using a *secure* local shell. You would then modify the ***mailcap*** (or other) file to direct execution to this safe shell. For example:

```
application/x-www-local-exec;  /usr/local/bin/secure_sh %s
```

where `secure_sh` is a special shell program that includes extra security features. This might run under the system's restricted shell (often called **rsh**), which restricts a program's ability to modify variables or files, or it might contain internal tests that check for dangerous commands. (The **vsafecsh** program, described in the following section, is one package that implements this type of protection; in this case, for scripts that are executed using the UNIX *csh* shell.)

The second option is to design special client-side programs, place them on every client and in a special directory, and allow the program passed from the server to access *only* these predesigned packages. For example, you could prepare small client scripts for starting up the newsreader or the teleconferencing package, and allow the script sent from the server to access these scripts only. Careful design of the small client scripts can significantly enhance the security of the local systems. This, in part, is the approach taken by **w3launch**.

The following are brief descriptions of the **vsafecsh** and **w3launch** utilities.

VSAFECSH

Vsafecsh is a C-language program that acts as a protective gateway for client-side **csh** executions on UNIX platforms. Ordinarily, you enable **csh** execution on a client with a ***mailcap*** entry, such as:

```
application/x-www-localc-exec; csh -f %s
```

As just described, this is not secure. Safety can be improved by replacing this entry with:

```
application/x-www-localc-exec; vsafecsh -f %s
```

which causes the **csh** script passed from the server to first be parsed by **vsafecsh**.

Vsafecsh forks and executes *only* those executable programs that are authorized by the user. Currently, the list of allowed programs is compiled into the code of **vsafecsh**, so the user or workstation administrator has to recompile **vsafecsh** for any newly allowed programs. However, this is a small price to pay in exchange for safe client-side execution.

Additional information about **vsafecsh** is found at:

```
http://www.eit.com/software/vsafecsh/vsafecsh.html
```

Vsafecsh was written by Vinay Kumar of Enterprise Integration Technologies.

W3LAUNCH

Much of the difficulty in setting up secure client-side execution is solved through a package called **w3launch**. Available for both UNIX and Windows PCs, **w3launch** allows a client site administrator to register certain special programs as *launchable* from a server script. This registration ensures that only secure programs are allowed to execute, and allows the administrator to control access to these programs. **W3launch** consists of client and server software for administering these registered programs and for creating server-side launch scripts that can access these programs.

Additional information about **w3launch** is found at:

```
http://www.leeds.ac.uk/bionet/w3launch/w3l-home.htm
```

The source codes and executables are available from this document or from the directory:

```
http://www.leeds.ac.uk/bionet/w3launch/
```

W3launch was written by Jon Maber of the University of Leeds.

SERVER-SIDE DOCUMENT INCLUDES

A recurring question by HTML authors is, Can I include a file within my HTML document in the same way I include an image? The answer is, in general, no. If you want to have documents that are created dynamically, you are supposed to use a CGI program. Needless to say, this can be annoying if all you want to do is patch a small piece of text into an otherwise stable document. Some kind of *include* HTML command would be a far easier mechanism than a full-blown CGI script.

The NCSA HTTP server allows for server-side file inclusion, using an extension to the HTML syntax that allows for *parsable* HTML. This allows you to include in your documents special HTML commands that can inline text files or other data, or even execute server programs and include the program output. This is a powerful feature but should not be abused, since every parsable file must be specially processed by the server, which significantly slows server response. You should reserve the use of parsable files to situations where it is the only realistic choice.

The server-side include feature is well documented in the NCSA HTTPD server online manuals. The following is a brief overview of this feature, and you are referred to the NCSA documentation and interactive tutorials for further details.

INCLUDE COMMAND FORMAT

The server-side include command is framed inside an HTML comment string:

```
<!-- include_command -->
```

When parsed by the NCSA HTTPD server, the entire comment string is replaced by the output of the `include_command`. Enclosing this command inside comment strings ensures that the document will not cause problems if it is served by another server. Other servers will simply deliver the document, including the comment line, and the client will treat the enclosed string as a comment and ignore it.

The general form for the include command is:

```
<!--#command arg1="value1" arg2="value2" -->
```

where `command` is the name of the command to be executed and `arg1` and `arg2` are arguments passed to the command. There must be no spaces between the hash sign (#) and the command name. The number and name of the arguments depends on the actual command; most commands take a single argument. Note that, despite this structure, this is *not* a comment statement. You *cannot* include comment descriptions inside an include command. Consequently, lines like:

```
<!--#command arg1="value1" This prints the time of day  -->
```

are invalid.

There are six possible commands: `config`, `include`, `echo`, `fsize`, `flastmod`, and `exec`. `Config` is used to configure the way the server will parse the document. `Include` is used to include another document (not a CGI program) at the indicated point, while `echo` is used to include the contents of one of the special environment variables set for parsed documents. `Fsize` and `flastmod` are similar to `echo`; `fsize` prints the size of a specified file, while `flastmod` prints the last modification date of a specified file. Finally, `exec` executes a single-line Bourne-shell command or a CGI program. For security reasons, the `exec` facility can be disabled in the server configuration files while leaving the include feature operational.

Here are the details of the six different commands, with examples.

INCLUDE

`Include` is used to include another document (or another parsed document) at the given location in the current document. `Include` can include only a document, and cannot include CGI program output. `Include` takes one argument that specifies the file to be included. The possible arguments and their values are:

`virtual="virtual/path"`—Virtual specifies the *virtual* path to the document relative to the server's document directory or a user's personal server

directory. For example, user `fosdick` with his or her own public HTML area would access files in this area with the virtual path:

```
virtual="~fosdick/path/file.html"
```

`file="relative_path/file"`—File specifies the path to the document relative to the current document. You cannot use `file` to move up in the hierarchy, only down (i.e., you can't use `../stuff.html`). For example, to include the file ***junk.html*** from the same directory you would type: `<!--#include file="junk.html" -->`

Examples of `include`:

```
<!--#include file="templates/template2.html" -->
<!--#include virtual="/path/subpath/templates/template2.html -->
```

Echo

`Echo` includes in the document the contents of a named environment variable. The variable name you wish to include is indicated via the argument `var="variable_name"`. `Variable_name` can be any of the CGI environment variables listed in Chapter 4, and can also be one of the following environment variables valid only in parsable files:

DOCUMENT_NAME—The name of the current file.

DOCUMENT_URI—The virtual path to the document, such as `~fosdick/path/file.html` or `/path/subpath/templates/template2.html`

DATE_LOCAL—The current date, using the local time zone. The format of this date can be controlled using the `timefmt` argument of the `config` command.

DATE_GMT—Same as *DATE_LOCAL*, but in Greenwich Mean Time.

LAST_MODIFIED—The last modification date of the current document. The format is specified by `timefmt`.

These environment variables are also available to any program executed using the `exec` command.

Examples of `echo`:

```
The file is found at: <!--#echo var="DOCUMENT_URI"  -->
This file was accessed at the time: <!--#echo var="DATE_LOCAL"  -->
```

FSIZE

`Fsize` includes the sizes, in bytes, of a specified file. The file is specified using the `file` or `virtual` arguments, as described with the `include` command. `Fsize` is useful for presenting information about a file to be downloaded, allowing the document to tell the user the size of the file even if the file size varies continuously (such as a mail archive). `Fsize` can also be useful for presenting file indexes that include file size information. The output format can be controlled using the `sizefmt` argument of the `config` command.

Example of `fsize`:

```
The size of the file main.html is: <!--#fsize file="main.html"  -->
```

FLASTMOD

`Flastmod` includes the last modification time of a specified file. The file is specified using the `file` or `virtual` arguments, as described with the `include` command. Like `fsize`, `flastmod` is useful for providing additional information about files that are periodically changed. The output format can be controlled using the `timefmt` argument of the `config` command.

Example of `flastmod`:

```
This file was last changed on:
<!--#flastmod virtual="/path/dir1/dir2/main.html"  -->
```

EXEC

`Exec` executes the given command or CGI program. `Exec` can have one of two possible arguments:

`cmd="cmd_string"`—Causes the string `cmd_string` to be executed using the Bourne shell **/bin/sh.** `Cmd_string` can be a simple one-line shell program

to do simple things, such as list directory contents, run a program to filter a data file for presentation, and so on.

`cmd="cgi_program"`—Executes the given CGI program, where the location of the program is given by the virtual path to the program. Note that the script must return a valid MIME-type. You cannot pass query strings or path information to the script using URL-type constructions. Thus, expressions like:

```
<!--#exec cmd="/cgi-bin/script.cgi/path1/path2?query" -->
```

are invalid. The only way you can access the script is with the command:

```
<!--#exec cmd="/cgi-bin/script.cgi" -->
```

However, suppose the parsable script ***stuff.shtml*** contained the command:

```
<!--#exec cmd="/cgi-bin/script.cgi" -->
```

and you access the file ***stuff.shtml*** via the URL:

```
...../stuff.shtml/extra/path?query_string
```

In this case, the query_string and /extra/path information *are* available to the script ***script.cgi*** called from ***stuff.shtml***. This is illustrated in an example at the end of this section.

CONFIG

The `config` command controls aspects of the output of the parsed commands, such as the formats of the date and size output strings, or the error message string to include if parsing fails. `Config` can take three different arguments, one argument per command. These are:

`errmsg="error_string"`—Gives the error string to use if there is an error in parsing an include statement.

`timefmt="format"`—`Timefmt` sets the format for printing dates. This format is specified as with the C strftime library call (strftime is commonly found on UNIX computers).

`sizefmt="bytes","abbrev"`—`Sizefmt` sets the format for the specification of file sizes. The value `"bytes"` prints file sizes in bytes, while `"abbrev"` uses kilobytes or megabytes as abbreviated forms, where applicable.

Examples of `config`:

```
<!--#config sizefmt="abbrev"  -->
<!--#config timefmt="%m%d%y"  -->
<!--#config errmsg="Unable to parse scripts"  -->
```

EXAMPLE OF SERVER-SIDE INCLUDES

The following example (shown in Figures 5.2 and 5.3) shows the use of server-side includes. The example consists of a main document ***stuff.shtml*** (the suffix ***.shtml*** is used to indicate parsable HTML documents) that includes a second parsable document ***inc_file.shtml*** and that also executes the CGI program ***test_script.cgi***. The listings for these examples are shown in Figure 5.2, while the browser rendering of the document ***stuff.shtml*** is shown in Figure 5.3.

In this example, the document ***stuff.shtml*** is accessed using the URL:

```
http://leonardo:8080/stuff.shtml/extra/path/info?arg1+arg2
```

Note that this passes query strings and extra path information to this document, just as if it were a gateway program (see Chapter 4 for more information about CGI programs and passed variables). This information should be contained in the environment variables *QUERY_STRING* and *PATH_INFO*. The documents are designed to print these variables. As you can see in Figure 5.2, these variables are empty inside the parsable HTML documents. However, as shown at the bottom of Figure 5.3, these variables *are* present inside the CGI program executed from within the parsable document. You can, therefore, access a parsable document and, through it, pass query information to a CGI program, just as if you were accessing the CGI program directly.

The include and echo commands are also illustrated in Figures 5.2 and 5.3. These can be used to print useful information about the current file. Remember, however, that you really want to avoid this type of include as much as possible, as every server execution slows the server response. For example, it is a waste of server resources to use the *LAST_MODIFIED* variable to display the last time you edited a simple HTML document, since you could just as easily add this information while editing it.

stuff.shtml

```
<html>
<head>
<title> Test of NCSA Server-side Includes </title>

<body>
<h1> Test of NCSA Server-side Includes </h1>

<pre>
Stuff.shtml was last modified: <!--#flastmod virtual="/stuff.shtml"  -->.
Size of stuff.shtml is:        <!--#fsize file="stuff.shtml"  -->.
DOCUMENT_NAME =                <!--#echo var="DOCUMENT_NAME" -->
DOCUMENT_URI =                 <!--#echo var="DOCUMENT_URI" -->
DATE_LOCAL =                   <!--#echo var="DATE_LOCAL" -->
QUERY_STRING =                 <!--#echo var="QUERY_STRING" -->
PATH_LOCAL =                   <!--#echo var="QUERY_STRING" -->
DATE_GMT =                     <!--#echo var="DATE_GMT" -->
LAST_MODIFIED =                <!--#echo var="LAST_MODIFIED" -->
</pre>

<!--#config errmsg="Unable to parse scripts"  -->

<p><em>....now include inc_example.shtml....</em>

<!--#include file="inc_file.shtml" -->
```

```
<p> <em>..... now include test_script.cgi CGI program output..... </em>

<!--#exec cgi="/cgi-bin/test_script.cgi" -->
</body>
</html>
```

inc_file.shtml

```
<pre>
Inc_file.shtml last modified: <!--#flastmod virtual="/inc_file.shtml"-->.
Size of inc_file.shtml is:    <!--#fsize file="inc_file.shtml"  -->.
DOCUMENT_NAME:                <!--#echo var="DOCUMENT_NAME" -->
DOCUMENT_URI:                 <!--#echo var="DOCUMENT_URI" -->
DATE_LOCAL:                   <!--#echo var="DATE_LOCAL" -->
DATE_GMT                      <!--#echo var="DATE_GMT" -->
LAST_MODIFIED                 <!--#echo var="LAST_MODIFIED" -->

</pre>
```

test_script.cgi (in the `cgi-bin` directory)

```
#!/bin/sh

echo "Content-type: text/html"
echo
echo "<pre>"
echo "This is  CGI script output."
echo "QUERY_STRING is \"$QUERY_STRING\"."
echo "PATH_INFO =  \"$PATH_INFO\". "
echo ""</pre>"
```

Figure 5.2 Example of NCSA server-side includes. The main file is ***stuff.shtml***, which *includes* the file ***inc_file.html*** and the output of the CGI program ***test_script.cgi***. The resulting HTML document upon accessing the URL:

```
http://leonardo:8080/stuff.shtml/extra/path/info?arg1+arg2
```

is shown in Figure 5.3.

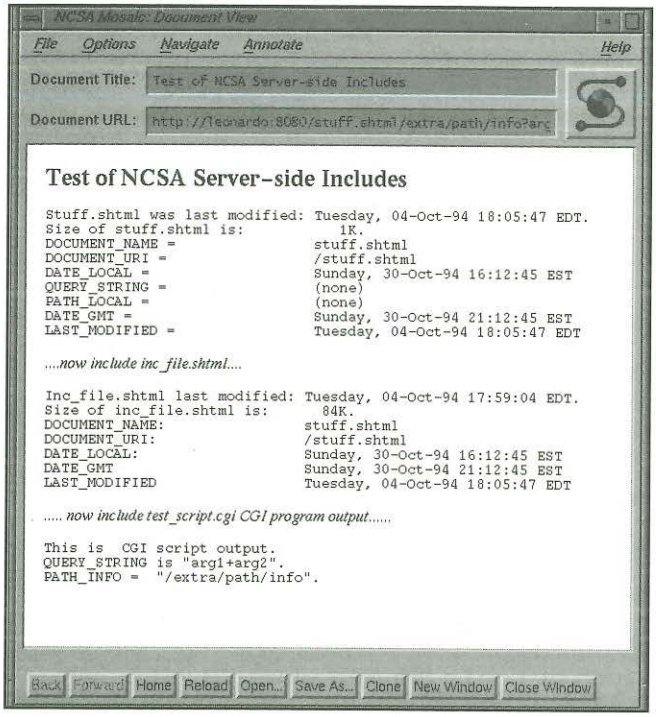


Figure 5.3 Browser rendering of the server-side executable document ***stuff.shtml***, when accessed using the URL:

```
http://leonardo:8080/stuff.shtml/extra/path/info?arg1+arg2
```

For additional information about server-side includes, and particularly for information on how to configure the NCSA HTTP server to allow server-side includes, consult the NCSA online documentation at:

```
http://hoohoo.ncsa.uiuc.edu/docs/tutorials/includes.html
```

HTML UTILITY PROGRAMS

This section lists several small utility programs that are useful in managing collections of HTML documents. Most of these are small perl programs and should run on any computer that has perl, although they were mostly developed on UNIX workstations.

DTD2HTML

Dtd2html is a perl program that takes an SGML Document Type Definition (DTD) file and generates a collection of HTML documents explaining the structural relationship among the elements defined in the DTD. The resultant documents contain hypertext links among these documents. This is useful if you are interested in learning more about the HTML DTD.

Dtd2html was written by Earl Hood. More information about **dtd2html** is available at:

```
http://www.oac.uci.edu/indiv/ehood/dtd2html.doc.html
```

HTML TABLE CONVERTER

As mentioned in Chapters 1 and 2, HTML2.0 does not contain elements for creating structured tables, although such elements are defined in HTML+. In fact, the specification for the **TABLE** elements is quite stable so that actual implementation of **TABLE** elements should be just around the corner. Indeed, some browsers, such as **viola**, **emacs-w3**, and the just released **Mosaic for X-Windows 2.5**, already support the **TABLE** elements.

The transition from the **TABLE**-less world of today to the **TABLE**-full world of tomorrow has been made easier by a small perl package written by Brooks Cutter. This package accepts documents containing HTML+ **TABLE** elements, and formats the table portions into an appropriate plain text table, surrounding the table with `<PRE>` and `</PRE>` tags. Using this package, you can write your documents using the **TABLE** elements and then quickly convert them into a form acceptable to today's browsers. Then, when browsers mature to the point of handling tables, you will already have tables in the correct format.

The package (a UNIX shell archive) is available at:

```
ftp://sunsite.unc.edu/pub/packages/infosystems/WWW/tools/html+tables.shar
```

HYPERMAIL

Hypermail is a C-language program that takes a file of mail messages in UNIX mailbox format and generates a set of cross-referenced HTML documents. **Hypermail** converts each letter in the mailbox into a separate HTML file, with links to other related articles. It also converts e-mail addresses and hypertext anchors in the original letters into HTML hypertext links. **Hypermail** archives can be incrementally updated, which significantly eases the updating of archives on a periodic basis.

Detailed information about **Hypermail** can be found at:

```
http://www.eit.com/software/hypermail/hypermail.html
```

The program itself is available from:

```
ftp://ftp.eit.com/pub/web.software/hypermail
```

The current C-language version of **Hypermail** was written by Kevin Hughes of Enterprise Integration Technologies Inc.

MHONARC: MAIL TO HTML ARCHIVE

MHonArc is a perl package for converting Internet mail messages, both plain text and MIME-encoded, into HTML documents. This can be extremely useful, for example, if you are archiving electronic mail messages or newsgroup postings and want to make them available on the WWW. The package uses the letter's subject line for the HTML TITLE and as an H1 heading in the HTML version of the letter, and converts relational headers, such as References or In Reply To, into the appropriate hypertext links if possible.

MHonArc can also sort letters according to their topical thread and connect them together with *Next* and *Previous* hypertext links. In addition, **MHonArc** creates an index of the letters or articles, and creates a link from each converted letter to this index.

The home page for **MHonArc** is:

```
http://www.oac.uci.edu/indiv/ehood/mhonarc.doc.html
```

This contains directions for obtaining the most recent version of MHonArc and pointers to extensive documentation. The author is Earl Hood.

TABLE OF CONTENTS GENERATOR

Htmltoc is a perl program that can automatically generate a Table of Contents (TOC) for a single HTML document or for a collection of related documents. **Htmltoc** uses the HTML **H1** through **H6** headings to locate sections within a single document, and uses the order of the heading (**H1**, **H2**, and so on) to determine the hierarchical relationship of the TOC. This, among many other features, can be significantly customized.

When **htmltoc** creates a Table of Contents, it creates hypertext links from the Table of Contents to the documents themselves. It does this by editing the original documents and adding the appropriate hypertext anchors. The original documents are backed up during this process so that you can easily recover the original material.

Additional documentation on **htmltoc**, including directions to the most recent version, can be found at:

```
http://www.oac.uci.edu/indiv/ehood/htmltoc.doc.html
```

Htmltoc was written by Earl Hood.

TREELINK

Treelink, written in tk and tcl, is a package that draws a hypergraph of the hypertext links, starting from a given hypertext document. **Treelink** analyzes the connections and draws a tree-like graph until it reaches a certain predefined depth (number of links). This graph often gives useful insights into the connection and arrangement of links to a particular document.

Additional information about **Treelink**, including instructions as to where to obtain the source code, can be found at:

```
http://aorta.tat.physik.uni-tuebingen.de/~gaier/treelink/
```

Treelink was written by Karsten Gaier.

CGI UTILITY FUNCTIONS

This section lists various CGI gateway programs and utilities. Some, such as the query and input parsing libraries, are of general use. Others, such as the e-mail handlers, may not be generally useful but do serve as valuable models for developing custom applications.

CGI EMAIL HANDLER

This package, due to Thomas Boutell, is a simple CGI program that allows WWW clients to send e-mail messages to a restricted set of allowed recipients. The package has three components. **Email.c** is the ANSI C program that does all the work. You also need to create FORM-based HTML documents for composing the letter: You need to create a separate FORM for each of the allowed recipients. Third, you need to create a server-side database of allowed recipients, used by **email** to verify that the user is sending mail to an authorized person.

This package is quite simple and has been used as a model for several other, more extensive implementations. Additional information about this package, including directions to the various component parts you will need, can be found at:

```
http://siva.cshl.org/email/index.html
```

CGI FEEDBACK FORM

This package of utilities by Arjan de Vet uses a FORM interface to let the client send feedback information to the server administrator. It consists of a

small C-language CGI program that reads the FORM data and converts the FORM information into a mail message to the site administrator.

The package is available at:

```
ftp://ftp.win.tue.nl/pub/infosystems/www/wwwutils.tar.gz
```

DETERMINING CLIENT SOFTWARE

Sometimes, it is nice to know the type of client accessing your server CGI program. Typically, this would be to offer different documents or database interfaces, depending on the capabilities of the client—there is no point in offering a clickable image interface to someone using **lynx**!

This requires a program that can parse the *HTTP_USER_AGENT* environment variable and break it into useful fields, including the client program field. A nice perl program demonstrating this facility is found at:

```
http://www.mps.ohio-state.edu/cgi-bin/clientinfo.pl
```

This includes directions for obtaining the perl source code and additional required perl libraries. This useful script was written by Doug Stevenson.

CONVERT UNIX MAN PAGES TO HTML

There are several CGI packages for searching through UNIX online manuals and converting the manual pages into viewable HTML documents. This section briefly describes three exemplary packages. For those familiar with UNIX man pages, these are useful models of CGI programming techniques.

MAN2HTML

Man2html is a perl program by Earl Hood that can read UNIX man pages, in formatted nroff form only, and produce HTML as output. Although you can do this on a per-file basis, **man2html** is designed to be used in a CGI program: Instructions are included with the distribution on how to do this, along with an example **man.cgi** script. This gives WWW clients access to all your UNIX online documentation. You can also configure **man2html** to include hypertext links to other man pages related to the one you are

reading. In addition, **man2html** supports the `man -k keyword` man page search facility.

Additional information about **man2html** is found at the URL:

```
http://www.oac.uci.edu/indiv/ehood/man2html.doc.html
```

Man2html was written by Earl Hood.

BBC_MAN2HTML

Bbc_man2html.pl is a perl program for distributing UNIX man pages via a World Wide Web server. **Bbc_man2html.pl** uses the `<ISINDEX>` search mechanism to prompt for a particular man page, where you enter the desired man page title in the query box. At present, `man -k keyword` searching is not implemented. **Bbc_man2html.pl** uses the **RosettaMan** program (discussed in the following) to format the man pages into HTML, so you also need this program.

Bbc_man2html.pl is available at:

```
ftp://src.doc.ic.ac.uk/computing/information-systems/www/tools/translators
```

Bbc_man2html.pl was written by Brooks Cutter.

ROSETTAMAN: UNIX MAN PAGE TO HTML

RosettaMan is a filter for UNIX-style manual pages. It takes as input man pages formatted for a variety of UNIX flavors (not just [tn]roff source) and produces as output a variety of file formats. RosettaMan is a tcl/tk script and requires the interpreters that come with the tcl/tk distribution.

The RosettaMan archive can be found at:

```
ftp://ftp.cs.berkeley.edu/ucb/people/phelps/tcltk/rman.tar.Z
```

RosettaMan is copyrighted by T. A. Phelps.

PROCESSING QUERIES AND FORM PACKAGES

One of the most tedious aspects of writing a CGI program is parsing the environment variables, command-line arguments, or standard input data to

the program: These data are encoded using the URL syntax so that a gateway program must first decode the information before it can be used. The following are some example programs and programming libraries that can help with this task.

Uncgi

Uncgi is a useful front-end package that handles this parsing for you. **Uncgi** decodes all the **FORM** fields and places them in environment variables. **Uncgi** then calls your processing program (C, perl, or whatever), passing these environment variables to it. Note that this may be a problem if the variable being passed in an environment variable becomes too large, since some systems place limits on the size of environment variables.

Additional information about **uncgi** is available at:

```
http://www.hyperion.com/~koreth/uncgi.html
```

Uncgi was written by Steven Grimm.

The EIT CGI Library

Enterprise Integration Technologies has made available a library of C functions that are extremely useful in constructing CGI programs. Included are routines for parsing input from FORM or ISINDEX queries, routines for generating MIME response headers, and much more. Details about the different available functions can be found at:

```
http://wsk.eit.com/wsk/dist/doc/libcgi/libcgi.html
```

The source code and executable libraries can be found at:

```
http://wsk.eit.com/wsk/dist/
```

in subdirectories corresponding to your operating system, and below that in the subdirectory ***libcgi***. For example, if you are using the IRIX operating system, the **libcgi** material would be at:

```
http://wsk.eit.com/wsk/dist/irix/libcgi/
```

If you don't find your operating system in this list, just go to one of the other directories and get the source code.

DATABASE CGI GATEWAY PROGRAMS

Needless to say, there is great interest in using the World Wide Web as a front end to sophisticated database packages. This requires the construction of gateway programs to connect the FORM- or ISINDEX-based query input mechanisms of the WWW with the back-end SQL (or other) mechanisms used by databases, such as Sybase, Oracle, or WAIS. The following is a list of current efforts at constructing these interfaces, including instructions on how to obtain more information. All these packages will require customization to your particular database requirements. Nevertheless, they provide a powerful starting point for integrating WWW applications with your database technologies.

WAIS GATEWAYS

WAIS, for Wide Area Information Servers, is an extremely popular Internet network publishing and textual database system. WAIS is designed as a client-server system, with WAIS clients able to interrogate WAIS databases using a well-defined protocol. This protocol is supported by a number of WWW clients so that it is often possible to directly interrogate a WAIS server by constructing a URL (and appropriate query strings) that points to that WAIS server.

However, many clients still do not support the WAIS protocol and, in many cases, this direct method is not ideal for accessing WAIS servers. Often, it is easier to interact with a FORM interface that is designed to construct WAIS queries in a manner more convenient to the user. For these reasons, several gateway programs have been constructed that allow non-WAIS-capable clients to access a WAIS server, and that allow for sophisticated FORM interfaces to WAIS servers. The following is a list of some of the server-side CGI packages designed to accomplish these tasks.

Basic WAIS

The NCSA HTTP server distribution comes with several small CGI programs, including *wais.pl*, a small perl program written by Tony Sanders.

Wais.pl uses the ISINDEX mechanism to obtain a query string from the user and passes this string to a local WAIS database query engine (**waisq**), and returns the results to the user as an HTML document. This is a simple and easy to understand perl script, and has served as the basis for a number of other WAIS gateway programs.

Wais.pl is available at the NCSA gateway program archive:

```
ftp://ftp.ncsa.uiuc.edu/Web/httpd/Unix/ncsa_httpd/cgi/wais.tar.Z
```

Son of WAIS

Son-of-wais.pl is a perl script based on the original **wais.pl** package from NCSA. **Son-of-wais.pl** adds some nice formatting of the returned data and more complete instructions to the user. The perl package is available at:

```
http://dewey.lib.ncsu.edu/staff/morgan/son-of-wais.html
```

(the perl code is part of the HTML document). **Son-of-wais.pl** was written by Eric Lease Morgan of North Carolina State University Libraries. However, Eric does point out that the package **kidowais.pl** (described next) is a significant improvement over **son-of-wais.pl**.

KidoWAIS

Kidowais.pl is another perl script WAIS gateway program based on **wais.pl**, but with significant functionality improvements over both **wais.pl** and **son-of-wais.pl**. **Kidowais.pl** also uses the ISINDEX query interface, but allows for pattern searches (`astro*` matches any word beginning with astro) as well as complex Boolean searches. The program returns search results complete with useful information about the resulting items, such as document type and length. In addition, when the documents are retrieved the search string is highlighted in boldface for easier identification.

Additional information is available at:

```
http://www.cso.uiuc.edu/grady.html
```

which includes pointers to the perl packages you will need (**kidowais.pl** and **print_hit_bold.pl**) and to other documents, with additional informa-

tion on installing and configuring the packages. **Kidowais.pl** was written by by Michael Grady.

SFgate

SFgate is another perl-based WAIS gateway program but is distinctly different from **wais.pl** or its derivatives. **SFgate** does not access a server-side WAIS query engine. Instead, it has WAIS client software built-in so that it can itself query any Internet-accessible WAIS database. In addition, it has a FORM query interface so that the user can select various databases (you can search more than one at the same time) and enter complex query information into text input boxes. You can also create customized FORM interfaces to make it easier for your users to access the database.

SFgate is particularly designed to work with **freeWAIS-sf** servers. **FreeWAIS-sf** is a WAIS variant modified to, among other things, allow for structured fields. Information about **freeWAIS-sf** can be found at:

```
http://ls6-www.informatik.uni-dortmund.de/freeWAIS-sf/README-sf.html
```

Additional information about **SFgate** can be found at:

```
http://ls6-www.informatik.uni-dortmund.de/SFgate/SFgate.html
```

while the perl source can be downloaded from:

```
ftp://ls6-www.informatik.uni-dortmund.de/pub/www/
```

The most recent version as of October 1994 was in the file *SFgate-3.1.tar.gz*, but this will no doubt have changed. **SFGate** was written by Ulrich Pfeifer.

WAISgate

WAIS Inc. is developing a WAIS to Web gateway program called WAISgate. Source code for this gateway is not available, but the gateway is accessible, for free, on the main WAIS server at *www.wais.com*. Additional information about **WAISgate** can be found at:

```
http://server.wais.com/waisgate-announce.html
```

WWWWAIS

WWWWAIS is a small ANSI C program that acts as gateway between **waisq** and **waissearch** (the WAIS programs that search WAIS indexes) and a FORM-capable World Wide Web browser. **WWWWAIS** allows for a customized FORM interface and a database access control mechanism (restricting access to certain Internet domains), and gives users the ability to search multiple databases. As with the preceding packages, **WWWWAIS** returns the search results as a hypertext document.

Additional information about **WWWWAIS** can be found at:

```
http://www.eit.com/software/wwwwais/wwwwais.html
```

The source code and support files are available from:

```
ftp://ftp.eit.com/pub/web.software/wwwwais/
```

Get the README file first to find out what you need. **WWWWAIS** was written by Kevin Hughes of Enterprise Integration Technologies.

GATEWAYS TO STRUCTURED QUERY LANGUAGE DATABASES

There are also several gateway packages for linking the World Wide Web to commercial database packages, such as Oracle or Sybase. The following sections list some of these efforts.

GSQL-Oracle Backend

GSQL-Oracle Backend is a CGI program that can link WWW applications to an Oracle database, using both the ISINDEX and FORM interfaces. **GSQL-Oracle Backend** is written in PRO-C, the C-language development environment for ORACLE, so you need this development option to compile **GSQL-Oracle Backend.** The package comes complete with installation instructions, and should compile easily on most UNIX machines.

Additional information about **GSQL-Oracle Backend** is available at:

```
ftp://ftp.cc.gatech.edu/pub/gvu/www/pitkow/gsql-oracle/oracle-backend.html
```

while an archive of the package is available from:

```
ftp://ftp.cc.gatech.edu/pub/gvu/www/pitkow/gsql-oracle/gsql-oracle.tar
```

GSQL-Oracle Backend was written by James Pitkow of the Graphics, Visualization, and Usability Center at the Georgia Institute of Technology. For further information, see:

```
http://www.cc.gatech.edu/gvu/
```

WEB/GENERA

Web/Genera is a software toolset for the integration of Sybase databases into the World Wide Web. **Web/Genera** can be used to retrofit a Web front end (FORM or ISINDEX) to an existing **Sybase** database, or to create customized interfaces. To use **Web/Genera**, you write a specification of the Sybase database and of the desired appearance of its contents on the Web using a simple, high-level schema notation. Various **Web/Genera** programs process this description file to generate SQL commands and formatting instructions that together extract objects from your database and format them into HTML. You don't have to write a single line of code to do this, and need learn only how to describe the database using the Genera schema notation, which is reputedly easy to use.

Genera supports URLs linked to specific database objects, as well as powerful FORM-based relational query construction. You can also use **Genera** to generate full-text searches of Sybase databases via Web/WAIS and Gopher/WAIS interfaces.

Additional information about **Web/Genera**, including links to the most recent version of the package and substantial documentation, can be found at:

```
http://cgsc.biology.yale.edu/genera.html
```

Web/Genera was developed by Stanley Letovsky, under work supported by National Science Foundation grant BIR-9201652. It is freely available to the public.

WDB—A Web Interface to Sybase

WDB is a CGI package similar to Web/Genera, and is also based on perl and sybperl. Like Web/Genera, WDB allows you to use high-level description files to specify the structure of the database and the format of the responses so that you can construct a generic WWW-Sybase interface without writing a single line of code. Notable is the ability to turn data from the database into hypertext links so that it is possible to access any database element directly via a URL.

Additional information about WDB, including a list of required software and pointers to the source code archive, is found at:

```
http://arch-http.hq.eso.org/bfrasmus/wdb/wdb.html
```

WDB was written by Bo Frese Rasmussen of the European Southern Observatory.

GSQL Gateway

GSQL is a C program that is invoked from the HTTP server via a shell script. GSQL is a simple gateway to Sybase or other SQL databases. It parses an SQL-specification file (called a PROC file) to create an HTML FORM, and uses the user input onto this FORM to call the database backend program to process the SQL query. Search query results are then returned to the client. The PROC file maps components of the SQL string to widgets (fields, buttons, pull-down menus, etc.) for user input or selection. Substantial documentation on GSQL and the construction of PROC files can be found at:

```
http://www.ncsa.uiuc.edu/SDG/People/jason/pub/gsql/starthere.html
```

The source code for GSQL can be found at:

```
http://base.ncsa.uiuc.edu:1234/gsqlsrc/gsql.tar
```

GSQL was written by Jason Ng of NCSA.

HTOracle

HTOracle was one of the first database gateway programs developed for the WWW. HTOracle is a small package of programs written in Pro-C and

C that can take ISINDEX-style queries and pass them to an Oracle database as an SQL SELECT statement, and then return the results to the client as a plain text table. Documentation on the package is somewhat sketchy, but what there is can be found at:

```
http://info.cern.ch/hypertext/WWW/RDBGate/ArthurNotes.html
```

while pointers to the various source-code components are at:

```
http://info.cern.ch/hypertext/WWW/RDBGate/Implementation.html
```

The **HTOracle** code has been used as the basis for a number of other database gateways, including **OraPlex**. **HTOracle** was written by Arthur Secret of CERN.

OraPlex

OraPlex is an adaptation of the **HTOracle** database gateway program. Originally designed for integration with the **Plexus** HTTP server, **OraPlex** is now available as a CGI-compliant package, so it can be used with other CGI-compliant servers. To use **OraPlex**, you need a local Oracle database and **oraPerl**, a perl package written by Kevin Stock that is designed to access Oracle databases.

