Using HP 3000 Systems with the World Wide Web

George Stachnik
Hewlett Packard

Oh no! It can't be! Please tell me it's not another paper about the World Wide Web!

Yes it is another paper about the Web. But don't stop reading yet - because this one is just a little bit different. If you're like most folks, you've started to get just a little bit sick of hearing about the Web. You've been subjected to dozens of papers and seminars on the subject. You've been preached at and lectured to in the "gee-whizziest" tones about the glories of the far-flung Internet future, when the Web will be the tool that we will be using to do everything from raiding our bank accounts to paying our bills to ordering out for pizza.

Well, this paper isn't going to be another sermon. Instead, we're going to hard nosed look at the Web today, and at the technologies that work today. We'll look at what you can do with the Web using today's technology. And because this kind of information tends to have a very short half-life, we'll also give you a number of addresses on the Web that you can visit with your Web browser so you can stay up to date. You can use these addresses to chase down more detailed information about any products and technologies in this paper that may interest you.

A year ago, if you wanted to be a Web-expert, there were no more than a few pieces software to learn about. Web technology was simple, and better still, it was free. Most web software was "shareware" - downloadable from the Internet at no cost. The Web began as a relatively simple technology intended to solve relatively simple problems. It was originally intended to be used in academia. In the 1990s, professors in every field from Anthropology to Zoology were learning to use the Internet to circulate technical papers and other publications. But most scientific papers contained many references to other papers, and chasing down those references could be tedious and difficult.

"Wouldn't it be nice," they reasoned, "if there were an easy way to link referenced papers to one another - so that a student reading a paper could simply click on the reference and automatically pull up the referenced publication?" Web technology was originally created to provide college students and professors with exactly these capabilities. A technical paper could be placed on the Web and viewed from a desktop PC or UNIX workstation, and the reader could run after references with one click of a mouse button.

The Web's dramatic growth period began when people began to use it as a business tool. After all, a technology that could be used to publish technical papers could also be used to publish product information, marketing brochures and so forth. And from there it seemed a small step to electronic commerce, (or e-commerce, as it's become known). Usage of the Web literally exploded. Most experts agree that in 1996, the number of people with Web access will exceed 16 million. By the end of 1997, that number could easily double, meaning that 25% of U.S. households will be "surfing". By the end of the century the Web's penetration into American households could approach that of televisions and VCRs.

There's no denying that Web technology has been enormously successful. It's also true that success can sometimes be the worst thing that can happen to a simple, useful technology. Today the Web is beginning to look like Frankenstein's monster as software vendors all over the world add their own pieces to the puzzle in an effort to cash in on the Web's success, and to broaden its appeal. Today, there seems to be no end to the list of new Web-related technologies and products hitting the market on a daily basis.

All this has added greatly to the complexity of the Web, in spite of the fact that the marketing people have been trying to maintain at least an appearance of simplicity by giving their products names that are more likely to appeal to the consumer market. Think about it - ten years ago the product we all know as "Netscape Navigator" would have entered the marketplace saddled with a moniker like: "Hypertext Internet Client Engine", and the biggest argument surrounding the technology would be whether HICE should be pronounced so that it rhymes with "nice" or with "hike".

Today the debate that surrounds the Web is more practical. What Web technologies work? Which are fastest? Which ones do I use to link my Web pages with databases like IMAGE/SQL on the HP 3000 or Sybase or Oracle on the HP 9000? Which are most (and least) secure? Which are the "flashiest", making it easy to incorporate features such as full motion video and animation? These are questions we'll explore in this paper.

Web Browsers

Let's begin with the software that lies at the heart of the Web experience: your browser. (OK - I know you already know what a browser is. You can skip ahead a paragraph or two if you really want to, but if I were you I'd stick around. You never know when you might learn something unexpectedly.)

A Web browser is a piece of software that reaches out across a network and retrieves pages of information. Browsers can work with your office's local area network, or they can work with the public Internet, or any sized network in between. To truly understand browsers, there are a couple of basic terms you need to understand.

- Browsers and servers are able to communicate with one another because they both use a common set of rules called Hyper Text Transfer Protocol (HTTP). The rules of HTTP make it possible for the browser to retrieve text and graphics from other computers called Web Servers, and to follow the links that tie those pages to one another (thus forming a "Web").
- Each page on the Web is identified by a special Web address called a Uniform Resource Locator, or URL. Most URLs start with the characters: "**http://**" - which means that the page addressed by the URL contains hypertext. For example, the character string "**http://www.hp.com**"[1] is the Web address of "Access HP", Hewlett-Packard's flagship Web page. Some URLs start with the characters "**ftp://**", which means that the page addressed by the URL contains files that can be transfserred to the client using file transfer protocol (ftp).

- The language that browsers and servers use to talk to one another is called the HyperText Markup Language, or HTML

Web users don't usually need to worry much about how their browsers work. All they know is that when they click on the Browser Icon, information is somehow magically retrieved from the Internet and displayed on their screen. But what if you're planning on running a Web server of your own? Then you need to understand a little of how browsers use the rules of HTTP to cooperate with Web servers. (See page 1: "**HTTP, Web Servers and HTML**").

