WHY JUST SURF THE NET WHEN YOU CAN MAKE WAVES?

BECOMING A WORLD WIDE WEB SERVER EXPERT

A LITA Regional Institute
http://sunsite.berkeley.edu/~emorgan/waves/
version 1.4

by

ERIC LEASE MORGAN

http://www.lib.ncsu.edu/staff/morgan/
eric_morgan@ncsu.edu

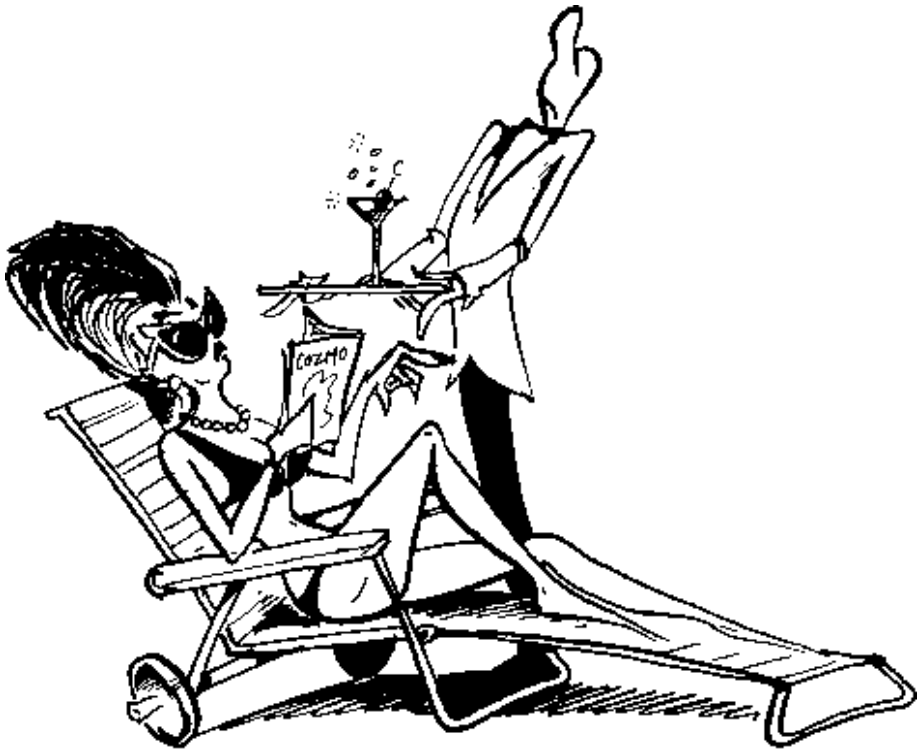


Table of contents

Preface

Through a series of presentations, demonstrations, group exercises, handouts, and video interviews, this one-day workshop will address the issues surrounding the initial development and ongoing maintenance of useful World Wide Web (WWW) servers.

Topics

Specifically, the workshop will focus on five major WWW server topics:

- Hardware and software
- Qualities of useful servers
- Content
- Server maintenance
- Staffing

Emphasis

In an effort to appeal to the widest audience and provide the foundation for individual growth, principles and guidelines will be emphasized over platform-dependent functions and features.

Who Should Attend

The workshop is intended for those considering or beginning the implementation of a WWW server. The only prerequisites are knowledge of basic computer operations (creating, deleting, and moving files and directories), a sense of curiosity, and a willingness to learn.

Acknowledgements

Some thank yous.

Few things of substance are accomplished without the cooperation of others. This workshop is no exception. First I would would like to acknowledge **Mark Beatty** and the people of the LITA Regional Institute. It was these folks' ideas that got the the whole thing started.

Thank you to **Jacqueline Zelman** and **Sulaiman Paperwalla** of **Florida International University**, the hosts of the second "Making Waves" workshop.

Thank you to **Berna Heyman** and the **College of William & Mary** who graciously hosted the first workshop.

Of course none of this wouldn't have happened without team LITA and specifically **Jacqueline Mundell** and **Elizabeth Cooper**.

Linda Van De Zande, who knows very litte about HTML, inspired the layout for the workshop's handouts. **David Cherry**, a childhood friend, is the artist whose illustrations grace the handouts and provide comic relief.

Thank you goes to **Nathan Turajski** of **Connectix** who sponsored the "virtual" attendence of **Roy Tennant** and **Jean Armour Polly** through the gift of digital cameras during the very first workshop in Williamsburg.

Last, but certainly not least, I would like to thank my wife, **Mary Grey**, who has been patient above and beyond the call of duty while I slavishly type away night after night after night...

Presenter

The workshop will be lead by Eric Lease Morgan, a systems librarian from the North Carolina State University Libraries. "I consider myself a librarian first and a computer user second whose professional goal is to discover new ways to use computers to provide better library services."

An Internet user since 1989, Eric has given dozens of presentations, written more than a few articles, and published two books (*WAIS and Gopher Servers* and *Teaching a New Dog Old Tricks: A Macintosh-based World Wide Web Starter Kit Featuring MacHTTP and Other Tools*).