OraPerl can be found in the directory:

```
ftp://ftp.demon.co.uk/pub/perl/db/oraperl
```

The original Plexus-only **OraPlex** programs are found in the directory:

```
http://moulon.inra.fr/oracle/
```

under the names ***oraplex_pl.html*** and ***oracle_search.html***, while the newer CGI-compliant package is found at:

```
ftp://moulon.inra.fr/pub/www-oracle
```

The C-language version of the database gateway package that uses the standard CGI interface and that can run under any server is discussed at:

```
http://moulon.inra.fr/oracle/www_oracle_eng.html
http://moulon.inra.fr/oracle/www_oraperl_eng.html
```

Documentation in French is available at:

```
http://moulon.inra.fr/oracle/www_oraperl.html
http://moulon.inra.fr/oracle/www_oracle.html
```

OraPlex was written by Guy Decoux of the *Institut National de la Recherche Agronomique.*

MACINTOSH SEARCH TOOLS: TR-WWW

TR-WWW is a Macintosh-based search engine that works only with the **MacHTTP** server. It provides a FORM-based interface to a search engine that dynamically searches (using Boolean searches) server document collections consisting of HTML, plain text, or Microsoft Word files. The documents do not have to be preindexed, as **TR-WWW** can index the documents in real time. **TR-WWW** is thus ideal for searching rapidly evolving collections of documents. Search results can be reviewed (as HTML documents) and sorted by keyword or by WAIS-style, relevance-ranked results.

Additional information about **TR-WWW** is found at:

```
http://informatics.med.monash.edu.au/tr-www.html
```

The latest beta version and manual updates are available by following the links at the top of the document at the preceding URL. The software itself can be found at:

```
http://informatics.med.monash.edu.au/tr-www.sit.hqx
ftp://ftp.cshl.org/transfer/tr-www.sit.hqx
```

CGI ARCHIVE SITES

The following URLs contain useful archives of CGI programs:

```
ftp://ftp.ncsa.uiuc.edu/Web/httpd/Unix/ncsa_httpd/cgi/
ftp://ftp.rz.uni-karlsruhe.de/pub/net/www/tools/cgi-src/
```

DATABASE GATEWAY REFERENCES

Additional information about setting up WAIS gateways can be found at:

http://wintermute.ncsa.uiuc.edu:8080/wais-tutorial/wais.html

http://wintermute.ncsa.uiuc.edu:8080/wais-tutorial/wais-and-http.html

Examples of useful WAIS gateway programs can be found at:

http://www.eit.com/software/wwwwais/wwwwais.html

http://www.cis.ohio-state.edu/hypertext/faq/usenet/wais-faq/getting-started/faq.html

Web Database Development Sites:

http://www.cs.vu.nl/~anne007/waissearch/pointers.html

http://www-rlg.stanford.edu/home/jpl/websearch.html

http://cgsc.biology.yale.edu/dbgw.html

http://oneworld.wa.com/htmldev/devpage/dev-page3.html

http://info.cern.ch/hypertext/WWW/RDBGate/Overview.html

http://www2.ncsu.edu/bae/people/faculty/walker/isindex.html

CHAPTER

6

HTML EDITORS AND DOCUMENT TRANSLATORS

This chapter provides an overview of the tools for creating HTML documents. This collection is constantly growing and evolving so that this list is certainly incomplete and not entirely up to date. For additional information, read the *comp.infosytems.www.users* and *comp.infosystems.www.misc* USENET newsgroups, and also the WWW Frequently Asked Questions (FAQ) list, posted regularly to these newsgroups and also found in hypertext form at:

```
http://sunsite.unc.edu/boutell/faq/www_faq.html
```

The descriptions in this chapter also include pointers to additional documentation on each package, along with directions for obtaining the software. These pointers can also be used to obtain the current status of a particular package.

This chapter is organized in three sections. The first section describes editors specifically designed for writing HTML documents. The second section looks at document translators/converters, which are programs designed to convert from one document format, such as WordPerfect, FrameMaker, or LaTeX, into HTML, and vice versa. The division is somewhat arbitrary, since the combination of a document processing system and a good HTML conversion program is often ideal for preparing HTML documents. Keep this in mind if you are looking for a tool related to a particular document processing system and do not see what you are looking for in the section on editors.

The third section lists HTML validation tools. These are programs that can check an HTML document for correct HTML syntax, and check that hypertext links reference valid URLs. These tools are very useful, as most HTML editors do not check for valid HTML syntax or the validity of the URLs. HTML syntax validation tools use the SGML *Document Type Definition (DTD)* for HTML so that you also need the HTML DTD file. This section includes suggestions of where to find these DTDs, and a description of the validation process.

HTML EDITORS

This section reviews HTML editors. Each editor is described briefly but with sufficient detail that you can see how it works and get an idea of the relative strengths and weaknesses.

At the beginning of each description, you will find the editor name and a list of supported platforms (PC, Macintosh, or UNIX). The subsequent description also notes if the product is commercial software or shareware. The bulk of the description briefly describes the overall layout and operation of the program. Also mentioned are known examples of problems with the current versions of these editors. You should regard this

information as an aid in using these editors and not as a judgment on the packages, since I have not tested these packages with equal detail. No attempt is made to describe the relative ease of use, as, in general, they all seem easy to use and straightforward to learn. Each description includes a list of URLs pointing to additional information about the editor, and to actual locations of the editor if available in this way.

Some of these packages are advertised as What You See Is What You Get (*WYSIWYG*) HTML editors. In the World Wide Web, this is a misleading term, since each Web browser will display the same HTML document in a different way, varying from the graphical rendering of browsers such as **MacWeb** or **Mosaic** to the text-only rendering of **lynx**. This does not mean that WYSIWYG capabilities are bad—they can be useful to give an idea of what the document can look like. However, the actual display on real browsers may be quite different from the editor's rendition. The only effective way to verify the clarity of your document design is to preview it on a few different browsers and adjust the design to ensure that the important information is clearly presented by all of them. This may sound like a lot of work but, after going through this procedure a few times, you will quickly develop a feeling for the types of document designs that are universally successful, regardless of the browser.

SIMPLE TEXT EDITORS (MACINTOSH, PC, UNIX)

Since writing HTML documents requires only the standard, printable ASCII characters, you can prepare and edit HTML documents using a plain text editor. For example, if you are using a PC, you can use the **EDIT** program that comes with DOS, or the **Notepad** editor that comes with Windows (note that you *cannot* use the Windows **Write** editor, as this produces documents in Microsoft's Rich Text Format (RTF), which is not standard ASCII). If you are using a Macintosh, you can use the **TeachText** text editor (limited to files smaller than 20K characters) or freeware editors,

such as **BBEdit-Lite**. On UNIX machines, you can use the ever-present editor **vi** or more sophisticated editors, such as **emacs** or **epoch**.

ALPHA (MACINTOSH)

The **Alpha** text editor provides a comprehensive HTML mode for creating new HTML documents. This is complete with editing menus, automatic character entity generation, and other features. The editor can be obtained from:

```
ftp://cs.rice.edu/public/Alpha
```

BBEDIT HTML EXTENSIONS (MACINTOSH)

There are two **BBEdit** HTML extension macro packages. **BBEdit** is a popular Macintosh text editor that comes in two versions: the freeware program **BBEdit-Lite** and the full-blown commercial package **BBEdit 3.0**. Both of these text editors allow for *extension* packages for customized editing and contain useful search and replace functions and tabbing controls that can be used when editing HTML documents. The **BBEdit** editor is a product of Bare Bones Software, which can be contacted at: *Bare Bones Software*, P.O. Box 108, Bedford, MA 01730-0108 (e-mail: ***bbedit@world.std.com***; phone: (508) 651-3561). The **BBEdit-lite** freeware version is available via anonymous FTP from various sites, including:

```
ftp://ftp.std.com/pub/bbedit/bbedit-lite-232.hqx
```

EXTENSIONS PACKAGE I

This macro package creates extension menus for all standard formatting commands, including FORMs. The current release of the package automatically inserts the **HTML**, **HEAD**, and **BODY** elements and can be configured to automatically insert paragraph tags at each carriage return.

More information is available at:

```
http://www.uji.es/bbedit-html-extensions.html
```

The macros themselves are found at either:

```
ftp://ftp.uji.es/pub/mac/util/bbedit-html-ext.sea.hqx
```

or:

```
ftp://sumex-aim.stanford.edu/info-mac/bbedit-html-ext-b3.hqx
```

This package was developed by Carles Bellver.

EXTENSIONS PACKAGE II

This package lacks the <HTML>, <HEAD>, and <BODY> tags, but does have other nice utilities, such as one for previewing the HTML code—this is not really a WYSIWYG mode but it does indent lists and enumerate list entries where applicable. In general, this package is often easier to use. Its biggest weakness is its lack of a FORM editing menu—all FORM tags must be typed in by hand. Additional information about this package is found at:

```
http://ctipsych.york.ac.uk/WWW/BBEditTools.html
```

while the macros themselves are at:

```
ftp://ctipsych.york.ac.uk/CTI_FTP/pub/BBEdit/BBEdit_HTML_Tools.sea.hqx
```

This second package was written by Lindsay Davies.

You can mix together the macros from these two packages, as the macro names are distinctly different. However, combining all the macros from both packages produces menus so long as to be quite unwieldy. It is best to pick and choose the macros you want and customize the editor to your taste. This is an easy way to blend the useful features of both extension packages.

CU_HTML.DOT (PC—WORD FOR WINDOWS)

CU_HTML.DOT is a Microsoft Word for Windows Version 2.0 and 6.0 document template that allows WYSIWYG editing and creation of HTML documents. The CU_HTML.DOT template adds special HTML styles for text processing along with a macro that takes a Word document, prepared

using these HTML styles, and converts it into the corresponding HTML file. You therefore cannot use this package to edit existing HTML documents, nor can it be easily used to convert existing Word documents into HTML. **CU_HTML.DOT** also allows you to insert inline GIF images where these images appear in the Word document so that the editor is almost WYSIWYG. You can also specify hypertext links by marking text and choosing the target file from a dialog box: Relative locations of files are converted into the correct URL notation. You can also use the normal editing features, such as spelling checking, printing, and cutting and pasting.

The documentation in HTML format is available at:

```
http://www.cuhk.hk/csc/cu_html/cu_html.htm
```

The ZIP file containing the template itself is found at:

```
ftp://ftp.cuhk.hk/pub/www/windows/util/cu_html.zip
```

This ZIP file also contains the hypertext documentation for **CU_HTML.DOT**. **CU_HTML.DOT** was written by Kenneth Wong and Anton Lam of The Chinese University of Hong Kong. The Computer Services Centre of The Chinese University of Hong Kong holds the copyrights of **CU_HTML.DOT** and **CU_HTML.DLL**.

EMACS (UNIX, PC)

The **emacs** editor, a product of the Free Software Foundation, is one of the most popular UNIX text editors. This is because **emacs** is available on almost any UNIX computer as well as on PCs, and because it is an extremely powerful editor that can be customized through its command language *elisp* (a version of lisp) for a variety of tasks. It is the editor of choice for many dedicated UNIX programmers, many of whom have been very active in the early development of the WWW. It is thus not a suprise that there are **emacs** elisp programs available to help with HTML editing.

The first package was written by Marc Andressen, then of NCSA. The name of this lisp program is **html-mode.el** and it is available, for example, from:

```
ftp://gatekeeper.dec.com/.3/net/infosys/Mosaic/elisp/
```

Html-helper-mode is a more recent and somewhat more full-featured **emacs** helper mode. Additional information about **html-helper-mode** is found at:

```
http://www.reed.edu/~nelson/tools/
```

while a complete tar archive of the elisp code and documentation is available at:

```
ftp://ftp.reed.ed/pub/src/html-helper-mode.tar.Z
```

Html-helper-mode was developed by Nelson Minar, based on the **html-mode.el** package.

GT_HTML.DOT (PC—WORD FOR WINDOWS)

This is another package of Microsoft Word for Windows macros developed by the Georgia Tech Research Institute (GTRI). These macros facilitate HTML document authoring by creating a template that provides a pseudo WYSIWYG authoring environment. Additional information is available at:

```
http://www.gatech.edu/word_html/release.htm
```

The authors of GT_HTML.DOT are Jeffrey L. Grover and John H. Davis III. You can contact the authors by electronic mail (*gt_html@gtri.gatch.edu*).

HOTMETAL (PC—WINDOWS, UNIX)

HoTMetaL is a shareware HTML editor from SoftQuad Inc. of Toronto, Canada. SoftQuad is a company specializing in SGML so that HoTMetaL is very careful about legal HTML markup.

HoTMetaL is context-sensitive and strongly guides you in creating new HTML documents—it absolutely refuses to let you put tags in the wrong place. HTML tags can be entered only by pull-down menus: The editor

will not let you type in HTML tags, and automatically converts typed left-angle brackets and ampersands into HTML entities to block you from doing so. The editor is semi-WYSIWYG, in that lists are indented and headings are emboldened and in a larger font. Images are not displayed inline. There is also a Publish command that changes **SRC** and **HREF** attribute values from local paths to full URLs.

The editor is supposed to be useful in editing any HTML document. Unfortunately, the current version interprets the HTML markup very strictly and often refuses to edit even slightly invalid HTML files. This interpretation is built into the *compiled* DTD file ***html.mtl***. SoftQuad is now distributing a second rules file, ***legacy.html***, that is more lenient in this regard and is more useful in editing older files.

HoTMetaL is currently available for PC Windows machines and for Sun SPARC computers running Motif. A Macintosh version is planned. You can obtain the program from:

```
ftp://gatekeeper.dec.com//pub/net/infosys/Mosaic/contrib/SoftQuad/hotmetal
```

A **HoTMetaL Pro** commercially supported version of the editor became available in late fall of 1994. This has many additional features, including spelling checking, thesaurus tools, and utilities for editing older HTML documents and correcting markup. This version also supports the new HTML **TABLE** elements and is able to display images inline with the text.

For more information contact SoftQuad by phone (416-239-4801) or e-mail (***hotmetal@sq.com***).

HTML ASSISTANT (PC—WINDOWS)

HTML Assistant is a Windows text editor with extensions to assist in creating HTML hypertext documents. Tags are inserted by selecting text with the mouse and choosing tags from pull-down menus. A useful utility allows the user to construct HTML hypertext anchors using URLs from **WinMosaic, Cello,** and **HTML Assistant** hotlist and bookmark files. There

is also a user-defined toolbox that allows the user to define special purpose tags.

Additional information about **HTML Assistant** can be found at:

```
ftp://ftp.cs.dal.ca/htmlasst/htmlafaq.txt
```

The program is located at:

```
ftp://ftp.cs.dal.ca/htmlasst/
```

The complete program suite is in the file *HTMLASST.ZIP*. You also need the file **VBRUN300.DLL**, which is found in the ZIP file *VBRUN300.ZIP*.

The author of HTML Assistant is Howard Harawitz.

HTMLED (PC—WINDOWS)

HTMLed is a customizable, text-mode HTML editor. As with other Windows editors, **HTMLed** allows the user to insert tags by marking sections of text and selecting the appropriate tag from pull-down menus. **HTMLed** also allows the user to save and print tag-free versions of the document; preview the document with an external viewer, such as **WinMosaic**; and insert URLs from WinMosaic hotlist files. The user can also create regular and floating toolbars to customize the behavior of the editor.

For more information about **HTMLed**, see:

```
http://info.cern.ch/hypertext/WWW/Tools/HTMLed.html
```

HTMLed is available at either:

```
ftp://pringle.mta.ca/pub/HTMLed/
```

or:

```
ftp://pringle.mta.ca/pub/HTMLed10.zip
```

or:

```
http://pringle.mta.ca/~peterc/
```

in the files *HTMED*.ZIP* (where * refers to the version number). HTMLed was written by Peter Crawshaw.

HTML.EDIT (MACINTOSH)

HTML.edit is a stand-alone HyperCard 2.2 application and does not require HyperCard to run. HTML.edit allows for standard graphical editing features: Items are marked with the mouse and the desired HTML element is selected from a pull-down menu. HTML.edit includes extra features such as multiple document editing and template files for automatic inclusion of headers and footers.

Additional information about HTML.edit can be found at:

```
http://nctn.oact.hq.nasa.gov/tools/HTMLedit/HTMLedit.html
```

while the program itself is available from:

```
ftp://ftp.oact.hq.nasa.gov/tools/HTMLedit/
```

HTML EDITOR (MACINTOSH)

A partially WYSIWYG editor, HTML Editor displays different level headings as different size boldface fonts, and shows italic, boldface, and underlined characters explicitly. Images, however, are not displayed, and lists are not explicitly formatted. There is also no tool for creating HTML FORMs, and some optional arguments are missing for certain elements (for example, there is no ALIGN=BOTTOM argument for the IMG element). However, the editor does allow you to add your own *user elements* to a user menu, so you can add many of these missing components to a customized selection list.

HTML Editor is a shareware program written by Rick Giles of Acadia University. Registered users receive free electronic updates. More information can be found at:

```
http://dragon.acadiau.ca:1667/~giles/HTML_Editor/Documentation.html
```

HTML FOR WORD 2.0 (PC—WORD FOR WINDOWS)

This is another package of Word for Windows templates that facilitates writing HTML documents. The template creates a structured HTML

document environment for Word 2.0. It has a simple user interface and preserves the Word environment as much as possible. It creates document instances that conform to the SGML standard, and is available from:

```
ftp://ftp.cica.indiana.edu/pub/pc/win3/uploads/html.zip
```

This software is an extended prototype of a production version that will be based on Word 6.0. Enough SGML functions have been implemented to ensure the creation of correct HTML instances. Performance of most functions will be greatly enhanced in the production version.

For additional information, contact: Eric van Herwijnen, NICE technologies, Chemin des Hutins, Veraz, 01170 Gex, France (phone (33)-50424940).

HTMLTEXT (UNIX)

Htmltext is a UNIX HTML text editor that supports a WYSIWYG editing environment and includes such useful features as spelling checking and automatic Table of Contents generation. Various menus are available for assigning attributes to different elements, inserting elements, and so on. It is X-Windows-based, but, more importantly, it is based on the graphical **Andrew Toolkit**, an object-oriented model for developing multimedia applications. Thus, to compile and install **Htmltext**, you will also need the **Andrew Toolkit.** Information on the Andrew Toolkit is found in the **Htmltext** documentation at:

```
http://web.cs.city.ac.uk/homes/njw/htmltext/htmltext.html
```

NEXTSTEP HTML-EDITOR (NEXT)

NextStep HTML-Editor is a graphical HTML editor for NeXT workstations. Information about the editor can be found at:

```
http://scholar.lib.vt.edu/jpowell.html
```

while the program itself is available at:

```
ftp://borg.lib.vt.edu/pub/next/HTML-Editor0.5.FAT.app.compressed
```

NextStep HTML-Editor was written by James Powell.

S.H.E. (MACINTOSH)

S.H.E., for Simple HTML Editor, is a text-mode editor written as a HyperCard stack, so you need HyperCard to run this program. **S.H.E.** allows you to create new HTML documents and edit old ones, and it has a palette for marking sections of text as bold, italic, list items, headings, and so on. It explicitly includes the **HTML**, **HEAD**, and **BODY** elements. However, it does not support some of the newer elements, such as **PRE** or **CODE**, and uses some obsolete elements, such as **PLAINTEXT**.

S.H.E. supports the **FORM** elements with a special **FORM** pull-down menu. **S.H.E.** also allows you to cut and paste between the **S.H.E.** window and other windows. **S.H.E.** does not have spelling checking and, because it is a HyperCard stack, it is limited to files smaller than around 30,000 characters (including the HTML markup).

Additional information about S.H.E. is found at:

```
http://dewey.lib.ncsu.edu/staff/morgan/simple.html
```

while the HyperCard stack can be obtained at either of:

```
ftp://ericmorgan.lib.ncsu.edu/Public/simple-http-editor.hqx
ftp://dewey.lib.ncsu.edu/pub/software/mac/simple-http-editor.hqx
```

S.H.E. was written by Eric Lease Morgan.

TKHTML (UNIX)

The **tkHTML** editor is actually a tk/tcl script and requires the interpretive shell **wwwish**. **TkHTML** is a graphical editor: Editing is accomplished by marking text and selecting markup tags from the pull-down menus. The editor is not WYSIWYG but comes with a preview mode that can preview most types of formatting.

Some elements are missing from the menus, such as the `<TITLE>` element and FORM support. The editor also does not include the `<HTML>`,`<HEAD>`, or `<BODY>` elements, so you have to insert these by hand.

Additional information about **tkHTML** is found at:

```
http://alfred1.u.washington.edu:8080/~roland/tkHTML/tkHTML.html
```

The program itself is available from:

```
ftp://ftp.u.washington.edu/public/roland/tkHTML
```

You should download the latest **tkHTML** tar archive and the latest **wwwish** tar archive. TkHTML was written by Liem Bahneman.

TKWWW (UNIX)

TkWWW is actually a UNIX WWW *browser* that allows the user to edit the retrieved document. Like **tkHTML**, **tkWWW** is based on the tk language, and the package also requires the interpretive shell **wwwish**. TkWWW allows you to edit active HTML documents in a manner similar to **tkHTML**, although with less able editing tools. An important difference is that the documents are *live*, so the hypertext links are real hypertext links and not just editable objects. You have to edit carefully, or else you will activate the links. TkWWW is discussed in more detail in Chapter 7.

DOCUMENT TRANSLATORS AND CONVERTERS

Often, you have a number of preexisting files that you would like to convert to HTML format. Alternatively, you may already be using a document preparation program, such as LaTeX, Scribe, or FrameMaker, and would like to be able to continue using these tools and convert the resulting documents into HTML. These are jobs for a document translator or converter: a program that can take your document in its original format and convert it to a close equivalent in HTML.

There are several packages that have been written to accomplish this task, including some powerful commercial products. As mentioned earlier, these conversions may not be ideal, since there is often no direct match between print markup languages and HTML. Consequently, you may have to edit

the resulting HTML documents to add elements that could not be translated. Nevertheless, editing a largely correct document is a much easier task than typing the whole thing in from scratch.

There are also several packages that can convert from HTML to another markup language more appropriate for printing. This is very useful for preparing printed versions of HTML documents for reading away from a computer. Some packages (such as **latex2html** and **WebMaker**) allow you to prepare parent manuscripts containing alternative blocks of markup: one block for printing and another for HTML. This is very useful for preparing documents that read well in either presentation format.

Many of these packages are script programs, often written in perl. Thus, these programs require that you have perl or another appropriate interpreter on your machine.

The programs are listed in the following section in alphabetical order. You are most likely interested in matching a document format or program name to a particular tool. Table 6.1 matches conversion programs to standard word-processor names and document formats.

Table 6.1 Conversion Programs for Different Document Formats

Format	Program
FrameMaker	Cyberleaf, FasTag, frame2html, mif2html, miftran, Webmaker
Interleaf	Cyberleaf, FasTag, TagWrite
LaTeX/tex	hyperlatex, latex2html, tex2rtf
Microsoft Word	FasTag, TagWrite, Cyberleaf
nroff/troff	mm2html, ms2html
Plain text	striphtml, asc2html, charconv
PostScript	ps2html

Continued

Table 6.1 Continued

Format	Program
Rich Text Format (RTF)	HLPDK, rtftohtml, RTFTOHTM, tex2rtf
Scribe	scribe2html
SGML (other than HTML)	TagWrite
Texinfo	texi2html
UNIX Man Pages	RosettaMan
Ventura Publisher (Corel)	TagWrite
WordPerfect	Cyberleaf, FasTag, TagWrite, wptohtml, wp2x

ASC2HTML

Asc2html.pl is a perl script for converting plain ASCII files into HTML. This script adds a `<TITLE>` element containing the name of the file that has been converted, and places the body of the document inside a `<PRE>. . . . </PRE>` element so that it reproduces the formatting of the typed text. However, the script also locates URLs embedded in the text and converts them to hypertext links.

Asc2html can be found at:

```
ftp://src.doc.ic.ac.uk/computing/information-systems/www/tools/translators
```

Asc2html was written by Oscar Nierstrasz of the University of Geneva, Switzerland.

CHARCONV

Charconv is a program filter (written in ANSI-C) that can transform one encoding of an extended character set into another; for example, transforming ISO-Latin I into MS-DOS or Macintosh encoding. This is important for conversion of text containing special characters, such as umlauts, diphthongs, and so on, since the ISO Latin-1 encodings of these characters used by Windows and UNIX machines are different from the encodings used by MS-DOS or the Macintosh operating system.

This program can also handle special character codes found in TeX or HTML documents, and the filtering process removes the HTML or TeX macros. The result is a plain text document.

Charconv is available at:

```
ftp://src.doc.ic.ac.uk/computing/information-systems/www/tools/
```

Charconv was written by Burkhard Kirste.

CYBERLEAF

Cyberleaf is a commercial document-conversion package from Interleaf Inc. planned for release in early 1995 on UNIX and Windows platforms. **Cyberleaf** is a stand-alone product (it does not require Interleaf) for converting documents authored in Interleaf, FrameMaker, Microsoft Word, and WordPerfect into HTML. In the conversion, **Cyberleaf** preserves hypertext links present in the original documents and converts native-format graphics into GIF images. For more information, contact your local Interleaf representative.

FASTAG

FasTag is an add-on package for HTML authoring packages, such as **HoTMetaL Pro**. **FasTag** is a conversion filter that can read documents in various formats (for example, Word, WordPerfect, formatted ASCII) and convert them into SGML encodings, including HTML. You can use **FasTag** to read a large WordPerfect document into **HoTMetaL Pro**, and then use the HoTMetaL editor to further process the HTML code to include those features that are not present in the original format. **FasTag** can handle formatted tables, and converts equations into inlined graphics. Optional add-on packages for other document formats, such as Interleaf and FrameMaker, are also available.

FasTag is a commercial product of Avalanche Development. For more information, contact Avalanche Development (phone: 303-449-5032). If

you are interested in purchasing the package as an add-on to **HoTMetaL Pro**, you can also contact SoftQuad (phone: 416-239-4801; e-mail: *hotmetal@sq.com*.).

FRAME2HTML

Frame2html is a collection of programs to facilitate the conversion of FrameMaker documents and books to HTML, primarily on UNIX platforms. It consists of several programs, some as perl scripts, and some as C-language programs that must be compiled.

The frame2html package is available at:

```
ftp://bang.nta.no/pub/
```

Frame2html also requires several other packages to work properly, including **gs** (**GhostScript**), **yacc**, **flex**, and **pbmplus** (**pnm** and **ppm** filters). Guides to obtaining these additional packages are included with the **frame2html** documentation.

Frame2html was written by Jon Stephenson von Tetzchner of Norwegian Telecom Research.

HLPDK

HLPDK is a Windows software package for developing online hypertext help documentation. **HLPDK** uses a native document format (HDF) that is then converted to the target format—HTML is one target format supported by HLPDK. The user front end is a Windows graphical editor comparable to some of the HTML editors discussed previously.

A shareware version of HLPDK is available at:

```
ftp://garbo.uwasa.fi/pc/programming
```

in the files *hdk115a.zip*, *hdk115b.zip*, and *hdk115l.zip*. There is also a professional supported version. For additional information, contact Ron Loewy (*rloewy@panix.com*).

HYPERLATEX

Hyperlatex is a package that allows you to use a subset of the LaTeX language to prepare documents in HTML and, at the same time, define a document that will be nicely typeset on paper. It consists of a lisp macro for the **emacs** editor and a LaTeX style file for printing the specially marked-up latex documents. **Hyperlatex** is a much simpler package than **latex2html** and lacks some of the **latex2html** features, such as the ability to display equations.

Additional information about **hyperlatex** can be found at:

```
http://www.cs.ruu.nl/people/otfried/html/hyperlatex.html
```

while the source for the program is available at:

```
ftp://ftp.cs.ruu.nl/pub/SGI/IPE/Hyperlatex-1.0.tar.gz
```

Hyperlatex was written by Otfried Schwarzkopf.

LATEX2HTML

Latex2html is a powerful perl package that can translate LaTeX documents into corresponding HTML documents. It has many controls for inlining equations (converting equations to GIF images and including them inline with the text); converting LaTeX cross-references into hypertext links; creating hypertext Tables of Contents and Lists of Figures; and breaking up large LaTeX files into smaller, more manageable HTML documents. It also allows for special LaTeX modes so that text can be marked for display in only the HTML version or only the printed version.

Latex2html requires several additional packages to function properly, as explained in the **latex2html** documentation.

The source archive for **latex2html** can be found at:

```
ftp://ftp.tex.ac.uk/pub/archive/support/latex2html
```

Latex2html was written by Nikos Drakos of the Computer Based Learning Unit, University of Leeds.

MIF2HTML

Mif2html is a commercial product of Quadralay Corporation. **Mif2html** provides users with an interface for converting existing FrameMaker documents to HTML documents suitable for use with the World Wide Web and with GWHIS, Quadralay's online, context-sensitive help system. It allows developers to create documentation within FrameMaker's Maker application, and to convert those files into HTML. This allows the user to maintain a single set of documentation for both printed and hypertext use.

For more information, contact Quadralay Corporation (*support@quadralay.com*).

MIFTRAN

Miftran is a general purpose MIF (FrameMaker's Maker Interchange Format) translation program. **Miftran** was designed primarily to translate to HTML, but is flexible enough that it could be used for other translations. It is a C-language program (along with some UNIX Bourne-shell scripts) with a single configuration file. It converts cross-references into HTML hypertext references, and special characters into the proper character or entity encodings, and can also generate Tables of Contents and Indexes. However, it cannot handle embedded image files or conditional blocks of text.

Miftran can be found at:

```
ftp://ftp.alumni.caltech.edu/pub/mcbeath/web/miftran/
```

Miftran compiles successfully on most UNIX workstations. Unfortunately, there isn't a version for PCs or Macintoshes. The author of **Miftran** is Jim McBeath.

MM2HTML

Mm2html is a perl script that converts nroff documents, written using the "mm" macros, into HTML. Its design is based heavily on the perl scripts

ms2html and **fm2html**. To properly display picture, equation, and table environments, you also need the **psroff** and the **pbmplus** packages.

Mm2html can be found at:

```
ftp://bells.cs.ucl.ac.uk/darpa
```

Mm2html was written by J. Crowcroft.

MS2HTML.PL

Ms2html.pl is a perl script that converts an annotated text file (annotations based on the troff "ms" macros) into the corresponding HTML markup. For **Ms2html.pl** to function properly, each ms macro command must be on its own line.

Ms2html.pl can be found at either of the following:

```
http://cui_www.unige.ch/ftp/PUBLIC/oscar/scripts/
ftp://cui.unige.ch/PUBLIC/oscar/scripts
```

You need the three files ***ms2html.pl***, ***button.pl***, and ***url.pl***.

Ms2html.pl was written by Oscar Nierstrasz.

PS2HTML

Ps2html is a perl package that converts arbitrary PostScript text to HTML. It is written for UNIX platforms, but should be adaptable to other systems. Because PostScript is a nontrivial language, there can be considerable variability in converting PostScript to HTML. This package provides a *types* specification file that allows you to control this conversion and make allowances for the underlying format of your PostScript documents. Additional information about **ps2html** can be found at:

```
http://stasi.bradley.edu/ftp/pub/ps2html/ps2html-v2.html
```

while the package itself can be obtained from:

```
ftp://stasi.bradley.edu/ftp/pub/ps2html/v2/ps2html-v2.tar
```

Ps2html was written by Jerry Whelan.

ROSETTAMAN

RosettaMan is a filter for UNIX-style manual pages. It takes as input manual pages formatted for a variety of UNIX flavors (not just troff or nroff source), and produces as output a variety of file formats. Currently, **RosettaMan** accepts manual pages formatted by the following flavors of UNIX: Hewlett-Packard HP-UX, AT&T System V, SunOS, Sun Solaris, OSF/1, DEC Ultrix, SGI IRIX, and Linux; and produces output for the following formats: printable ASCII only, headers only, TkMan, [tn]roff, Ensemble, (not now but soon) SGML, HTML, LaTeX, and RTF. **RosettaMan** is a tcl/tk script and requires the interpreters that come with the tcl/tk distribution.

The **RosettaMan** archive can be found at:

```
ftp://ftp.cs.berkeley.edu/ucb/people/phelps/tcltk/rman.tar.Z
```

RosettaMan is copyrighted by T. A. Phelps.

RTFTOHTML

Rtftohtml is a program that converts Microsoft Rich Text Format (RTF) documents into their HTML equivalents, and produces HTML containing graphics, tables, and equations along with the regular text. It supports RTF 1.2 and is compatible with Word 6.0 RTF output, as well as RTF output from WordPerfect, FrameMaker, and Interleaf.

Additional information about **rtftohtml** can be found at:

```
ftp://ftp.cray.com/src/WWWstuff/RTF/
```

Executable versions are available for Sun and Macintosh platforms. These are found at:

```
ftp://ftp.cray.com/src/WWWstuff/RTF/latest/binaries/
```

while the source code is found at:

```
ftp://ftp.cray.com/src/WWWstuff/RTF/latest/src/unix.tar
```

Rtftohtml was written by Chris Hector.

RTFTOHTM

RTFTOHTM is a Word 2.0 for Windows extension package that allows conversion of RTF documents to HTML, and also allows you to save edited files in an HTML format. **RTFTOHTM** can be found at:

```
ftp://oak.oakland.edu/SimTel/msdos/windows3/htmtl060.zip
```

(or higher numbers for more recent revisions). **RTFTOHTM** was written by Jorma Hartikka.

SCRIBE2HTML

Scribe is a document markup and production system similar to markup languages such as [tn]roff or LaTeX. The **Scribe2html** package allows you to use Scribe to generate HTML documents from a Scribe manuscript. This requires special markup codes in the Scribe manuscript, so you cannot simply convert a preexisting Scribe document to HTML. However, smaller manuscripts can be converted quite easily.

Scribe2html consists of Scribe macros for introducing special HTML commands, and a small **awk** program that breaks up the large HTML file produced by **Scribe** into several smaller files linked together by hypertext links. The package is free and is available at:

```
ftp://gatekeeper.dec.com/pub/DEC/NSL/www/
```

The package comes with documentation, as a collection of HTML documents.

STRIPHTML

Striphtml is a small perl program that strips HTML markup from a text document to create a plain text version. It also converts the entity references <, >, and & into the correct printable characters.

Striphtml was written by Earl Hood and is available at:

```
http://www.oac.uci.edu/indiv/ehood/perl/striphtml
```

TAGWRITE

TagWrite is a commercial document-conversion tool produced by Zandar Corp. **TagWrite** is a Windows package that can read Microsoft RTF, WordPerfect, ASCII files, and other formats to produce tagged output in various markup formats, including HTML. Untagging is also possible so that you can convert in either direction. **TagWrite** is also able to convert into other markup languages, such as Ventura Publisher, Quark, Interleaf, and troff. The package can also convert from HTML to these other formats. Indeed, Corel has licensed **TagWrite 3.1 for Windows** and includes it as a document-conversion tool with its new release of the Ventura line of document-processing tools. For more information (including pricing), contact Zandar Corp. (phone: 802-365-9393; FAX: 802-365-4974).

TEX2RTF

Tex2rtf is a utility for converting from a simple LaTeX subset to HTML, RTF, and Windows Help RTF formats. The input document is a LaTeX subset combined with some additional hypertext macros. LaTeX equations are not supported, and there is minimal support for tables.

Compiled versions of **tex2rtf** are available for Windows and for Sun SPARC stations, or you can obtain the source code. All these different versions can be found at:

```
ftp://skye.aiai.ed.ac.uk/pub/tex2rtf
```

Tex2rtf was written by Julian Smart.