Today's Web "surfers" have many browsers to choose from, including Netscape's Navigator and Microsoft's Internet Explorer. One of the first Web browsers to become commonly available is a program called "Mosaic". Mosaic was originally written for UNIX workstations, but it has been ported to a variety of desktop platforms including MS/Windows. It is still still being distributed today as shareware. The National Center for Supercomputing Applications (NCSA) is an organization that was heavily involved in the development of Mosaic and the Web. You can download a copy of Mosaic for MS/Windows from NCSA's Webite at **http://www.ncsa.uiuc.edu/SDG/Software/WinMosaic/HomePage.html**.

Is One Browser As Good As Another?

All Web browsers are based on HTTP and HTML, and they all perform fundamentally the same job, but that doesn't make them equivalent to one another. Web technology is changing constantly. The HTML language is constantly being enhanced. And when one company brings a new technology to the table, you can't assume that every browser on the market is going to support it.

Java, Shockwave and framing are three examples of technologies that serve to differentiate browsers from one another. Java is an object oriented programming language developed at Sun Microsystems that can be used to add certain kinds of animation to Web pages. ShockWave is a technology from a company called Macromedia that is used for similar purposes. If your browser supports Java, then when you use it to look at a Web page that contains Java programming, you'll see Java animation. If your browser is not "Java-enabled", then you'll see static images in its place. Shockwave works much the same way. For more information on Java, read the "Frequently Asked Questions (FAQ)" file found at this URL:
HTTP://www-net.com/java/faq/faq-java.txt

Framing is another example of a technology that is not supported by all browsers. HTML framing statements can be used to divide a Web page into areas called frames. These frames can be handled independently by the browser. So for example, imagine a Web server that contains maps of the City of Chicago and its suburbs. The first Web page that you see when you access this site (the "home page") might be divided into two frames, with one frame containing a map of the City, and the other frame containing an index of the names of the suburbs. If the user selects a suburb (say, Naperville) from the index frame, a map of Naperville will

replace the map of Chicago in the map frame. But the frame containing the index remains on the screen, unchanged. So if you change your mind and decide you want to view a different suburb, you can select it from the index without having to return to the the home page. Both frames will remain on the screen at all times, but only ***if*** your browser supports framing. If your browser does ***not*** support framing, then selecting Naperville from the index will simply cause your browser to jump to the Naperville Web page. But without framing, the map of Naperville will replace not only the map of Chicago, but also the index of its suburbs. If you change your mind and decide you want to look at a map of another suburb (say Oak Park), you must first go back to the Chicago home page to display the index. Then you may choose Oak Park from that list.

In other words, when a user looks at a web page, what the user sees depends to some degree on what browser he or she is using. Framing, Java and ShockWave represent differences between browsers that affect what users see on the screen. Today, NetScape/Navigator 2.0 supports all three of these technologies. Other browsers may or may not. Some shareware browsers probably never will. NetScape/Navigator can be found at the following URL:
http://home.netscape.com/

So far, the differences between browsers that we've seen have been mostly cosmetic. At worst, a user might not see an animation, or have to bounce around a few extra pages to find the desired information. There are other differences that may be more important. As people move into electronic commerce there is an increasing level of interest in security and encryption. When data is passed between a Web server and a browser, the data can be encrypted before it is transmitted across the network. This is particularly important for electronic commerce applications, which require the transmission of sensitive pieces of information such as credit card numbers. But in order for this to work, the browser and the server must use the same encryption algorithm. Today, there are multiple "standards" for data encryption, and it's not clear which one will eventually be accepted. (See page 1 **Encryption and Security Protocols**)

These are differences that should be taken into account when selecting a browser, whether it's for personal or business use. Today, shareware browsers such as Mosaic represent the least common denominator of Web technology. Commercial products provide a richer level of functionality. Today's most popular commercial browsers include Netscape's Navigator, Microsoft's Internet Explorer, and the browsers provided by the various Internet service providers. Each of these have unique features that differentiate them from the others, and especially from Shareware.

Keeping up to date with changes in Web technology can be daunting. Web technology is changing very quickly, and books and papers that explain how it works (including this one) can become obsolete very quickly. For this reason, the Web iteself is one of your best sources for Information about the Web. For general information about the Web and

new technologies that relate to it, point your Web browser at the following URL:
http://www.w3.org/hypertext/WWW/TheProject.html.
This site will keep you up to date on the latest technical developments relating to the Web.

Web Server Software

Virtually any computer can act as a Web Server as long as it is connected to a network and is running a Web server "daemon". In general, "daemons" are processes that run on servers independent of a user. On MPE/iX systems, the closest things to daemons are system processes such as the spooler. A Web server "daemon" is a program that accepts and responds to requests from Web browsers.

The first Web server programs to become commonly available were distributed as shareware, much like the Mosaic browser. Shareware server software can still be downloaded from the Web at no charge . For example, you can retrieve source code for a Web Server Daemon program by pointing your browser at this URL:
http://www.w3.org/pub/WWW/Daemon/

This software was written by NCSA for use on UNIX servers, including the HP 9000. Table 1 shows a list of some of the operating systems to which the NCSA software has been successfully ported. The daemon software has also been successfully ported to many other platforms, including the HP 3000. The MPE/iX based version of this softwarwe can be downloaded from the HP 3000 website. It's URL is:
http://jazz.external.hp.com.