Eric's home page is located at <URL:http:// www.lib.ncsu.edu/staff/morgan/>.

Introduction

These sections outline a recent history of the World Wide and a simple description of how much of the Internet is "put together."

Welcome! This one-day workshop will give you an overview of what it means to be a World Wide Web server administrator. It covers:

- fundamental hardware and software issues
- specific server software for Unix, Windows, and Macintosh computers,
- access control issues
- log file analysis and URL integrity
- design issues for creating useful information systems
- the basics of CGI scripting using Perl
- specific library issues like staffing, and your OPAC

The keyword in the workshop's title is becoming. This workshop will not make you an expert overnight. Rather, this workshop is intended to give you an introduction of the technology in terms of its strengths and weaknesses.

The World Wide Web (and the Internet in general) is in its infancy. Consequently, there is so much to learn and so much change that it is impossible to bestoe on you everything you needed to know about WWW server administration in one day. For this reason, the handout is riddled with URLs and you are encouraged to follow up on topics of interest with these pointers.

I hope you enjoy the workshop, and I hope you all go away ready to "make some waves." Let's begin with a bit of a history lesson and background on how the technology works.

History

The WorldWideWeb (W3) is the universe of network-accessible information, an embodiment of human knowledge. It is an initiative started at CERN, now with many participants. It has a body of software, and a set of protocols and conventions.

> W3 uses hypertext and multimedia techniques to make the web easy for anyone to roam, browse, and contribute to. --Tim Berners-Lee 1993

Essentially, World Wide Web (WWW) servers are information dissemination tools. They allow you to save information or data on your hard disk and allow others to access and read that information. The information you can disseminate can be simple ASCII text, formatted hypertext markup language (HTML) documents, graphics, sounds, and/or movies.

In 1989, Tim Berners-Lee of CERN (a particle physics laboratory in Geneva, Switzerland) began work on the World-Wide Web. The Web was initially intended as a way to share information between members of the high-energy physics community. By 1991, the Web had become operational. The World Wide Web is a hypertext system, a concept originally described by Vannevar Bush. The term "hypertext" was coined by Theodore H. Nelson. In a hypertext system, a document is presented to a reader that has "links" to other documents that relate to the original document and provide further information about it.

Not only does the hypertext feature work within documents, but it works between documents as well. For example, by clicking on Table Of Contents a new document will be presented to you, the table of contents of this handout. If you ever get lost, you can always use your WWW browsing software to go back to where you came because there is always a "go back" button or menu choice.

Scholarly journal articles represent an excellent application of this technology. For example, scholarly articles usually include multiple footnotes. With an article in hypertext form, the reader could select a footnote number in the body of the article and be "transported" to the appropriate citation in the notes section. The citation, in turn, could be linked to the cited article, and the process could go on indefinitely. The reader could also backtrack and follow links back to where he or she started.

Here are just a couple examples of electronic journal/magazine articles employing hypertext features:

- Vannevar Bush, "As We May Think," *Atlantic Monthly* 176 (July 1945): 101-108.
- Mac Net Journal

The Hypertext Transfer Protocol (HTTP) that allows this technology to happen is older than the gopher protocol. The original CERN Web server ran under the NeXTStep operating system, and, since few people owned NeXT computers, HTTP did not become very popular. Similarly, the client side of the HTTP equation included a terminal-based system few people thought was aesthetically appealing. All this was happening just as the gopher protocol was becoming more popular. Since gopher server and client software was available for many different computing platforms, the gopher protocol's popularity grew while HTTP's languished.

It wasn't until early 1993 that the Web really started to become popular. At that time, Bob McCool and Marc Andreessen, who worked for the National Center for Supercomputing Applications (NCSA), wrote both Web client and server applications. Since the server application (httpd) was available for many flavors of UNIX, not just NeXTStep, the server could be easily used by many sites. Since the client application (NCSA Mosaic for the X Window System) supported graphics, WAIS (see WAIS, Inc., CNIDR's freeWAIS, and Ulrich Pfeifer's freeWAIS-sf), gopher, and FTP access, it was head and shoulders above the original CERN client in terms of aesthetic appeal as well as functionality. Later, a more functional terminal-based client (Lynx) was developed by Lou Montulli, who was then at the University of Kansas. Lynx made the Web accessible to the lowest common denominator devices, VT100-based terminals. When NCSA later released Macintosh and Microsoft Windows versions of Mosaic, the Web became even more popular. Since then, other Web client and server applications have been developed, but the real momentum was created by the developers at NCSA.