TEXI2HTML

Texi2html is a perl script that converts texinfo files to HTML. The quality of the HTML documents is close to that of the printed output. The program understands most Texinfo Version 2 commands, and runs without problem on big texinfo files, such as the GNU Emacs manual.

Texi2html can be obtained at:

```
ftp://src.doc.ic.ac.uk/computing/information-systems/www/tools/translators
```

Texi2html was written by Lionel Cons.

WPTOHTML

This is a WordPerfect macro package for converting WordPerfect 5.1 for DOS and WordPerfect 6.0 for DOS files into HTML. The macro packages are available at:

```
ftp://oak.oakland.edu/SimTel/msdos/wordperf/
```

in the files **wpt60d10.zip** (for WordPerfect 6.0) and **wpt50d10.zip** (for WordPerfect 5.1).

WPTOHTML was written by Hunter Monroe.

WP2X

Wp2x is a C-language program for converting WordPerfect 5.1 files into other formats, including HTML. This is a UNIX package but should compile on a PC, provided you have a standard ANSI-C compiler. Additional information can be found at:

```
http://journal.biology.carleton.ca/pub/software/
```

This includes pointers to the up-to-date archive for the package. **Wp2x** was written to convert WordPerfect documents into a form suitable for a local *Conservation Ecology* online journal and, consequently, does some rather specific things to the document.

Wp2x was originally created by Raymond Chen.

WEBMAKER

WebMaker is a package for creating FrameMaker documents and converting them to HTML. **WebMaker** includes special FrameMaker macro definitions to facilitate writing HTML documents, and programs for converting

the resulting MIF files into a hypertext network of HTML files. **WebMaker** is an actively supported product of the Programming Techniques Group at CERN and currently runs only under UNIX.

Additional information about **WebMaker**, including information about how to obtain the package, can be found at:

```
http://www.cern.ch/WebMaker/
```

or by writing (*webmaker@cern.ch*.).

HTML VERIFIERS

There are two simple tools for checking the validity of HTML elements in a document. These both use the HTML Document Type Definition (DTD) file as the definition of the HTML syntax, so you will need this file to use these tools. The HTML DTD can be obtained from:

```
http://www.hal.com/%7Econnolly/html-spec/html.dtd
http://www.hal.com/%7Econnolly/html-spec/html.decl
```

(the second file is the SGML *declaration* for HTML and should be appended to your ***html.dtd*** file). New DTD files will appear with each revision of the language. To test your documents' compatibility with this new standard, you need only download the revised DTD and plug it into your verification program.

SGMLS

Sgmls is an SGML syntax-checking program. This program takes as input an SGML file and checks the document structure against the relevant document type definition (DTD). URLs pointing to the current HTML DTD are listed in the foregoing and in the References section at the end of Chapter 2. As output, the program prints a list of syntax errors and the line number at which the errors occurred.

As an example, consider the file ***test.html*** shown in Figure 6.1, which shows the output of **sgmls** after *testing* this file against the HTML DTD.

```
<HTML>
<HEAD>
<TITLE> <em> Instructional </em>and Research Computing Home Page</TITLE>
</HEAD>
<BODY>

<h1> Instructional and Research Computing </H1>

<hr>

This is the Instructional and Research Computing Group <B>(IRC)</B>
World Wide Web home page. If you get lost try the
<a href="big%20dog.html"> big dog help </a> or
<a href="http://www.university.ca/</a>home.html"> right here </a>

<hr>

<oL>
<LI> consulting services in <A HREF="InsT/intro.html"> instructional
technology and applications</A>
applications</A>.<P>

</ol>
<HR>

</BODY>
</HTML>
```

Figure 6.1 Example HTML document ***test.html***.

The following command on a UNIX computer will validate the file *test.html*:

```
sgmls -s html.dtd test.html
```

On a PC running DOS, the command is:

```
sgmls.exe -s html.dtd test.html
```

The output lists the errors and the line numbers at which they occurred. Figure 6.2 shows the **sgmls** output for the preceding file.

```
sgmls:SGML error at test.html, line 3 at ">":
       EM end-tag ignored: doesn't end any open element (current is TITLE)
sgmls:SGML error at test.html, line 3 at ">":
       Bad end-tag in R/CDATA element; treated as short (no GI) end-tag
sgmls:SGML error at test.html, line 3 at "d":
       HEAD end-tag implied by data; not minimizable
sgmls:SGML error at test.html, line 3 at ">":
       TITLE end-tag ignored: doesn't end any open element (current is HTML)
sgmls:SGML error at test.html, line 4 at ">":
       HEAD end-tag ignored: doesn't end any open element (current is HTML)
sgmls:SGML error at test.html, line 21 at ">":
       A end-tag ignored: doesn't end any open element (current is OL)
```

Figure 6.2 Sgmls error output, after parsing ***test.html*** (shown in Figure 6.1).

The errors at line 3 are due to the illegal character markup inside the TITLE element. The subsequent errors at lines 3 and 4 are a result of this same mistake. The error at line 21 is a very typical error: The file has a duplicate </A> ending tag.

Where do you find **sgmls**? It is a public-domain package, available from many anonymous FTP sites. An example location is:

```
ftp://jclark.com/pub/sgmls/
```

The source code archive is in the files *sgmls*.tar.Z*, while the PC executables are in the files *sgmls*.zip* (where * is a version number). **Sgmls** was written by James Clark (*jjc@jclark.com*).

HTML VALIDATION SITE

If you want to check your files but do not feel comfortable downloading the **sgmls** package, you can use it remotely instead. Dan Connolly has developed a FORM-based HTML validation service accessible from anywhere on the World Wide Web. The URL for this useful resource is:

```
http://www.hal.com/%7Econnolly/html-test/service/validation-form.html
```

The FORM interface contains an HTML **TEXTBOX** into which you *paste* (or type if you do not have cut-and-paste editing) your entire HTML document. When you submit the form, the submitted HTML file is checked by **sgmls** and the results are returned in a subsequent document.

LINK VERIFIERS

Sgmls verifies that your HTML tags are correctly placed, but cannot ensure that the hypertext links go to valid locations. To check hypertext links you need a *link verifier*, which is a program that reads your document, extracts the hypertext links, and tests the validity of the URL.

LINKCHECK

Linkcheck is a perl program written by David Sibley of Pennsylvania State University. It is designed to run on UNIX machines but may run on other platforms, given some effort.

Linkcheck is a program for checking **gopher**, **ftp**, and **http** URLs in a document. It cannot check other URLs, such as **mailto**, **wais**, **telnet**, **rlogin**, or **tn3270**. **Linkcheck** tests **gopher** URLs by fully accessing the indicated URL, which can be slow if the URL links to a large file. It tests **ftp** URLs by listing

directory contents rather than fetching the document, which is a lot nicer. **Http** URLs are checked by using the HTTP HEAD method, which is just as nice. If the HEAD method access fails, **linkcheck** tries the GET method (some servers do not understand the HTTP HEAD method).

Linkcheck does have difficulties with relative URLs. You must specify the URL of the directory *containing* the file to allow checking of relative URLs, so the document must be in an HTTP server-accessible directory. This makes it impossible to check the file prior to placing it on the server. In addition, **linkcheck** cannot validate partial URLs containing relative paths: Any URL starting with the string ../ is labeled as invalid.

Linkcheck is available from:

```
ftp://ftp.math.psu.edu/pub/sibley/
```

You need the files ***linkcheck***, ***about.linkcheck*** (documentation for the package), ***ftpcheck*** (for checking FTP links), and ***mconnect.pl***.

Linkcheck also requires a program called **mconnect**, which allows an interactive connection with a remote mailer program. If your system does not have **mconnect**, you can use the **mconnect.pl** perl program indicated in the preceding list.

If you use **mconnect.pl**, you also need the two additional perl packages **sock.pl** and **telnet.pl**. These are found in the files:

```
ftp://anubis.ac.hmc.edu/pub/perl/scripts/netstuff/sock.pl.gz
ftp://ftp.cis.ufl.edu/pub/perl/scripts/telnet.shar.gz
```

If you use **ftpcheck**, you need the auxiliary **ftplib.pl** library. This can be found at:

```
ftp://ftp.cis.ufl.edu/pub/perl/scripts/ftplib.pl
```

VERIFY_LINKS

Verify_links is a FORM-based CGI program for validating hypertext links in an HTML document. **Verify_links** is limited to verifying **http** URLs and

cannot check FTP, Gopher, telnet, and other links. In this respect, it is not as useful as **linkcheck**. However, it is more reliable than **linkcheck** in validating **http** URLs and particularly local or relative URLs. It also can check POST actions to a limited degree, and will *push the buttons* in a FORM to verify the existence of the attached CGI program. Like **linkcheck**, **verify_links** attempts to use network-friendly requests, such as the HTTP HEAD request, to minimize the load on the network, and falls back to full file retrieval only if this fails.

Additional information about **verify_links** can be found at:

```
http://wsk.eit.com/wsk/dist/doc/admin/webtest/verify_links.html
```

Verify_links is available in executable program. Instructions for obtaining and installing **verify_links** is found at:

```
ftp://eit.com/pub/wsk/doc/README.verify_links
```

Verify_links is a product of Enterprise Integration Technologies Corporation.

CHAPTER

7

WEB BROWSERS AND HELPER APPLICATIONS

This chapter reviews the WWW browsers available for UNIX VAX/VMS and NeXT workstations; PCs running DOS, Windows, or OS/2; Macintoshes; and Amigas. Almost all of these browsers have documentation available on the WWW, and corresponding URLs are given when applicable. Monitoring these URLs and reading the various WWW newsgroups are the best ways of keeping up to date with the changes and improvements to a particular package. These tools are evolving quickly, and it is impossible for any book to be completely current.

The reader will also find some helper applications that can be used to decompress, unzip, or unstuff files; play movies or sound files; or view images. I also suggest sites on the Internet where these programs can be obtained.

ACCESSING SOFTWARE

Most browsers are freely available on the Internet; a number of URLs for the executable versions or source code have been included in this chapter. These are not exclusive locations, and you may find that a program is no longer available at the site indicated here. In this case, use **archie** to locate an alternative site. The **archie** client program is available for almost all computer platforms and is available for downloading from many anonymous FTP sites. A detailed description on using **archie** to search for programs is given in Appendix C, along with instructions for obtaining an **archie** client.

PC PLATFORM BROWSERS

Running any browser requires an Internet connection, either an ethernet (or other) card connecting your computer to a local network or a telephone dial-up SLIP or PPP network link. In both cases, you also need software that allows your computer to *talk* to the connection using TCP/IP, which means you must install TCP/IP networking software. Some free TCP/IP software is available on the Internet, but you must really understand what you are doing to set this up properly. There are also a number of commercial packages that provide TCP/IP functionality at modest cost. Consult your local computer shop if you are uncomfortable with installing these packages.

PCs running DOS or Windows can obtain essentially free TCP/IP software by installing the **trumpet winsock** package. This is found at many FTP sites; for example:

```
ftp://ftp.utas.edu.au:/pc/trumpet/wintrump/
ftp://ftp.csn.net/pub/dos/trumpet/
```

A word of warning: There is also a USENET newsreader program named **trumpet**, so be careful not to get the two mixed up.

If you are running a Windows application, you may also need the Windows *WINSOCK.DLL* program library. This is a package of routines that act as an interface between your Windows application (such as **Mosaic**) and the actual TCP/IP software. This library is free and comes as part of the **trumpet winsock** distribution. Many browser distributions also include this library with the browser.

Finally, your Windows application may be a 32-bit application and require the ***win32s*** package, which allows you to run proper 32-bit applications under Windows. ***Win32s*** is currently a Beta release package and is undergoing constant changes, so you may have to download the most recent version to get your browser to work (this is true for **Mosaic 2.0A7** and later, and perhaps for others). The most recent release, ***win32s1.15***, can be found at the URLs:

```
ftp://ftp.ncsa.uiuc.edu/Mosaic/Windows/win32s.zip
ftp://ftp.microsoft.com/developr/win32dk/sdk-public/
```

If you are running OS/2 Version 2.1, you will need to purchase TCP/IP networking software from IBM. OS/2 Version 3.0, on the other hand, has TCP/IP software built into the operating system.

MSDOS BROWSER: DOSLYNX

DosLynx is a DOS-based version of the full-screen text browser **lynx**. It is a full-screen text-mode browser, so inlined images are not displayed; however, images can be displayed by separate helper graphics programs if you have them on your machine. **Lynx** is a useful browser, and ideal for users who are running systems that are unable to run Windows. **DosLynx** can even run on old 8086-based computers.

Additional information about **DosLynx** can be found at:

```
ftp://ftp2.cc.ukans.edu/pub/WWW/DosLynx/readme.htm
```

The program itself is found at the URL:

```
ftp://ftp2.cc.ukans.edu/pub/WWW/DosLynx/
```

The current package is found in the archive *DLX0_8A.EXE*. You may find a more recent version in this directory, with a later (i.e., larger) version number.

WINDOWS AND OS/2 BROWSERS

Almost any 16-bit Windows application will run under OS/2, including browser software. The situation is more problematic with 32-bit Windows applications, since Microsoft keeps changing its special *win32s* 32-bit library package. IBM can't keep up, so newer 32-bit Windows applications, written for the newer *win32s* libraries, will not work on OS/2. This means that newer versions of **WinMosaic** and possibly the 32-bit version of **Air Mosaic** will not run under OS/2 Version 2.1. This has hopefully changed with the release of Version 3.0. But then, Version 3.0 comes with a WWW browser built in!

AIR MOSAIC

Air Mosaic from Spry is a commercially supported version of **Mosaic** that has all the features of the NCSA program, plus several enhancements. Spry currently produces 16- and 32-bit versions of **Air Mosaic**. The 16-bit version runs easily under Windows and OS/2, while the 32-bit version requires an up-to-date *win32s* library and may have problems running under OS/2.

Air Mosaic understands all URLs, including **wais** and **mailto**. It also has support for proxy servers, and a kiosk mode that hides the navigation and control buttons and menus. There are also well-organized pull-down menus for editing helper application lists, for adding new MIME-types, and for editing hotlists. **Air Mosaic** also allows for secure online financial transactions using the company's *Secure Encryption Transaction* (SET) architecture, but you will, of course, also need appropriate server software.

You can obtain a fully functional demonstration copy of the 16-bit version from:

```
ftp://ftp.spry.com/AirMosaicDemo/
```

Spry can be contacted at 1-800-SPRYNET (1-800-777-9638), valid in the U.S. and Canada only, or via e-mail (*info@spry.com*). You can also find out more information at:

```
http://www.spry.com/
```

CELLO

Cello is an elegant Windows 3.1 graphical interface to the WWW. Cello also runs under the most recent release of OS/2; a patch is needed to the TCP/IP software for earlier OS/2 releases. IBM has a patch for this, so if you find that **Cello** does not work, you can contact IBM Service and ask for the fix (for APAR #PN52335). This should not be a problem with the current TCP/IP package, or with OS/2 Version 3.0.

Cello is easy to use and simple to configure, and can use external viewers to display noninlined images and movies, and to play audio files. It can access HTTP, Gopher, FTP, telnet, CSO/ph/qi, and USENET (NNTP) servers directly, and requires a gateway to access WAIS.

Cello has a flexible bookmark utility, and you can easily reconfigure the interface colors and fonts. **Cello** does not support proxy servers or SOCKS, so it is of limited use inside a firewall-protected local network. It also does not support the HTML **FORM** elements, although an upcoming release will do so.

Additional information about **Cello** can be found at:

```
ftp://fatty.law.cornell.edu/pub/LII/Cello/default.htm
http://www.law.cornell.edu/cello/cellofaq.html
```

You can find the archive for the **Cello** program at the URL:

```
ftp://fatty.law.cornell.edu/pub/LII/Cello/
```

in the file ***cello.zip***. The additional files ***cellofaq.zip*** (or ***cellofaq.tar.gz***) contain useful notes about **Cello**. This directory also contains the helper programs ***lview31.zip*** (for images), ***wham131.zip*** (for audio), and ***gswin.zip*** (for PostScript files).

Cello was written by Thomas R. Bruce.

SPYGLASS

Spyglass produces commercial versions of **Mosaic** but sells its product to Original Equipment Manufacturers (OEMs) only. Spyglass also distributes the commercial licenses to **Mosaic**: NCSA ceded these Licensing Rights to Spyglass in the fall of 1994. Currently the **Mosaic** software is licensed to IBM, FTP Software, Digital Equipment Corp., O'Reilly, and Firefox. Spyglass is located in Illinois, at:

Spyglass, Inc., 1800 Woodfield Drive Savoy, IL 61874 (phone: 217-355-6000).

WINMOSAIC: NCSA MOSAIC FOR WINDOWS

WinMosaic was one of the first Windows World Wide Web browsers and spurred the development of several others. The current Version 2.0A7 does not run under OS/2, as it requires up-to-date ***win32s*** support. **WinMosaic** Version 2.0A2 is a 16-bit application and does run under OS/2.

WinMosaic is a well-designed WWW browser for Windows PCs, with support for **http**, **gopher**, **file**, **news** (USENET), and **telnet** URLs, as well as support for WAIS access via a gateway. The **mailto** URL is not supported. There is also a bookmark/hotlist facility; a *stop* button to stop file downloads; and pull-down menus for configuring the display. Unfortunately, configuring the helper applications requires you to edit, by hand, the *MOSAIC.INI* file. This is complicated but is explained in the documentation that comes with the program. **WinMosaic** also comes with built-in proxy support. There are also several commercially available derivatives of **WinMosaic**, which are described next.

Additional information about **WinMosaic** is found at:

```
http://www.ncsa.uiuc.edu/SDG/Software/WinMosaic/HomePage.html
http://www.ncsa.uiuc.edu/SDG/Software/WinMosaic/Bugs.html
```

Instructions for obtaining **WinMosaic** executables are found at:

```
http://www.ncsa.uiuc.edu/SDG/Software/WinMosaic/General.html
```

Versions are available for Intel processors running Windows and Windows NT, as well as for DEC ALPHA and MIPS machines running Windows NT.

WinWeb

WinWeb is a graphical Windows WWW browser for PCs currently at release Version 1.0Alpha2; this version runs well under OS/2.

The current version (1.0A2) of **WinWeb** has some important limitations. It currently supports only **http**, **file**, **ftp**, and **gopher** URLs—all other URLs are ignored. It does, however, have proxy support for access through a firewall, and includes useful features, such as bookmarks and easy configuration of fonts and background screen colors. It also has a *stop* button for interrupting document retrieval.

WinWeb is under rapid development—Up-to-date information is found at:

```
http://galaxy.einet.net/EINet/Winweb/DeveloperNotes.html
http://galaxy.einet.net/EINet/Winweb/Winweb Home.html
```

while the most recent **WinWeb** program release can be found in:

```
ftp://ftp.einet.net/einet/pc/Winweb/Winweb.zip
```

WinWeb uses the *VBRUN.DLL* library to access helper programs (for playing movies, sounds, and so on), so you will also need the ***vbrun300.zip*** package. This is found in the same directory as ***winweb.zip***.

WinWeb is being developed at the Microelectronics and Computer Technology Corporation (MCC) by the Enterprise Integration Network (EINet) Group. EINet is a trademark of the Microelectronics and Computer Technology Corporation (MCC).

HELPER APPLICATIONS

The following is a list of some typical shareware or freeware packages for viewing alternative image formats, playing audio and movie files, decompressing downloaded files, and so on, on PCs. There are many other packages that can do these things, so you should not feel restricted to this

collection. The Cello program archive site (noted earlier) has several of these helper applications. Here is a list of the most common helper applications and the names to look for when doing **archie** searches.

Audio Files (*.au*, *wav*, *.aiff*, *.iff*): **archie** search string: `wham`

This can be found at:

`ftp://ftp.ncsa.uiuc.edu/Web/Mosaic/Windows/viewers/`

The most recent version number is 1.3, found in the file. **Wham** works with most standard audio cards, including Soundblaster.

Gnu Zip File (*.gz*): **archie** search string: `gzip`

Note that **gzip** is also common on UNIX computers, so make sure you are accessing a DOS/Windows version. You can find many different PC-archiving and -unarchiving programs, including **gzip**, at:

`ftp://ftp.std.com/src/pc/archivers/`

Image Files (*.gif*, *.jpeg*, *.jpg*, *.tif*, *.tiff*): **archie** search string: `lview`

Lview can display JPEG, JFIF, GIF, Truevision Targa, Windows, and OS/2 BMP image files, among others. A registered version of the shareware package *WECJLIB.DLL* from Express Compression Laboratories is also recommended for fast JPEG image file decompression. These can both be found at:

`ftp://akiu.gw.tohoku.ac.jp/pub/network/www/Web/Windows/viewers/`

Lview will run without *WECJLIB.DLL*; it will just be slower.

MPEG Video (*.mpeg*, *.mpg*): **archie** search string: `mpegw`

This is a shareware **mpeg** movie player that can be found at:

`ftp://alfred.ccs.carleton.ca/pub/civeng/viewers/`

Look for files *mpegw32e.zip* and *mpegw32e.txt* (installation instructions).

PostScript Files (*.ps*, *.eps*): archie search strings: `gsview` and `gs261exe` (you need both)

The Ghostscript/Ghostview PostScript previewer can be found at:

`ftp://ftp.ncsa.uiuc.edu/Web/Mosaic/Windows/viewers/`

QuickTime Video (*.mov*) : archie search string: `qtw`

Qtw stands for QuickTime for Windows. This can be found at:

`ftp://winftp.cica.indiana.edu/pub/pc/win3/desktop/qtw111.zip`

MACINTOSH PLATFORM BROWSERS

Similar to the PCs, you must have appropriate SLIP, PPP, and TCP/IP software on your Macintosh to run these browsers. All Macintosh browsers require at least System 7, and you will also need MacTCP 2.0.2 (and preferably 2.0.4) to provide the TCP/IP software. There is also a MacPPP package for PPP support. Contact your local Macintosh software outlet to obtain these packages.

MacMosaic: Mosaic for Macintosh

MacMosaic is a Macintosh version of the Mosaic program developed by the NCSA development team. The last stable and fully supported release is Version 1.0.3. This is a good browser, but cannot interpret the **FORM** elements, which severely limits its usefulness. It also cannot communicate with proxy servers, so it is of limited use inside a firewall. The latest release is **MacMosaic** 2.0A8; this version supports the FORM elements as well as proxy servers and SOCKS firewall access. Therefore, if you need these facilities, you will need this (or a later) version. **MacMosaic** 2.0A8 tends to be slow and takes up a lot of memory (3.0 MB minimum). **MacWeb**, another Macintosh browser, is generally smaller and faster.

Both versions of **MacMosaic** support all URLs, except for **mailto**, and support WAIS access only through a WAIS gateway. **Telnet**, **rlogin**, and **tn3270**

URLs are enabled using telnet/rlogin tools already present on your Macintosh, so you also need a telnet/rlogin package. **MacMosaic** has an extensive hotlist interface and built-in lists of useful Web starting points, including pointers to **MacMosaic** information pages. **MacMosaic** also has a *stop* button to interrupt file downloads. Menus are available for configuring helper applications.

NCSA documentation on **MacMosaic** is found at:

```
http://www.ncsa.uiuc.edu/SDG/Software/MacMosaic/MacMosaicHome.html
```

and the online **MacMosaic** users guide is at:

```
http://www.ncsa.uiuc.edu/SDG/Software/MacMosaic/Docs/MacMosa.0.html
```

The Mosaic executables (in binhexed files) are found at:

```
ftp://ftp.ncsa.uiuc.edu/Mosaic/Mac/
```

Several versions are available; the version number is part of the filename. The current 2.0A8 versions are found in the files *NCSAMosaic A8.68k.hqx* (for Macintoshes with 680x0 processors) and *NCSAMosaic A8.PPC.hqx* (for PowerPC machines). Another site containing **MacMosaic** executables is:

```
ftp://sunsite.unc.edu/pub/packages/infosystems/WWW/clients/MacMosaic/
```

MACWEB

MacWeb is a fine Web browser with features both similar and complementary to **MacMosaic**. It supports most URLs, but does not support **rlogin**, **mailto**, or **wais**. Telnet and tn3270 connections are provided through external programs. Helper application configuration is easily done using editable pull-down menus. There is a very convenient hotlist editor for storing and sorting hotlists, as well as a history mechanism for switching randomly between previously visited pages. **MacWeb** supports the CERN Proxy protocol, but you cannot configure this from the pull-down menus: You must use the Macintosh **ResEdit** resource editor to edit the proxy information.

A nice feature is the ability to access hypertext link information: If you place the mouse over a hypertext anchor and hold down the mouse button, you are presented with a menu of choices for what to do with the linked item (get information about the link, retrieve and save to disk, etc.).

Unfortunately, **MacWeb** does not have a *stop* button—once a transfer is started there is no way of interrupting it. This is an unfortunate weakness in an otherwise fine program.

Additional information about **MacWeb** is found at the URLs:

```
http://galaxy.einet.net/EINet/MacWeb/MacWebHome.html
http://galaxy.einet.net/EINet/MacWeb/DeveloperNotes.html
```

while the latest version of the program executable is found at:

```
ftp://ftp.einet.net/einet/mac/macweb/macweb.latest.sea.hqx
```

The EINet MacWeb software was developed at the Microelectronics and Computer Technology Corporation (MCC) by the Enterprise Integration Network (EINet) Group. EINet is a trademark of the Microelectronics and Computer Technology Corporation (MCC). For additional information, you can write to EINet (***macweb@einet.net***).

MACINTOSH HELPER APPLICATIONS

Helper applications display image formats that cannot be displayed inline; play movies or sound files; unpack binhexed files; and so on. Here is a list of the most common helper applications and the names to look for when doing **archie** searches. Several of these applications can be found at:

```
ftp://boombox.micro.umn.edu/pub/gopher/Macintosh-TurboGopher/helper-applications/
ftp://ftp.ncsa.uiuc.edu/Mosaic/Mac/Helpers/
```

Basic Audio (*.au*, *.aiff*, *.aifc*): **archie** search string: `soundmachine`

This is found at the preceding URLs, in the file ***SoundMachine.sit.hqx***.

Expander Enhancer (*.sit*, *.zip* ...): **archie** search string: `enhanc`

This add-on package enables **StuffIt Expander** to expand files compressed with virtually all compression formats used on Macintosh, UNIX, or PC

systems, including ZIP (*.zip*), ARC (*.arc*), AppleLink (*.pkg*), gzip (*.gz*), UNIX compress (*.Z*), and UUencode (*.uu*), not to forget StuffIt archive (*.sit*, *.sea*).

Image Files (*.gif, .jpeg, .jpg, ...*): archie search string: `jpegview`

This is found at the preceding URLs, in the file *JPEGView-3.1.sea.hqx* (or a newer version).

MPEG Video (*.mpeg*, *.mpg*): archie search string: `sparkle`

This is found at the preceding URL, in the file *Sparkle16.sit.hqx*.

PostScript Files (*.ps*, *.eps*): archie search string: `mac-ghost`

This can be found, for example, at:

```
ftp://sumex-aim.stanford.edu/infor-mac/grf/util/
```

QuickTime Video (*.mov*) : archie search string: `simpleplayer`, or `simple_player`

This is found at the preceding URL, in the file *Simple_Player.hqx* (you also need *QuickTime1.5.hqx* if you don't already have it).

StuffIt Archives (*.sit.hqx*, *.cpt.hqx*): archie search string: `sit-expand`

There are many packages that can expand StuffIt archives: **StuffIt Expander** is one popular version.

Zip Files (*.zip*): archie search string: `mac-unzip`

This can be found, for example, at:

```
ftp://sumex-aim.stanford.edu/infor-mac/cmp/
```

Many of these packages may be available as shareware at your local Macintosh software store.

UNIX AND VAX/VMS PLATFORM BROWSERS

UNIX and VAX/VMS computers come with TCP/IP software as part of the operating system, which is a nice improvement over the PC and Macintosh platforms. Unfortunately, you will often have to compile the browsers for your platforms, as executable versions are often not available. This is usually not difficult, and most programs include compilation and installation instructions. You will need to be a system administrator to complete the installation.

BATCH MODE BROWSER (UNIX)

Url_get is a perl program that operates like a batch-mode browser. It allows you to retrieve a document via any of the WWW protocols simply by specifying the document URL. This is useful for downloading WWW documents, or for use in automated processing, such as UNIX cron jobs.

Additional information can be found at:

```
http://wwwhost.cc.utexas.edu/test/zippy/url_get.html
```

A tar archive of the package is found at:

```
http://wwwhost.cc.utexas.edu/test/zippy/perl/url_get/url_get.tar.Z
```

Url_get was written by Jack Lund.

CHIMERA (UNIX)

Chimera is a UNIX X-Windows-based WWW browser. Most UNIX browsers user the Motif X-Windows widget set, which is not available on some systems. **Chimera** uses the public-domain Xaw Athena widget set, so it can be compiled almost anywhere. You will need, at a minimum, a C compiler and the standard X11 libraries. This is not a task for the timid, as **chimera** does not always compile without prodding. Still, an experienced systems manager should be able to get the package running in a couple of hours.

Chimera has a look and feel similar to **Mosaic**, but with a different arrangement of buttons and menus. **Chimera** is fast and, on slow workstations, can be significantly faster than **Mosaic for X-Windows**. This may be important if you are using an older, underpowered workstation. **Chimera** supports most URLs, but not **wais**, **mailto**, **news**, or **rlogin**. **Chimera** requires the **pbmplus** library to display inlined images. If you don't have **pbmplus** on your system, images are replaced by a default *image not available* bitmap.

The distributed **chimera** package does not support proxy servers or SOCKS. You can patch **chimera** to use SOCKS firewall access by following the instructions in the **chimera** distribution documentation.

The **chimera** home page is:

```
http://www.unlv.edu/chimera/
```

This gives a brief description of the program and some pointers to anonymous FTP sites housing the **chimera** source code. The original home of the **chimera** source code is:

```
ftp://ftp.cs.unlv.edu/pub/chimera/
```

EMACS-W3 MODE (UNIX, VMS, OTHERS)

Emacs-w3 is a WWW browser mode for the **emacs** text editor. If you are using the **emacs** variants **lemacs** or **epoch**, **emacs-w3** will display HTML documents using multiple fonts for the headings, titles, and so on, and can also display inlined graphics. **Emacs-w3** mode supports all URL forms and implements many of the HTML+ features. If you are familiar with **emacs**, you should give this package a try.

The **emacs-w3** package is available at the URL:

```
ftp://moose.cs.indiana.edu/pub/elisp/w3/
```

You will need the three files: *w3.tar.gz*, *extras.tar.gz*, and *icons.tar.gz*. You should also read the *README* file for the latest details and installation instructions.

Emacs-w3 mode was written by William Perry, with help from a large and supportive cast.

LINEMODE BROWSER (UNIX, VMS)

This program, developed at CERN, was one of the first WWW browsers. The **LineMode browser** allows anyone with a dumb terminal to view WWW documents, but is not as flexible as such programs as **lynx**. It can also be run as a background process to download documents.

Up-to-date information is found at:

```
http://info.cern.ch/hypertext/WWW/LineMode/Defaults/CommandLine.html
```

Executables (for various platforms) are found at:

```
ftp://info.cern.ch/pub/www/bin
```

in the subdirectory appropriate to your computer type, and in files with names *www*.tar.Z* (where * is a version number). The source code is located at:

```
ftp://info.cern.ch/pub/www/src/WWWLineMode.tar.Z
```

LYNX (UNIX, VMS)

Lynx is a full-screen hypertext browser that can run on any dumb terminal. It uses the arrow and tab keys and single letter keycodes to navigate around the document and for executing program commands. It is the best character-based browser and should definitely be part of your browser collection. Recent versions support proxy access, which means that if you are inside a secure local network, you can use **lynx** to access external documents via a *proxy* hypertext server that you must install on your own network (the CERN HTTPD server can do this). When you request a URL, your client passes the URL on to the proxy server inside your network, which makes the request of the machine outside your network on your behalf, and then passes you the returned result. Proxy service is discussed in more detail in Chapter 3.

Lynx was written by Lou Montulli (***montulli@ukanaix.cc.ukans.edu***) of the University of Kansas. Sources and precompiled binaries for several machines are available at:

```
ftp://ftp2.cc.ukans.edu/pub/lynx/
```

Lynx online documentation is available from:

```
http://www.cc.ukans.edu/lynx_help/Lynx_users_guide.html
```

This set of documents comes with the **lynx** distribution.

MIDASWWW (UNIX, VMS)

The MidasWWW browser is based on the Midas hypertext library. The browser has undergone substantial revisions in its short life and is currently at Version 2.1. It still has a few bugs, but appears generally stable.

MidasWWW is an elegant graphical browser supporting inlined images and clear marking of hypertext links. In addition, visited links are marked differently from unvisited ones to help you navigate an unfamiliar collection of documents. This list is lost when you quit the program. MidasWWW supports all the standard HTML markup, including the FORM elements. MidasWWW supports all URLs except **tn3270**, **login**, and **mailto**. MidasWWW comes with a built-in default configuration that can be changed by creating an ***app-default*** file. MidasWWW does not appear to support proxy server or the SOCKS package, and is, consequently, of limited use inside a firewall.

MidasWWW has extensive online help, but this is downloaded from the home site in Stanford, California, which can be slow. There is a Cancel Download pop-up menu that appears when downloading a new document. However, on my system, this menu does not always appear, which is annoying, given there is no other way of canceling a file download. On one workstation (Sgi Indigo2), the program occasionally locks up the X terminal when downloading files. The X terminal or workstation eventually recovers, but it is not a pleasant experience.

Additional information about MidasWWW can be found at:

```
http://www-midas.slac.stanford.edu:80/midasv21/overview.html
http://www-midas.slac.stanford.edu:80/midasv21/about.html
```

Executables and the source code are found at:

```
ftp://freehep.scri.fsu.edu/freehep/networking_email_news/midaswww/
```

Binaries are recommended, as compiling this program is nontrivial. Precompiled programs for DEC Alpha, AIX, HPUX, SGI, and Sun SPARC systems are available in the *binaries/* subdirectory. There is also a Vax VMS binary, found in *binaries/vms*.

MidasWWW was created by Tony Johnson and Chung Huynh of the Stanford Linear Accelerator Laboratory.

MOSAIC FOR X-WINDOWS (UNIX)

Mosaic for X-Windows is a full-featured WWW browser distributed by NCSA. This browser supports all URLs except **mailto**, and has many nice features, including hotlists, document source viewing and printing, FORMs support, hypertext help, and document annotation. Hypertext links are also specially marked when accessed so that you can keep track of previously explored links. There is also a tool for making local HTML annotations to documents and a text search tool for searching for words in the displayed document.

Mosaic for X-Windows does not have proxy server or SOCKS support. If you can obtain the Mosaic source code and the SOCKS library, you can recompile **Mosaic** to support SOCKS firewall access, but this is not supported by NCSA. For more information on this option, contact Ying-Da Lee (*ylee@syl.dl.nec.com*). In some cases, the precompiled binaries distributed by NCSA do not have native WAIS support. If you obtain the **Mosaic** source code, you can recompile it to include WAIS support, provided you also obtain the **freeWAIS** software.