H/W Vendor or Platform	O/S	Version	Support for NCSA Server Software?
IBM	AIX	3.2	YES
HP	HP-UX (SNAKES)	9.0	YES
SGI	IRIX	5.2	YES
PC	LINUX	1.2.10	YES
NEXT, PC, HP	NeXTSTEP	3.3	YES
DEC	OSF/1	3.0	YES
SUN	SOLARIS	2.3	YES
INTEL	SOLARIS	2.4	YES
SUN	SunOS	4.1.3	YES
DEC	ULTRIX	4.3	YES
PC	SCO	3.0/3.2	YES
Intergraph	Clipper	C300/C400	YES
DEC	VMS		Independent Distribution

Table 1

If you have no experience with the Web, (and no budget!), then the NCSA shareware might be a good way for you to get your "feet wet". But keep in mind that this software is shareware: there is no support for it available

from HP. If you plan on running a Web site for commercial purposes, you probably will not want to use the shareware as anything other than a learning tool.

Fortunately, there are commercial server packages packages available from a variety of companies, including Open Market, Netscape and Microsoft, among others. As is the case with browsers, the commercial server packages provide functionality (security, encryption, support for languages such as Java, etc) that may not available with the shareware. Open Market provides commercial Web Server software products for the HP 3000. Table 2 shows some URL's to visit for more information about commercial Web Server software:

Open Market	http://www.openmarket.com
Microsoft	http://www.microsoft.com/infoserv
Netscape	http://home.mcom.com/comprod/server_central/index.html

Table 2 - URLs for Commercial Web Servers

Creating Web Pages

When a browser accesses a server, it displays a page of text and graphics. Each server can hold multiple pages (many have hundreds or even thousands of them). The rules of HTTP are used to link pages together, which is how your browser can jump from one page to another. The administrator of a Web Server, (sometimes called a "WebMaster"), may link a server's pages to one another, or to pages on other servers. The linking is defined in HTML using "anchors". (See page 1 "**Anchors"**).

The term "World Wide Web" was chosen to denote the fact that although the pages making up the Web are linked together, they are not necessarily linked in an orderly fashion - (otherwise I suppose the Web might have been named something like "the grid"). The pages and anchors that hold the Web together do not follow any pattern beyond the whims of the WebMasters. You could find information on the Web by following the anchors from page to page, from server to server, until you happened upon what you were looking for. This would be a tedious task. Fortunately, there's an easier way. Responding to an obvious need, a number of Web servers have been set up whose sole purpose is indexing and cross referencing other servers. Today, the busiest Web servers on the Internet include Netscape's Net Search page (**http://home.netscape.com/home/internet-search.html**), Yahoo (**http://www.yahoo.com/**) and WebCrawler, (**http://webcrawler.com/**) the three largest servers devoted to helping people find other Web servers.

Designing web pages is as much an art as a science. The HTML language provides a rich set of capabilities for incorporating graphics, animation, and other capabilities. As you design your web pages, however, you want to be selective about which of these capabilities you use. Graphics and animation take up a lot of bandwidth on the network that you're using, and many users are already joking that "WWW" used to stand for "World Wide Web", but now it stands for "World Wide Wait". If you're going to incorporate graphics or animation in a Web page, you want to do it

sparingly to minimize the wait. One way to do this is to use compressed bitmapped files whenever possible.

There are multiple ways to store bitmapped images on computers, and some ways offer more compression than others. Typically, the filename extension, (the characters at the end of the filename following the period) indicates how a bitmapped image was stored, and how it is to be decoded. For example, HP sells scanners, which are devices that are used to make digital copies of printed material or photographs. The digital files produced by scanners are typically "Tag Image Files" or TIF files. TIF files are not compressed, and as a result they can very large. The exact size of a TIF file depends on the size of the image being scanned, and the amount of detail you choose to include. A color snapshot can easily produce a file of five to ten million bytes! This is too large to be practical for most Web applications, so the TIF files produced by most scanners are rarely found on the Web. Instead, they are compressed before they are incorporated into Web pages.

In the 1980s, CompuServe developed a format for bitmapped images called the "Graphics Interchange Format (GIF)". By converting a TIF file to a GIF file, the size of the file can be reduced dramatically with no apparent loss in image quality. This is highly desirable for files that are going to be transmitted across a network.

An even more compressed bitmapped file format is the JPG format. When TIFs or GIFs are converted to JPGs, the amount of compression can be varied. Specifying a low level of compression results in little or no loss of image quality, but the resulting JPG file will be relatively large (approaching the size of the corresponding GIF file). Specifying a high level of compression results in a much smaller file, but with a loss of detail in the image. There are a variety of packages that can be used to convert bitmapped data from one format to another. One such package is LVIEW, available as Shareware from

MMedia Research
1501 East Hallandale Beach Boulevard, #254
Hallandale, FL 33009
USA

Today, most Web browsers are capable of displaying bitmapped data in the GIF and JPG formats. If your web page contains graphic data in other formats, the graphics files can still be viewed using a "helper application". For example, the Paintbrush program used by Microsoft Windows can be used to view bitmapped images in files ending with the extension .BMP. BMP files can be incorporated into Web pages. If your browser does not support BMP files, then viewing the page will simply cause the BMP file to be downloaded to your computer, where the browser will execute a "helper application", (such as Microsoft's Paintbrush program), to view it.