See Also

1. "World Wide Web" - This URL will take you to a terminal-based WWW browser. <URL:telnet://telnet.w3.org/>
2. "World Wide Web Consortium [W3C]" - The Consortium provides a number of public services: 1) A repository of information about the World Wide Web for developers and users, especially specifications about the Web; 2) A reference code implementation to embody and promote standards 3) Various prototype and sample applications to demonstrate use of new technology. <URL:http://www.w3.org/pub/WWW/>
3. Alan Richmond, "WWW Development" <URL:http://www.charm.net/~web/Vlib/>
4. Bob Alberti, et al., "Internet Gopher protocol" <URL:ftp://ftp.lib.ncsu.edu/pub/stacks/finding/protocol.txt>
5. CERN European Laboratory for Particle Physics , "CERN Welcome" - CERN is one of the world's largest scientific laboratories and an outstanding example of international collaboration of its many member states. (The acronym CERN comes from the earlier French title: "Conseil Europeen pour la Recherche Nucleaire") <URL:http://www.cern.ch/>
6. CNIDR, "freewais Page" <URL:http://cnidr.org/cnidr_projects/freewais.html>
7. Daniel W. Connolly, "WWW Names and Addresses, URIs, URLs, URNs, URCs" - "Addressing is one of the fundamental technologies in the web. URLs, or Uniform Resouce Locators, are the technology for addressing documents on the web. It is an extensible technology: there are a number of existing addressing schemes, and more may be incorporated over time." <URL:http://www.w3.org/hypertext/WWW/Addressing/Addressing.html>
8. Distributed Computing Group within Academic Computing Services of The University of Kansas, "About Lynx" <URL:http://kufacts.cc.ukans.edu/about_lynx/about_lynx.html>
9. Internet Engineering Task Force (IETF), "HTTP: A protocol for networked information" - HTTP is a protocol with the lightness and speed necessary for a distributed collaborative hypermedia information system. It is a generic stateless object-oriented protocol, which may be used for many similar tasks such as name servers, and distributed object-oriented systems, by extending the commands, or "methods", used. A feature if HTTP is the negotiation of data representation, allowing systems to be built independently of the development of new advanced representations. <URL:http://www.w3.org/hypertext/WWW/Protocols/HTTP/HTTP2.html>
10. Karen MacArthur, "World Wide Web Initiative: The Project" - [This site hosts many standard concerning the World Wide Web in general.] <URL:http://www.w3.org/>
11. Mary Ann Pike, et al., Special Edition Using the Internet with Your Mac (Que: Indianapolis, IN 1995)
12. N. Borenstein, "MIME (Multipurpose Internet Mail Extensions)" - "This document is designed to provide facilities to include multiple objects in a single message, to represent body text in character sets other than US-ASCII, to represent formatted multi- font text messages, to represent non-textual material such as images and audio fragments, and generally to facilitate later extensions defining new types of Internet mail for use by cooperating mail agents. <URL:http://www.oac.uci.edu/indiv/ehood/MIME/1521/rfc1521ToC.html>
13. National Center for Supercomputing Applications, "A Beginner's Guide to URLs" <URL:http://www.ncsa.uiuc.edu/demoweb/url-primer.html>
14. NCSA, "NCSA Home Page" <URL:http://www.ncsa.uiuc.edu/>
15. NCSA, "NCSA Mosaic Home Page" <URL:http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/help-about.html>
16. NCSA, "NCSA Mosaic for the Macintosh Home Page" <URL:http://www.ncsa.uiuc.edu/SDG/Software/MacMosaic/MacMosaicHome.html>
17. NCSA, "NCSA Mosaic for Microsoft Windows Home Page" <URL:http://www.ncsa.uiuc.edu/SDG/Software/WinMosaic/HomePage.html>
18. NCSA HTTPd Development Team, "NCSA HTTPd Overview"

<URL:http://hoohoo.ncsa.uiuc.edu/docs/Overview.html>
19. Software Development Group (SDG) at the National Center for Supercomputing Applications, "SDG Introduction" <URL:http://www.ncsa.uiuc.edu/SDG/SDGIntro.html>
20. Thomas Boutell, "World Wide Web FAQ" - "The World Wide Web Frequently Asked Questions (FAQ) is intended to answer the most common questions about the web." <URL:http://www.boutell.com/faq/>
21. Tim Berners-Lee, Roy T. Fielding, and Henrik Frystyk Nielsen, "Hypertext Transfer Protocol" - "The Hypertext Transfer Protocol (HTTP) has been in use by the World-Wide Web global information initiative since 1990. HTTP is an application-level protocol with the lightness and speed necessary for distributed, collaborative, hyper media information systems. It is a generic, stateless, object-oriented protocol which can be used for many tasks, such as name servers and distributed object management systems, through extension of its request methods (commands). A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred." <URL:http://www.w3.org/hypertext/WWW/Protocols/Overview.html>
22. Ulrich Pfeifer, "FreeWAIS-sf" <URL:http://ls6-www.informatik.uni-dortmund.de/freeWAIS-sf/>
23. University of Kansas, "KUfact Online Information System" <URL:http://kufacts.cc.ukans.edu/cwis/kufacts_start.html>
24. University of Minnesota Computer & Information Services Gopher Consultant service, "Information about gopher" <URL:gopher://gopher.tc.umn.edu/11/Information%20About%20Gopher>
25. URI working group of the Internet Engineering Task Force, "Uniform Resource Locators" <URL:http://www.w3.org/hypertext/WWW/Addressing/URL/URL_TOC.html>
26. Vannevar Bush, "As We May Think" Atlantic Monthly 176 (July 1945): 101-108 <URL:http://www.isg.sfu.ca/~duchier/misc/vbush/>
27. WAIS, Inc., "WAIS, Inc." <URL:http://www.wais.com/>

Client/Server Model of Computing

This is client/server computing described.