Precompiled binaries of **Mosaic for X-Windows** Version 2.4 (the current supported version) are available at many sites, including:

```
ftp://ftp.ncsa.uiuc.edu/Mosaic/Unix/binaries/2.4/
```

In mid-October 1994, NCSA announced the release of NCSA **Mosaic 2.5b2 Beta**. This is an enhanced program that includes support for **TABLE**,

SUP, and SUB HTML elements; nested hotlists; and a kiosk mode. This new release still has some bugs and is available as an executable only at:

```
ftp://ftp.ncsa.uiuc.edu in/Mosaic/Unix/binaries/2.5/
```

MOSAIC (TUEV) 2.4.2

The Computing Center (ZDV) of the University of Tuebingen in Germany has developed an extensively modified version of **Mosaic**, known as **Mosaic (TueV) 2.4.2**. This is a multilingual update to NCSA **Mosaic 2.4 for X-Windows**; it supports a multilingual user interface and **mailto** URLs, and incorporates many other patches and improvements. The source code is available at either:

```
ftp://ftp.uni-tuebingen.de/pub/WWW/Mosaic-TueV/source
http://ftp.uni-tuebingen.de/pub/WWW/Mosaic-TueV/source
```

while precompiled binaries are found at the following URLs:

```
ftp://ftp.uni-tuebingen.de/pub/WWW/Mosaic-TueV/binaries
http://ftp.uni-tuebingen.de/pub/WWW/Mosaic-TueV/binaries
```

QUADRALAY GWHIS BROWSER (UNIX)

Quadralay markets a *Global Wide Help & Information System (GWHIS)* that allows application developers to add online documentation to applications via a WWW hypertext system. The browser portion of this package is derived from NCSA **Mosaic**. The **GWHIS** package combines this browser with hypertext servers, database search engines, a **GWHIS** API, and auxiliary tools for converting existing documents into HTML format.

GWHIS is currently available for several different UNIX platforms, but not for Macintoshes or PCs (Macintosh and PC platform support is planned). Additional information can be found at:

```
http://www.quadralay.com/www/ProductInfo/gwhis/pr05.html
```

RASHTY VMS CLIENT (VMS)

This text-based browser is based on the VMS SMG screen management routines, and has an interface similar to a Gopher client.

The **Rashty VMS client** was written by Dudu Rashty and is available at:

```
ftp://vms.huji.ac.il/www.dir/vms_client.dir/
```

TKWWW (UNIX)

Tkwww is a WWW browser based on the tk and tcl toolkits. **Tkwww** compiles easily on many UNIX systems, but you must first install the tk and tcl libraries.

Tkwww's most interesting use is as an HTML editor. **Tkwww** contains many editing commands for marking emphasized text, creating lists, and so on, which allow you to edit and dynamically create HTML documents. You can even create hypertext links and immediately test that they work.

Tkwww is somewhat weak as a browser. It can display inline images (only in a separate window); does not support **mailto, rlogin,** or **wais** URLs; and does not understand the **FORM** elements. There also is no *stop* button to interrupt transfers that have gone bad, or that are simply taking too much time. **Tkwww** does have the native ability to display almost any image format, and has built-in hooks for playing movies and sounds.

There is a useful if somewhat out-of-date collection of **tkwww** documentation at:

```
http://tk-www.mit.edu:8001/tk-www/help/
```

The current version of **tkwww**, 0.12 Beta, can be found at:

```
ftp://nntp.cs.ubc.ca/mirror1/x-contrib/misdirected
```

Tkwww was written by Joseph Wang (*joe@athena.mit.edu*).

VIOLAWWW (UNIX)

ViolaWWW is an X-Windows-based browser based on the Viola *interactive media* scripting language/toolkit. Once again, this is a package you will likely have to compile yourself; there is an executable available only for Sun SPARC platforms. Compilation is a difficult task, as **ViolaWWW** is a complex package. This should not be attempted by the timid!

Viola can be compiled with or without Motif—Motif compilation is recommended. **Viola** supports most standard URLs except **wais** and **mailto**.

Viola has several nice features—notably, its high level of HTML+ support: **Viola** supports **TABLES**, paragraph formatting control, and **FORMs**, and also experimental equation formatting elements.

Additional and up-to-date information on **Viola** is found at:

```
http://xcf.berkeley.edu/ht/projects/Viola/docs/Viola/about.html
http://xcf.berkeley.edu/ht/projects/Viola/README
```

Source code for **Viola** is available at:

```
ftp://ora.com/pub/www/Viola
```

while binaries for Sun SPARC systems are found at:

```
ftp://ora.com/pub/www/Viola/old/
```

Viola was developed by Pei Y. Wei.

UNIX HELPER PACKAGES

Most of the following helper packages will have to be compiled and installed, as was the case for the browsers. In some cases, you will find executable program files, which is a bonus.

If you are using a UNIX system, you are encouraged to obtain the help of a systems manager when installing these tools.

Audio Files (*.au, .aiff,* etc.): **Showaudio** (found inside the **metamail** package—see the following)

Metamail contains the script **showaudio**, which can be used on several UNIX computers to play audio information.

Control Panel (for Movie Players): **Archie** search string: `xplaygizmo`

The `xplaygizmo` is found, for example, at:

```
ftp://ftp.ncsa.uiuc.edu/Mosaic/Unix/viewers/xplaygizmo
```

General Video (*.mpeg*, *.mov*): Archie search string: `xanim`

The **xanim** program is found, for example, at:

`ftp://crl.dec.com/pub/X11/contrib/applications`

Image Files (*.gif*, *.jpeg*, *.tiff*, etc.): Archie search string: `xv-`

The **xv** program is found, for example, at:

`ftp://ftp.cis.upenn.edu/pub/xv`

Metamail (MIME Mail Viewer): Archie search string: `mm`

Metamail, a multimedia mail package, contains the **showaudio** audio player script. **Metamail** is found, for example, at:

`ftp://thumper.bellcore.com/pub/nsb`

MPEG Video (*.mpeg*, *.mpg*): Archie search string: `mpeg_play`

The **mpeg_play** program is found, for example, at:

`ftp://tr-ftp.cs.berkeley.edu/pub/multimedia/mpeg`

PostScript Files (*.ps*, *.eps*): Archie search string: `ghost`

You need the three components *ghostview*, *ghostscript*, and the ghostscript *fonts*. These can be found at:

`ftp://ftp.cs.wisc.edu/pub/ghost/gnu/`

in the files ***ghostscript-2.6.1.tar.gz***, ***ghostscript-fonts-2.6.1.tar.gz***, and ***ghostview-1.5.tar.gz***.

TeX .dvi Files (Display *.dvi* Files): Archie search string: `xdvi`

The **xdvi** *.dvi* file previewer is found, for example, at:

`ftp://export.lcs.mit.edu/contrib/applications`

NEXT PLATFORM BROWSERS

The NeXT platform comes with TCP/IP software as part of the standard operating system. Browsers also come as ready-to-install packages so that compilation is not necessary.

THE CERN NEXT BROWSER

CERN has developed a NeXT browser that is also a WYSIWYG HTML editor. The current Version 2.02 is relatively stable, although not bug-free. Additional information about CERN can be found at:

```
http://info.cern.ch/hypertext/WWW/NextStep/Status.html
http://info.cern.ch/hypertext/WWW/NextStep/Menus.html
```

The executable file is found at:

```
ftp://info.cern.ch/pub/www/bin/next-fat/
```

in the archive file ***WWW.app_n.nn.tar***, where ***n.nn*** is the version number.

The program has a rich collection of commands for editing HTML documents. You can, for example, create links within the document and then immediately test that they work. This browser supports most URL protocols and can support proxy servers. It does not support the FORM elements.

OMNIWEB

Omniweb gets good reviews from NeXT users, who describe it as "pretty cool." It can handle most URLs, including **mailto**. However, it does not support the **FORM** elements and does not have proxy support.

By default, **Omniweb** can display only *.tiff* or *.eps* image formats inline (these are the native image formats for the NeXT windowing system). For the standard WWW formats, you need a helper application. Omni provides an image filter that allows **Omniweb** to view inline GIF and other image formats. This **OmniImageFilter** package is found at:

```
http://www.omnigroup.com/Software/OmniImageFilter/OmniImageFilter.pkg.tar
```

You need this to view most HTML documents, since the default Web image formats are GIF or X-Bitmap.

Additional information about **Omniweb** can be found at:

```
http://www.omnigroup.com/Software/Omniweb/
```

The executable program is available at:

```
http://www.omnigroup.com/Software/Omniweb/Omniweb.app.tar.gz
```

AMIGA BROWSER: AMOSAIC

AMosaic is an Amiga browser based on the NCSA Mosaic package. The browser requires Amiga OS 2.0 or greater, and preferably 3.0. You also need TCP/IP software.

The home page for additional information about AMosaic is:

```
http://insti.physics.sunysb.edu/AMosaic/home.html
```

the installation page, at:

```
http://insti.physics.sunysb.edu/AMosaic/Installation.html
```

which also explains how to obtain Amiga TCP/IP software.

COMING ATTRACTIONS

Various companies are preparing or have just announced new WWW browser products. This is a brief list of these companies and their plans.

IBM OS/2 BROWSER: WEB EXPLORER

IBM intends to bundle a browser with the release of OS/2 Version 3.0, sometime in early 1995. This browser will be based on the Mosaic program. Early reviews of the beta release compare this product favorably with Netscape (see following).

MICROMIND SLIPKNOT

This is a Windows 3.1 WWW browser that can function via a telephone dial-up connection to a UNIX account, without requiring SLIP or PPP access. It provides background document loading (over the phone connection), multiple document windows, and other features. It is currently available as a shareware package.

NETSCAPE COMMUNICATIONS CORP.

Netscape Communications Corp. was founded in part by Marc Andressen, the primary developer of the original NCSA Mosaic. Netscape

Communications has designed an all-new WWW browser, **Netscape**, that has significant enhancements over the original Mosaic program. The beta version of **Netscape** was released in mid-October 1994, to much acclaim and controversy. Acclaim over its elegant design and new features, such as the ability to view documents while they are loading, its elegant USENET news interface, and a common look and feel across PC Macintosh and UNIX platforms. Controversy over some new and **Netscape**-specific HTML extensions that include flowing text around images, changing font sizes inside regular text, and the centering of text. The controversy comes about because several of the HTML extensions are not compatible with the SGML model for HTML. Whatever comes of the controversy, it is clear that Netscape Communications has jumped to the front with some exciting new ideas, and will certainly spur the development of WWW browsers and the HTML language.

You can obtain executable **Netscape** binaries for PC, Macintosh, and various UNIX platforms from:

```
ftp://ftp.mcom.com/pub/netscape/
```

or from other sites listed at:

```
http://Mosaic.mcom.com/info/how-to-get-it.html
```

General information about **Netscape** can be found at:

```
http://home.mcom.com/welcome.html
```

You can also send inquiries by e-mail (***info@mcom.com***) or by old-fashioned mail to:

Mosaic Communications Corporation, 650 Castro Street, Suite 500, Mountain View, CA 94041 (phone: 415 254-1900).

CHAPTER

8

HTTP SERVERS AND SERVER UTILITIES

So, you are thinking of setting up your own server. You have a connection to the Internet (hopefully, a fast one!) and perhaps a computer. Now, you want to get some server software and get things running. At this stage, a little advance planning can help you avoid much future grief. The following guidelines, followed by brief descriptions of the different servers available, should help you make your decision. The chapter concludes with a list of small applications that can help you analyze the server log file output to determine average usage and other diagnostic information.

BASIC SERVER ISSUES

The assumption is that you want to set up a reliable server and that you are not doing this just for fun (although fun should be a component in this process). You then need a server that is fast, easy to maintain, and reliable (the machine does not crash), since you want it to run 24 hours a

day (your server can be accessed at any hour, day or night). Given these criteria, there are a number of options available. In order of decreasing desirability, these are: UNIX servers, Windows NT or OS/2 servers, and other platform servers.

UNIX SERVERS

The consensus among WWW administrators is that a UNIX machine, combined with either the CERN or the NCSA HTTP servers, is the ideal package for distributing hypertext materials. There are several reasons for this choice. First, the UNIX operating system is designed to run many simultaneous processes (multitasking) so that a well-designed UNIX server can almost effortlessly respond to many simultaneous HTTP service requests. Also, UNIX is designed to isolate the server and user processes from the management level of the operating system, which ensures reliable operation even under heavy loads. Last, the powerful HTTP servers from CERN and NCSA were originally written for UNIX machines and are generally the most full-featured servers available. They are also freely available with precompiled executable versions available for UNIX computers from *Sun*, *Hewlett-Packard*, *Silicon Graphics*, and others. The source code is also publicly available, so you can download and compile it yourself.

If you use a UNIX server but have other machines, such as Macintoshes or PCs, you can make the server filesystems accessible to these users by remote mounting of the HTTP server directories on the PCs or Macintoshes. For example, **PC-NFS** allows PCs to mount UNIX filesystems, while **CAP** (Columbia Appletalk Protocol) allows similar access for a Macintosh. Setting up a development environment such as this requires some thought, as you must take into account the peculiarities of each system (for example, PCs allow only 8-character filenames and 3-character filename extensions); but, in general, this networked approach is useful in integrating a WWW development environment into a heterogeneous collection of computers.

Setting up a UNIX server need not be expensive. Although buying a machine from a vendor such as *Sun* or *Hewlett-Packard* is one option, you

can also purchase a standard PC clone and install the freeware UNIX clone **linux**. The NCSA server has been successfully ported to **linux**, so you can set up a **linux**-based HTTP server for little more than the cost of a PC.

The downside of the UNIX approach is complexity. UNIX is a sophisticated operating system and can be difficult to learn and manage. If you do not have someone in your organization familiar with UNIX, you should consider hiring someone who is; learning UNIX and installing an HTTP server are not things you want to be doing at the same time.

VMS SERVERS

The VMS operating system, which runs on VAX and some DEC ALPHA machines, is a multitasking operating system that, like UNIX, can also efficiently support server software, such as an HTTP server. There are currently two HTTP server packages for VMS. The CERN server, ported to VMS from UNIX, retains the features of the UNIX package but is reputedly slow. The **DECthreads** HTTP server is faster but does not have all the capabilities of the CERN package.

WINDOWS NT OR OS/2

Windows NT and OS/2 are the next best choice for hosting an HTTP server. Both Microsoft's Windows NT and IBM's OS/2 efficiently run simultaneous processes, and also carefully separate user processes (such as an HTTP server) from the management level of the operating system. These systems are also generally easier to configure than a UNIX system, particularly for those coming from a PC background (this may be more perspective than fact!). It is certainly true, though, that both OS/2 and Windows NT are easily integrated into PC networks, since this was a major goal of their system designs.

There are two servers available for these systems. **OS2HTTPD** is a port of the NCSA HTTPD Version 1.3 server to OS/2. **HTTPS** is a similarly powerful server written expressly for Windows NT.

WINDOWS OR MACINTOSH

There are a number of packages available for PCs running Windows, and for Macintoshes. Several of these are quite full-featured. The Windows or Macintosh option is at the bottom of the list because of the limitations of the operating systems. DOS, Windows, and the Macintosh System-7 operating systems are not designed for multitasking, nor are they designed to safely separate the management level of the operating system from user processes. The first limitation means that it is difficult to write an efficient HTTP server—in general, they quickly bog down when the number of HTTP requests grows large. The second limitation is the reason PC and Macintosh computers often crash while running innocuous programs, such as spreadsheets or word processing programs (or HTTP servers!). For a Macintosh or PC user, this is an irritation. For World Wide Web users trying to access your now *dead* server, this is a major annoyance.

If you are going to use a Windows or Macintosh HTTP server for serious dissemination of material on the Web, you should dedicate a machine specifically to that task. This means not letting anyone use it to do other tasks, since extra user programs significantly increase the risk of a system crash. In addition, you will want to regularly test that the system is *alive*. This is true of any server but even more so for platforms for which crashes are more likely. Nothing will impress prospective clients or customers less than a fancy service that does not work properly or is unavailable.

BEHIND A FIREWALL?

Finally, just a reminder that if you are behind a firewall and want to both run a server and access the outside world, then you need the CERN HTTP server. This is the only server providing the *proxy support* you need.

LIST OF SERVER SOFTWARE

The following is a list of the different server packages available for different platforms and operating systems. This includes URLs pointing to additional documentation and to places where you can obtain the actual programs.

CERN HTTP SERVER (UNIX, VMS)

The CERN HTTP server is a publicly available, full-featured server developed by Ari Luotonen, Henrik Frystyk, and Tim Berners-Lee of CERN. It supports the HEAD, GET, POST, and PUT methods, is fully CGI/1.1-compliant, and comes with several useful CGI applications, including a program for handling active images. It also supports file and directory access control and encrypted user-authentication. The CERN server can also act as a *caching* proxy server, which is ideal if you want to both run a server and also access other servers through an Internet firewall. There is both printable and online documentation. By reputation, it is somewhat more complicated to install and configure than the NCSA server.

Information about the current version can be found at:

```
http://info.cern.ch/hypertext/WWW/Daemon/Status.html
```

The current release is Version 3.0, which was made publicly available in late September 1994. Precompiled executables can be found at:

```
ftp://info.cern.ch/pub/www/bin/
```

in subdirectories corresponding to the name of your operating system. You will want the files *cern_http_x.x.tar.Z* and *cern_httpd_utils_x.x.tar.Z*, where *x.x* is the version number (currently 3.0). VMS software is located in the *vms* subdirectory. Information on installing the VMS version is found at:

```
http://delonline.cern.ch/disk$user/duns/doc/vms/www_run_httpd.html
```

The source code for the CERN HTTP server can be found at:

```
ftp://info.cern.ch/pub/www/src/cern_httpd.tar.Z
```

A VMS source code archive is found in the file *cern_httpd_vms.zip*. This directory also contains the online documentation in the archive file *cern_httpd_docs.tar.Z*. A PostScript version of the documentation is found in *cern_httpd_guide.ps.Z*.

CL-HTTP (SYMBOLICS LISP MACHINES)

CL-HTTP is an HTTP server written using the Common Lisp Object System (CLOS). It runs on Symbolics LISP Machines under Genera 8.3, although ports of the package to Common LISP running on other machines and operating systems are expected in the near future (most notably, Mac Common Lisp).

CL-HTTP supports the HEAD, GET, and POST methods. Gateway programs are possible but do not use the CGI mechanism, because, with lisp, an arbitrary lisp function can generate returned HTML for a given URL. This has two advantages: First, lisp is a much more powerful language for writing response functions than the typical scripting languages used with the CGI interface. Second, all these run as compiled and dynamically linked objects.

CL-HTTP uses a sophisticated object-oriented approach to pass information between the server and query engines. As a result, high-level abstractions are available for automatic form processing and other gateway applications. Access control through domain name or IP addresses is also possible at the level of the entire server or of any specific URL.

Additional documentation on the CL-HTTP server can be found at:

```
http://www.ai.mit.edu/projects/iiip/doc/cl-http/server.html
```

The source code for the package is available at:

```
ftp://ftp.ai.mit.edu/pub/users/jcma/cl-http/sources.tar.Z
```

while the most recent revisions are always available at:

```
http://research.ai.mit.edu/code/server/
```

A Lisp Machine distribution file is available at:

```
ftp://ftp.ai.mit.edu/pub/users/jcma/cl-http/lispm.reel-1
```

CL-HTTP was written by John C. Mallery of the Artificial Intelligence Laboratory, Massachusetts Institute of Technology. Support for the MIT Artificial Intelligence Laboratory's artificial-intelligence research is provided, in part, by the Advanced Research Projects Agency of the Department of Defense under contract number MDA972-93-1-003N7.

CMS HTTPD (VM/CMS)

There are still a lot of IBM mainframes about, and a lot of organizations are still built around the services provided by these machines. CMS HTTPD is ideal for these communities and allows them to implement a WWW server with a minimum of fuss.

CMS HTTPD supports the GET and HEAD methods and is fully CGI-compliant. At present, there are a limited number of CGI programs written for

the CMS system—the CMS HTTPD server author is currently writing a program to handle active images.

Additional information about CMS HTTPD is available at:

```
http://ua1vm.ua.edu/~troth/rickvmsw/rickvmsw.html
```

The source code is available in both VMARC and tar archive formats at the URLs:

```
http://ua1vm.ua.edu/~troth/rickvmsw/cmshttpd.vmarc
http://ua1vm.ua.edu/~troth/rickvmsw/cmshttpd.tar
```

CMS HTTPD was created by Rick Troth and is written entirely in REXX.

DECTHREAD HTTP SERVER (VAX/VMS)

The DECthread HTTP server is a native VAX/VMS HTTP server that uses the DECthreads model for multithreaded processing. As a result, this server should be significantly faster than the straightforward port of a UNIX server (such as the CERN HTTP server port), which is forced to run under the `inetd` to service more than one request at a time.

The DECthread HTTP server fully implements the HEAD and GET methods and supports CGI-compliant gateway scripts. The server comes equipped with example CGI scripts, including one to handle active images. There is also a facility to restrict file access based on the domain name of the client making the request; more sophisticated authentication schemes are also possible.

Additional information can be found at :

```
http://kcgl1.eng.ohio-state.edu/www/doc/serverinfo.html
```

which provides pointers to the server source code.

GN GOPHER/HTTP SERVER (UNIX)

GN is a single server supporting both the Gopher and HTTP protocols. It allows both protocols to access the same data so that GN is extremely useful for Information Services managers making a transition from a Gopher-based system to an HTTP-based one. GN is a freeware program, subject to the GNU general-public license.

GN fully implements the HEAD, GET, and POST methods, and supports CGI-compliant gateway programs. Access control is possible on a directory basis only, and is implemented by restricting access to particular Internet domain names or IP addresses. User authentication is not supported. The distributed package comes with installation instructions written from the point of view of Gopher service, so installation should be easy for those with this background.

The home page for GN information is:

```
http://hopf.math.nwu.edu/
```

This includes links for accessing the most current version of the source code. At present, the source code for GN and its support programs is found at:

```
ftp://ftp.acns.nwu.edu/pub/gn/gn.tar.Z
```

A GN server FAQ list is available at:

```
http://gopher.cwru.edu:70/0/about.gopher/gn.faq
```

GWHIS HTTP SERVER (UNIX)

The GWHIS HTTP server is a commercial server package distributed by Quadralay Inc. It supports the GET, HEAD, and POST methods, and is CGI-compliant. It also includes special components for data encryption, user authentication, and gateways to special Quadralay database software. Additional information is found at:

```
http://www.quadralay.com/
```

HTTPS (WINDOWS NT)

HTTPS is the only server that runs under native Windows NT. HTTPS supports the GET, HEAD, and POST methods, and is fully CGI-compliant. The server does not support access control or user authentication. These features are expected to be added to a planned commercial version.

HTTPS comes with extensive installation instructions and has convenient window-driven configuration menus. Additional information about HTTPS can be found at:

```
http://emwac.ed.ac.uk/html/internet_toolchest/https/contents.htm
```

The executable program, in zipped format, is found in the directory:

```
ftp://emwac.ec.ac.uk/pub/emwac/
```

There are three **.zip** files for the three different Windows NT hardware architectures. These are ***hsalpha.zip*** (DEC ALPHA), ***hsi386.zip*** (Intel 80386 or higher), and ***hsmips.zip*** (MIPS R4000 or higher).

HTTPS was written at the European Microsoft Windows NT Academic Centre (EMWAC) of Edinburgh University.

JUNGLE (UNIX)

Jungle is an HTTP server written in tk and tcl. This server supports the GET and POST methods. Additional information is available at:

```
http://catless.ncl.ac.uk/Programs/Jungle/
```

Jungle is under continuous development by the program author, Lindsay Marshall. **Jungle** will be of interest to those familiar with the tk/tcl languages who might be interested in participating in the development of a tk/tcl server. The author intends to implement all the features of the CERN HTTP server.

MACHTTP (MACINTOSH)

MacHTTP is the only Macintosh HTTP server. **MacHTTP** supports the GET and POST methods, and is CGI-compliant. Instead of invoking UNIX-shell scripts, MacHTTP can invoke and pass arguments to **AppleScript, MacPerl, HyperCard,** or any custom application written to support **MacHTTP's** custom suite of AppleEvents. Sample code is available.

MacHTTP also has an interface to **AppleSearch** that allows the server to present searchable indexes. This interface is provided by the companion application **AppleWebSearch**, which makes any **AppleSearch** source-accessible on the World Wide Web.

MacHTTP supports access control on a per-machine or per-domain basis by denying (or allowing) access to machines or groups of machines with specific IP addresses. **MacHTTP** also supports the Basic authentication scheme that allows access to be controlled document by document.

Additional information about **MacHTTP** is found at:

```
http://www.uth.tmc.edu/mac_info/machttp_info.html.
```

which includes pointers to useful **AppleScript** and other support programs. Another useful site is:

```
http://www.uwtc.washington.edu/Computing/Internet/MacintoshWebDevelop.html
```

The **MacHTTP** package is available from:

```
ftp://oac.hsc.uth.tmc.edu/public/mac/MacHTTP/machttp.sit.hqx
```

This executable will run on all Macintoshes, including those using the PowerPC chip. The archive includes hypertext documentation and installation instructions. **MacHTTP** was written (and is being continually improved) by Chuck Shotton.

NCSA HTTPD (UNIX)

The NCSA public-domain HTTP server implements the HEAD, GET, and POST methods, is fully CGI-compliant, and supports access control through encrypted user-authentication (although you must supply the authentication routines separately) and domain-name restrictions (restricting access to users connecting from registered Internet domains). The distribution package comes with several useful gateway programs, including a program for handling active images (*imagemap*).

The NCSA server also permits executable HTML documents, which allows you to create parsable HTML documents that *include* other documents on-the-fly. This is often called the *server-side include* feature. The server can also be configured to allow users to have executable CGI programs in their own home directories.

The NCSA server comes with online documentation and installation instructions. Printed documentation is also available. The NCSA server is easy to install and configure, and is an ideal choice if you don't require proxy server capabilities.

Additional information is found at:

```
http://hoohoo.ncsa.uiuc.edu/docs/Overview.html
http://hoohoo.ncsa.uiuc.edu/docs/FAQ.html
```

while a PostScript version, can be found at:

```
http://hoohoo.ncsa.uiuc.edu/docs/postscript-docs/Overview.html
```

The current release of the source code is available at:

```
ftp://ftp.ncsa.uiuc.edu/Web/httpd/Unix/ncsa_httpd/current/httpd_source.tar.Z
```

while executables for various platforms are found at:

```
ftp://ftp.ncsa.uiuc.edu/Web/httpd/Unix/ncsa_httpd/current/
```

OS2HTTPD (OS/2)

OS2HTTPD is a port of the **NCSA HTTPD** Version 1.3 server to OS/2. **OS2HTTPD** supports virtually all the features of NCSA HTTP Version 1.3, including the HEAD, GET, and POST methods; CGI gateway programs; file and directory access control; and server-side includes.

Additional information can be found at:

```
ftp://ftp.netcom.com/pub/kfan/overview.html
```

and the executable is found in the same directory:

```
ftp://ftp.netcom.com/pub/kfan/
```

The current release is found in the file *web2-101.zip*, but this release number will change with updates and improvements.

The author, Frankie Fan, has released **OS2HTTPD** to the public as a freeware package for noncommercial use only.

PLEXUS (UNIX)

Plexus is an HTTP server written in the perl language. It is a public-domain server designed to be fast, extensible (for prototyping custom servers with new protocols and interfaces), and easy to use. **Plexus** requires perl 4.0 patchlevel 36 or later, and runs on almost any UNIX platform. **Plexus** fully implements the HEAD and GET methods, and is CGI-compliant. **Plexus** has a limited facility for user authentication and access control.

Gateway programs are possible using the CGI specification and also using the **Plexus** native gateway interface (which is around ten times faster than the CGI, since the **Plexus** interface does not require an exec()). Native gateway programs are treated as additional perl modules to the server. An explanation of how this works, along with example gateway programs, are included with the online documentation.

Additional information about **Plexus** can be found at:

```
http://www.bsdi.com/server/doc/plexus.html
```

The source code is available at:

```
ftp://ftp.earth.com/plexus/3.0-beta/prerelease/Plexus-3.0m.tar.Z
```

Plexus was written by Tony Sanders, a member of the technical staff at Berkeley Software Design, Inc.

SERWEB (WINDOWS 3.1)

SerWeb is a simple, easy-to-use HTTP server that supports the GET method only. It does not support gateway programs or access control. However, it is easy to install and run, and comes with source code (in C++) for those interested in developing their own server software.

SerWeb can be found at:

```
ftp://128.83.218.12/pub/windows/winsock/serweb03.zip
```

and a version for Windows NT is available at:

```
ftp://emwac.ed.ac.uk/pub/serweb/serweb_i.zip
```

SerWeb was written by Gustavo Estrella.

WEB4HAM (WINDOWS 3.1)

Web4Ham is a Windows 3.1 server that supports the GET and HEAD methods only. The most recent version (0.17) does have some support for gateway programs, but the documentation is limited. However, the package has been improving rapidly since its debut (March 1994).

Additional information is available at:

```
ftp://ftp.informatik.uni-hamburg.de/pub/net/winsock/
```

The server software is found in the file ***web4ham.zip***, while documentation is found in ***web4doc.zip***. **Web4Ham** was written by Gunter Hille.

WINHTTPD: NCSA HTTPD FOR WINDOWS (WINDOWS 3.1)

WinHTTPD (also called **WHTTPD**) is a port of the NCSA UNIX server to Windows. It retains most of the features of the UNIX version: It supports the GET, HEAD, and POST methods, and is fully CGI-compliant.

The distribution archive comes with several example gateway applications, including programs for handling active images. **WinHTTPD** also supports file and directory access control, and allows for sophisticated user-authentication—you have to provide the authentication programs separately.

Additional information about **WinHTTPD** can be found at:

```
http://www.alisa.com/win-httpd/
```

while the program archive is found at:

```
ftp://ftp.ncsa.uiuc.edu/Web/httpd/Unix/ncsa_httpd/contrib/winhttpd/
```

in the file ***whtp13p1.zip*** (Version 1.3 Prelease 1). Note that the revision numbers change rapidly. The NCSA FTP server is often busy: If you are having trouble accessing it, use **archie** to find another location.

COMING ATTRACTIONS

The following servers are announced packages that will be available in the near future.

BASIS WEBSERVER

The BASIS WEBserver, a product of Information Dimensions Inc., will be an HTTP server with added BASISplus capabilities. BASISplus is a relational database system proprietary to Information Dimensions. The BASIS WEBserver will be equipped with gateway programs to allow clients to interact with the BASICplus system and databases.

MDMA: MULTITHREADED DAEMON FOR MULTIMEDIA ACCESS

MDMA is a new HTTP server written explicitly for the Sun Solaris operating system. This server contains several exciting new features, including multithreading for speed and dynamic extensibility, to allow for dynamic inclusion of gateway-like functionality directly into the server.

MDMA is currently available as a Solaris executable and as source code, but is still a bug-filled alpha release. Additional information is available at:

```
http://elanor.oit.unc.edu/mdma.html
```

NETSCAPE NETSITE

Netscape Communications Corp. is designing new WWW servers that will have significant enhancements over current first-generation servers. The first product, **Netscape Netsite**, was introduced in late 1994, and is designed to be significantly faster than the NCSA server. The commercial version implements RSA Data Security, Inc. cryptographic technology to allow for secure commercial transactions over the Internet. For additional information, you can send e-mail (*info@mcom.com*), or look at:

```
http://mosaic.mcom.com/
```

SERVER SUPPORT PROGRAMS

Most server packages produce log files that record information about each server request. Server support programs are designed to read these log files and to convert the raw log file data into useful summaries of server usage. Not all HTTP servers use the same log format, so some of the support programs are specific to particular servers. The more recent CERN and NCSA servers (and the various ports of these packages) use a *common log file format*, and most newer support programs are designed around this standard.

GETSTATS

Getstats is a C-language program that can produce log summaries for the CERN and **NCSA HTTP** servers, **GN**, **Plexus**, **MacHTTP**, and UNIX **Gopher** servers. The **getstats** package is exceptionally well documented at:

```
http://www.eit.com/software/getstats/getstats.html
```

while the source code can be found at:

```
http://www.eit.com/cgi-bin/viewsource/software/getstats/getstats.c
```

There is an HTML **FORM** interface that allows users to access server statistics using **getstats**. Information about this interface is found at:

```
http://www.eit.com/software/getstats/getstats.html#form
```

Getstats was written by Kevin Hughes of Enterprise Information Technologies, Inc.

GWSTAT

Gwstat is a UNIX package of programs and scripts that can convert HTML output from the program **wwwstat** into GIF format graphs of server statistics. These can be inserted inline in HTML documents. **Gwstat** does not do all this itself, and requires the packages **Xmgr** (a data plotting package), **ImageMagick** (an image format conversion package), and **ghostscript** (a PostScript interpreter).

Additional information on **gwstat** can be found at:

```
http://dis.cs.umass.edu/stats/gwstat.html
```

This includes general information about **gwstat**, as well as directions for finding the applications **xmgr**, **ImageMagick**, and **ghostscript**. **Gwstat** itself can be found at:

```
ftp://dis.cs.umass.edu/pub/gwstat.tar.gz
```

Gwstat was written by Qiegang Long.

WEBSTAT

Webstat can read common format log files (NCSA HTTP Version 1.2 (or later) and the CERN server), and return statistical information about server usage. The main features of **webstat** are:

- Generation of usage report aggregated per service and requesting domain
- Generation of usage report per connecting country
- Generation of usage report per document
- Generation of reports on usage per year, month, week, or day
- Automatic generation of summaries as HTML documents

Additional information about **webstat** can be found at:

```
http://www.pegasus.esprit.ec.org/people/sijben/statistics/advertisment.html
```

while the program itself (written in **python**) is available at:

```
ftp://ftp.pegasus.esprit.ec.org/pub/misc/webstat.tar.gz
```

You will also need the **python** program by Guido van Rossum. This is found at:

```
http://www.cwi.nl/ftp/python/index.html
```

Webstat was written by Paul Sijben of the University of Twente.

WUSAGE

Wusage is a C program that generates simple weekly-usage reports in the form of HTML documents, including inline image graphs displaying server usage and the distribution of accesses by continent. A particularly nice feature is the ability to exclude irrelevant document retrievals (of inline images, from local machines, etc.) from the analysis. **Wusage** is able to read log files in the new, common log format, as well as older log files produced by the NCSA Version 1.1 server.

Additional information about **wusage** can be found at the **wusage** home page:

```
http://siva.cshl.org/wusage.html
```

The source code for **wusage** can be found at:

```
ftp://isis.cshl.org/pub/wusage/
```

The current release is Version 3.2, found in the file *wusage3.2.tar.Z*.

Wusage was written by Thomas Boutell. **Wusage** 3.2 is copyrighted by the Quest Protein Database Center, Cold Spring Harbor Labs.

WWWSTAT

Wwwstat is a **perl** program that can read common log file formats (NCSA Version 1.2 or newer, CERN HTTP servers, and others) and produce a log summary file as an HTML document suitable for publishing on your server. The package is available at:

```
ftp://liege.ics.uci.edu/pub/arcadia/wwwstat/wwwstat-1.01.tar.Z
http://www.ics.uci.edu/WebSoft/wwwstat/wwwstat-1.01.tar.gz
http://www.ics.uci.edu/WebSoft/wwwstat/wwwstat-1.01.tar.Z
```

The distribution consists of several components, including installation instructions. It also contains the perl program **oldwwwstat**, which can be used to analyze log files produced by the NCSA Version 1.1 server.

Wwwstat was written by Roy Fielding of the University of California at Irvine.