Similarly, the Web can be used in this way to distribute files that were created with commercial software packages such as MS/Word, Lotus's Freelance or Microsoft's PowerPoint. The important thing to understand

is that if the Web browser does not contain logic to view a file, then it spawns a process that runs the appropriate application software for viewing it. This only works, of course, as long as the user has a license to use the helper application on the client computer.

This technique has also been used to make full motion video images available to web users. When a video image is incorporated into a Web page, viewing the web page can cause the video image file to be downloaded to the client computer. There, a helper application will display the video clip on the screen. Today, there are at least three separate ways of digitally storing moving video images on a computer.

1. Video clips that are stored in Apple's QuickTime format typically use a filename extension of MOV. QuickTime software for displaying MOV files is available for Apple Macintosh computers as well as for MS/Windows and most UNIX workstations.
2. Microsoft has a format of it's own for video images. The extension used is AVI (for Audio Visual Information). AVI files can be played using the Media Player supplied with MS/Windows, as long as the appropriate drivers are in place.
3. Finally the MPG format is perhaps the most highly compressed of the three. MPG software is available for most UNIX workstations, as well as for MS/Windows.

There is a good deal of work going on to allow full motion video without the use of helper applications. Future enhancements to Java and Shockwave might make this possible. But at this writing, video images are being distributed through the use of helper applications, many of which are available on the Web at no charge.

Data Base Integration

We've seen how the HTML language can be used to display simple text and graphics on a browser, and to link pages of text and graphics together. Inevitably, people wanted to use HTML to do more. The Common Gateway Interface, or CGI, is a standard that allows Web servers to do much more than simply display static pages of text and graphics. CGI scripts can be used to allow a Web Server to interface with databases and with other computers.

For example, using CGI scripts, an HTML page can be set up to display forms on a browser, prompting the user for database information and passwords. When the user fills out the form and transmits it back to the server, the server uses the information to access a database and transmit the results back to the user. Without CGI scripts, HTML pages are static. Anybody who accesses a Web page will see precisely the same information. With CGI scripts, HTML pages can be made dynamic. So each user who accesses an HTML page will have a different experience.

CGI scripts present the WebMaster with some unique system administration problems. When a browser sends a request to a Web server, the user running the browser does not log on to the server in the traditional sense of the word. For example, suppose you set up an HP

3000 to be used as a Web Server. The Web server daemon proces (typically executing under the control of a batch job) receives requests from browsers and responds to them. The user at the browser doesn't have to enter a password or logon to the HP 3000 system in any way. (The same is true of browsers running on UNIX or other operating systems). By running the Web server daemon, you are implicitly granting anybody with a Web browser and network access permission to access the Web pages on your system.

A CGI script is basicly a link between an HTML page and a program that's located on the Web server. CGI programs can be written in C, C++ or Perl. When you put HTML pages with CGI programs on a Web Server, you are implicitly granting Web users permission to run these programs.

In other words, CGI scripts allow people to run programs on your system without ever logging on! The security and performance implications of using CGI scripts are serious. (See page 1 **Database Integration, CGI scripts and Stateless Applications**). It's critical that the system administrator maintain tight control over any Web-accessable software. CGI programs must be stored in a single directory usually /cgi-bin. The system administrator should make it a point to maintain strict controls over access to this directory. It's also important that any sensitive data (whether it's user data from a database or database passwords) be encrypted before it is transmitted across the network. Encryption is a feature of many of the commercial Web server packages. But it's important to ensure that the encryption features of your server are also supported by whatever browser you are using.

Since database integration is such a critical part of the future growth of the Web, there are a number of technologies being developed to make it easier and more manageable than today's CGI scripting allows. For example, there is a Web-based 4GL called Autobahn available from Speedware, which solves many of the problems associated with CGI scripts. Also, there are extensions to Java being developed which may address many of these issues.

Internet vs Intranet

All the tools and technologies that we've discussed can be used to make applications available to the general public using the Internet. It's possible to work directly with the federal government, and have high speed T1 communications lines run directly to your site. But many companies have found that a more cost effective option for making systems available to public Internet users is to use an Internet service provider. These are companies that assume the responsibility for the physical link to the Internet. There are many such companies incorporating today. The following URL can be referenced to view a list of Internet Service providers (ISPs):
http://www.thelist.com

Users who are thinking about using an internet service provider should evaluate them carefully, checking with reference accounts to ensure that their service is reliable and that their network bandwidth is adequate.

One of the fastest growing areas for Internet applications is the use of so-called “Intranet” applications. An intranet application is one that is based on the same technologies used on the Internet, but which is entirely contained on a company’s internal network. Many companies have found that Web browsers can be used internally for the distribution of information and literature. Company phone books, marketing brochures and technical documents can all be kept up to date easily by distributing them via internal HTML documents.

HTTP, Web Servers and HTML.