To truly understand how much of the Internet operates, including the Web, it is important to understand the concept of client/server computing. The client/server model is a form of distributed computing where one program (the client) communicates with another program (the server) for the purpose of exchanging information.

The client's responsibility is usually to:

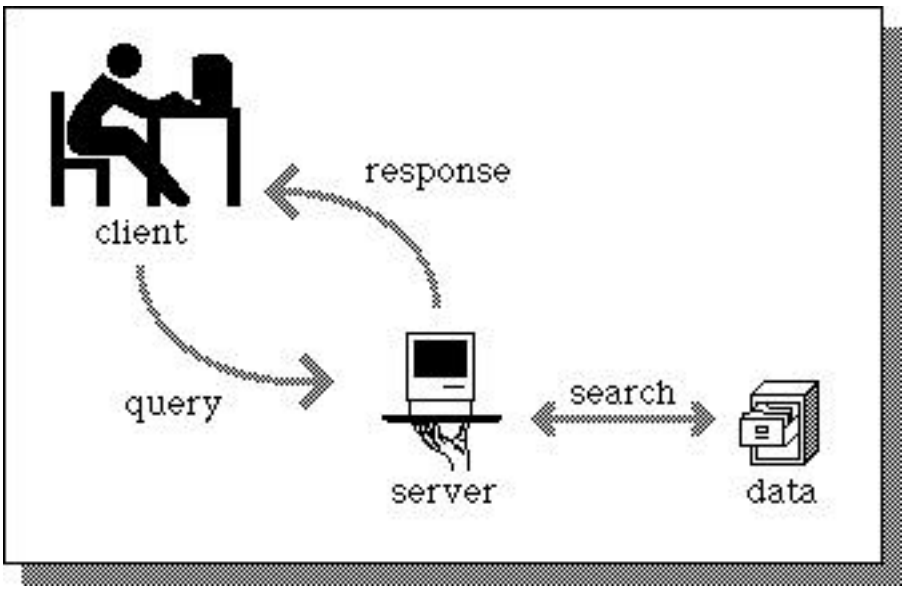
1. Handle the user interface.
2. Translate te user's request into the desired protocol.
3. Send the request to the server.
4. Wait for the server's response.
5. Translate the response into "human-readable" results.
6. Present the results to the user.

The server's functions include:

1. Listen for a client's query.
2. Process that query.
3. Return the results back to the client.

A typical client/server interaction goes like this:

1. The user runs client software to create a query.
2. The client connects to the server.
3. The client sends the query to the server.
4. The server analyzes the query.
5. The server computes the results of the query.
6. The server sends the results to the client.
7. The client presents the results to the user.
8. Repeat as necessary.



A typical client/server interaction

This client/server interaction is a lot like going to a French restaurant. At the restaurant, you (the user) are presented with a menu of choices by the waiter (the client). After making your selections, the waiter takes note of your choices, translates them into French, and presents them to the French chef (the server) in the kitchen. After the chef prepares your meal, the waiter returns with your diner (the results). Hopefully, the waiter returns with the items you selected, but not always; sometimes things get "lost in the translation."

Flexible user interface development is the most obvious advantage of client/server computing. It is possible to create an interface that is independent of the server hosting the data. Therefore, the user interface of a client/server application can be written on a Macintosh and the server can be written on a mainframe. Clients could be also written for DOS- or UNIX-based computers. This allows information to be stored in a central server and disseminated to different types of remote computers. Since the user interface is the responsibility of the client, the server has more computing resources to spend on analyzing queries and disseminating information. This is another major advantage of client/server computing; it tends to use the strengths of divergent computing platforms to create more powerful applications. Although its computing and storage capabilities are dwarfed by those of the mainframe, there is no reason why a Macintosh could not be used as a server for less demanding applications.

In short, client/server computing provides a mechanism for disparate computers to cooperate on a single computing task.

Hypertext Transfer Protocol

Once you look under the hood, there is no magic to HTTP.

HTTP stands for Hypertext Transfer Protocol. The protocol, designed by Tim Berners-Lee as early as 1989, is really rather simple. In a nutshell, one application hosts data and "listens" for connections on TCP port 80. (Think of a "port" like a telephone number extension.) Another application then opens a connection to the host on the same port and initiates a dialog. The dialog is simply a request for data (by the client) and response by the host (server). In many ways, the protocol is much simplier than FTP or SMTP. Both of these protocols require there to be a true dialog between client and server. The current implementation of HTTP only requires one request and one response.

Connecting

One of the best ways to learn about the protocol itself is through the use of your telnet application. All you have to do is telnet on port 80 to a known HTTP server and initiate a request.

For example, try telneting to Netscape at home.netscape.com on port 80. Once connected, simply press your return key twice and see what happens. You should get something like this:

```
HTTP/1.0 400 Bad Request
Server: Netscape-Enterprise/2.0b4

Your browser sent a message this server could not understand.
```

Here the server replied with an error, specifically error 400. This is correct because you did not send a valid request. At the very least, a WWW browser should specify a "method" and a document in a request. The simplest method is GET and the simplest document is "/". (By the way, method names are case-sensitive.)