CHAPTER

9

REAL-WORLD EXAMPLES

Chapter 9 is a change from the rest of the book. This chapter contains some examples of World Wide Web sites that illustrate the types of things that can be accomplished on the Web, and that also illustrate aspects of the document design process. These examples cover areas ranging from scientific research to comic strips. I asked the authors of these collections to write a brief description of their site and of the design issues they found important in developing these materials. I encourage you to read these brief descriptions and then to visit their sites—I can guarantee that you will find the exercise enjoyable and educational.

ELECTRONIC E-PRINT SERVERS

http://xxx.lanl.gov/

Paul Ginsparg
Los Alamos National Laboratories
http://xxx.lanl.gov/pg.html

The Los Alamos National Laboratory *e-print* server is an automated archive for electronic communication of research information. In many fields of physics, e-print servers have become the primary means of communicating ongoing research work, since they are inexpensive, democratic (anyone with Internet access can access the work), and fast. There are currently eight different physics e-print databases, archiving several tens of thousands of electronic articles. These articles are all written in TeX or LaTeX, which have become the text formatting languages of choice in the physics community.

E-print articles are submitted by electronic mail to an archive site, using a special electronic letter format containing both the article itself and an abstract. An automatic program running at the archive site (rather like a LISTSERV program) automatically unpacks the letter and archives the abstract and the full article, and adds both to a searchable database.

Originally, this archive was accessible only by e-mail so that accessing the archive required sending an e-mail letter containing search and command strings (such as "find all papers by Jones written this year"). The results of the search were then mailed back to the person who sent the request. This access method still works, but has been largely supplanted by a more efficient World Wide Web search interface. Submission of articles, however, still must be done by electronic mail.

Figure 9.1 shows the home page for the Los Alamos e-print server. Note that this allows for both FORM and non-FORM access to the database. When the project started, the FORM element didn't exist. While everyone should, in principle, now have access to a FORM-compliant client, the lower-end clients are frequently painful to use (the more sophisticated the use of a graphical interface, the harder it is to duplicate in a terminal mode). Therefore, it is deliberate that everything possible from the FORM page is also possible via ordinary pages. Note that the button that leads to the FORM interface is itself a FORM, so it functions only for clients that understand FORMs.

The home page design is designed to put near the top the things people access most frequently so that you need only to scroll down for secondary

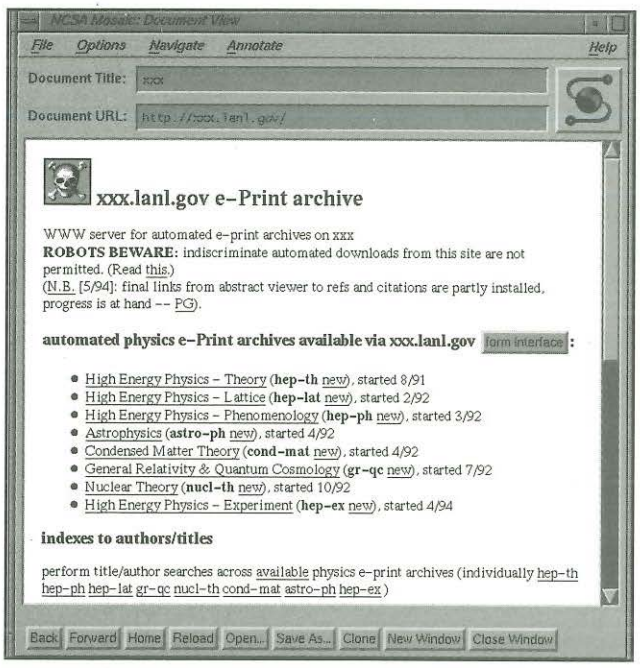


Figure 9.1 The home page for the Los Alamos e-print server.

information. Since this was an overlay to a preexisting database, already divided into separate *archives* corresponding to different subfields of physics, it was natural to branch from the home page to a separate page for each of the archives. On the other hand, it was important to avoid the common error of slavishly reproducing exactly the functionality familiar from the previous interface.

In particular, I wanted the *new* link to daily abstract listings to be immediately accessible in order to convey the ease with which these listings could be obtained, and to eventually help people move away from the less efficient e-mail interface.

The home page is one of the few static HTML documents in the entire interface—the rest of the pages are generated by CGI scripts, and all the HTML is added dynamically at request time to the preexisting plain text files used by the e-mail interface. This has two advantages: Only a single set of files has to be maintained for both formats; and any change in the interface (e.g., a new link on an abstract page) can be made globally by changing a single

script. In the next revision, even this front page will be generated by a gateway to make it easier to maintain and update.

The skull icon was originally put in as a joke during testing (it was grabbed from the *kill process* button on the processes panel from NextStep—a 2-bit monochrome image, colorized using **xv**). It was originally there to evoke the skull that appears together with *xxx* on bottles of poison, but later came to symbolize the death-defying survival of the system in the absence of funding support.

Figure 9.2 shows a typical FORM page. The FORM page is generated by a script that knows where you came from (the default selectors are *hep-th* from the front page, but if you come from the *cond-mat* page, they would be *cond-mat*). This page contains four different archive selectors in the four different FORMs. It is unfortunate that I had to use separate archive selectors, due to the limitations of the original NCSA FORMs implementation—a FORM could have only a single submit button, so different actions from different FORM submit buttons were not possible. I chose not to use horizontal line `<hr>` elements between the separate **FORMs**, since that stretched things out vertically and sometimes required users to scroll to see the full **FORM**.

Mosaic for X-Windows Version 2.5 does allow multiple submit buttons supporting different actions from within a single **FORM**, but it will be some time before all browsers upgrade to this feature.

The basic methods of obtaining information from the e-mail interface—find, list, get, and help—are also the basic functionalities in the FORM interface, except that *get* is restricted to *get abstract*. This allows the user to read the abstract and then decide if he or she wants to download the full article, and ensures that large file downloads are not made inadvertently.

The Find and Listings commands use the same output as the e-mail interface, except that the document is piped through **sed** (the UNIX stream editor) to automatically reformat the text, and to pattern-match appearances of the form "arch-ive/9401001" and convert them to hypertext anchors in the displayed document. The result of a Find search is shown in Figure 9.3, searched from the past year's articles containing the name "graham."

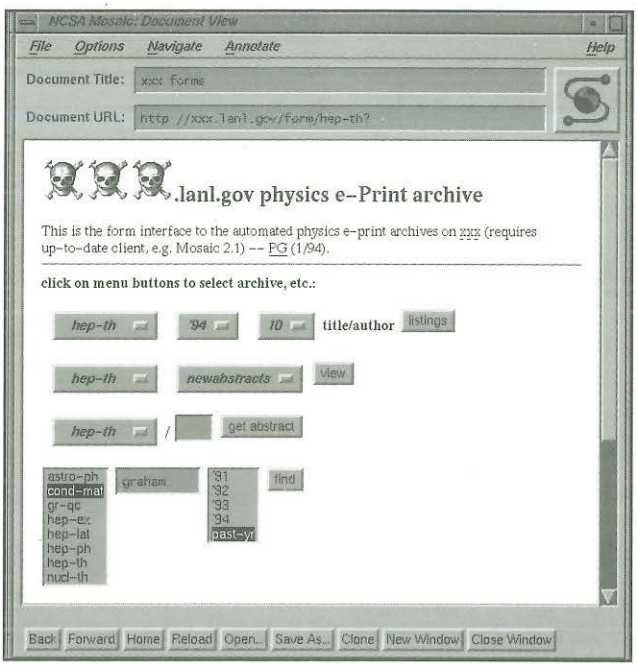


Figure 9.2 The HTML **FORM** database interface.

Clicking on the *abs* anchor downloads the article abstract. This is shown in Figure 9.4

The abs script just pipes the conventional e-mail abstract (in this case, ***cond-mat/papers/9404/9404016.abs***) through **sed** to reformat the document, maintaining the basic feel of the plain text abstract but changing the title to a header, and, of course, adding the hypertext link to the source.

The document source interface is similar to the e-mail interface, but takes advantage of HTTP's ability to transmit binary data files. Thus, the returned articles are sent in compressed formats when possible, significantly reducing the sizes of the transmitted data—most clients read the HTTP content-encoding MIME header and automatically decompress the data upon receipt. In contrast, the e-mail interface requires decompressed and specially coded data to ensure safe communication of the document.

One overall design choice was to keep the URL names as simple and logical as possible, using the NCSA HTTP ScriptAlias feature. For example, articles,

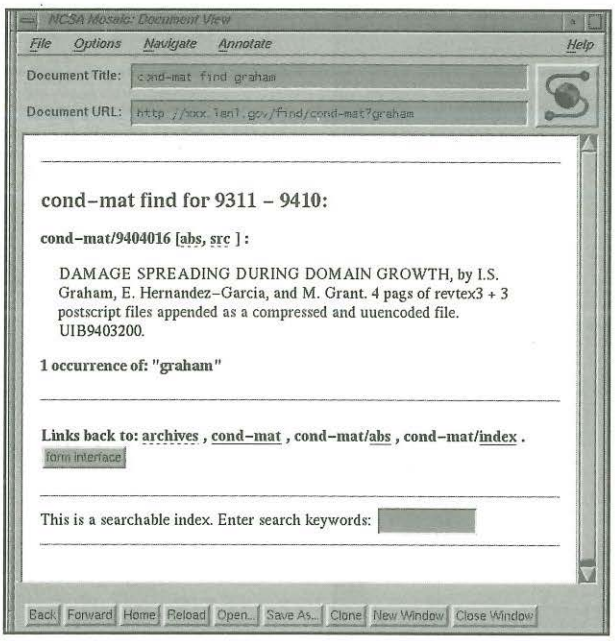


Figure 9.3 Result of searching with the bottom form from Figure 9.2 for the name "graham."

abstracts, and gateway programs are referenced via directory and filenames, such as */list/hep-th/9401*, */abs/hep-th/9401001*, */find/hep-th?*, */e-print/hep-th/9401001*, and so on. Since the entire package is generated by scripts, the leading field is *aliased* to the appropriate script, which parses the rest of the URL to determine archive and other information on which to operate.

ONCOLINK

`http://cancer.med.upenn.edu:/`

E. Loren Buhle, Jr., Ph.D.
University of Pennsylvania School of Medicine
P.O. Box 7806, Philadelphia, PA 19101
(E-mail: *BUHLE@XRT.UPENN.EDU*)
Phone: 215-662-3084
Fax: 215-349-5978

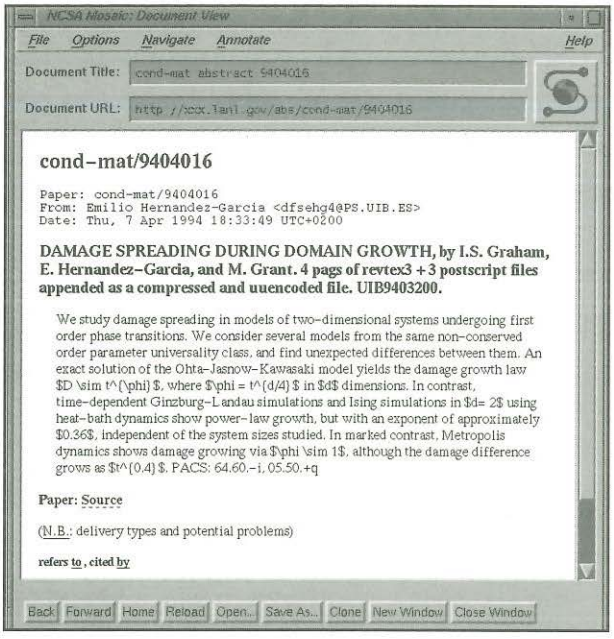


Figure 9.4 Abstract for the article found in Figure 9.3.

INTRODUCTION

OncoLink was launched on the Internet on March 7, 1994 to provide comprehensive and timely information regarding cancer to patients, physicians, health care workers, and other interested parties via either Gopher or World Wide Web clients. The type of information on this server ranges from methods of diagnosis and treatment of cancer, to the many psychosocial elements a patient and his/her family may experience throughout the cancer experience.

The material on OncoLink must be traversed with ease, regardless of topic or educational background of the user. OncoLink users range from those with no prior knowledge of biology to practicing clinical oncologists. The material on OncoLink must be both comprehensive and easy to navigate. Thus, many of the documents have extensive hypertext links to allow inexperienced users to learn as they go, while not encumbering the more experienced users. Some of the material on OncoLink is explicit, such as the stereotactic radiosurgical approach to brain tumors. Other material deals with more amorphous issues, such as quality of life, and death and dying.

THE ONCOLINK IMPLEMENTATION

The World Wide Web server GN (discussed in Chapter 8) was implemented on a DEC 3000-800 computer running the OSF/1 operating system. The GN server allows rapid dissemination of both hypertext and plain text information to both World Wide Web and Gopher clients, respectively. The GN server software was selected on the basis of its simplicity of design; efficiency of serving documents to both the World Wide Web and Gopher clients; and the security it afforded the server. The GN server software was felt to be the most secure WWW server in that the resources served to the client must be defined within the server in a predetermined menu file. While this menu file is cached to provide speedy service, no externally suggested documents or actions are supported if they are not explicitly stated in this menu. This arrangement tightly controls the behavior of the WWW server and appears to preserve the integrity of the computer and its resources. The GN software is quite portable, making the transfer of OncoLink to an HP 9000/735 computer, running HPUX for purposes of experimental mirroring, a trivial exercise.

The OncoLink home page (Figures 9.5 and 9.6) is accessed by Wide World Web clients at the URL:

```
http://cancer.med.upenn.edu/
```

It can also be accessed from a Gopher client via the command:

```
gopher cancer.med.upenn.edu 80
```

since the server runs on port 80 rather than the usual default port of 70.

The actual name `cancer.med.upenn.edu` is an alias for an actual numerical Internet (IP) address of a machine on the Internet. Should OncoLink move to another machine, the name `cancer.med.upenn.edu` would remain the same, and the alias would be changed in the Internet nameserver tables to reflect the transition.

To track usage patterns, GN generates a detailed log of all access transactions. This log is a very important and powerful aspect of information publishing via the Internet, providing valuable feedback to OncoLink's maintainer. In addition to providing an insight to OncoLink's most popular

resources, the time between hypertext key requests and the path between the selection of one key and the next hypertext key gives an insight to the information sought; the portions of the document read; and the time the requester spent on each section of the document. Studying this log also revealed how many requests were from graphical WWW clients, text-based hypertext-capable WWW clients, and Gopher clients.

USE OF ONCOLINK

OncoLink was released on the Internet as a WWW and Gopher server on the evening of March 7, 1994. In the first eight months of operation, OncoLink has logged a half million transactions from more than 80 countries throughout the world. Approximately 80 percent of these accesses were from WWW clients. OncoLink access grows on an average of 10 percent per month, though a growth of 30 percent has occasionally been observed. The average weekday access rate is generally three times heavier than weekend use. Within a given day, OncoLink is accessed most heavily between 8 A.M. and 11 P.M. EST, though use between 1 and 6 A.M. EST is typically heaviest from Australia, New Zealand, and Japan, reflecting their working hours.

From the Internet domain names, we have determined the scope of OncoLink users to be international, with accesses from essentially all settled continents, even if English is not the primary language of the country (e.g., Japan, Germany, Poland, Israel, Brazil, etc.). OncoLink has been advertised only in the electronic environment, with minimal conventional print-media announcements.

```
<HTML>
<HEAD>
<TITLE>Welcome to OncoLink</TITLE>
<LINK REV="MADE" HREF="mailto:buhle@xrt.upenn.edu">
</HEAD>
<BODY>
<H2>Welcome to OncoLink</H2>
<H1><img alt="Oncolink logo" src="/I/docs/images/oncolnk2.gif">
<A HREF="/s/sounds/welcome.au">
<img alt="" src="/I/docs/images/www/sound.xbm"></A></H1>
```

```
<p>
<H2>The University of Pennsylvania Multimedia Oncology Resource</H2>
<HR>
<A HREF="/0h/stuff/upmc">
<img alt="Info about Penn" src="/I/docs/images/penn1.gif"></A>
<A HREF="/I/docs/images/child.gif">
<img alt="Children's artwork" src="/I/docs/images/child_sm.gif"></A>
This artwork was donated by a pediatric cancer patient and <b>will change
</b>at frequent intervals!
<i>(Click here to see the gallery of children's artwork)</i>
<A HREF="/0h/docs/images/child/gallery1">
<i>in Part 1</I></A> or <A HREF="/0h/docs/images/child/gallery2"><i>Part
2</i></A><p>
<HR>
<img src="/I/docs/images/light_bulb.gif">
<A HREF="/1s/stuff/latest_stuff">
What's <i>NEW </i> on OncoLink?</A><p>
<img src="/I/docs/images/question.gif">
<A HREF="/7wc/stuff/search_wais/OncoLink.inv">
<b>WAIS Search</b> of OncoLink</A> or a <A
HREF="/7g/stuff/search_dir1"><b>single-word SEARCH</b></A> of OncoLink.<p>
<img alt="" src="/I/docs/images/news1.gif">
<A HREF="/1s/buhle/cancer_news">
CANCER news, warnings, etc.</A>
<P>
<HR><img alt="" src="/I/docs/images/menat.gif">
<i>This cancer information resource is under continuous development.
OncoLink is best used with a World-Wide Web client (e.g. Mosaic, lynx,
etc.), but can also be reached via a gopher server, using
port 80.</i><BR><b>October has been designated as Breast Cancer
Month in the United States of America and Canada.</b><p>
<HR>
.
.
.
<b>OncoLink</b> was awarded the International
<b>Best of the Web '94
Award</b> for <b>Best Professional Service</b>. To see the other award
winners, please
<A HREF="http://wings.buffalo.edu/contest/awards/index.html">click
here</A>.
 Thank you for your support and we look forward
to making this resource even more valuable!
</BODY>
</HTML>
```

Figure 9.5 HTML document of OncoLink home page (this file has been truncated to save space).

Gopher access could be discerned by the selection of the non-HTML files. In the early days of OncoLink, the cancer resources served by Gopher and WWW were essentially identical, the Gopher resources lacking only the multi-

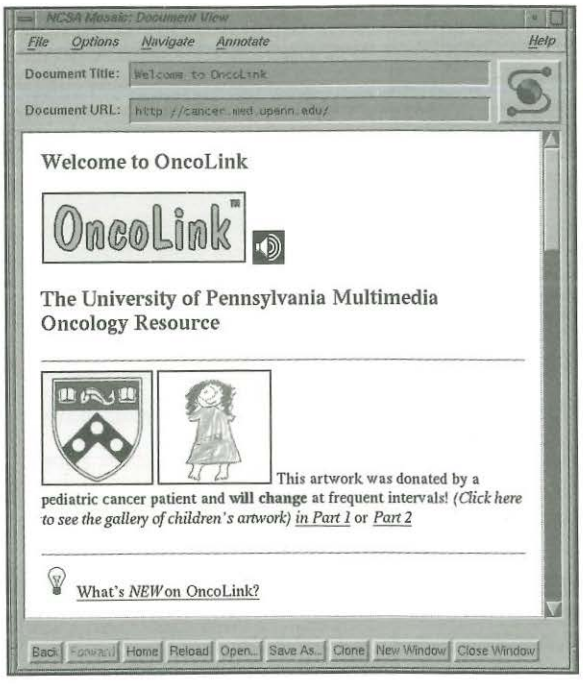


Figure 9.6 Home page to the OncoLink resource site.

media and the hypertext. As OncoLink developed, the Gopher menus were extended to allow access to contents formerly available only within the hypertext keys of HTML documents. Documents such as the *Pediatric Oncology Case of the Month* progressed in a more nonlinear direction, requiring access to hypertext keys to access the treatment given. While still represented in the classical linear or sequential fashion of printed literature (i.e., the entire document is laid out in a chronological sequence), the hypertext documents such as the *Case of the Month* remains the most popularly requested document. An example of a *Case of the Month* is shown in Figure 9.7.

A critical element to disseminating information on the Internet is to listen to the user and generate a flexible and intuitive interface that accommodates the user's needs. OncoLink began with discrete menus defining specialties within the practice of oncology (cancer) treatment. Thus, there were menus pertaining to Pediatric Oncology and Gynecologic Oncology, as well as Medical Oncology, Surgical Oncology, and Radiation Oncology. Our users

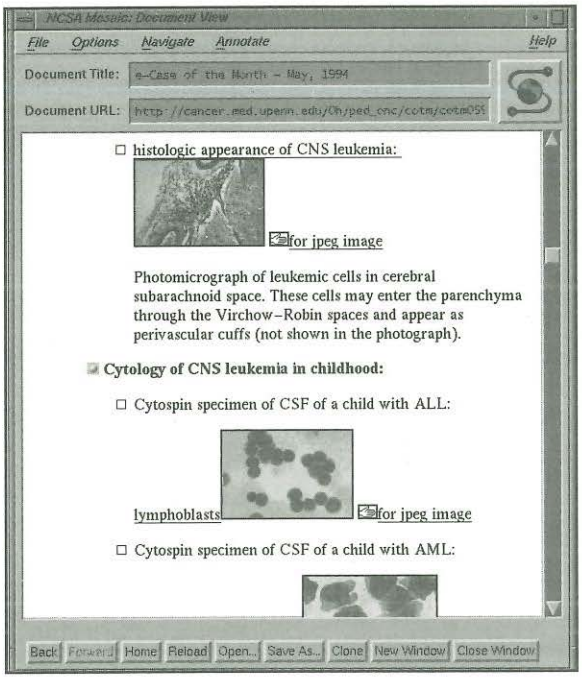


Figure 9.7 Case of the Month page. This is a detailed scientific document of a particular cancer case, including hypertext links to related documents and to full-scale images of the scientific images and figures.

quickly requested a disease-oriented selection of menus, as a breast cancer patient would not necessarily know which menu item to select first. We rapidly discovered that a certain class of OncoLink users rarely used the menus at all, preferring to move from document to document via the hypertext links, reading very small portions of the documents they accessed. When we looked at what portions of the documents the users read, we realized there was a tight focus on information sought. Thus, some users browsed the many items on OncoLink, while others focused their attention very sharply for certain pieces of information.

World Wide Web documents that used icons of images for hypertext keys were more apt to be selected than merely hypertext documents. Using several pediatric oncology *Case of the Month* presentations, where the user is led through the initial presentation of the patient to the patient's treatment,

we recorded the sequence of hypertext selections as a given user (tracked by following an IP address in a contiguous block of time). While physicians usually selected the hypertext keys in the order of their presentation, the completeness of the selection was higher when the hypertext anchors were miniaturized icons of the full-size figure. In documents targeting patients, iconic hypertext keys for procedures were often selected over text-based hypertext keys.

OncoLink must also allow inexperienced users, who are not exactly sure what their questions are, to make use of the many navigational aids to find what they are looking for in the OncoLink resources. One of the design criteria behind OncoLink is to allow users to find something relevant to their queries within five minutes of their first interaction with OncoLink. By using the search mode (Figure 9.8), a user can rapidly examine pertinent articles contained anywhere on OncoLink. When OncoLink numbered just a few hundred documents, an internal searching function (using a fast *grep*) was used. While fast, this search procedure returned an unweighted list of OncoLink documents containing a case-insensitive match. This was useful for examining potential hypertext links from the perspective of generating new hypertext links, but was hardly useful for users unfamiliar with the contents of OncoLink. When OncoLink grew to over 1,500 documents, this type of search became quite slow.

OncoLink now employs WAIS and a similar *glimpse* search to give the user a prioritized list of likely resources. The interface to these tools is shown in Figure 9.8, while the results of a typical search (for *Tamoxifen* and *breast*) is shown in Figure 9.9. Soundex and phonix algorithms were employed to help users who could not spell the medical terminology. In the case of exact matches, if users could not spell the search word (e.g., rhabdomyosarcoma), they had great difficulty finding relevant material. The search facility also allows Boolean query syntax (e.g., red and green, not blue), as well as partial search requests (hum* locks the search on any word starting with hum).

The ability to use search tools was codified into previously *static* links to provide *dynamic* links. An example of a simple dynamic link would be to

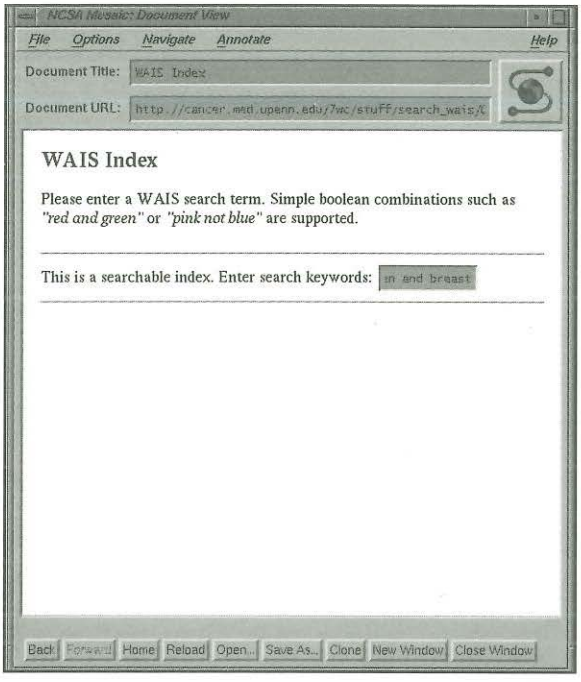


Figure 9.8 OncoLink WAIS search page. The documents at OncoLink are indexed by a WAIS database. These can be searched to find useful documents. In this case, the search is for *"Tamoxifen"* and *"breast."*

request "All the recent papers on breast cancer" from the National Institute of Cancer. The result of this hypertext selected query would be different from month to month. A more complicated example is the selection of the hypertext link *taxol.* OncoLink has dozens of different articles discussing taxol. Is the user interested in the isolation of taxol from the Yew tree; the patient side effects of taxol; clinical protocols using taxol; the molecular action of taxol; or the organic synthesis of taxol? A static link to only the side effects of taxol addresses only some of the issues. If the server tracks the type of documents requested on OncoLink, the server may be able to steer patient-oriented articles to the patient, clinical articles to the oncologist, and so on.

OncoLink strives to present *one-stop shopping* for people involved with cancer. Clearly, all the knowledge of cancer will never be stored on OncoLink. Much of the information on cancer will be shared elsewhere on the Internet.

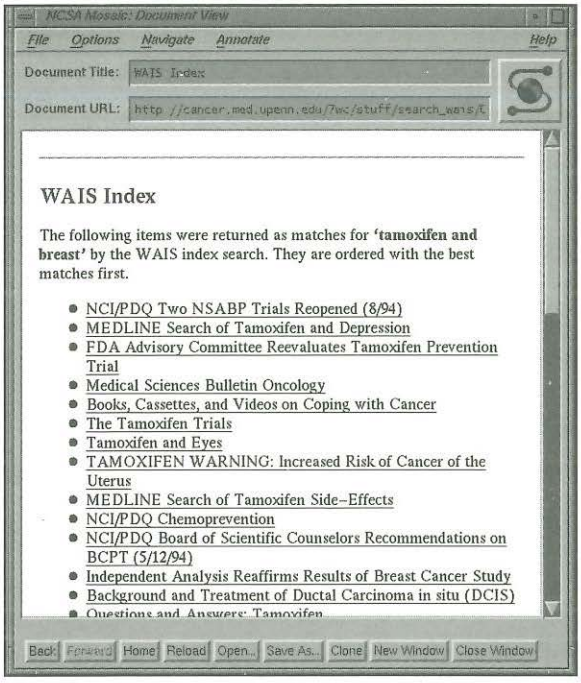


Figure 9.9 OncoLink WAIS search results. The documents listed contain information related to *"Tamoxifen"* and *"breast."* Clicking on the highlighted text downloads the document for viewing. This list is created by a gateway program that sorts the search results and inserts the hypertext links.

The ability to reproducibly find this information, as well as monitor the issues of concern of OncoLink's users and the Internet cancer population, is an area for future work using active and passive computer agents. Active agents, also called KnowBots and World Wide Web Worms (WWWW), are useful for indexing the contents of other information services on the Internet. Passive agents, also called sentries, can monitor the USENET newsgroups, Internet LISTSERVers, and other forums of electronic cancer information (e.g., CompuServe's CANCER forum).

OncoLink is developing interactive sessions much like the multithreaded bulletin board services found on CompuServe. The developers see their goal as an interactive television service.

VIEWS OF THE SOLAR SYSTEM

http://www.c3.lanl.gov/~cjhamil/SolarSystem/homepage.html

Calvin J. Hamilton
Los Alamos National Laboratory
Mail Stop 3265
Los Alamos, New Mexico 87545

BACKGROUND

While browsing the Internet during the fall of 1993, I ran across some images of planets in our solar system, which sparked in me a latent interest in astronomy. I subscribed to *sci.astro* and related astronomy USENET newsgroups and, after a short time, found the major Internet repositories of deep space and solar system images.

I found many nice images, but the selection was not very complete. In particular, there were very few images of Pluto, Uranus, Mercury, and of the satellites of the planets. Some of the images were well done; others were only sloppy scans of photographs. I could also tell that many of the images had been subsampled or reduced in size from the original data that was available.

It is understandable why larger images were not provided on the Internet. Large images require a lot of disk space, and individuals who want to download them to their personal computers may not have the ability to handle or view such large files. However, I considered the selection and resolution of images unacceptable. I had seen very high-quality photographs in books about astronomy and I wanted those images, in high-quality formats, to be available to the public via the Internet.

I located the raw image data taken by the Voyager and Viking spacecraft at the anonymous FTP site ***explorer.arc.nasa.gov***, but, without a lot of image processing, these images displayed poorly on a computer. Using my image-processing background, I began filtering and enhancing the raw data and produced some high-resolution image mosaics and image files. When I created inline images, I tried to make them display well on 8-bit color computer

monitors (recall that an 8-bit monitor can display only 256 different colors at the same time).

At about this same time, I was introduced to NCSA **Mosaic** and the World Wide Web. I could see that this would be an ideal way to make my images available to the public. Although I did not know it, this was the beginning of Views of the Solar System (Figure 9.10). I started by making an HTML document for each planet, with links to my images and the Jet Propulsion Laboratory/NASA (JPL/NASA) images. Then, I decided that it would be nice to have some statistical information about each planet. I went to the library, located some books on astronomy, and added a few selected statistics to each of the planetary pages. By this time, I had become fascinated with the solar system and with astronomy. Every couple of weeks, I checked out a stack of astronomy-related books. From these books, I got the idea of including a short description for each of the planets. I also found that there was a lot of information from JPL/NASA available on the Internet, which I also incorporated into the tour.

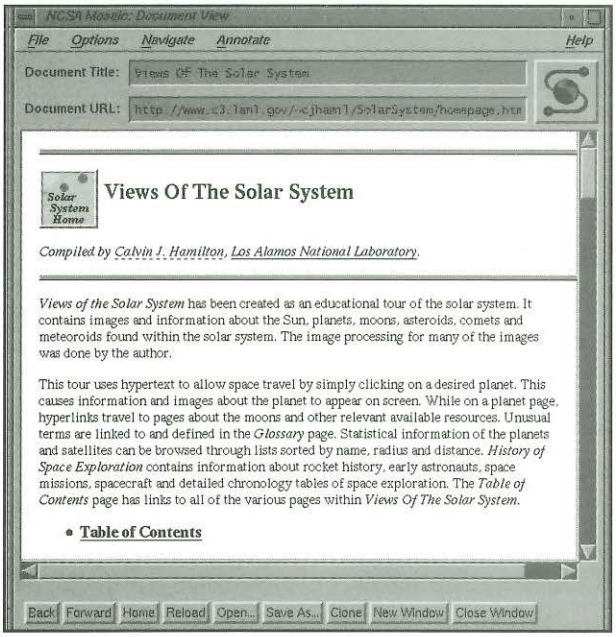


Figure 9.10 Top of the home page for Views of the Solar System.

As time went on, I decided to add separate WWW pages for each planetary satellite for which I had an image. In my free time, I would process new images and add them to my pages. Often, I would look through the astronomy books I had checked out from the library and, when I found an image I liked, I would locate the raw Voyager/Viking data and process it to give computer images of quality comparable to those found in the book (Figure 9.11). I subsequently added sections on asteroids, comets, and the history of space exploration.

At the present time, Views of the Solar System has the greatest variety of planetary images available on the Internet. It is being used by several schools in New Mexico and elsewhere. There is a German publisher writing a book on the Internet who is publishing a CD-ROM with examples of WWW pages. This project will include the entire tour on the CD-ROM. In addition, the University of Southwestern Louisiana is using information from the tour for a NASA contract to develop multimedia courseware for high-school students.

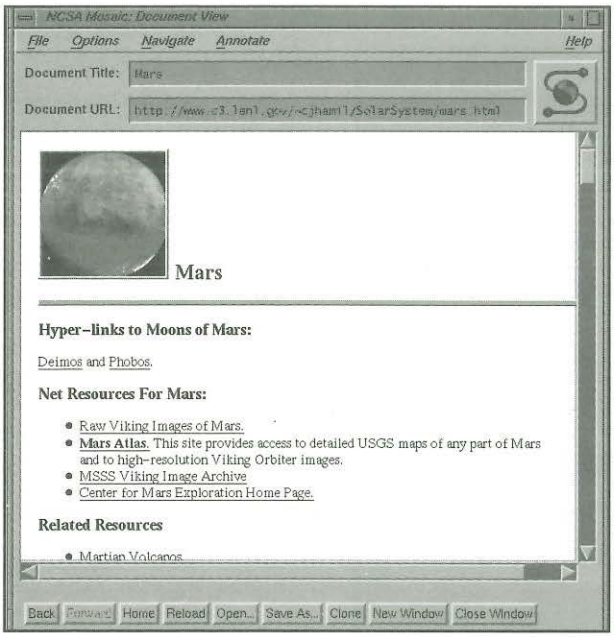


Figure 9.11 Top of the home page for the planet Mars. The document contains several icons of Mars images, along with details about the martian surface, atmosphere, and so on.

HTML ISSUES

When I started writing HTML pages, my biggest obstacle was finding proper documentation. I eventually found that the best way to learn the HTML syntax was to look at examples on the Internet. One problem I found was that my **Mosaic** browser overlooked certain errors that other browsers did not. I found out about these problems only when someone on the Internet was kind enough to point them out.

Since Views of the Solar System uses a lot of inline images, I found that certain pages used up all of the available colors on my 8-bit color monitor and that, as a result, some of the images didn't look good at all. I resolved this problem by using image color compression software developed at Los Alamos National Laboratories. Whenever I found an HTML page that used too many colors, I compressed the colors in the images so that they used far fewer colors, typically 128. This greatly improved the appearance of the pages when displaying on 8-bit color monitors.

Another important issue was the size of the inline images. On my home page, I used 68 × 68-pixel images, and, on my other pages, I chose a maximum width and height of 150 pixels. This tends to be a bit larger than many sites on the Internet. The advantage of the larger inline image is that the details are much clearer. The disadvantage is that it takes longer to transfer these large images across the Internet. A 150 × 150-pixel GIF image is approximately 12,000 bytes, whereas a 68 × 68-pixel image is only about 2,700 bytes.