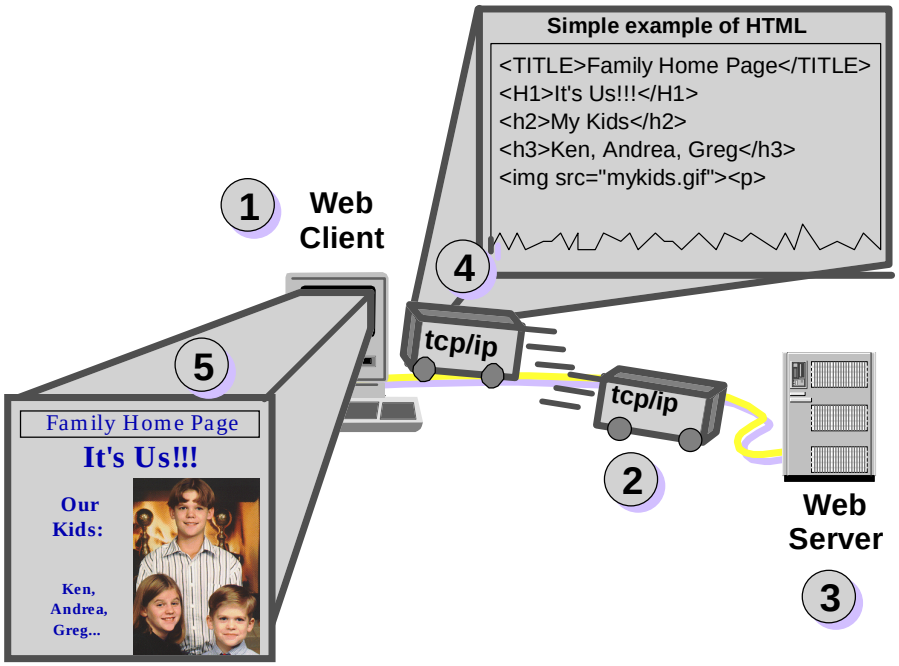


Figure 1

The World Wide Web is possible because of two pieces of software. One of them is called a Web browser, and it typically runs on a desktop computer. (Actually, there are very simple browsers that can be used from ASCII terminals, but they have no graphics capabilities and are therefore of little use on today's Web). When you run a browser on a UNIX Workstation or Personal Computer, the desktop computer becomes a Web client (1), which is to say you can use it to display the text and graphical information stored on Web servers.

Most browsers have a window near the top of the screen where you can enter the URL address of the Web page that you want to access. When you specify an address in this window, it is transmitted across the network using Internet Protocol (TCP/IP). TCP/IP (2) is a network transport mechanism that can be used on virtually any network - from your office's local area network (LAN), to the public internet. So the server that you are requesting can be a machine in your own company, or it can be a machine on a public network located anywhere in the world.

When your request arrives at the Web server (3), the server responds by sending back a string of text. This text is encoded using a language called Hyper Text Markup Language (HTML). The HTML language describes the layout and contents of the Web page. It's not our intent to teach the whole HTML language in this paper, but a brief explanation of the elements found in the above example (4) will serve to show that HTML is a language that's relatively easy to understand and learn.

The sample HTML code shown defines this Web page's title, three headers and a bitmapped photograph. The first line of the example reads:
<TITLE> Family Home Page </TITLE>

The character strings <TITLE> and </TITLE> are used to mark the beginning and end of the title of this page. When this page is displayed using a browser, the words "Family Home Page" will be displayed in the bar at the top of the browser's window on the web client.

The next line in the example defines the first header. It reads:
<H1>It's Us!!!</H1>

The designator "H1" indicates that this is a level one header - the largest used with HTML. The words "It's Us" will be displayed in this level one header. The next two lines define a level two and level three header respectively.
<H2>My Kids</H2>
<H3>Ken, Andrea and Greg</H3>
The level two header contains the words "My Kids" and the level three header contains the words "Ken Andrea and Greg". The next part of this example is an "image source". It reads this way:
<img src="mykids.gif">

This indicates that there is a file called "mykids.gif" stored on the server. This file contains a bitmapped image which is to be displayed on the screen. This line of HTML code will cause this file to automatically be downloaded from the Web Server. The browser will display the image as part of the web page. The last line of this example reads:
<p>

This indicates the end of a paragraph. In the HTML language, you need to use <p> statements to format the text that's going to be displayed by the browser. If you want a blank line to appear in the middle of a screen of text when it's displayed by the browser, you must use a <p> to force a paragraph break. If you insert blank lines (carriage returns) in the middle of the text in your HTML code, they will simply be ignored. When the text is displayed by the browser, the text before the blank line will be displayed right next to the text after the blank line. The browser will word-wrap the text into a single paragraph. The length of each line will depend on the size of the window being used to run the browser. Re-size the browser window, and the text will automatically be rewrapped. If you want the browser to skip a line at a particular point, then you must use a paragraph break ("<p>").