Simplest Request

Armed with this newly found information, telnet to www.apple.com on port 80 and when the connection opens enter "`GET /`" and two carriage returns. This time you should see a whole lot of HTML being sent back to you. This HTML is what your WWW browser would interpret and render on your screen.

Qualifying Requests

If you specify a version of HTTP in your request, then the responses from servers are more complete and sometimes quite interesting. For example, telnet to www.microsoft.com on port 80 and request "`GET / HTTP/1.0`". If Microsoft has not changed its page, then new information, called the header, will be displayed before the HTML:

```
HTTP/1.0 302 Object moved
Server: Microsoft-IIS/3.0
Date: Tue, 28 Jan 1997 04:03:55 GMT
Location: http://msid.msn.com/mps_id_sharing/redirect.asp?
```

```
www.microsoft.com/default.asp
Connection: Keep-Alive
Content-Length: 199
Content-type: text/html
Set-Cookie: MC1=ID=45e7b538788b11d0a7550000f8485726;
expires=Wed, 15-Sep-1999 19:00:00 GMT; domain=.microsoft.com;
path=/
Cache-control: private

<head><title>Object moved</title>
</head><body><h1>Object Moved</h1>
This object may be found <a
HREF="http://msid.msn.com/mps_id_sharing/redirect.asp?
www.microsoft.com/default.asp">here</a>.</body>
```

This header information outlines various information about the reply being returned such as what server is being used, the date, and the length of the data being sent. The most important information is the Content-type. This piece of information tells your browser the MIME type of the data being returned. This information is critical. Otherwise, your browser would not know how to interpret the returned data.

The Point

The point of this demonstration is two-fold. First, it is a demonstration of the client/server model of computing. One application connects to another application and makes some sort of request. The second application processes the request and returns some sort of results.

Second, the demonstrations illustrate the specific sort of dialog that takes place between an HTTP client and server. These examples have not been exhaustive. There are quite a number of other values the client can send in a request, and there are quite a number of other values a server can send in a response.

Keep in mind, one of the most important pieces of information being returned by the server is the Content-type. This information is sometimes called the "magic line" and specifies the MIME type of information being returned. It will also be necessary in any CGI scripting you may do.

Comparing HTTP to Other Services

> *This section compares and contrasts HTTP with telnet, FTP, SMTP, WAIS, and gopher.*

Just about every Internet service relies on the client/server model of computing. One application sends a request to another application and the second application replies.

Telnet

Telnet is a protocol allowing you to log into another computer and interact with that computer as if your computer were physically attatched to it like a terminal. Essentially, telnet applications are

terminal-like applications. Since most implementations of telnet allow you to specify the TCP/IP port for communications, the use of telnet is an excellent way to learn the fundamentals of other TCP/IP services. Telnet is a protocol that "retains state" with the remote computer. This means that your connection is always active with the remote computer as long as you are logged on and even when you are performing no actions. On the other hand, HTTP is a "stateless" protocol meaning connections between client and host are created, a dialog insues, and the connection is then broken.

FTP

An acronym for File Transfer Protocol, FTP has two functions. The first is to copy files from one computer to another. Second, and less commonly known, FTP allows clients to do rudimentary directory creation and deletion on remote computers. Like HTTP, the primary purpose of FTP is to send and retrieve data between computers. FTP does not support MIME, nor do FTP clients interpret any of the data they recieve. FTP uses two TCP/IP ports for its communications. One for the dialog and a second for the file transfer. HTTP relies on one port.

SMTP

SMTP is the email protocol (Simple Mail Transfer Protocol). This protocol is almost exclusively handled without human intervension. Messages are passed between client and server in a sometimes humorous fashion if transactions are done by hand. The headers of SMTP messages formed the basis of headers in HTTP requests and replies. Similarly, both protocols use a blank line to delimit the header from the content of the message. With the recent introduction of MIME into SMTP headers, some email clients are now doing rudimentary interpretation of incoming data, just like HTTP clients.

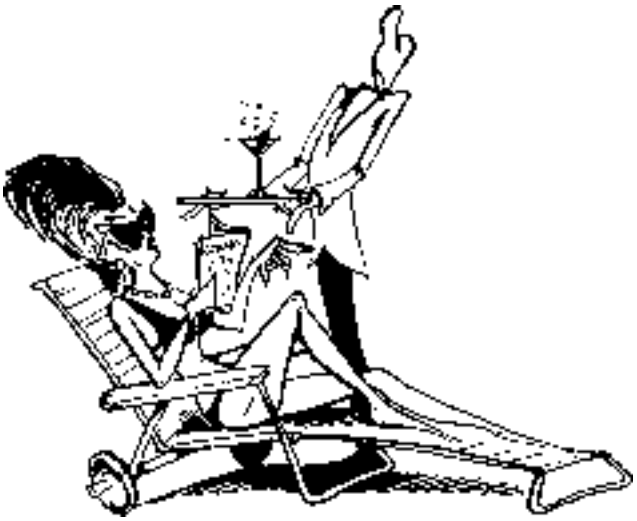
WAIS

WAIS (Wide Area Information Server) was originally designed to demonstrate the capabilities of Thinking Machine, Inc. computer hardware. It is loosely based on an old version of the Z39.50 protocol and allows clients to query remote WAIS databases, interpret returned results, and request specific items from these results. The various WAIS distributions also include software for creating WAIS databases. Early NCSA Mosaic browsers for Unix computers could perform WAIS queries without the need of proxy servers. HTTP has little in common with HTTP except both protocols are ultimately used for the purposes of sending files from server to client.