I also had to decide what types of image files and formats to use when transferring noninlined images across the Internet. Should a small representative image be returned, or the best image available? I chose the latter option. In certain cases, I have very large mosaics, in which case, I provide both a small and a large version of the same image (Figure 9.12). I also avoid lossy compression methods, such as JPEG, unless I receive the raw image in that format. The JPEG lossy image-compression technique tends to leave the image with unsightly blocks scattered throughout the image. Once lossy compression is used, further image processing or enhancement is useless. I

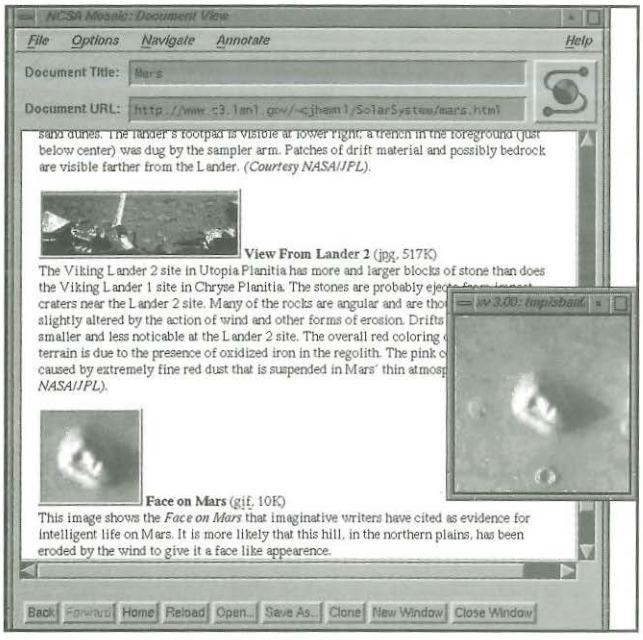


Figure 9.12 Detail from the Mars page. Clicking on the Face of Mars downloads the larger GIF image of the same feature.

save all of my images in GIF format if I need 8 bits of color, or in TIFF format if I need 24 bits.

Until recently, I used the HTML <HR> tag for separator bars dividing document sections. I now use a three-dimensional-looking bar (a GIF image) as a separator. This bar uses three colors and takes up 579 bytes. I tried separators with a rainbow appearance but they required too many colors: I needed these colors for the actual images instead.

I organized Views of the Solar System somewhat like a book. The home page is like an introduction, with links to the major portions of the tour. The home page also contains links to educational and other space-related resources. Organizationally, the home page links to a cover page; a Table of Contents; a solar system introduction; each planet in our solar system; asteroids; comets; meteoroids; and the history of space exploration. The planet pages have further links to each of their satellites and to other related

resources. Many of the terms used in Views of the Solar System are linked to an extensive glossary page. There is also a section on people who have made a contribution to the understanding of the world and universe we live in.

SUMMARY

I am not an astronomy professional but, during the last year, I have read many books and have learned much about our solar system. Perhaps this demonstrates that you don't have to be an expert in a particular field to make a significant contribution. If time and money permit, I would enjoy making continual improvements to this tour. Views of the Solar System has been an evolving and somewhat experimental project that I hope will continue to be educational and beneficial for others.

NETBOY—CHOICE OF AN ONLINE GENERATION

http://www.interaccess.com/netboy.html

Stafford Huyler
E-mail: ***netboy@interaccess.com***

I had been looking for a way to publish a comic years before I came into contact with the Web. I spent all my time in school drawing cars and spaceships, while my teachers sneered that my work wasn't *real art*. I knew that the standard was to get a little strip going in a free edition of some college rag, and nurture it for years while praying that it would be noticed. This wasn't the way for me; it was just not dynamic enough.

In my job as the marketing director for InterAccess, a Chicago-based Internet provider, I thought a lot about the possibilities of self-publishing on the Net. There was a tremendously intelligent audience out there, hungry for good entertainment. I knew that I should approach them. At the time, however, there was no way to distribute a comic effectively.

When I discovered the Web, I realized that it was a very promising forum. I felt that it was the up-and-coming interface with the Net—yet, there was so little of interest on it! I wanted to read a comic that was cool enough to joke about the standard knowledge that all onliners have. It didn't exist, so I decided to create it. NetBoy is the first Internet-topical comic.

Between my own experiences online and behind the scenes at InterAccess, I observed a lot of funny stuff. I also had the resources to produce a comic—a scanner, technical help, and a company willing to give my comic a chance. If I were producing this alone, it could get quite expensive because of the amount of traffic moving through the site. However, I have worked out a reciprocal arrangement with InterAccess. They feel that the publicity that they get from NetBoy's success is a good reward for assisting me in getting the comic out.

I played with different styles and felt that a black-and-white stick figure column would serve several purposes. It would play with the fact that the delivery medium was colorful. It would be easy to download. It would be simple for me to draw, so I could keep my day job. It seemed pleasingly goofy to me that my technological heroes were mere stick figures. Finally, the medium meant that I could choose any size and format.

I draw the comic by hand, as it gives the best control. Once it is scanned in, all the touch-ups and lettering are done in Photoshop with a custom font. For file formats, I chose GIF over JPEG because of the inline capability. Format is very important; you must plan your readers' traffic well in advance.

I created a beta in May 1994 to get a feel for my capabilities and audience. I experimented for a couple of months, and the current version of NetBoy debuted on July 1, 1994. The home page for NetBoy is shown in Figure 9.13.

The goal of NetBoy is really to create a hip place to be online. There is the comic, but there are also back issues, feedback, N-Mart, contests, and links to other comic pages (Figure 9.14). It's paying off; NetBoy has been growing with the Web. In just the first three months, readership went from 500 per month to over 90,000. Over a million people have read it now, and the audience is growing by 20 percent a month. This is thanks to enthusiastic word of mouth, comics being tacked up in computer labs, links from many pages, and media coverage.

Of course, I do all this for free, but I have been thinking of ways to make money from my project. For a time, I flirted with the idea of offering a paid

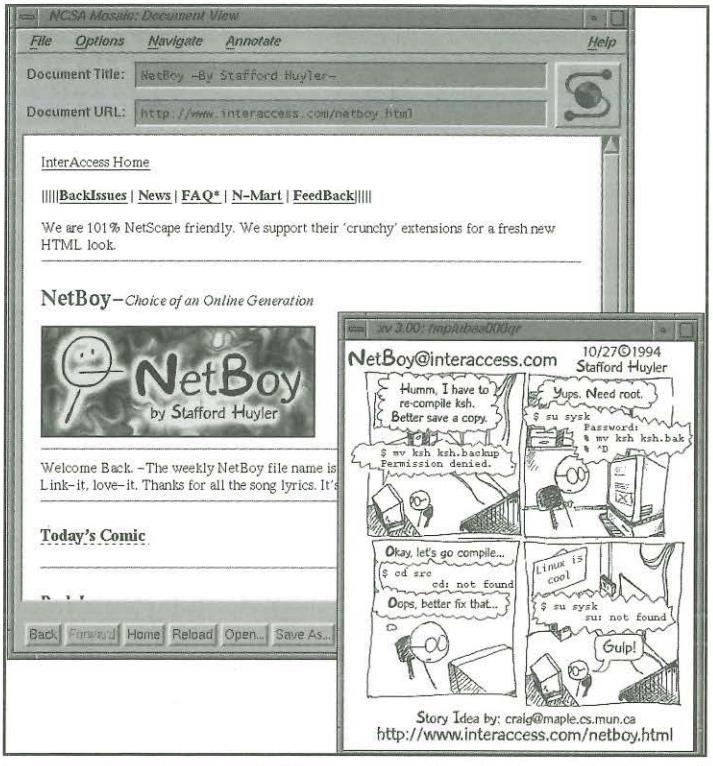


Figure 9.13 Home page for NetBoy. Clicking on Today's Comic downloaded the displayed GIF image; this image is new every day.

daily subscription to NetBoy via e-mail as a MIME-encoded document, but I realized that this move would greatly restrict my readership (even though I had planned to keep the large Saturday editions free). I am now taking the Microsoft approach—making NetBoy the standard against which all others are judged (and copied). I know that I can build up a loyal readership by giving them a strong product for free.

With this readership growing every day, I feel the future is in merchandising. Because so many people are reading NetBoy, there is a growing market for NetBoy books, mugs, and so on. I have already done a good business selling NetBoy t-shirts.

My advice to anyone else starting out would be to learn good HTML and provide a solid service. If you do a weekly or daily, keep it up religiously. If

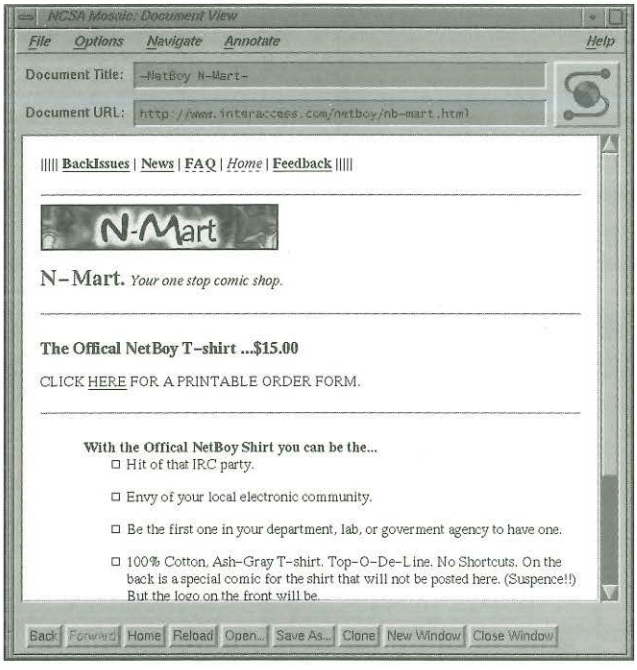


Figure 9.14 Want to buy a NetBoy t-shirt? This is the page to look at.

the competition is 100-percent reliable in their new pages, and you're not, then you may lose.

With strong perl programming skills, you can create forms to monitor reader feedback, which is most helpful. You can see what's working and what's not—everything from a certain vein of humor to the layout of the page.

Another pointer is that you should give your feature a consistent name each day it is released. For example, I call each new comic "today-netboy.gif." This way, readers can link the GIF image into their home page. People want their reading delivered with a minimum of fuss.

The Web is the most effective publishing medium that I have ever encountered. I am very pleased with it, and if I had to do it again, I would only have started sooner.

SAN FRANCISCO RESERVATIONS' WORLD WIDE WEB PAGE

http://www.hotelres.com/

Eric J. Fraser
Programmer/Analyst
University of California
Computer Science Division
e-mail: *fraser@CS.Berkeley.EDU*

INTRODUCTION

San Francisco Reservations (SFR) is a central hotel reservations agency serving the San Francisco Bay area. SFR agents reserve hotel rooms at volume discount rates, and monitor seasonal specials and promotions, passing on the savings to the public. Before the WWW page was set up, business was handled by a toll-free telephone number and by fax. While much of the business infrastructure was done on computers (a UNIX network), there was no previous need for an Internet connection.

In June of 1994, at an informal dinner party, a discussion began on ways to improve the computing environment at SFR. On that day's front page of the Business section of *The New York Times*, there was an article of a florist that had set up a WWW page:

`http://florist.com:1080/flowers/bf1006.html`

We decided that the Net had definitely "gone commercial" and that if a florist shop could make it on the Internet, then surely SFR should have a place as well.

In the following month, the details were worked out on how much it would cost to get an Internet connection, on which local service provider to use, whether to get a dedicated SLIP-type connection or to get a dedicated digital line, and what additional hardware might be needed. After much boring deliberation, it was decided that SFR would get a dedicated frame-relay (56kbps) connection to a small co-op style Internet provider, and to use one of the already owned 486 machines running **linux**.

SFR's WWW server was up and operational in the beginning of September 1994. Even before we announced the page to the relevant newsgroups, users

were accessing the documents. As it went, first someone logged in from Stanford, then from Japan, and then from Australia, all within an hour. It was very exciting to see that quick a response. Since that day, the number of people accessing the page has grown quite rapidly. On average, there are around 150 users per day, more than we expected in our first month.

WHY ON THE WORLD WIDE WEB?

Designing a system using HTML/HTTP seemed like the natural choice for SFR. In fact, it was decided that it was the only way to go in connecting to the Internet. Some of the deciding factors were:

- HTML/HTTP was quickly becoming the standard of information servers
- Ease of programming
- Several easy-to-use browsers, including Mosaic, lynx, tkwww, and so on
- Nonintrusive nature
- Support for graphics, sounds, and MPEG movies
- CGI and FORMs capabilities

The foremost appeal of WWW is its widespread use. WWW is rapidly becoming the standard for information servers around the world, replacing the text-only Gopher servers. SFR's hope was to allow access to the greatest number of people. WWW seemed to be the most popular.

Another advantage is that the WWW offers a way that people can do business in a nonintrusive manner on the Internet. Unlike mailing lists and postings to USENET newsgroups—which take up disk space and network bandwidth on computers, where it is unwanted—the WWW gives users the ability to browse only those remote servers that they choose. SFR is generally perceived as a service-oriented business, since clients are not charged for making reservations. The notion of invasive business is not as relevant to SFR as it may be to someone selling life insurance, for instance, but it is still in any company's best interest to play by the rules of the Internet.

The most desirable quality about HTML and the World Wide Web, for someone creating pages, is the ability to copy something and use it as a model. Browsers such as **Mosaic** allow users to view the source code of the document they are presently looking at. By piecing together ideas from several

different pages, you can quickly get something presentable up and running. In the creation of SFR's pages, many ideas were found in other documents.

Mosaic has become the WWW viewer of choice. There are, however, several others that are available, giving users the ability to view the same information in several different ways. **Lynx** allows users to get a text-only view of WWW. This is quite useful to those who do not have a fast connection, or for those who do not have an IP address (or TIA).

In addition, the glory of sight and sound is available to those who have a fast connection. HTML offers an easy way to include graphics, sounds, and MPEG movies with the documents. As you will see in the following, the ability to display images was a key design element in SFR's WWW pages.

Finally, HTML includes CGI and HTML FORMs capability (although not all viewers support this). These give HTML the flexibility and power to handle almost any task. If a script can be written that sends output to the standard output, then it can be incorporated into an HTTP server and be accessed from your HTML documents. There are several cases where SFR's WWW page requires this sort of flexibility, including the ability to check availability of all hotels for any given day, the ability to search hotel lists for those that match specific criteria, and the ability to process reservation requests.

DESIGNING THE WWW PAGE

The overall design goal for San Francisco Reservations' World Wide Web page (Figure 9.15) was to create an easy-to-use system that would give users the ability to virtually browse through hotels before making a reservation: Trying to describe, in words, the way a hotel lobby looks or feels is difficult. Given this desire, there were other important considerations.

First of all, information should be clear and concise. The most important place for this is the home page. Here, it should be clear to the user what exists on the server. For the design of SFR's page, we asked ourselves the following two questions: "What do users know when they get here?" and "What do they want to find out now that they are here?" This helped shape the notion of having several search paths for finding hotels: by hotel location, price, type, and keyword (if the user has a specific hotel in mind). To

maintain conciseness, depth was always chosen before breadth whenever possible. By this, I mean it is better to have a single link to a page listing other WWW pages than to list all the pages on the front page (which I have seen done on several pages). Also, we included a Help button in as many places as possible, allowing us to keep many details off of the main pages (see, for example, Figure 9.16).

One of SFR's key design constraints was to give users the ability to get partial lists of hotels based on certain criteria. Originally, we had thought we would build static lists of hotels for these criteria: After all, hotels do not change their location, name, type, or prices very often. For the sake of longevity and robustness, however, we decided to build a single database of hotel data and write perl scripts to extract the desired information from this database.

Similarly, we needed to have a standardized reservation request form for users to complete, plus the ability to get up-to-the-hour hotel availability information. Perl scripts were written to handle these situations, although in an imperfect manner. Availability of San Francisco hotels can change suddenly, especially in high tourist season. We had hoped that when users completed their search of hotels, a list would be provided that matched their criteria, along with the availability information for each hotel. Users would then be restricted from submitting a reservation request for any hotel that was not available. We determined, empirically, that this method could be ineffective, since hotels could quickly become *unavailable* during the time the user was browsing the *available* hotels list. We maneuvered around this problem by checking availability only at the time a user is ready to make a reservation. However, it is rather inconvenient to find a few hotels that you like only to discover, one by one, that they are sold out. To handle this, we allow the user to view all hotel availability for an entire month when a hotel is deemed sold out (we also have a link to this information on the home page, for those who know exactly what days they are to be in San Francisco).

Inline images in HTML pages should be small, should have a minimal number of colors, and should be used sparingly. While it was important for SFR to have a flashy WWW server, it was realized early on that there is a beauty versus usability trade-off, and that usability is far more important. Sending

large pictures over busy networks will test the patience of any user. After all, there are thousands of other pages they could be looking at instead of waiting for your GIF image to arrive. In SFR's case, the details about each hotel include pictures of sample hotel rooms, exterior photographs, hotel lobbies, and local area maps that the user might wish to view. We made these available without sacrificing usability by making a small-size copy of the image (an *icon*), and linking the small images to the full-size versions. By doing so, users can get an idea of what the picture is like from the small version, and if they are interested in seeing the details, they can click on the image (see, for example, Figures 9.17 and 9.18). Here is some example HTML code of this:

```
<A HREF="majes_1.gif">
<IMG ALIGN=middle SRC="majes_1_small.gif">
lobby photo</A>
```

If users like the look of a hotel they will most likely want to book a room or check room availability. This information can be accessed directly from the Hotel page (Figure 9.18) by pressing the *Submit Reservation Request* button. This returns the document shown in Figures 9.19 and 9.20, which allows you to book and confirm a reservation or check room availability.

A final constraint was to make our server both **Mosaic**- and **lynx**-friendly. Many WWW servers cater to only one or the other. **Mosaic** users love all kinds of graphical buttons, big pictures, and clickable imagemaps. **lynx** users, however, cannot see the pictures and are sometimes left helpless on what to do. An example of this is the use of graphical buttons. SFR's home page uses clickable buttons for the different search paths, with descriptions of the button on the button itself. **Mosaic** users will see the nice descriptive button (e.g., Hotel Location), while, if you are not careful, **lynx** users will see only the text string "[IMAGE]". To solve this problem, the ALT attribute should be used. Here is an example:

```
<A HREF="/srchloc.html">
<IMG SRC="srchloc.gif" ALT="[Hotel Location]">
</A>
```

With the ALT command, **lynx** users will see "[Hotel Location]" in place of the image, providing a useful and aesthetically pleasing page to view. In general, it is good practice to view each page in several formats before you are

done. You should also realize that there are many people using **Mosaic** who have the inline image loading turned off. Try this out. SFR is still working on how to make this look nice.

```
<title> San Francisco Reservations</title>

<img src="sfrheader.gif" ALT="SAN FRANCISCO RESERVATIONS">
<hr>

Welcome to <b>San Francisco Reservations</b>' (SFR) World-Wide Web
page! This service provides hotel reservations in the San Francisco
area at no cost to the user. SFR rates are, in most cases, the lowest
rates available. Feel free to browse around and send us
<A href="/feedback.html">
  <img align=top src="icons/email.gif" ALT="[e-mail]">
</a>
if you have any questions or concerns.

<h2>Search San Francisco Hotel List by:</h2>

<A href="/srchloc.html">
<img src="icons/srchloc.gif" ALT="[Hotel Location]"></a>
<A href="/srchprice.html">
<img src="icons/srchprice.gif" ALT="[Hotel Price]"></a>
<A href="/srchtype.html">
<img src="icons/srchtype.gif" ALT="[Hotel Type]"></a>
<A href="/srchkey.html">
<img src="icons/srchkey.gif" ALT="[Key Word]"></a>

<h2>Other Information</h2>

<A href="/sfrinfo.html">
<img align=middle src="icons/sfrinfo.gif" ALT="[Help]"></a>
For more information about San Francisco Reservations. <br>
<A href="/availability.html">
<img align=middle src="icons/availability.gif" ALT="[Availability]"></a>
Examine hotel availability.<br>
<A href="/otherwww.html">
<img align=middle src="icons/otherwww.gif" ALT="[Elsewhere]"></a>
Other points of interest on the World Wide Web.<br>

<hr>

<b>San Francisco Reservations</b><br>
22 Second Street, 4th Floor<br>
San Francisco, California 94105<br>
<img src="icons/phone_num.gif" ALT="(800) 677-1550 or (415) 227-1500">
<address>sfr@hotelres.com</address>
```

Figure 9.15 HTML document for the San Francisco Reservations' home page.

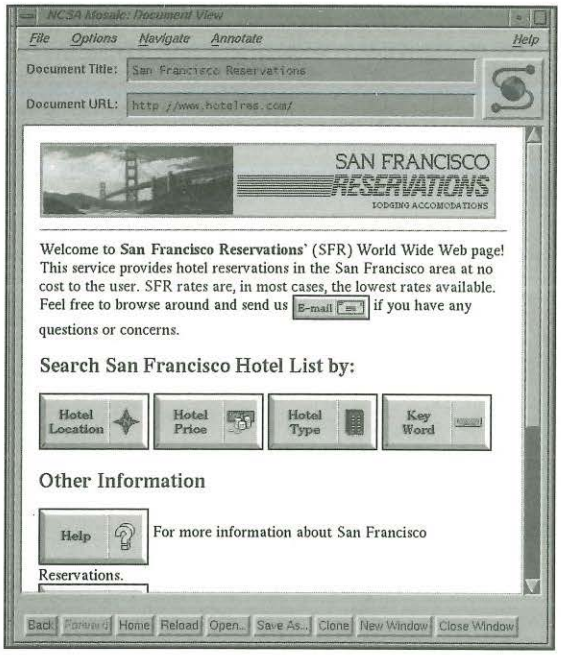


Figure 9.16 Home Page for San Francisco Reservations.

```
<title>Cartwright Hotel</title>

<h1>Cartwright Hotel</h1>
524 Sutter Street /Powell<br>
San Francisco, CA 94102
<br>
    <FORM METHOD="POST" ACTION="/cgi-bin/hoteldates.pl">
    <input type="hidden" name="H_code" value="`CARTWRIGHT HOTEL'">
    <input TYPE="submit" value="Submit Reservation Request"><p>

  <A href="cartw_l.gif">
  <img align=middle src="cartw_l_small.gif">
  lobby photo</a>

  <A href="cartw_r.gif">
  <img align=middle src="cartw_r_small.gif">
  sample room photo</a>

  <A href="cartw_m.gif">
  <img align=middle src="cartw_m_small.gif">
  local area map</a><br>

<hr>
<pre>      1994 Published Rates |  SFR Value Rates (based
                           |on availability)
```

```
=============================================================================
                              | (4/1-10/31/94)      (11/1-12/31/94)
 Dbl. 1-2 Pers.    $119/$129  | 1-2 Pers.     $79   1-2 Pers.        $69
 Queen 1-2 Pers.   $119/$129  | 1-2 Pers.     $79   1-2 Pers.        $69
 2 Twin 1-2 Pers. $119/$129   | 1-2 Pers.     $79   1-2 Pers.        $69
 Extra Pers.       $ 12/$ 12  |               $12                    $12

  Children 12 yrs. & under Free</pre>
<hr>

  <img align=middle src="../icons/question.gif">
  What are<A href="../sfrvaluerates.html"> SFR Value Rates</a>?<p>

<b>Check In</b>: [Open] <b>Check Out</b>: [1:00pm]<br>
<b>Accepted Credit Cards</b>: [AX, MC, Visa,
                Diners, Discover, JCB].<p>

<em>              Credit Card Required to Guarantee
                  Arrival. To Change or Cancel a
                  Reservation, Notify Us 48 hrs Prior to
                  Date of Arrival to Avoid a Charge.
</em>
<pre> [N] Airport Shuttle
 [Y] Parking [$12 self-parking nearby]
 [Y] Restaurant [Teddy's 7am-11am Bkfst]
 [N] Room Service
 [Y] Continental Bkfst [$5]
 [N] In-Room Movies
 [N] VCR
 [N] View
 [N] Fireplace
 [N] Air Conditioning
 [Y] Roll Away/Crib [$10/Free]
 [Y] Suites [1 Bedroom]
 [N] Kitchen
 [N] Handicap
 [N] Business Center
 [N] Pool
 [N] Fitness Center
 [N] Pets</pre>

<b>Neighborhood</b>: Comfortable downtown location. 1 1/2 blocks from Union
Square, 1/2 block to Cable Car.<br>

<b>Lobby</b>: Light, bright lobby with overstuffed sofas and chairs.
Reading room/library adjacent. Traditional style decor. <br>

<b>Guest Rooms</b>: 114 small homey rooms and suites, fresh flowers, reading
pillows, clock radios.<br>

<b>Suites</b>: 1 bedrooms done with antiques.<br>

<b>Services</b>: Personalized service, meeting facilities available.
Turndown service.<br>
```

Figure 9.17 HTML document for the Cartwright Hotel page.

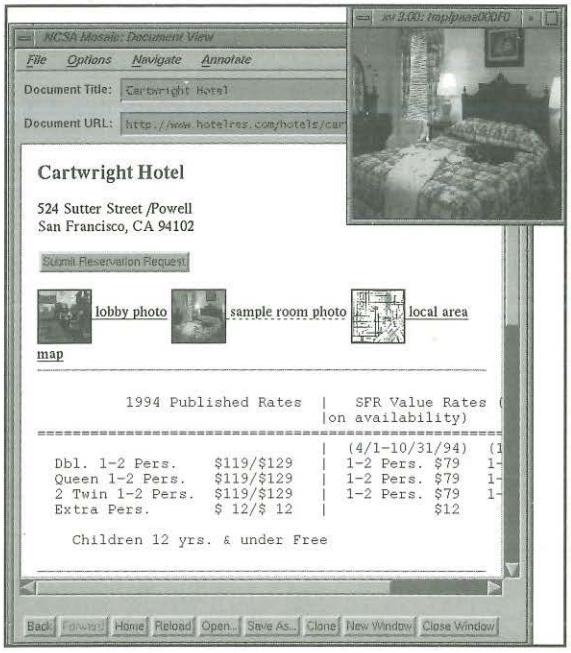


Figure 9.18 Typical page describing a San Francisco hotel. Clicking on the image icons yields larger, more detailed images of the hotel or of the rooms, as shown in the inset. Note the use of the FORMs "hidden" INPUT element.

FINAL NOTES ON SFR

The creation of SFR's WWW page was completed in about a month. It was decided that it should be *finished* before it was officially announced so that it could be thoroughly tested. We have found, however, that it is a continually changing project. By allowing users to send us feedback, we have received several suggestions that have shaped the way the pages look. The helpful and supportive atmosphere on the Internet is a vital resource that is invaluable.

```
<html><head>
<title>Travel Dates</title>
</head>
<body>
<h1><img src="../icons/box.gif" ALT="">Travel Dates
<img src="../icons/box.gif" ALT=""></h1>
<b>Hotel:</b> 'CARTWRIGHT HOTEL'
<p>In order for us to determine the availability of this hotel, please
fill out the following information. If you are making a reservation for
```

```
tonight, please call us at (800) 677-1550 or (415) 227-1500 so that we
can be sure that your request is processed in time. Our hours are 7am
to 11pm Pacific Standard Time, seven days a week.
<FORM METHOD="POST" ACTION="/cgi-bin/grind.pl">
<hr>Enter Dates in the form: MM/DD/YY
<p>Arrival Date:   <input NAME="Arrdate" SIZE="8">
<p>Number of Nights: <input NAME="Night_stay" SIZE="2">
<p><hr>
<input TYPE="submit" value="check availability">
<input TYPE="reset" value="clear">
<p><pre>
</pre>
<input type="hidden" name="H_code" value="'CARTWRIGHT HOTEL'">
</FORM>
<p> Generated by: <var>/home/http/cgi-bin/hoteldates.pl</var><br>
Date: 8:51:49 PST on Wed 19 Oct 94.<p>
</body>
</html>
```

Figure 9.19 Reservations and availability HTML document. This page was generated by the server-side perl script ***hoteldates.pl***. Note the use of the FORMs `<input type="hidden"...>` element to store the name of the hotel. This is how the server-side script knows what hotel is being considered.

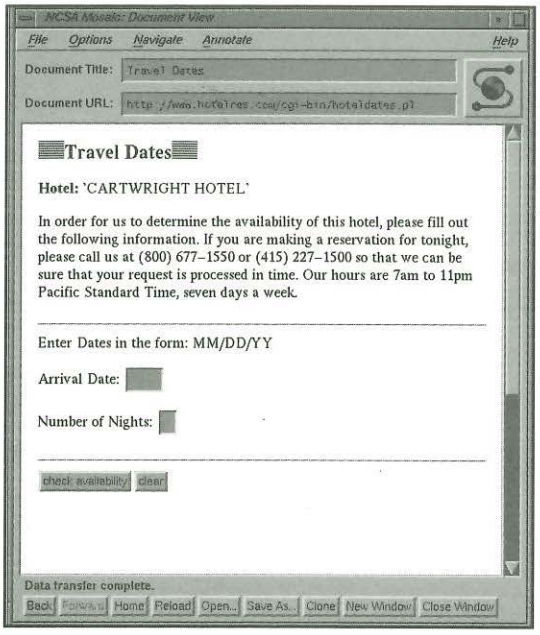


Figure 9.20 Reservations and availability query page for the Cartwright Hotel. This is accessed by clicking on the Submit Reservation Request page shown in Figure 9.18.

APPENDIX

A

THE ISO LATIN-1 CHARACTER SET

A computer character set is simply an agreed-upon relationship between computer binary codes and a set of letters or graphical characters. Since almost all computers use 8-bit bytes as the basic storage unit, standard character sets are designed to use a single 8-bit byte to store a single character. Since there are 8 bits in a byte, each byte can represent 256 possible characters ($256=2^8$).

Character sets for international use are specified by the International Standards Organization (ISO). The ISO has specified several different character sets that can fit inside 8-bit characters. The default set for World Wide Web applications is the ISO Latin-1 character set, shown in Table A.1. The first 128 characters of the ISO Latin-1 character set are equivalent to the 128 characters of the US-ASCII (also known as ISO 646) character set. US-ASCII is known as a 7-bit character set, since it consists of only 128 characters and can be represented using just 7 bits (2^7). Of these 128 characters, 33 are special characters for controlling printing devices and communications lines. These characters are not printable and are indicated in the table by two- or three-letter character sequences that are mnemonically designated to represent their function. For example, NUL is a null character, BEL is the bell character (rings a bell), CR is carriage return, BS is the backspace character, and so on. In addition, the table marks the space character (decimal 32) with the symbol SP, as, otherwise, this would be invisible.

There are 128 additional characters in the ISO Latin-1 character set. The first 32 are nonprintable control characters: These are marked in Table A.1 by a double dash (—). The remaining characters are printable and consist of many of the accented and other special characters commonly used in western European languages.

URL CHARACTER ENCODINGS

As discussed in Chapter 2, any ISO Latin-1 character can be represented in a URL by indirect references or encodings. In a URL, any ISO Latin-1 character can be represented by the special character sequence:

```
%xx
```

Where xx is the *hexadecimal* or *hex* code corresponding to the character. Table A.1 shows both the regular decimal and hexadecimal codes for all the ISO Latin-1 characters. As an example, the URL *encoding* for the percent character itself would be:

```
%25
```

since the percent character is the thirty-seventh (hexadecimal 25) character in the character set.

HTML CHARACTER AND ENTITY REFERENCES

In HTML, any ISO Latin-1 character can be represented by either a *character reference* or an *entity reference*. A character reference represents each character through its decimal code. Thus, the character reference for a capital U with an umlaut (Ü) is:

```
&#220;
```

since this is the 220th character in the ISO Latin-1 character set. Numerical references are awkward and difficult to remember, so HTML also allows entity references for some of these characters. For example, the entity reference for a capital U with an umlaut (Ü) is

```
&Uuml;
```

In HTML, the four ASCII characters (>), (<), ("), and (&) are interpreted to have special meanings. Therefore, to display them in text as ordinary characters, you must use character or entity references. The special entity names for these characters are given in Table A.1. For example, the entity reference for the less-than sign is `<`. Figures A.1 and A.2 show the use of entity references in an HTML document.

Not all computers use the ISO Latin-1 character set. UNIX computers and PCs running Windows use the ISO Latin-1 character set by default, while Macintoshes and PCs running DOS do not, although the first 128 characters of the Macintosh and DOS character sets are the same as the US-ASCII characters, as with ISO Latin-1. You therefore must take care on these platforms when preparing documents for use on the World Wide Web—it is always safe to use the character or entity references mechanisms for non-ASCII characters.