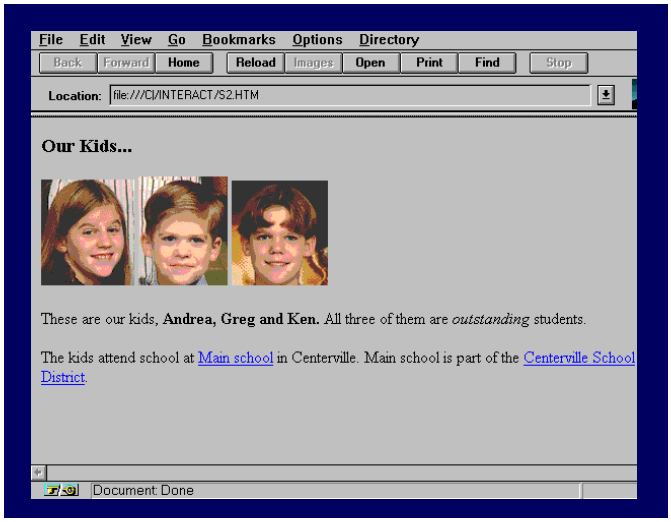
When the html code and the graphics file shown in the figure arrive at the browser (5), the browser will display them together. The details of the format of the displayed Web page will depend on the size of the window and the limitations of the browser being used. .

More information on HTML is available from a variety of sources, but the Web itself is probably your best source for information. Use your Web browser to visit this address for general information on HTML:
http://info.cern.ch/hypertext/WWW/Protocols/HTTP/HTTP2.html

Information on the technical specifications of HTML can be found at:
http://www.w3.org//hypertext/WWW/MarkUp/MarkUp.htm

A tutorial on the use of HTML is available at:
http://web66.coled.umn.edu/Cookbook/HTML/MinutePrimer.html

Anchors




```
<H3>Our Kids...</H3>
<IMG SRC="andi.gif">
<IMG SRC="greg.gif">
<IMG SRC="ken.gif">
<P> These are our kids, <B>Andrea, Greg and Ken.</B>
All three of them are <I>outstanding</I> students.</P>

<P>The kids attend school at
<A HREF="main.html">Main school</A> in
Centerville.

Main school is part of the
<A HREF="http://centerville.edu">Centerville School
District</A>.</P>
```

Figure 2

The HTML language can be used to display text in a variety of ways. In Figure 2 above, the HTML language on the right results in the Web page displayed on the left. The words "Andrea, Gregand Ken" are displayed in boldface on the Web page because they are sandwiched between the "<B>" and "</B>" in the HTML example. Similarly, the word "outstanding" is italicized in the web page because the HTML code has it surrounded by the "<I>" and "</I>".

Text may also appear to be displayed with enhancements because it's used to define an anchor. Anchors are the "glue" that hold the Web together. An anchor is typically displayed on the browser screen as a few words of underscored text. Pointing to an anchor with the mouse, a user can jump to a different web page simply by clicking the button.

The web page shown on the left in figure 2 contains two anchors. The words "Main School" and "Centerville School District" are each associated with other web pages. The box at the right of figure 2 shows some of the HTML statements that make up the Web page on the left. The first anchor references a Web page that's on the same server as the current page. It is defined in the following line of code:

<A HREF="main.html">Main school</A>

This indicates that the words “Main school” will be displayed on the screen underscored, so that the user will see that these two words represent an anchor. If the user selects this anchor with the mouse, the browser will automatically send a request to the server, asking to see the Web page defined in the file named **main.html**.

To define an anchor that references a Web page that’s on a different server, you use an HTML statement much like the following example:
<A HREF="http://centerville.edu">Centerville School District</A>.

This statement displays the words “Centerville School District” in the underscored text associated with anchors. But if the user clicks on this anchor, the browser will send a request to the server at this Web address: “http://centerville.edu”. The only difference between this statement and the previous one is the format of the address.

Anchors can be defined as text items, as in these examples. But other elements, such as pictures, can also be used.

CGI Scripts, Database Integration and Stateless Applications

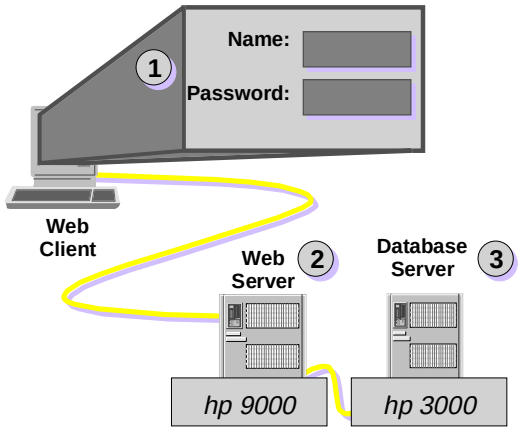


Figure 3

Many companies are beginning to investigate the use of the Web as a "front end" to legacy applications based on existing databases. As shown in figure 3, this typically involves the use of two servers - a Web Server, which runs the Web daemon, and a database server, where the database resides. The figure shows an HP 3000 which is being used to run an application that is based on an IMAGE/SQL database. This HP 3000 is networked with an HP 9000 which is being used as a Web server. Web-based transaction begin with Web clients (browsers). The browsers pass requests to the Web server, which in turn interacts with the Database server. This configuration is typical, but by no means is it unique.

For example, the Web server could just as well be an HP 3000 using Open Market's Server software. The Database server could have just an easily been an HP 9000 running an Oracle or Sybase application. Or a single machine could be used for both purposes.