Gopher

The gopher protocol, developed by the University of Minnesota Microcomputer, Workstation Networks Center, initially became more popular than HTTP because initially there were more implementations of clients and servers for different operating systems. It was originally designed to allow people to seamlessly tranfer ASCII files from one computer to another without understanding the necessesities of FTP. Thus, like FTP and HTTP, gopher's purpose was the transfer of files. Unlike FTP, gopher clients did attempt to interpret incoming data, but the data types were too limiting. Gopher worked. It did what it was designed to do. It is just that with the development of Mosiac, HTTP did it better.

One significant difference between the gopher protocol and HTTP is how files got transfered. In the gopher protocol, the server did all the work getting and transfering files. HTTP clients, on the other hand, are much more independent and do their own retrieveal. In short, HTTP distributes the processing load much more effectively than gopher.



Bringing Up a WWW Server

This section describes the fundamentals of bring up a WWW server.

Believe it or not, bringing up a WWW server is simple compared the work involved to add content and maintain it. Important considerations in this area include the hardware where your server will run, the software implementing HTTP, and configuring the software in terms of MIME types, and access control.

See Also

1. "Apache HTTP Server Project" - "The Apache project has been organized in an attempt to answer some of the concerns regarding active development of a public domain HTTP server for UNIX. The goal of this project is to provide a secure, efficient and extensible server which provides HTTP services in sync with the current HTTP standards." <URL:http://www.apache.org/>
2. "WebSite Central" - This is the official home page of WebSite. <URL:http://website.ora.com/>
3. Brian Behlendorf, et al., "Running a Perfect Web Site with Apache" (Indianapolis, IN: Que, 1996) - "This book is designed for those who are new to setting up a Web server on a UNIX platform. The featured Web server is Apache, though many of the subjects covered are applicable to other Web servers."
4. CERN, "[Summary of HTTP Error Codes]" <URL:http://www.w3.org/pub/WWW/Protocols/HTTP/HTRESP.html>
5. David Strom, "WebCompare" - "[T]he leading site for in-depth information on server software for the World Wide Web." <URL:http://webcompare.iworld.com/>
6. NCSA, "NCSA HTTPd Overview" - These pages document the NCSA HTTPd server, the server WebSite is based upon. <URL:http://hoohoo.ncsa.uiuc.edu/docs/Overview.html>
7. Roy Fielding, "WWW Protocol Library for Perl" - "libwww-perl is a library of Perl packages/modules which provides a simple and consistent programming interface to the World Wide Web. This library is being developed as a collaborative effort to assist the

further development of useful WWW clients and tools." <URL:http://www.ics.uci.edu/pub/websoft/libwww-perl/>

8. StarNINE, "WebSTAR Product Information" - "WebSTAR(TM) is the industry standard for transforming your Mac into a powerful Web server. WebSTAR can serve millions of connections per day, and is fully extensible through WebSTAR plug-ins." <URL:http://www.starnine.com/webstar/webstar.html>
9. StarNine, "WebSTAR" - Based on Chuck Shotton's MacHTTP, WebSTAR(TM) helps you publish hypertext documents to millions of Web users around the world, right from your Macintosh. You can also use WebSTAR to put any Macintosh file on the Web, including GIF and JPEG images and even QuickTime(TM) movies. And yet, using WebSTAR is as easy as AppleShare(r). Plus, it's faster than many Web servers running on UNIX. <URL:http://www.starnine.com/webstar/webstar.html>
10. Stephen Turner, "Analog" - "Fast, professional WWW logfile analysis for Unix, DOS, NT, Mac and VMS." <URL:http://www.statslab.cam.ac.uk/~sret1/analog/>

MIME Types

This subsection describes MIME types and why they are important.

MIME is an acronym for Mulitpurpose Internet Mail Extensions. As RFC 1521, it represents a standard for describing data types. It is information about data. As originally concieved, it was designed as an extention to the Simple Mail Transfer Protocol (RFC 822) allowing people to send binary data (like pictures and sounds) as enclosures with email messages.

The standard was flexible enough that it could be easily incorporated into HTTP. Consequently, MIME provides the mechanism for seemlessly tranfering non-text data easily to your browser. (Incidentally, the gopher protocol did not support MIME. If it had, then maybe it would have proved a more valuable tool.)