Table A.1 ISO Latin-1 Characters, Showing Decimal Codes, Hexadecimal Codes, and HTML Entity References

Character	Decimal	Hex	Entity Reference	Character	Decimal	Hex	Entity Reference
NUL	0	0		SOH	1	1	
STX	2	2		ETX	3	3	
EOT	4	4		ENQ	5	5	
ACK	6	6		BEL	7	7	
BS	8	8		HT	9	9	
NL	10	a		VT	11	b	
NP	12	c		CR	13	d	
SO	14	e		SI	15	f	
DLE	16	10		DC1	17	11	
DC2	18	12		DC3	19	13	
DC4	20	14		NAK	21	15	
SYN	22	16		ETB	23	17	
CAN	24	18		EM	25	19	
SUB	26	1a		ESC	27	1b	
FS	28	1c		GS	29	1d	
RS	30	1e		US	31	1f	
SP	32	20		!	33	21	
"	34	22	"	#	35	23	
$	36	24		%	37	25	
&	38	26	&	'	39	27	
(	40	28		)	41	29	
*	42	2a		+	43	2b	
,	44	2c		-	45	2d	
.	46	2e		/	47	2f	
0	48	30		1	49	31	
2	50	32		3	51	33	
4	52	34		5	53	35	
6	54	36		7	55	37	
8	56	38		9	57	39	
:	58	3a		;	59	3b	
<	60	3c	>	=	61	3d	
>	62	3e	<	?	63	3f	
@	64	40		A	65	41	
B	66	42		C	67	43	
D	68	44		E	69	45	
F	70	46		G	71	47	
H	72	48		I	73	49	

Character	Decimal	Hex	Entity Reference	Character	Decimal	Hex	Entity Reference
J	74	4a		K	75	4b	
L	76	4c		M	77	4d	
N	78	4e		O	79	4f	
P	80	50		Q	81	51	
R	82	52		S	83	53	
T	84	54		U	85	55	
V	86	56		W	87	57	
X	88	58		Y	89	59	
Z	90	5a		[	91	5b	
\	92	5c		]	93	5d	
^	94	5e		_	95	5f	
`	96	60		a	97	61	
b	98	62		c	99	63	
d	100	64		e	101	65	
f	102	66		g	103	67	
h	104	68		i	105	69	
j	106	6a		k	107	6b	
l	108	6c		m	109	6d	
n	110	6e		o	111	6f	
p	112	70		q	113	71	
r	114	72		s	115	73	
t	116	74		u	117	75	
v	118	76		w	119	77	
x	120	78		y	121	79	
z	122	7a		{	123	7b	
|	124	7c		}	125	7d	
~	126	7e		DEL	127	7f	
—	128	80		—	129	81	
—	130	82		—	131	83	
—	132	84		—	133	85	
—	134	86		—	135	87	
—	136	88		—	137	89	
—	138	8a		—	139	8b	
—	140	8c		—	141	8d	
—	142	8e		—	143	8f	
—	144	90		—	145	91	
—	146	92		—	147	93	
—	148	94		—	149	95	
—	150	96		—	151	97	
—	152	98		—	153	99	
—	154	9a		—	155	9b	
—	156	9c		—	157	9d	
—	158	9e		—	159	9f	
	160	a0		¡	161	a1	
¢	162	a2		£	163	a3	
¤	164	a4		¥	165	a5	
|	166	a6		§	167	a7	
¨	168	a8		©	169	a9	
ª	170	aa		«	171	ab	
¬	172	ac		-	173	ad	
®	174	ae		–	175	af	

Character	Decimal	Hex	Entity Reference	Character	Decimal	Hex	Entity Reference
°	176	b0		±	177	b1	
²	178	b2		³	179	b3	
´	180	b4		µ	181	b5	
¶	182	b6		·	183	b7	
¸	184	b8		¹	185	b9	
º	186	ba		»	187	bb	
¼	188	bc		½	189	bd	
¾	190	be		¿	191	bf	
À	192	c0	À	Á	193	c1	Á
Â	194	c2	Â	Ã	195	c3	Ã
Ä	196	c4	Ä	Å	197	c5	Å
Æ	198	c6	Æ	Ç	199	c7	Ç
È	200	c8	È	É	201	c9	É
Ê	202	ca	Ê	Ë	203	cb	Ë
Ì	204	cc	Ì	Í	205	cd	Í
Î	206	ce	Î	Ï	207	cf	Ï
Ð	208	d0		Ñ	209	d1	Ñ
Ò	210	d2	Ò	Ó	211	d3	Ó
Ô	212	d4	Ô	Õ	213	d5	Õ
Ö	214	d6	Ö	×	215	d7	
Ø	216	d8	Ø	Ù	217	d9	Ù
Ú	218	da	Ú	Û	219	db	Û
Ü	220	dc	Ü	Y	221	dd	Ý
Þ	222	de	Þ	ß	223	df	ß
à	224	e0	à	á	225	e1	á
â	226	e2	â	ã	227	e3	ã
ä	228	e4	ä	å	229	e5	å
æ	230	e6	æ	ç	231	e7	ç
è	232	e8	è	é	233	e9	é
ê	234	ea	ê	ë	235	eb	ë
ì	236	ec	ì	í	237	ed	í
î	238	ee	î	ï	239	ef	ï
∂	240	f0	ð	ñ	241	f1	ñ
ò	242	f2	ò	ó	243	f3	ó
ô	244	f4	ô	õ	245	f5	õ
ö	246	f6	ö	÷	247	f7	
ø	248	f8	ø	ù	249	f9	ù
ú	250	fa	ú	û	251	fb	û
ü	252	fc	ü	y	253	fd	ý
þ	254	fe	þ	ÿ	255	ff	ÿ

```
<html>
<head>
<title> Entity References </title>
</head>
<body>
<h1> Example 12 - Entity References </h1>
<PRE>
<B>"Special" Character References</B>

&amp;lt;      -  &lt;      &amp;gt;      -    &gt;
```

```
&amp;amp;      -  &amp;       &amp;quot;     -    &quot;

<B>ISO Latin-1 Entity References </B>

&amp;AElig;    -  &AElig;     &amp;Aacute;   -  &Aacute;     &amp;Acirc;    -  &Acirc;
&amp;Agrave;   -  &Agrave;    &amp;Aring;    -  &Aring;      &amp;Atilde;   -  &Atilde;
&amp;Auml;     -  &Auml;      &amp;Ccedil;   -  &Ccedil;     &amp;ETH;      -  &ETH;
&amp;Eacute;   -  &Eacute;    &amp;Ecirc;    -  &Ecirc;      &amp;Egrave;   -  &Egrave;
&amp;Euml;     -  &Euml;      &amp;Iacute;   -  &Iacute;     &amp;Icirc;    -  &Icirc;
&amp;Igrave;   -  &Igrave;    &amp;Iuml;     -  &Iuml;       &amp;Ntilde;   -  &Ntilde;
&amp;Oacute;   -  &Oacute;    &amp;Ocirc;    -  &Ocirc;      &amp;Ograve;   -  &Ograve;
&amp;Oslash;   -  &Oslash;    &amp;Otilde;   -  &Otilde;     &amp;Ouml;     -  &Ouml;
&amp;THORN;    -  &THORN;     &amp;Uacute;   -  &Uacute;     &amp;Ucirc;    -  &Ucirc;
&amp;Ugrave;   -  &Ugrave;    &amp;Uuml;     -  &Uuml;       &amp;Yacute;   -  &Yacute;
&amp;aacute;   -  &aacute;    &amp;acirc;    -  &acirc;      &amp;aelig;    -  &aelig;
&amp;agrave;   -  &agrave;    &amp;aring;    -  &aring;      &amp;atilde;   -  &atilde;
&amp;auml;     -  &auml;      &amp;ccedil;   -  &ccedil;     &amp;eacute;   -  &eacute;
&amp;ecirc;    -  &ecirc;     &amp;egrave;   -  &egrave;     &amp;eth;      -  &eth;
&amp;euml;     -  &euml;      &amp;iacute;   -  &iacute;     &amp;icirc;    -  &icirc;
&amp;igrave;   -  &igrave;    &amp;iuml;     -  &iuml;       &amp;ntilde;   -  &ntilde;
&amp;oacute;   -  &oacute;    &amp;ocirc;    -  &ocirc;      &amp;ograve;   -  &ograve;
&amp;oslash;   -  &oslash;    &amp;otilde;   -  &otilde;     &amp;ouml;     -  &ouml;
&amp;szlig;    -  &szlig;     &amp;thorn;    -  &thorn;      &amp;uacute;   -  &uacute;
&amp;ucirc;    -  &ucirc;     &amp;ugrave;   -  &ugrave;     &amp;uuml;     -  &uuml;
&amp;yacute;   -  &yacute;    &amp;yuml;     -  &yuml;

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

Figure A.1 An HTML document showing the use of HTML entity references.

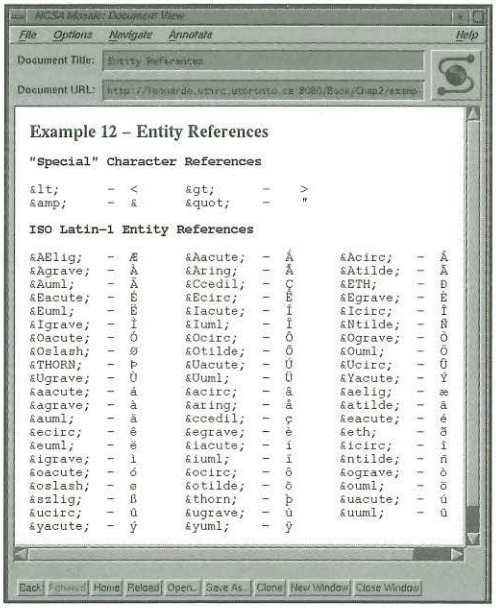


Figure A.2 **Mosaic for X-Windows** rendering of the HTML document listed in Figure A.1.

APPENDIX

B

MULTIPURPOSE INTERNET MAIL EXTENSIONS (MIME)

MIME, for Multipurpose Internet Mail Extensions, is an extension to the traditional Internet mail protocol to allow for multimedia electronic mail. The original Internet mail protocol, defined in the document RFC822, was designed with simple text messages in mind—it defined a number of message headers defining routing information about the message, but said little about the message content. At the time (which was not that long ago!), most electronic mail messages were plain text files, so concerns about other formats were unwarranted.

Today, however, there is enormous demand for more sophisticated messages. For example, today's electronic mail often contains many parts, such as a Rich Text or HTML document, encoded image files, and sometimes sound files. Such messages can be easily communicated only if all mail programs share a standard for constructing and transporting such multipart multimedia messages.

The MIME protocol is designed to allow this type of mail message by providing an open protocol for multimedia/multipart mail messages. MIME defines several new mail document headers that specify if a message consists of multiple parts and how those parts are separated, and includes additional headers within each subpart of the message to specify what each part contains. Thus, a message might look like:

```
MIME-Version: 1.0 Content-type: multipart/mixed; boundary=23
--23 Content-type: text/html
.... html document ....
--23
Content-type: audio/aiff
..... audio file ....
--23--
```

This simple example leaves a great deal out, but gives the general idea of the approach. The message contains two parts, a text file in HTML format and an audio file in AIFF format, with the MIME *multipart* message surrounding it, indicating that there is more than one component to the message and specifying the string used to divide the message parts.

THE MIME CONTENT-TYPE

The preceding was, of course, an overly simplified description of the MIME protocol; for further details, you are referred to the relevant documentation (RFC 1521). Of primary interest to us is the MIME content-type header. This should be already familiar to you, since this is the header used to indicate the contents of files being transferred using the HTTP protocol. Whenever you request a document from an HTTP server, the server must first determine the type of the document and send the appropriate content-type header ahead of it. Similarly, when a client uses the POST method to send data to the server, the data is preceded by a content-type header that tells the server the format of the data arriving, namely:

```
Content-type: application/x-www-form-urlencoded
```

(see Chapter 4); and when a CGI program generates data to be sent to the client, it must send the appropriate content-type header ahead of the data it returns to the client.

How do content-type headers work? Each header has a minimum of two parts giving the data *type* and *subtype*, using the format:

```
Content-type: type/subtype
```

Type can be any of `image`, `audio`, `text`, `video`, `application`, `multipart`, `message`, and `extension-token` (these names, like the string `"content-type"`, are case-insensitive). The meanings of the first four are obvious and indicate the overall type of the data. The `application` type is for other data

(perhaps binary) that needs to be processed in a special way. This could be a program to run, or maybe a PostScript document to be executed by a PostScript previewer. `Multipart` indicates a message containing more than one part, while `message` refers to an old-fashioned RFC822 message body. `Extension-token` is any name beginning with `X-`, and refers to experimental data types.

Subtype gives the specifics of the contents. Thus, `text/html` means a text file that is an HTML document, `application/postscript` means a PostScript file to be run through a PostScript interpreter, and so on. There are lots of content-types: Table B.1 lists the ones most important in WWW applications. Subtypes can also be experimental extension types, such as the `x-www-form-urlencoded` subtype shown previously.

Because of the flexibility of the HTTP client-server interaction, there are many content-types used by WWW applications that are not commonly used in electronic mail messages. Therefore, you should not see the list in Table B.1 as appropriate for electronic mail messages.

HOW DOES THE SERVER DETERMINE CONTENT-TYPE?

For the server to be able to send a content-type header, it must somehow know what a document contains. The convention is to use the filename extension or suffix to indicate the content type. Thus, files with the extension ***.mpeg*** are assumed to be MPEG movies, files with the ***.html*** extension are assumed to be HTML documents, and so on. With PC servers, these names have to be shortened to three letters; for example, ***.mpg*** for MPEG movies or ***.htm*** for HTML documents. You can specify more than one extension for each type if that is more convenient. When you place a document on a server, you must be sure to give it the filename extension matching the contents of your file. At the same time, you must be sure to update your server's extension-to-MIME-type database if you add a previously unknown type.

If the server does not know the type of a file, it assumes a default content-type—often, `text/plain`.

HOW DOES THE CLIENT DETERMINE CONTENT-TYPE?

If a client receives a file from a Gopher or HTTP server, these servers themselves indicated the contents of the data: HTTP servers send a content-type header, while Gopher servers have another mechanism for indicating the data contents. With FTP or local file access, this support is not available, and the client must itself be able to determine the file content. Again, this is done by the filename extensions so that the client must *also* have a database matching filename extensions to data types. The location of this database varies from client to client, but, in all cases, it involves matching a filename extension to a file type/subtype. For example, with **Mosaic for X-Windows**, the database is found in a file called *mime.types*; with **WinMosaic**, the database is part of the *MOSAIC.INI* file; and with **MacWeb**, the database is part of the program resource fork and can be edited from the browser using a pull-down menu.

Table B.1 MIME-Types Commonly Used in the World Wide Web, Showing Corresponding Filename Extensions and Data Types

MIME-Type/Subtype	Typical Filename Extensions	Description of Data Contents
message/rfc822	mime	MIME message
application/postscript	ai eps ps	PostScript
application/rtf	rtf	MS Rich Text Format
application/x-tex	tex	Tex/LateX
application/x-texinfo	texinfo texi	TexInfo format
application/x-troff	t tr roff	troff document
application/x-troff-man	man	troff with MAN macros
application/x-troff-me	me	troff with ME macros
application/x-troff-ms	ms	troff with MS macros
application/x-gtar	gtar	gnu tar format
application/x-tar	tar	4.3BSD tar format
application/x-ustar	ustar	POSIX tar format
application/x-bcpio	bcpio	Old CPIO format
application/x-cpio	cpio	POSIX CPIO format
application/x-shar	shar	UNIX sh shell archive
application/x-pdf	pdf	Adobe Acrobat pdf format
application/zip	zip	Pkzipped files
application/mac-binhex40	hqx	Macintosh Binhexed archives
appication/x-stuffit	sit sea	Macintosh Stuffit Archive
application/macwriteii	??	MacWrite file
application/msword	??	MS word document
application/octet-stream	tar dump readme	tar, binary dump, trick extension to force a save to disk
application/octet-stream	bin uu	Binary, UUencoded
application/octet-stream	exe	PC executable

MIME-Type/Subtype	Typical Filename Extensions	Description of Data Contents
application/x-dvi	dvi	TeX dvi format file
application/x-wais-source	src wsrc	WAIS "sources"
application/hdf	hdf	NCSA HDF data format
audio/basic	au snd	"Basic" SUN audio 8-bit u-law PCM encoding
audio/x-aiff	aif aiff aifc	Macintosh audio format
audio/x-wav	wav	Microsoft audio format
image/gif	gif	GIF format
image/xbm	xbm	X-Bitmaps
image/xpm	xpm	X-Pixelmaps
image/ief	ief	Image Exchange Format
image/jpeg	jpeg jpg jpe	JPEG images
image/tiff	tiff tif	TIFF images
image/rgb	rgb	RGB images
image/x-xwindowdump	xwd	X-Windowdump image
image/x-pict	pict	Macintosh PICT images
text/html	html htm	HTML document
text/plain	txt	Plain text
text/plain	c c++ pl cc h	Program listings
text/x-setext	etx	Structure enchanced text
text/richtext		
video/mpeg	mpeg mpg mpe	MPEG Movie
video/quicktime	qt mov	Macintosh QuickTime
video/x-msvideo	avi	Microsoft Video Format
video/x-sgi-movie	movie	SGI Movie Format
Special HTTP/WWW types		
application/x-www-form-urlencoded		See Chapter 4
application/x-www-pgp-request		See Chapter 4
application/x-www-pgp-reply		See Chapter 4
application/x-www-local-exec		See Chapter 5

REFERENCES

The Internet Mail Protocol is defined in RFC822, available at:

```
ftp://ds.internic.net/rfc/rfc822.txt
```

(this is an archive site for all Internet RFC documents).

MIME is defined in RFC1521, available at:

```
http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/rfc1521.txt
http://www.oac.uci.edu/indiv/ehood/MIME/MIME.html
ftp://thumper.bellcore.com/pub/nsb
```

Introductory documentation, in PostScript, is also available:

```
ftp://ftp.uu.net/networking/mail/mime/mime.ps
```

while a definitive list of registered MIME-types is found at:

```
ftp://gum.isi.edu/share/in-notes/media-types/
```

Note that this list of electronic mail MIME-types is much shorter than the list used by WWW applications.

APPENDIX

C

FINDING SOFTWARE USING ARCHIE

Most of the software mentioned in this book is freely available on the Internet, and we have included URLs pointing to typical sites archiving the executable versions or source codes for these programs. However, you may find that a program has moved and is no longer available at the indicated location. In this case, you should not panic, but should use **archie** to locate an alternative archive site.

STANDARD ARCHIE

Archie clients (the software you run on your own PC or workstation) are available for almost all computer platforms. DOS, UNIX NeXT, and VMS versions can be found at:

```
ftp://ftp.cs.widener.edu/pub/archie/
```

Versions compiled for particular operating systems are indicated by the filename; for example: *archie-dos.zip* for PCs running DOS, and *archie-vms.com* for VAX/VMS computers, while the source code itself is found in the file *archie-1.4.1.tar.Z*. The source code compiles simply and easily on most UNIX workstations. Other sites containing **archie** are:

```
ftp://ftp.unipg.it/pub/unix/infosys/archie/
ftp://unix.hensa.ac.uk/pub/uunet/networking/info-service/archie/clients/
ftp://ftp.mr.net/pub/Info/archie/clients/
```


395

This command-line **archie** client works identically on DOS, UNIX, or VAX/VMS computers: You control the program through a series of command-line options you type after the name of the program, and **archie** prints the results of the search to the screen. Most often, you will want to make a *case-insensitive substring* search for filenames that contain the substring of characters that you type in. In this case, you give **archie** a character substring that is contained in the name of the package or program you are looking for: Giving a case-insensitive substring and not the entire name makes allowances for changing version numbers, for possible changes in capitalization, and for site-to-site variations in the name. For example, if you are looking for the program WinWeb, you would type at the command line prompt:

```
archie -s WinWeb
```

The string `-s` indicates a case-insensitive substring search, so **archie** will return all file or directory names containing the pattern `WinWeb` anywhere in the name, and with any combination of upper- and lowercase letters. For example, here is just part of the output from the preceding search:

```
Host ftp.einet.net
        Location: /einet/pc/winweb
                FILE -rw-rw-rw-      424263  Aug 10 10:21  winweb.zip
```

This gives the Internet domain name where the package was located (***ftp.einet.net***), the directory on this server that contains the file (***/einet/pc/winweb***), and the name of the file found (***winweb.zip***). You can then use anonymous FTP to access this site, or just use your WWW browser, pointing it to the URL:

```
ftp://ftp.einet.net/einet/pc/winweb/winweb.zip
```

Your **archie** client gets the information about file locations from an archie *server*. Each **archie** client has a built-in server name that it uses by default. Sometimes, this server is busy and does not return a prompt response. You can select another server using the command-line option `-h`. For example:

```
archie -h archie.sura.net -s winweb
```

means to search for the substring winweb on the alternate server ***archie.sura.net***. Typing the command:

```
archie -L
```

gives a short list of **archie** servers. Here is a typical list:

```
archie.ans.net (USA [NY])
archie.rutgers.edu (USA [NJ])
archie.sura.net (USA [MD])
```

```
    archie.unl.edu (USA [NE])
    archie.mcgill.ca (Canada)
    archie.funet.fi (Finland/Mainland Europe)
    archie.au (Australia)
    archie.doc.ic.ac.uk (Great Britain/Ireland)
* archie.rutgers.edu is the default Archie server.
* For the most up-to-date list, write to an Archie server and give it the command
  'servers'.
```

Archie servers generate their own databases of files so that they do not all contain the same information. Therefore, if one machine does not give you a useful response, it is a good idea to try another.

ARCHIE FOR MACINTOSHES

Macintosh users have access to an elegant shareware program called **Anarchie**, which combines archie searches with an FTP retrieval package for retrieving files from amongst the returned list. To do an archie search with **Anarchie**, you select Archie... from the pull-down File... menu. You then select the archie server you want, type the search string into the Find: box, select the type of search to use, and press the Find button to start the search. **Anarchie** then displays a small window telling you it is working, and even gives an estimate of the time required to complete the search. You can even start up multiple parallel archie searches if you wish. When finished, **Anarchie** produces a window containing the list of successful results. Double-clicking on an item in this list starts up an FTP connection to the relevant site, and automatically downloads the file to your computer.

Anarchie is available at many sites. Some examples are:

```
ftp://ftp.cac.psu.edu/pub/mac/comm/anarchie/Anarchie-121.hqx
ftp://sumex-aim.stanford.edu/info-mac/comm/net/anarchie-110.hqx
ftp://sics.se/pub/info-mac/comm/net/anarchie-110.hqx
```

DOWNLOADING FILES USING FTP

If you have a WWW browser, you can just type in the URLs to access files: Browsers come with the FTP protocol built in. Thus, to access the file ***winweb.zip*** from the server ***ftp.einet.net*** in the directory ***/einet/pc/winweb***, you would type in the URL:

```
ftp://ftp.einet.net/einet/pc/winweb/winweb.zip
```

If you leave out the filename, that is:

```
ftp://ftp.einet.net/einet/pc/winweb/
```

you get a content listing for the directory */einet/pc/winweb/*. You can then click on the file you want to download.

Sometimes, the preceding procedure doesn't work, and you get a terse message stating that that the browser was "unable to connect to the URL." This often means that the server you are trying to connect to already has too many people making FTP connections, and that it has refused you access. To test this, you can make the FTP access the old-fashioned way by running the standard FTP program. Here is an example FTP session (the string `green:~>` is the command-line prompt). Recall that, after making the FTP connection, you log in as user ***anonymous*** and give your ***e-mail address*** as a password. At this point, the FTP server may refuse your connection: Most FTP servers put a limit on the number of people who can access the server, and if the server is already full, your connection will be refused. But, at least now you know what is going on.

Here is an example FTP session to the site ***ftp.einet.net***. I am trying to get the ***winweb.zip*** program I found with the earlier **archie** search. The user's input is boldfaced, while the computer response is in regular font; *commentary is added in italics*.

```
green:~> ftp ftp.einet.net    (start ftp, pointing to the desired Domain name)
Connected to ftp.einet.net.
220 ftp.einet.net FTP server (Version 6.42 Wed Jul 6 12:37:10 CDT 1994) ready.
Name (ftp.einet.net:igraham): anonymous
331 Send e-mail address as password (for example: joe@green.utirc.utoronto.ca).
Password:  (type in your e-mail address--it will not be visible)
230-
230- Welcome to the EINet public FTP server.  For information about
230- EINet, send e-mail to einet-info@einet.net.
230-
230- If you encounter any unusual problems, please report them via e-mail
230- to ftp@ftp.einet.net.  If your client hangs on multi-line responses,
230- please try using a dash (-) as the first character of your user name.
230- This will turn off the continuation messages that may be confusing
230- your client.
230-
230- All connections and transfers are logged.
230-
230-
230 Anonymous login ok; access restrictions apply.
Remote system type is UNIX.
Using binary mode to transfer files.
ftp> cd /einet/pc/winweb    (change into the directory containing what we want)
250 CWD command successful.
ftp> ls    (list the directory contents, just to see what's there)
total 1888
```

```
drwxr-xr-x   2 6522      2000          512 Aug  1 19:43 debug
drwxr-xr-x   2 6522      2000          512 Aug 19 15:36 parts
 rw-r--r--   1 6522      2000        22022 Jul 27 14:39 pkunzip.exe
 rw-r--r--   1 6522      2000       245756 Jul 27 14:40 vbrun300.zip
 rw-r--r--   1 6522      2000         7680 Aug  2 15:26 winvoke.wri
 rw-r--r--   1 6522      2000       596336 Sep 28 19:15 winweb.zip
 rw-r--r--   1 6522      2000       596336 Sep 28 19:15 winweb1.00A2.1.zip
 rw-r--r--   1 6522      2000       424263 Jul 26 19:50 winweb1.00A2.zip
226 Transfer complete.
ftp> bin     (set FTP session to binary mode to transfer zip files or programs)
200 Type set to I.
ftp>
get winweb.zip          (get the file from the server)
200 PORT command successful.
150 Opening BINARY mode data connection for winweb.zip (596336 bytes).
596336 bytes received in 122.55 seconds (4.75 Kbytes/s)
ftp> quit               (we're finished, so quit)
221 Goodbye.
green:~>
```

In the preceding session, the command `bin` was used to set *binary transfer* mode. This is needed for transferring binary data (such as images, movies, or sound files) and compressed program or archive files. If you are transferring plain text files, such as README files, or plain PostScript files (with the extension *.ps*), you should type the letter `a` to select *ASCII mode*.

VIRUS PROTECTION

Anytime you download programs from the Internet onto PCs, Macintoshes, or Amigas, you run the risk of importing a computer virus. You must be very cautious about this. Make sure that you have virus-detection software on your computer, and that you keep it up to date; computer viruses are evolving almost as fast as the World Wide Web. To date, there have been few cases where WWW software has been contaminated by a virus, but this is no reason to relax. Computer viruses are a major problem, and you should always be on guard.

APPENDIX

D

LISTENING AT A TCP/IP PORT

This appendix contains the source code for the program **listen**, which is a simple program that *listens* at a TCP/IP port and prints to the screen (or terminal) any characters received at the port (Figure D.1). This program was used to observe the data that WWW client browsers send to HTTP servers.

Listen was written for UNIX machines, particularly for DECstations running the operating system Ultrix 4.3A, and some work may be needed to port this program to System V-based machines. Porting to PCs or Macintoshes may require a bit more work.

To run the program, simply type the program name at the shell prompt. **Listen** then prints the port number it is listening at, and then goes silent. Here is typical output:

```
% listen
listening on 2055
```

At this point, any data sent to port 2055 on this computer will be printed on the screen just below the string `listening on 2055`. You terminate the program by typing CTRL-C.

Listen was written by Norman Wilson of the Instructional and Research Computing Group at the University of Toronto.

401