The interaction between database server and Web server is managed through the use of CGI scripts and programs. There are a number of issues raised by the use of CGI scripts. Many of these issues are related to the fact that the Web is a "*Stateless*" application. Most traditional database applications are not stateless. This means that when the user logs onto the system, he gains access by means of a user-id and password. Users are in one of two states - they're either logged on or they're not. Similarly, when a user gains access to a database by opening it, (typically supplying a database password), they maintain access until they close the database. Again their in one of two states - they either have the database open or they don't.

The Web, by contrast is a stateless application. This means that every interaction between browser and server is independent. The browser doesn't "log on" to the server. For example, when a browser sends a request to a Web Server, there is no password or logon session established. The Web daemon running on the server responds to the request, and when it's finished, the connection to the browser is closed. The transaction is forgotten. Each dialog between browser (client) and daemon (server) is completely independent of prior or subsequent transactions.

If you're going to use CGI scripts to build a Web-based application, the stateless nature of the web can present some problems. Suppose a browser is used to display a form (1) on a Web client. The form provides places to fill out the database name and password.. The request arrives at the Web Server (2), which uses a CGI script to pass the information provided by the user to a CGI program. This program opens a database on the database server (3). If the information provided by the user is correct, then the database is opened successfully. CGI programs cause a new Web page to be transmitted back to the browser to inform the user of this event. However, as soon as this new page is sent back to the browser, the rules of HTTP state that the transaction is complete. The connection between the browser and the Web server is broken, and the Web Server maintains no memory of the transaction.

This is a problem because the user has not yet retrieved any data from the database. Typically, a database user will now want to provide the Web server with some kind of key value, which will then be passed to the database server to retrieve a particular record (or records) from the database. Since the database has already been closed, many of today's Web-based database applications will actually open and close the database with every transaction. This has serious implications for performance and security. The performance questions arise because opening and closing a database usually has a lot of overhead associated with it. And the security questions stem from the fact that sensitive database names and passwords must therefore be passed back and forth with each transaction.

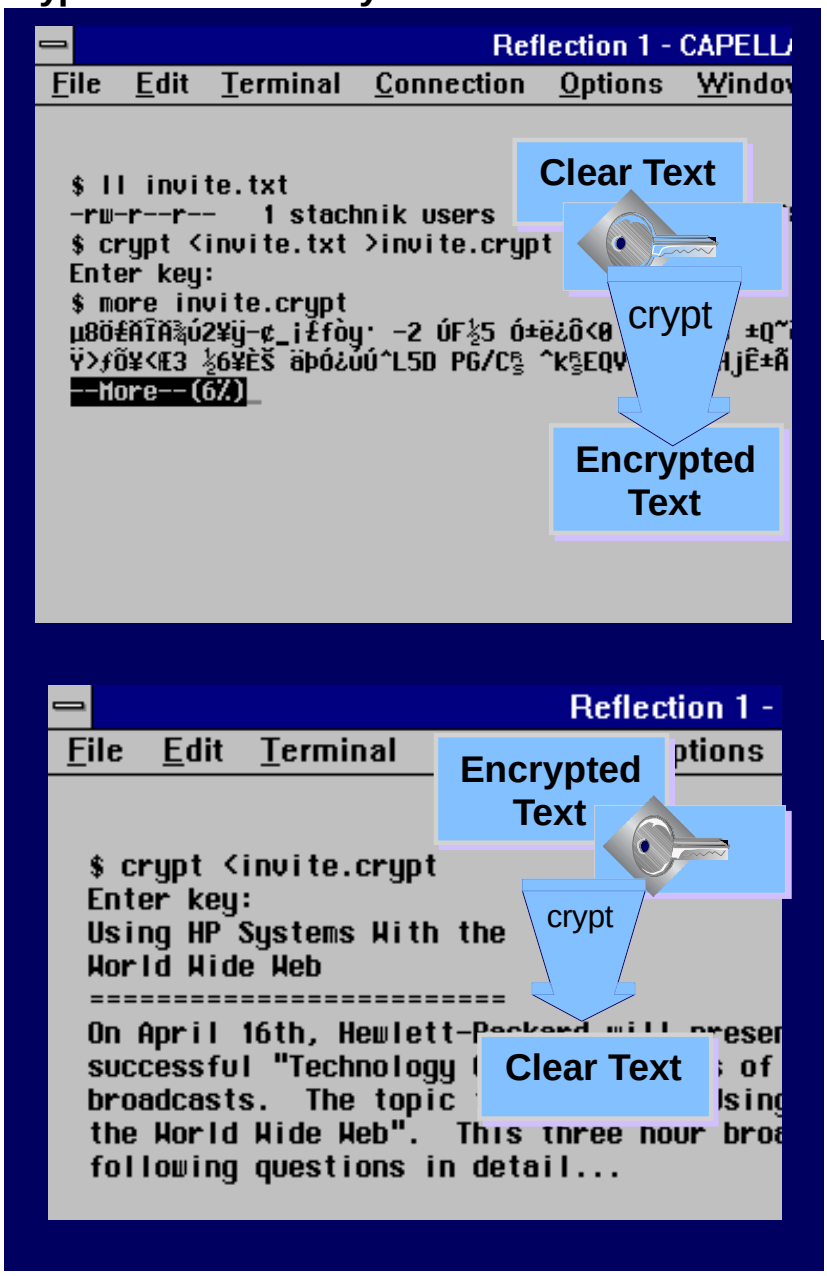


Figure 4 - Encryption using Secret Keys

Encryption is a key technology for commercial use of the web. Basicly, there are two kinds of encryption commonly used with Web applications. The first involves the use of a secret key, and is illustrated in figure 4. The second involves the use of public and private keys, and is illustrated in figure 5.