A particular MIME type is a pair of elements delimited by a slash ("/"). The first element describes the "type" of data. Examples include but are not limited to:

- application
- audio
- image
- text
- video

The second element describes the format of the type such as:

- msword
- gif
- jpeg
- mpeg
- quicktime
- html
- plain

Complete MIME types then include some of the following examples. You have probably seen these sorts of things while configuring your WWW browser for helper applications:

- application/msword
- application/macwriteii
- application/postscript
- application/zip
- application/x-tar
- audio/basic
- audio/x-wav
- text/html
- text/plain
- image/gif
- image/jpeg
- video/mpeg
- video/quicktime

Thus, when your browser (or email application) recieves a MIME type of application/msword it is expected to know what application to run (Microsoft Word) and handle it accordingly. Simiarly, when your browser recieved some data of type text/html (the most common WWW MIME type), then your browser knows to interpret the incomming data as HTML and display it.

As a WWW server administrator, MIME types are important because you have to know what kinds of data you are serving to your constituents. These kinds of data are described with MIME. All WWW server applications come configured to handle most MIME types, but when a new data type becomes available, you will have to know a bit about MIME in order for you to server it correctly.

Hardware

It almost doesn't matter what hardware you use.

The ideal way to select the hardware for any computing task is to first ask yourself what software you need to satisfy the task. The software should drive the hardware. In the real world, this is not always the case. Consequently, you may not have to ask yourself what hardware to use. Rather you may be asking yourself, "What hardware is available?"

If you do have a choice of hardware, then first articulate and enumerate the purpose(s) of your server. Analyse who your audience is and try to guesstimate how many hits your server will get per day. If your server is intended to disseminate lists of services or act as a conduit for your usual print publications, then a microcomputer-based server will do just fine. If you are planning for a departmental Intranet, again, a microcomputer-based server will fit the bill.

Consider exactly what any HTTP server is doing. It is disseminating small (maybe 35 K) files. Thirty-five kilobytes of data is tiny. It does not take a large computer to serve this amount of data over and over again. Furthermore, it has been proven microcomputer-based servers can handle thousands of hits per hour, if not tens of thousands of hits. The use of your microcomputer as a server may also be determined by what information in databases you have that you would like to share. If your data resides in a microcomputer, then you will be using the same microcomputer to serve this data (unless you want to move it to another machine).

In short, microcomputer-based servers offer a number of distinct advantages. First, the use of an operating system you are already familiar with; you will not have to learn something like Unix and all of its administrative overhead. Second, the microcomputer hardware is readily available. Starting out, consider using one of those computers in the back room that may be gathering dust.

On the other hand, the use of a Unix, Windows NT, or even a VMS-based server may be something to consider. This is especially true if you desire to serve the data from your OPAC through a WWW interface. These multiuser, rock-solid operating systems come at a cost. One is administrative support. You will almost necessarily have to have some sort of network "guru" managing your system. For you there may be a learning curve while you get familar with the Unix, Windows NT, or VMS environment. Another cost is distance. "Out of sight, out of mind." In other words, if your HTTP resides on a computer in the back room that only a few people have direct access to, then the computer turns into a mystery. This is not the best way to perceive of computers and makes them seem far away and out of your control.

At the same time, these bigger computers were designed to run client/server applications. TCP is a fundamental part of the Unix operating system, and consequently there is a lot of support for TCP networking built in. Also, there is no denying it, these bigger computers are faster. But remember, the speed of any network connection is only as fast as the slowest link in the chain. If you users are all using 28.8 modems, then it doesn't matter whether or not your server is a blazingly fast minicomputer.

In the end, you will probably find yourself having an mixture of micrcomputer-based and minicomputer-based HTTP servers in your institution. When you are just learning, use your desktop computer as a test bed and grow from there.

Server Software

Features to look for in HTTP server software are described here.

By now, there must be more than a hundred different HTTP servers to choose from for just about every operating system. They range from things promising you the Moon to absolutely free applications where you get exactly what you pay for. Obviously you want to choose something in between.

Features

If you are just starting out, here is a list of features you should look for when selecting HTTP server software:

- Access control via IP or domain name
- Adequate technical support from the Internet
- CGI scripting compatible
- Configurable error file definitions
- Flexible log file creation
- Security through passwords
- Supports server side includes (SSI)

More advanced features include:

- Built-in imagemaping
- Implements byte serving
- Implements the PUT method
- Simple logfile analysis
- Specialized "hooks" to the server's operating system

- Supports the Secure Socket Layer (SSL) protocol
- Technical support from the software's vendor
- Tight integration with database applications
- Web access to administrative functions

Specific Choices

Each of the three HTTP servers outlined in the following sections support all of the features in the first list and a few of the features in the second. They are all free, and they are all very stable applications.

Apache

Apache, a server for Unix computers, is the most popular and is considered one of the most robust implementations. It does not do any logfile analysis. There is no vendor so there is no formal technical support. There is no real way to administrate the computer through a Web interface.

On the other hand, Apache's modular approach allows for a great deal of customization. It includes strong links for database applications. It allows you to save your logfiles in formats you define. But most importantly, it runs on any Unix computer, and Unix is an operating system designed for client/server and TCP/IP applications.