```
#include <stdio.h>
#include <sys/socket.h>
#include <netinet/in.h>
main(argc, argv)
int argc;
char **argv;
{
        int verbose;
        int lfd, sfd;
        struct sockaddr_in myaddr, claddr;
        struct in_addr ia;
        int inane;
        int pid;
        verbose = 0;
        if (argc > 1 && strcmp(argv[1], "-v") == 0) {
                verbose = 1;
                argc--;
                argv++;
        }
        if ((lfd = socket(AF_INET, SOCK_STREAM, 0)) < 0) {
                perror("socket");
                exit(1);
        }
        myaddr.sin_family = AF_INET;
        myaddr.sin_port = 0;
        myaddr.sin_addr.s_addr = INADDR_ANY;
        if (bind(lfd, &myaddr, sizeof(myaddr)) < 0) {
                perror("bind");
                exit(1);
        }
        inane = sizeof(myaddr);
        if (getsockname(lfd, &myaddr, &inane) < 0) {
                perror("getsockname");
                exit(1);
        }
        if (listen(lfd, 5) < 0) {
                perror("listen");
                exit(1);
        }
        fprintf(stderr, "listening on %d\n", ntohs(myaddr.sin_port));
        fflush(stdout);
        while (inane = sizeof(claddr), (sfd = accept(lfd, &claddr, &inane)) >= 0) {
                if ((pid = fork()) < 0)
                        perror("fork");
                else if (pid == 0)
                        copydata(sfd, verbose);
                /* else parent */
                close(sfd);
                if (verbose)
                        fprintf(stderr, "accept %s:%d, pid %d\n",
                                inet_ntoa(claddr.sin_addr), ntohs(claddr.sin_port),
                                pid);
        }
        exit(0);
}
copydata(fd, verbose)
int fd;
int verbose;
{
        char buf[200];
        int mypid;
        FILE *fp;
        mypid = getpid();
        if ((fp = fdopen(fd, "r")) == NULL) {
                fprintf(stderr, "%d: can't fdopen\n", mypid);
                exit(1);
        }
        while (fgets(buf, sizeof(buf), fp)) {
                if (verbose)
                        printf("%d: ", mypid);
                fputs(buf, stdout);
                fflush(stdout);
        }
        exit(0);
}
```

Figure D.1 Program listing for Listen.c

anchor The location of a hypertext link in a document. An anchor can be either the start of a hypertext link, or the destination of a hypertext link.

archie Archie is a system that automatically generates and maintains a contents database for anonymous FTP servers. An **archie** server accesses information from FTP servers and archives the directory listings. An **archie** client can access these databases and search for programs or files matching a particular name.

archive file An archive is a single file that contains a collection of different files and/or directories. Archive files are often used to transport collections of files across the Internet, since you can tranport a large collection in a single archive file. UNIX archives have the extension *.tar* (for Tape ARchive). **PKZIP** if often used to create archives on DOS computers (suffix *.zip*), while **StuffIt** is often used to create Macintosh archives (suffixes *.sea* or *.sit*). **PKZIP** and **StuffIt** archives are also *compressed*.

ASCII ASCII stands for the American Standard Code for Information Interchange. This is a 7-bit character code capable of representing 128 characters. Many of these characters are special control characters used in communications control, and are not printable.

attribute An attribute is a quantity that defines a special property of an HTML element. Attributes are specified within the start tag. For example, `<IMG SRC="image.gif">` means that the element **IMG** has an attribute **SRC**, which is assigned the indicated value.

browser A browser is any program used to view material prepared for the World Wide Web. **Mosaic, MacWeb**, and **lynx** are examples of browsers. Browsers are able to interpret URLs and HTML markup, and also understand several Internet protocols, such as HTTP, FTP, and Gopher.

CERN *Centre Européen pour la Récherche Nucleaire*. CERN is a large physics particle-accelerator laboratory located in Geneva on the French-Swiss border. The World Wide Web originated here, largely due to the efforts of Tim Berners-Lee.

CGI Common Gateway Interface. This is the specification for how an HTTP server should communicate with server gateway programs. Most servers support the CGI specification.

client A client is a program used to extract information from a server. For example, a browser, such as **MacWeb**, is a client that can access data from HTTP (and other) servers.

compressed Many files on the Internet are compressed—this reduces the space taken up by the files and makes transmitting them over the Internet faster. The user must then have a program to decompress the files.

CRLF The combination of a carriage-return (CR) and a linefeed (LF) character. This combination is used by several Internet protocols to denote the end of a line.

domain name Computers on the Internet can have *domain names* that are mapped onto the computer's formal numeric Internet (IP) address. Domain names allow you to reference Internet sites without having to know the numerical address.

download You are *downloading* a file whenever you transfer it from a remote computer to your computer.

DTD Document Type Definition. An SGML document type definition is a specific description of a markup language. This description is written as a plain text file, often with the filename extension ***.dtd***. The HyperText Markup Language, HTML, has its own Document Type Definition file, often called ***html.dtd***.

element An element is a basic unit of an HTML document. HTML documents use start and stop *tags* to define structural elements in the document. The name of the element is given in the name of the tag, and specifies the meaning associated with the block. Some elements are *empty*, since they don't affect a block. Elements that have contents are also often called *containers*.

firewall A *firewall* is used to separate a local network from the outside world. In general, a local network is connected to the outside world by a *gateway* computer. This gateway machine can be converted into a firewall by installing special software that does not let unauthorized TCP/IP packets pass from inside to outside, and vice versa. You can give users on the local network, and *inside* the firewall, access to the outside world by using the **SOCKS** package or by installing the CERN HTTP *proxy server* on the firewall machine.

FTP File Transfer Protocol, or FTP, is an Internet client-server protocol for transferring files between computers.

GIF GIF, for Graphics Interchange Format, is a format for storing image files. It is one of only three formats that can appear inline in an HTML document, the other two being X-Bitmaps and X-Pixelmaps.

Gopher Gopher is a protocol for distributed information delivery commonly used in distributed information systems. Gopher clients give you access to this information. Gopher is a menu-based delivery system and does not have hypertext capabilities.

helper A helper application is a program launched or used by a browser (such as **Mosaic** or **MacWeb**) to process files that the browser cannot handle internally. Thus, you have helpers to view JPEG images or play sound files, and also to decompress compressed files, or to unstuff archives.

home page The home page is the introductory page for a World Wide Web site. A home page usually provides an introduction to the site, along with hypertext links to local resources.

HTML Abbreviation for HyperText Markup Language. HTML is a markup language defined by an SGML Document Type Definition (DTD). To a document writer, HTML is simply a collection of tags used to mark blocks of text and assign them special meanings.

hypertext A hypertext document is any document that contains *hypertext links* to other documents. HTML documents are almost always hypertext documents.

inline image An inline image is merged with the displayed text. This is often described as *inlining* the image.

Internet provider An Internet provider is the company from which you purchase your Internet connectivity. This could either be a dedicated connection (for example, a telephone connection that stays open 24 hours a day) or a dial-up connection. Usually, you run software, such as PPP or SLIP, to allow Internet connectivity across the line.

Internet resources The collection of data, documents, and databases available on the Internet.

ISO International Standards Organization. An international organization responsible for setting international standards, such as the ISO Latin-1 character set.

ISO Latin-1 This is an 8-bit character code developed by the International Standards Organization. An 8-bit code contains 256 different characters. In the ISO Latin-1 code, the first 128 characters are equivalent to the 128 characters of the US-ASCII character set (also called the ISO 646 character set). The remaining 128 characters consist of control characters plus a large collection of accented and other characters commonly used in European languages.

JPEG From Joint Photographic Experts Group, JPEG is another image format. In general, JPEG allows for higher-quality images than GIF. Browsers cannot display JPEG images inline and, instead, must display them using *helper* programs.

LAN LAN is an acronym for Local Area Network.

link A *link*, or *hypertext link*, is the connection between one hypertext document and another. Links are associated with hypertext anchors.

linux **Linux** is a freeware clone of UNIX for 386-based PC computers. **Linux** consists of the linux kernel (core operating system), originally written by Linus Torvalds, along with utility programs developed by the Free Software Foundation and by others. Since PC hardware is inexpensive and **linux** is essentially free, the combination of the two is a practical way of developing inexpensive and reliable HTTP service.

Lynx **Lynx** is a very popular character-mode World Wide Web browser.

MIME MIME, or Multipurpose Internet Mail Extensions, is a scheme for allowing electronic mail messages to contain mixed media (sound, video, image, and text). The World Wide Web uses the MIME content-type to specify the the type of data contained in a file or being sent from an HTTP server to a client.

Mosaic **Mosaic** is a graphical browser for the World Wide Web, developed at NCSA. There are also several commercial versions of **Mosaic**.

MPEG MPEG stands for Motion Picture Experts Group, and is an acronym for a common video file compression method.

NCSA National Center for Supercomputing Applications. The NCSA is situated at the Urbana-Champaign campus of the University of Illinois. The NCSA software development team developed the programs **Mosaic** and the NCSA HTTPD server.

packet A packet is a small package of data. The TCP/IP Internet protocol breaks messages up into packets, and sends each packet independently to the message destination. The protocol ensures that there is no error in transmission and that the entire message arrives.

perl Practical Extraction and Reporting Language. A scripting language written by Larry Wall. Because powerful data and text manipulation programs can be written quickly and easily using perl, it has become a popular language for writing CGI applications.

port number Any Internet application communicates at a particular port number specific to the application. For example, FTP, HTTP, Gopher, and telnet are all assigned unique port numbers so that the computer knows what to do when contacted at a particular port. There are accepted standard numbers for these ports so that computers know which port to connect to for a particular service. For example, Gopher servers generally *talk* at port 70, while HTTP servers generally *talk* at port 80. These default values can be overridden in a URL.

PPP Point-to-Point Protocol. Software that allows you to turn a dial-up telephone connection into a point-to-point Internet connection. This is commonly used to run WWW browsers over a phone line.

provider See *Internet provider*.

protocol A protocol is simply an agreed convention for intercomputer communication. Thus, the TCP/IP protocol defines how messages are passed on the Internet, while the FTP protocol, which is built using the TCP/IP protocol, defines how FTP messages should be sent and received.

proxy server A *proxy* server acts as an intermediary between your computer and the computer you want to access. Thus, if you make a request for a resource from computer *A*, this request is directed to a proxy server, which makes the request, gets the response from computer *A*, and then forwards the response to you. Proxy servers are useful for accessing World Wide Web resources from inside a *firewall*.

RFC An RFC, or Request for Comments, is a document, written by groups or individuals involved in Internet development, that describes agreed-upon standards or proposes new standards for Internet protocols. For example, the rules for electronic mail message composition are specified in the document RFC822.

server A server is a program, running on a networked computer, that responds to requests from client programs running on other networked computers. The server and client communicate using a client-server *protocol*.

SGML Standard Generalized Markup Language. This is a standard for describing markup languages. HTML is defined as an instance of SGML.

shell The UNIX *shell* is the program that interprets the commands typed at the terminal. A shell can also be used to run simple script programs called *shell scripts*. There are several different shells, with slightly different commands and syntax. The most common are the Bourne shell (sh), the C shell (csh), and the Korn shell (ksh).

SLIP Serial Line Internet Protocol. Software that allows you to turn a dial-up telephone connection into an Internet connection. This can be to run WWW browsers over a phone line, but is less stable than a PPP connection.

SOCKS SOCKS is a package that allows hosts inside a firewall to communicate with the outside world. To allow access to the outside world, a secure network must run a SOCKS server on its gateway/firewall machine and must configure all its networking software to talk to the SOCKS server. SOCKS is a proxy server without the special caching capabilities of the CERN HTTP server.

tag (HTML) HTML marks documents using *tags*. A tag is simply typed text surrounded by the less-than and greater-than signs, for example: <TAG>. An end tag is similar but has a slash in front of the tag name, such as </TAG>.

TCP/IP Transmission Control Protocol/Internet Protocol. TCP/IP is the basic communication protocol that is the foundation of the Internet. All the other protocols, such as HTTP, FTP, and Gopher, are built on top of TCP/IP.

telnet Telnet is a terminal emulation protocol that allows you to make a terminal connection to other computers on the Internet. This requires that you run a telnet client on your computer and connect to a telnet server on the other machine.

TIA TIA stands for The Internet Adapter. This is a program that you can run on a dial-in UNIX account and that allows you to create a SLIP-like connection between your home computer and the dial-in site. TIA is useful if you have a UNIX account with a company that does not provide PPP or SLIP service.

TIFF TIFF, for Tag Image File Format, is a graphic file format developed by Aldus Corporation. TIFF is the standard format of many graphics and desktop publishing programs.

URL URL stands for Uniform Resource Locator, and is the scheme used to address Internet resources on the World Wide Web.

USENET USENET is the Internet's worldwide bulletin-board system, consisting of over 5,000 topical discussion groups, called newsgroups. The newsgroups related to the World Wide Web are: ***comp.infosystems.www.misc***; ***comp.infosystems.www.providers***; ***comp.infosystems.www.users***; and ***bionet.software.www*** (for biological applications only).

viewer A viewer is a program launched by a browser (such as **Mosaic** or **MacWeb**) to view files that the browser cannot handle internally. Thus, you have viewers for JPEG images, sound files, and MPEG movies. Viewers are also often called *helpers* or *helper applications*.

visit When you access a World Wide Web document, you are said to be *visiting* the site.

whitespace A *whitespace* refers to any combination of space or tab characters that separate two character strings.

WWW The World Wide Web. Also called the Web or W3.

& (ampersand), 74-75, 196, 384-85
<> (angle brackets), 74-75
\ (backslash), 25, 166
: (colon), 25, 166
" (double quotes), 74-75, 79, 163, 384-85
/ (forward slash), 2, 32, 164, 165, 166
> (greater-than sign), 75, 76, 384-85
(hash sign), 164, 175-76, 242, 254
< (less-than sign), 76, 384-85
% (percent sign), 164, 177
+ (plus sign), 173, 196, 213-14
? (question mark), 165, 174
; (semicolon), 73
. (single dot), 26-27, 32
[] (square brackets), 20
~ (tilde character), 176

A

ABBREV element, 139
ABSTRACT element, 155
Acadia University, 288
access control, 201-4
ACRONYM element, 139
ACTION attribute, 67
ADDRESS element, 43, 49, 94-95
addresses, in URLs, 165-66
Adobe Photoshop, 234
A element, 31, 129-34
Air Mosaic, 34-35, 37, 312-13
ALIGN attribute, 19
alldocs.sit, 53
alldocs.tar, 53
alldocs.tar.Z, 53
alldocs.tar.z, 53
alldocs.zip, 52
Alpha, 282
ALT attribute, 20, 22, 31
AMosaic, 331
anchors, basic description of, 21-22, 403
Andressen, Marc, 285
Andrew Toolkit, 289
archie, 310, 316, 319-20
  basic description of, 403
  finding files with, overview of, 395-99
  for Macintoshes, 397
  standard, 395-97
archives, 52-55, 276, 403
ARG element, 139
Asc2html.pl, 293
ASCII code, 161-65, 170
  basic description of, 403
  converting, into HTML, 293
  US-ASCII character set, 73-75, 162-63, 383, 385
attribute(s). *See also* specific
attributes, 77-82
  basic description of, 3, 77, 403
  case-sensitivity and, 78-79
  names, 78-79
  unknown, 80
  values, 79
audio files, 316, 319, 328. *See also* sound files
authentication, user, 201-4
Avalanche Development, 294-95

B

BACK attribute, 48
badlinks.html, 28-29
Bahneman, Liem, 291
Bare Bones Software, 282
BASE element, 86-88, 206
Basic Audio, 319
BASIC WEBserver, 245
Batch mode browser, 321
Bbc_man2html.pl, 267
BBEdit, 282
BBEdit-Lite, 282
B element, 11, 146
Bellver, Carles, 283
binary mode, 170
bionet.software.www, 69
blobby.map, 242, 244-45
BLOCKQUOTE element, 14-15, 95-99
BODY element, 9-11, 14-15, 48, 94-95
  comments and, 75
  introduction to, 80-81
Bourne-shell script, 210-15, 220, 249, 250
Boutell, Thomas, 265
BR element, 79, 137-38
browsers. *See also* specific browsers
  Amiga, 331
  basic description of, 309-32, 403
  Macintosh platform, 317-19
  NeXT, 329-30
  OS/2, 312-15, 331
  PC platform, 310-12
  UNIX, 321-28
  VAX/VMS, 321-28
  Windows, 312-15
Bruce, Thomas R., 313
Buhle, E. Loren, Jr., 354-63
button icons, 22

C

C (high-level language), 208, 241, 251-52, 275-76
C++ (high-level language), 208
caching, server, 203-4
Can Be Inside field, 83-84
Can Contain field, 83-84
capitalization, 2, 79
CAPTION element, 149-50
case-sensitivity, 3-4, 20, 78-79
Cello, 7-11, 13, 313, 316
CERN, 188, 202, 204, 205
  basic description of, 403
  HTTP server, 337
  NeXT browser, 330
CGI (Common Gateway Interface), 182, 184, 191-92, 199, 201-77
  archive sites, 276
  basic description of, 208-77, 403
  client-side executable programs and, 248-49
  database programs, 269-76, 277
  EIT library, 268
  email handler, 265
  feedback form, 265-66

409

imagemap and, 241
includes and, 253-61
proxy servers and, 203-4
utility functions, overview of, 265-68
characters (character sets). *See also* symbols
disallowed, 163
encoding schemes and, 162
hexadecimal code for, 163
highlighting elements and, 138-48
Latin-1 character set, 73-74, 84, 162-63, 182, 383-88, 405
special, 74-75
tags as strings of, 76-77
US-ASCII character set, 73-74, 162-63, 383, 385
valid printable, 73-74
Charconv, 293-94
Chen, Raymond, 302
chimera browser, 55, 321-22
CITE element, 139, 141
CL-HTTP, 337-38
Clark, James, 206
client-side programs, 247-71
configuring the client and, 250
security issues and, 250-52
sending scripts and, 248-49
CMD element, 139
CMS HTTPD, 338-39
CODE element, 11, 139, 142
collections, document, 44-56
archiving, 52-55
designing, 44-45
organization of, 51-52
color
bullets, 235
dithering, 234
home pages and, 38-39
image downloading problems and, 41, 57
indexing schemes, 233
map tables, 233-34
RGB values for, 237
comments, basic description of, 75-76
comp.infosystems.www.misc, 69
comp.infosystems.www.providers, 68
comp.infosystems.www.users, 69
Computer Based Learning Unit, 296
Computer Services Centre, 284
config, 257-58
Connolly, Dan, 306
Cons, Lionel, 302
containers, 2, 77
Control Panel, 329
converters, 279-80, 291-303
Cornell University Medical College, 235-36
CR characters, 170
Crawshaw, Peter, 287
CRLF (carriage-return line-feed character pair), 187, 189, 404
Crowcroft, J., 298
CU_HTML.DOT, 283-84
Cutter, Brooks, 267
Cyberleaf, 294

D

databases
CGI, 269-76, 277
FORMs and, 62
image map, 242-47
data.html, 176
Davies, Lindsay, 283
Davis, John H., 285
DD element, 115-17
DECthread HTTP server, 339
Delay Image Loading button, 21
DIR element, 13, 15, 119-21
directories
archiving, 52
cgi-bin, 26, 42, 175, 229, 241-42, 260
document, 24-26, 42
home, 170
information on, in URLs, 79, 169
personal HTML, 176-77
special program, 176
dithering, 234
DL element, 113-14
document
converters/translators, 279-80, 291-303
structure, overview of, 80-81
type definition (DTD), 72, 159, 262, 280, 303-8, 404
domain names, 42, 162, 165, 166, 404
DOS (Disk Operating System), 25, 73-74, 166
DosLynx, 311-12
Drakos, Nikos, 296
DRN element, 139, 142
DT element, 114
DTD (Document Type Definition), 72, 159, 262, 280, 303-8, 404
Dtd2html, 262

E

eavesdropping, 184-86
echo, 255-56, 259
editors, overview of, 280-91
electronic e-print servers, 349-54
element(s). *See also* elements (listed by name)
character-related, 134-38
basic description of, 2, 4, 7-11, 404
complete listing of, 81-159
empty, 3-4, 69-70, 77
highlighting, 10-11, 138-48
markup tags and, 76-81
nesting, 4, 80
overall document structure and, 80-81
rules for using, 3-4, 83-84
unknown, 80
elements (listed by name)
ABBREV element, 139
ABSTRACT element, 155
ACRONYM element, 139
ADDRESS element, 43, 49, 94-95
A element, 31, 129-34
ARG element, 139
BASE element, 86-88, 206
B element, 11, 146
BLOCKQUOTE element, 14-15, 95-99
BODY element, 9-11, 14-15, 48, 75, 80-81, 94-95
BR element, 79, 137-38
CAPTION element, 149-50
CITE element, 139, 141
CMD element, 139
CODE element, 11, 139, 142

DD element, 115-17
DIR element, 13, 15, 119-21
DL element, 113-14
DRN element, 139, 142
DT element, 114
EM element, 10, 48, 66, 139, 143
FIG element, 155-56
FOOTNOTE element, 148, 156
FORM element, 14, 15, 61-68, 99-101, 182-84, 186, 194-200, 202, 208-10, 223, 225-29, 265-66, 268, 306-8
H1 element, 9, 15, 37-40, 77, 84, 263, 264
H2 element, 37-39, 84, 264
H3 element, 84, 264
H4 element, 84, 264
H5 element, 84, 264
H6 element, 15, 84, 264
HEAD element, 8-11, 15, 81, 83, 85-87
HN element, 111-13
HREF element, 31, 79, 87
HR element, 14-15, 79, 113
HT element, 77
HTML element, 85
I element, 11, 146, 147
IMG element, 14, 17, 20-21, 31, 47, 60, 77, 79, 134-37, 240
INPUT element, 64, 65, 67, 101-6
ISINDEX element, 86-89, 182, 194, 208-15, 218, 223, 225
KBD element, 11, 139, 143
LI element, 14, 15, 77-78, 83-84, 123-24
LINK element, 86-87, 89-90, 205, 207
LIST element, 113-34
LIT element, 157
MARGIN element, 148, 157
MATH element, 158
MENU element, 13, 15, 121-23
META element, 86-87, 90-91, 201
NEXTID element, 86-87, 92
NOTE element, 158
OL element, 13, 15, 53, 117-18
ONLINE element, 159
OPTION element, 64, 107-8
P element, 124-26
PERSON element, 139
PRE element, 48-51, 126-29
Q element, 139
SAMP element, 139, 143-44
SELECT element, 64, 67, 106-7
S element, 146
STRIKE element, 139, 144
STRONG element, 10-11, 14, 48, 66, 139, 144-45
SUB element, 146
SUP element, 146
TABLE element, 148, 262
TD element, 153-54
TEXTAREA element, 108-11
TEXTBOX element, 64, 67
TH element, 150-53
TITLE element, 8-9, 15, 37-39, 86-87, 92-93, 207
TR element, 155
TT element, 11, 146, 147
U element, 80, 146, 147
UL element, 13-15, 53, 77, 118-19
VAR element, 11, 139, 145
emacs, 284-85
emacs-w3, 262, 322
email (utility program), 265
EM element, 10, 48, 66, 139, 143
encryption, 201-4. *See also* security
Enterprise Integration Network Group (EINet), 315, 319
Enterprise Integration Technologies, 252, 263, 268, 308
environmental variables, 209, 218-24, 255, 258, 266-68
errors
  config and, 257
  "document not available," 31
  the HTTP protocol and, 181, 193
  nesting, 80
  validation tools and, 303-6
European languages, 73. *See also* characters (character sets)
European Southern Observatory, 274
ex1.html, 5-15
ex2a.html, 15-32
ex2b.html, 15, 17, 23-25
ex2c.html, 24, 25, 26
ex2d.html, 24, 25, 26
ex_prog, 208-9
exec, 256-57
Expander Enhancer, 319
Extensions Package I, 282
Extensions Package II, 283

F

FAQ (Frequently Asked Questions) lists, 69, 279
FasTag, 294
FIG element, 155-56
file extensions
  .gif, 17, 22, 31, 61
  .htm, 5, 391
  .html, 5, 391
  .mpe, 61
  .mpeg, 61, 391
  .mpg, 61, 391
  .xbm, 21, 22, 31
  .xpm, 22, 31
  .zip, 52
file.html, 167-68
firewalls, 203, 336, 404, 407
flastmod, 256
fonts, 72. *See also* highlighting
FOOTNOTE element, 148, 156
foreign languages, 73. *See also* characters (character set)
FORM element, 14, 15, 61-68, 99-101, 182-84, 186, 194-200, 202, 208-10, 223, 225-29, 265-66, 268, 306-8
form.html, 63-64, 65, 66, 68
fragment identifiers, 175-76
Frame2html, 295
FrameMaker, 280, 291, 292
Fraser, Eric J., 373-82
Free Software Foundation, 284-85
freeWAIS-sf, 271
fsize, 256

FTP (File Transfer Protocol), 43, 61, 161, 165, 168-70
archie and, 397-99
basic description of, 404
ftplib.pl, 307

G

Gaier, Karsten, 265
gateway programs, 175, 176, 181-230. *See also* CGI (Common Gateway Interface); HTTP protocol
General Video, 329
Georgia Institute of Technology, 273
GET method, 183, 186-98, 205-6, 209, 210, 222, 225, 228, 240, 307
Getstats, 346
GIF images
active images and, 239
basic description of, 404
file extension for, 17, 22, 31, 61
home pages and, 38
linking large, 56-61
mapedit and, 246
transparent, 233, 235-38
Giftrans, 236-37
Giles, Aaron, 235
Giles, Rick, 288
Ginsparg, Paul, 349-54
GMT (Greenwich Mean Time), 191, 206
GN Gopher/HTTP server, 339-40
Gopher, 43, 161, 339-40
basic description of, 405
resource identifier codes, 172
URLs and, 165-66, 168, 171-74
Graphic Converter, 234
graphics. *See also* images
document collections and, 55
home pages and, 33, 38-39
grep, 210, 215
Grimm, Steven, 268
Grover, Jeffrey L., 285
GSQL, 274
GSQL-Oracle Backend, 272-73
GT_HTML.DOT, 285
GWHIS HTTP server, 340
Gwstat, 347
gzip, 53, 316

H

H1 element, 9, 15, 37-40, 77, 84, 263, 264
H2 element, 37-39, 84, 264
H3 element, 84, 264
H4 element, 84, 264
H5 element, 84, 264
H6 element, 15, 84, 264
Hamilton, Calvin J., 364-69
Harawitz, Howard, 287
Hartikka, Jorma, 300
HEAD element, 8-11, 15, 85-87
basic description of, 81, 86-87
rules applying to, 83
HEAD method, 186, 188, 200-201, 205, 208, 222, 307
Hector, Chris, 299
helper applications, 60-61, 309, 315-17
basic description of, 405
Macintosh, 319-20
Unix, 328-29
Help icons, 40
Herwijnen, Eric van, 289
highlighting, 10-11, 84, 138-48
High Performance Computing home page, 40
HLPDK, 295
HN element, 111-13
home_bad.html, 34-35
home.gif, 17
home.html, 36-37, 38
Home Icons, 19
home pages
basic description of, 405
designing, 32-33
examples of, 30-44
icons and, 33, 40-41
information provided on, 33
organization of, 39-40
text portion of, 39
titles and headings for, 37-39
Hood, Earl, 262, 266, 300
HoTMetaL, 285-86
HoTMetaL Pro, 294-95
HREF attribute, 22, 77
HREF element, 31, 79, 87
HR element, 14-15, 79, 113
hrule.html, 46-47, 51
HT element, 77
HTML (HyperText Markup Language)
basic outline of, 4
HTML+ features, 82, 84, 148-59, 262
simple example document, 5-15
standard version (Version 2.0), 72, 81, 84, 87, 159, 262
HTML assistant, 286-87
html.dtd, 72, 303
HTMLed, 287
HTML.edit, 288
HTML Editor, 288
HTML element, 85
HTML for Word 2.0, 288-89
htmlindex.html, 50
Htmltoc, 264
HTOracle, 274-75
HTTP_Method, 188
HTTP protocol, 181-230. *See also* HTTP servers
adding header information with, 215-18
basic description of, 182-83
four stages of, 183
GET method, 183, 186-98, 205-6, 209, 210, 222, 225, 228, 240
HEAD method, 186, 188, 200-201, 205, 208, 222
monitoring client-server interaction and, 184-86
POST method, 183, 188, 192, 198-200, 205, 208-10, 222, 224-29
PUT method, 205, 224
redirection transactions, 192, 193
request headers, 183, 187, 193-94, 204-8
response headers, 183, 189-92, 194, 204-8
TEXTSEARCH method, 194, 195, 205
HTTPS, 340-41

HTTP servers, 5, 62, 333-48. *See also* HTTP protocol
  archiving and, 53
  client-side executable programs and, 249
  home pages and, 39
  image map databases and, 243, 245
  URLs and, 25, 42, 161, 162, 165, 166, 168, 174-77
HTTP_Version, 188, 189
Huang, Z., 59
Hughes, Kevin, 263
Huyler, Stafford, 369-72
Huynh, Chung, 325
Hyperlatex, 296
Hypermail, 263
hypertext links
  active images and, 239, 240
  anchors and, 21-22
  basic description of, 15-32, 405
  creating, 26-27
  document collections and, 44, 47-48, 55
  good and bad examples of, 27-31
  home pages and, 39-40, 43
  the HREF element and, 77
  linking large images and, 57, 60
  partial URLs and, 22-26
  utility programs and, 263, 264
  validation tools and, 306-8

I

icons
  audio/movie files and, 57-59, 61
  home pages and, 33, 40-41
  images and, 57-59, 61, 247
  navigation, 22, 33, 40-41, 44-45, 55
  thumbnail sketches as, 57-59, 61
I element, 11, 146, 147
IETF (Internet Engineering Task Force), 73, 179, 182
image.gif, 237
imagemap, 175, 240-44
image(s). *See also* specific types
  active, 238-42
  database, creating, 242-47
  helper applications and, 60-61, 316, 320, 329
  icons and, 57-59, 61, 247
  linking large, 56-61
  map file, 175, 240-44, 246-47
  size, reducing, 233-35
  types, overview of, 232-47
  viewing program (UNIX), 60, 237, 238, 246
IMG element, 14, 17, 20-21, 31, 47, 60, 77, 79, 134-37, 240
inc_file.shtml, 258, 260
includes, server-side document, 253-61
INPUT element, 64, 65, 67, 101-6
Instructional and Research Computing Group, 32
Interleaf, 292
ISINDEX element, 86-89, 182, 194, 208-15, 218, 223, 225
ISMAP attribute, 240
ISO Latin-1 character set, 72-73, 84, 162-63, 182, 383-88, 405

J

Johnson, Tony, 325
JPEG (Joint Photographic Experts Group), 76, 405
Jungle, 341
junk.html, 255

K

Kawall, J. G., 59
KBD element, 11, 139, 143
Keffer, J. F., 59
keywords, searching for, 44
kidowais.pl, 270-71
Kirste, Burkhard, 294
Kumar, Vinay, 252

L

Lam, Anton, 284
languages, foreign, 73. *See also* characters (character sets)
LANs (local area networks), 203, 204, 406
LaTeX, 280, 291, 292
Latex2html, 296
leonardo.subnet.ca, 185
Letovsky, Stanley, 273
LI element, 14, 15, 77-78, 83-84, 123-24
LineMode browser, 323
Linkcheck, 306-7
LINK element, 86-87, 89-90, 205, 207
linux, 406
LIST element, 113-34
listen (program), 184-85
lists
  FAQ lists, 69, 279
  unordered lists, 12, 13-14, 53
LIT element, 157
Loewy, Ron, 295
Los Alamos National Laboratory, 349-54, 364-69
lynx, 5, 7-11, 13, 311-12, 323
  active images and, 239
  basic description of, 406
  editors and, 281
  ex2a.html and, 16, 17, 19-20, 22, 23
  FORMs and, 62, 66, 67
  home pages and, 35, 38

M

Maber, Jon, 252
McBeath, Jim, 297
MacHTTP, 341-42
Macintosh, 25, 166, 170, 336. *See also* MacWeb browser
  archie for, 397
  archiving and, 52
  character sets and, 73-74
  client-side executable programs and, 250
  GIF images and, 235-36
  helper applications and, 61, 319-20

the HTTP protocol and, 185
platform browsers, 317-19
search tools, 276
space characters and, 163
MacMosaic, 62, 67, 68, 317-18
MacWeb browser, 5, 79, 317-19
editors and, 281
ex2a.html and, 16, 17, 19, 21
ex2b.html and, 24
protocols and, 43, 195
mailcap, 250, 251-52
mailto, 169, 177
main.html, 168
man.cgi, 266
Man2html, 266-67
mapedit, 246
MARGIN element, 148, 157
MATH element, 158
MCC (Microelectronics and Computer Technology Corporation), 315, 319
mconnect, 307
MDMA, 345
MENU element, 13, 15, 121-23
META element, 86-87, 90-91, 201
Metamail, 329
method fields, 187-88, 193-94, 195
MHonArc, 263
Micromind Slipknot, 331
Midas WWW, 324-25
Mif2html, 297
Miftran, 297
MIME (Multipart Independent Mail Extensions) type, 60-61, 76, 170
basic description of, 389-94, 406
client-side executable programs and, 248, 250
the HTTP protocol and, 188, 191, 194, 199, 205-7, 214, 217, 224, 228
server-side document includes and, 257
utility programs and, 263
Mm2html, 297-98
Monroe, Hunter, 302
Montulli, Lou, 323
Morgan, Eric Lease, 290
Mosaic, 281. *See also* Mosaic for X-Windows
basic description of, 406
Mosaic (TueV) version, 326
Mosaic for X-Windows, 15-18, 43, 325-26, 250
document collections and, 47
ex2a.html and, 16, 21, 23, 25
ex2b.html and, 25
FORMs and, 62, 64
the HTTP protocol and, 186-87, 196, 197, 198, 202
hypertext links and, 28-29
linking large images and, 58-59
URLs and, 170, 178
utility programs and, 262
movies, 56-61, 316, 320, 329, 406
MPEG video, 57, 60, 316, 320, 329, 406
Ms2html.pl, 298
my_database, 241

N

name(s)
domain, 42, 162, 165, 166, 404
element, 77, 78-79
/value pairs, 196-97
NAME attribute, 64, 65, 175
National Science Foundation, 273
navigation icons, 22, 33, 40-41, 44-45, 55
NCSA (National Center for Supercomputing Applications), 202, 241, 243-45, 261
description of, 406
HTTPD server, 253-54, 342-43
nesting, 4, 67, 80
Netboy, 369-72
Netscape Communications Corporation, 331-32
Netscape Netsite, 346
NEXTID element, 86-87, 92
NextStep HTML-Editor, 289
NICE Technologies, 289
Nierstrasz, Oscar, 298
nonparsed headers, 192, 210, 217-18, 219, 245-46
North Carolina State University, 270
Norwegian Telecom Research, 295
NOTE element, 158
NotePad, 2, 281
nroff/troff, 292

O

OL element, 13, 15, 53, 117-18
Omniweb, 330
OncoLink, 354-63
ONLINE element, 159
OPTION element, 64, 107-8
OraPerl, 275
OraPlex, 275
OS/2, 312-15, 331, 335
OS2HTTPD, 343

P

Paintshop Pro, 234
paragraphs, 1, 11-13, 32
Pascal, 208
passwords, 165-66, 169, 184, 201-4
path information, 166
pbmplus, 234
P element, 124-26
Pennsylvania State University, 306
perl (Practical Extraction and Reporting Language), 262-63, 267, 268, 321, 406
Perry, William, 322
PERSON element, 139
Phelps, T. A., 299
Pitkow, James, 273
PKZIP, 52, 53
Plexus, 343-44
port numbers, 42, 162, 165-66, 406
POST method, 183, 188, 192, 198-200, 205, 208-10, 222, 224-29
PostScript, 292, 317, 320, 329
Powell, James, 289

PRE element, 48-51, 126-29
proxy servers, 203-4
Ps2html, 298
PUT method, 205, 224

Q

Q element, 139
Quadralay Corporation, 297, 326
query strings (search strings), 166, 165
  client construction of, 173
  the HTTP protocol, 194-95, 197, 208, 210-15
QuickTime, 317, 320

R

Rashty VMS client, 326-27
Rasmussen, Bo Frese, 274
request headers, 183, 187, 193-94
response headers, 183, 189-92, 194, 204-8
resource identifiers, 171-72
resource location, 166
rlogin, 178
RosettaMan, 267, 299
RTF (Rich Text Format), 293, 299, 300
RTFTOHTM, 300
Rtftohtml, 299
rules, for elements, 3-4, 83-84

S

SAMP element, 139, 143-44
San Francisco Reservations (SFR), 373-82
Sanders, Tony, 269-70
Schwarzkopf, Otfried, 296
Scribe, 291, 293
Scribe2html, 300
search strings. *See* query strings (search strings)
security, 201-4, 229-30, 250-52
SELECT element, 64, 67, 106-7
selector strings, 171-72
S element, 146
Send Message button, 62, 65
send_note, 62-63
server
  directives, 216-17
  -side document includes, 253-61
  utilities, overview of, 333-48
SerWeb, 344
SFgate, 271
SGML (Standard Generalized Markup Language), 72, 159, 262
  basic description of, 407
  validation tools and, 293
Sgmls, 303-6
S.H.E. (Simple HTML Editor), 290
Sibley, David, 306
Smart, Julian, 300
SOCKS proxy support, 204
SoftQuad, 285
solar system, views of, 364-69
son-of-wais.pl, 270
sound files, 56-61. *See also* audio files
space characters, 15, 48, 163
Spyglass, 314
srch-example, 210-15, 217, 218, 220
src_link.html, 54-55
sright.xbm, 21
standard input, 209, 225-27
Stanford Linear Accelerator Laboratory, 325
STRIKE element, 139, 144
Striphtml, 300
STRONG element, 10-11, 14, 48, 66, 139, 144-45
StuffIt, 52-53, 319-20
stuff.shtml, 257, 258, 259-61
SUB element, 146
SUP element, 146
symbols. *See also* characters (character sets)
  & (ampersand), 74-75, 196, 384-85
  <> (angle brackets), 74-75
  \ (backslash), 25, 166
  : (colon), 25, 166
  " (double quotes), 74-75, 79, 163, 384-85
  / (forward slash), 2, 32, 164, 165, 166
  > (greater-than sign), 75, 76, 384-85
  # (hash sign), 164, 175-76, 242, 254
  < (less-than sign), 76, 384-85
  % (percent sign), 164, 177
  + (plus sign), 173, 196, 213-14
  ? (question mark), 165, 174
  ; (semicolon), 73
  . (single dot), 26-27, 32
  [] (square brackets), 20
  ~ (tilde character), 176

T

TABLE element, 148, 262
Table of Contents, 37, 231
  document collections and, 45, 50-52, 55
  generating, 264
tags, 2-4
  capitalization and, 2
  comments and, 75
  end, 77, 79-80, 83
  introduction to, 2, 4, 7-11, 76-81
  paragraph, 11-13
  special characters denoting, 74-75
  start, 77, 79-80
  unordered list, 12, 13-14
TagWrite, 300
tcl, 208
TCP/IP networks, 203, 310-11, 317, 401-2, 408
TD element, 153-54
TeachText, 2, 281
telnet, 165, 169, 178, 185
test.html, 304-5
test_script.cgi, 258, 260
Tetzchner, Jon Stephenson von, 295
TeX, 10, 329
Tex2rtf, 300
Texi2html, 301-2
Texinfo, 293
TEXTAREA element, 108-11
TEXTBOX element, 64, 67
TEXTSEARCH method, 194, 195, 205
TH element, 150-53

thumbnail sketches, 57-59
time zones, 191, 206
TITLE element, 8-9, 15, 37-39, 86-87, 92-93, 207
tkHTML, 290-91
TkWWW, 291, 327
TR element, 155
TR-WWW, 276
translators, document, 279-80, 291-303
transparency, 233, 235-38
Treelink, 264-65
trumpet, 310, 311
TT element, 11, 146, 147

U

U element, 80, 146, 147
UL element, 13-15, 53, 77, 118-19
Uncgi, 268
University Home Page, 42
UNIX, 1-2, 5-15, 24, 170, 177-78, 293
  archiving and, 53
  client-side executable programs and, 250
  document collections and, 55
  editors and, 280
  helper programs, 328-29
  home pages and, 35
  the HTTP protocol and, 185
  image viewing program (xv), 60, 237, 238, 246
  servers, 334-35
  w3launch and, 252
UNLINK, 205
UNSET, 332
uoft.html, 42
Up button, 48
Url_get, 321
URLs (Uniform Resource Locators), 3, 22-27, 30-32
  active images and, 241, 242
  allowed characters in, 162-63
  basic description of, 42-43, 161-80, 408
  directory/filename information in, 79
  disallowed characters in, 163
  document collections and, 47
  editors and, 281
  fragment identifiers and, 175-76
  the HREF element and, 77
  the HTTP protocol and, 182-83, 186, 188, 190, 192-94, 199-200, 205-6, 208-9, 211-15, 217-18, 221, 223, 225, 228-29
  image maps and, 242, 243, 244, 245
  partial, 22-26, 167-68
  passing parameters to servers in, 174-75
  server-side document includes and, 257
  special characters in, 164-65
  specifications for, 168-79
  typical, example of, 165-67
  validation tools and, 280, 303, 306-8
USENET newsgroups, 68-69, 166, 167, 177-78, 247, 279, 310, 408
utilities, 261-65, 333-48

V

validation tools, 280, 303-8
VALUE attribute, 64
VAR element, 11, 139, 145
Ventura Publisher, 293
Verify_links, 307-8
video, 56-61, 316, 317, 320, 329
Views of the Solar System, 364-69
viola, 262
ViolaWWW, 327-28
virus protection, 399
VMS server, 335
vortex.html, 58
vsafecsh, 251-52
VT-100 browser, 1-2

W

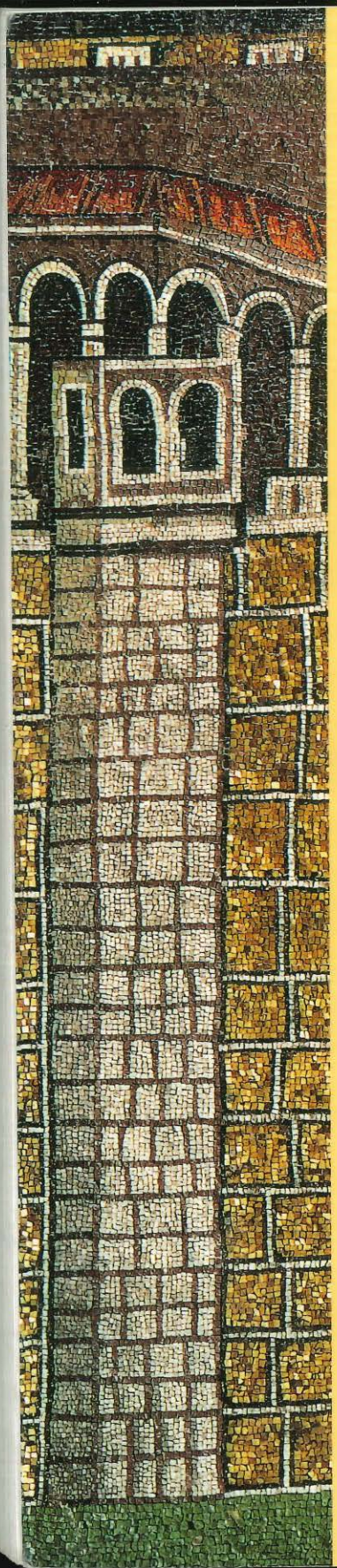
w3launch, 251-52
WAIS, 161, 165-66, 169, 178, 269-72
Wang, Joseph, 327
WDB, 274
Web4Ham, 344
web_admin.html, 43
Web Explorer, 331
Web/Genera, 273-74
WebMaker, 302-3
Webstat, 347-48
Wei, Pei Y., 328
Whelan, Jerry, 298
Wilson, Norman, 401
win32s, 311
Windows (Microsoft), 25, 61, 166, 252, 312-15, 335-36
WinHTTPD, 344-45
WinMosaic, 314-15
WINSOCK.DLL, 311
WinWeb, 315
Wong, Kenneth, 284
Word for Windows, 1, 283-85, 288-89, 292
WordPerfect, 1, 280, 293
Wp2x, 302
WPTOHTML, 302
Write, 281
Wusage, 348
Wwwstat, 348
WWWWAIS, 272

X

X-Bitmaps, 21, 22, 31, 57, 232, 235, 239
X-Pixelmaps, 22, 31, 232-33, 239
xdvi.dvi file previewers, 329

Z

Zandar Corporation, 300
Ziegler, Rudy, 59
Zip files, 320



Communications/Internet **$29.95**

Your total guide to online publishing using

HTML

The HTML Sourcebook is an indispensable resource for authors, publishers, marketers, educators, and virtually anyone interested in publishing on the World Wide Web. Written by the author of one of the most accessed HTML online tutorials, it provides all the essentials for preparing character-based, as well as graphically based, hypertext documents for online publication. With the help of Ian S. Graham's clear, step-by-step guidelines and priceless pointers you'll quickly learn how to:

- **Master all HTML commands and the URL syntax**
- **Create and edit HTML documents for Mosaic, Netscape, Lynx, and all other WWW browsers**
- **Translate documents from other formats such as Word or WordPerfect into HTML documents**
- **Link your hypertext documents to graphics, video, and sound files**
- **Choose hypertext servers for UNIX, Macintosh, and PC platforms**
- **Construct "cgi-bin" programs to create sophisticated interactive documents**
- **Access and use all the browsers, editors, translators, and other valuable World Wide Web tools currently available online**

The HTML Sourcebook also features an array of examples taken from business, education, science, and other fields. Including both code and screen captures of the resulting pages as they appear on the Web, the examples let you see firsthand what well-designed hypertext documents and their underlying HTML structures look like.

Whether you want to deliver online documentation for software packages, distribute sales catalog information, access library databases, or expand the audience for a hypertext comic strip, *The HTML Sourcebook* is your complete guide to composing and distributing documents on the World Wide Web.

Ian S. Graham, Ph.D., is an instructional technology specialist with the Division of Computing and Communications at the University of Toronto.

Cover Design: Adrienne Weiss
Cover Illustration: Art Resource
Porto di Classe, S. Apollinare Nuovo Ravenna

John Wiley & Sons, Inc.
Professional, Reference and Trade Group
605 Third Avenue, New York, N.Y. 10158-0012
New York • Chichester • Brisbane • Toronto • Singapore

ISBN 0-471-11849-4

52995

9 780471 118497



GRAHAM

THE HTML SOURCEBOOK

WILEY