Text which is not encrypted in any fashion is sometimes referred to as “clear text”. There are many algorithms that can be used to encrypt clear text. As kids many of us played with “decoder rings” which simply substituted different characters for one another. This substitution algorithms are very easy to break. In fact, any encryption algorithm that has

the clear text to be encrypted as its only input can be broken. For this reason, the encryption algorithms typically used with the web have two inputs: the clear text to be encrypted, and a second character string called a “secret key”.

The use of secret keys can be demonstrated using the crypt command, which is available on most UNIX systems (including the HP 9000). Suppose you have a file called “invite.txt”, which contains some ASCII text. This file can be encrypted using the crypt command. The file, invite.txt, is used as input to the command. The output is directed to another file, called invite.crypt (see figure 4, left). The crypt command will prompt the user for a key. This can be any character string. The crypt command is also used to decrypt the file by simply reversing the process. The input of the crypt command is redirected to reference the encrypted file (invite.crypt) (see figure 4, right). Once again, the user is prompted for a key. The same character string that was used to encrypt the file must be specified to decrypt the file.

The use of secret keys results in an encryption algorithm that is very difficult to break. But it may not be practical for Web applications. Secret keys depend on an agreement between the client and the server to use the same key. If the client and server “meet” on the internet for a single transaction (to order a pizza, for example), there may be no opportunity to establish an agreed upon key. Fortunately, there’s another way. It involves the use of public and private keys.

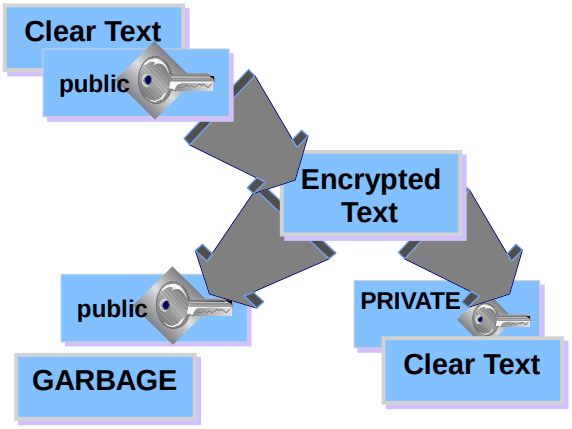


Figure 5 - Public and Private Keys

Figure 5 shows how public and private keys are used to encrypt and decrypt information. In this scheme, information is encrypted and decrypted using keys, but the key that's used to encrypt the data is different from the key that's used to decrypt it. An example will serve to show how this can be useful.

Suppose a bank starts to allow its customers to use the Web to access their accounts. They might provide their customers with a browser, and a public encryption key. This key (which could even be hard coded inside the browser) would be used to encrypt information passed from the browsers to the server (located at the bank). The public key can be used to encrypt information, but as shown in figure 5, it cannot be used to decrypt it.

Therefore it doesn't matter if an unauthorized person finds out what the public key is, because all the banks customers use the same key, and it cannot be used to decrypt anything. When the encrypted messages arrive at the bank's server, they are decrypted using a second key called a private key. This key is known only to the bank, and must be protected from unauthorized persons.

When the bank is ready to send information back to its customers, it can be encrypted using the private key, and decrypted using the public key (exactly the reverse of the process that the customers used to send encrypted messages to the bank). Messages encrypted using the private key are not very secure, because they can be decrypted using the public key. And the public key is hardly a secret; it's in the hands of all the bank's customers. Therfore this kind of encryption allows customers to send secure messages to the bank - ("My account number is 12345, and my password is ABCDE"). But the bank must assume that its replies could be intercepted and read by unauthorized persons. Therefore the replies should not contain any sensitive information. (That is, the bank should reply "Your password is correct", not "ABCDE is the correct password.")

These encryption techniques are used in a variety of ways to ensure that the data transmitted across the Web is safe from prying eyes. Perhaps the most common security protocols are the Secure Sockets Layer (SSL) and Secure HTTP (S-HTTP). SSL is a transport level or session encryption protocol which enables a secure channel in which messages are encrypted. This provides private and reliable communication between client and server, and server authentication. S-HTTP is a document level protocol to provide security of documents on the WWW. S-HTTP provides the capability to authenticate clients and servers, support digital signatures, negotiate security levels based on application needs and provide secure communication through existing firewalls. Digital signatures provide message authenticity (eg: entire e-mail msg) using a sophisticated hash algorithm based on the sender's private key. There are other security protocols including the "Pretty Good Privacy" program (PGP), Smart Cards and Kerberos. These protocols are not

interchangable, (so, for example, a server using S-HTTP will not work properly with a browser using SSL). At this time, no single security protocol has achieved the level of acceptance required to make it a defacto standard.

It's important to investigate security protocols thoroughly before investing in a Web server package for use with commercial applications, or before standardizing on a browser for use in a large enterprise.