Quid Quo Pro

This server is extraordinarily simple to bring up and maintain. It does not support a close relationship with any database applications, but it is integrated with the Macintosh OS through AppleEvents. Unlike most other server software implementations, there are absolutely no text files write and save; everything is done through dialog boxes. It does include administrative functions and has built-in imagemapping. In short, Quid Quo Pro support the vast majority of features of the most populare Macintosh HTTP server but it is infinitely cheaper, $0.

WebSite

WebSite comes with the most bells and whistles of the servers mentioned here. It supports all the features in the first list as well as the second list with the exception of SSL. It comes with extensive examples of imagemapping techniques, CGI scripting, server side includes, and administration features. Designed to run under Windows95 and Windows NT, this server would fit most people's needs especially considering the proliferation of the Windows platform.

Apache

> *This section describes how to bring up an Apache HTTP server.*

According to Netcraft, Apache has been the most popular HTTP server for quite some time. This is understandable since it, like the other servers described here, is "as free as a free kitten", runs

under any flavor of Unix, is very extensible, and is just about as robust a server you will find anywhere. As of version 1.3b, Apache now also runs under Windows95 and NT. It will also be ported to Rhapsody (Macintosh) when that operating system becomes available this year. Consequently, Apache represents a good, all-around HTTP server. One that you can/will be able to take to just about any computer.

Apache, based on the original NCSA httpd application, got its name from when its developers where trying to break httpd down into its original parts. Thus, it was "a patchy server."

There are two alternatives for acquiring the Apache software. One, you can download a precompiled binary for your particular version of Unix or Windows. Or two, you can download Apache's source code and compile it yourself. This section outlines the second option for Unix since it offers you greater flexibility.

Compiling

Compiling our own version of Apache is a 4-step process:

1. downloading the archive and uncompressing it
2. editing the src/Configuration file
3. running src/Configure
4. running make

Begin by downloading the archive from www.apache.org. If you are adventurous, then download the latest beta version. Otherwise grab the version without any b's listed in its name. (Notice how small the archives are. They could easily fit on a single floppy disk!)

Next, uncompress and untar the downloaded file with gunzip and tar. This should result in a directory named `apache_1.2.5` or something similar.

Change directories to the `src` directory within the apache_1.2.5 directory. Use your favorite text editor to edit the file named Configuration. The Configuration file lists the modules ("patches") you would like to incorporate into your httpd binary. All of these modules are described in the online documentation. Consider uncommenting the module named `status_module`. This module will allow you to monitor the status of your server from a WWW browser. To uncomment a module remove the pound sign (#) preceeding its definition.

Next, enter the Configure command. This command very quickly edits a few files in your distribution and returns. Now, enter make and wait. If everything goes well, you should see a lot of text scrolling up your terminal but finally you will get your prompt back and your Apache HTTP server is compiled.

Configuration

Configuring the Apache server requires the following steps:

1. copying the compiled binary up one directory level
2. editing conf/httpd.conf
3. editing conf/srm.conf
4. editing conf/access.conf

Copy your newly created Apache server application up one directory out of the `src` directory. To make your life easier, this directory should also contain the `htdocs`, `logs`, `conf`, and `cgi-bin` directories.

Change to the conf directory and `cp httpd.conf-dist httpd.conf`. This will make sure you have an original version of the `httpd.conf` file. Using your text editor again, edit httpd.conf. In this file you will want to change the following definitions:

`Port`
Enter an integer. The default port of HTTP servers is 80. If you are the superuser of your Unix computer, then you will be able to run the server on this port. If you are just exerimenting, then use a port like 8000. Example: `8000`

`User`
Enter a username. The user variable is used to tell the server who owns the httpd process once it it running. Since the httpd application will have to read and write files on your computer, it will be necessary to select a username that has just the right number of priveleges. It is best to create a bogus user named "nobody", give them few priveleges, and have the httpd application run under that user. Example: `nobody`

`Group`
Enter an integer. Every username should be assigned to at least one usergroup. Like usernames, groups help define priveleges on the computer and you should have a group defined that has limited authority. Enter a pound sign followed by an integer denoting the group the httpd application should run under. Example: `#-1`

`ServerAdmin`
Enter here the email address of the person who should get messages when things go wrong. (Obviously this will never be needed.) Example: `webmaster@lib.univ.edu`

`ServerRoot`
Enter the full path to the httpd application. Example: `/usr/local/apache`

`ServerName`
Enter here the IP address or fully qualified domain name of your Unix computer. This address or name will be the name returned to any clients connecting to your server. Do not make up a name. Example: `dewey.lib.univ.edu`

Next, cp srm.conf-dist srm.conf so you have a backup of the file. Here you will want to make sure you edit the following values:

`DocumentRoot`
Enter the full path to the location where you will be saving your HTML documents. Example: `/usr/local/apache/htdocs`

`ScriptAlias`
If you plan on using CGI scripts, then you will want to change this configuration to map a virtual directory to a real directory containing the scripts. Example: `/cgi-bin/ /usr/local/apache/cgi-bin/`

`AddHandler`
Again, if you want to run CGI scripts from your server, it is convienent to create a

filetype here with the .cgi extension. Example: `cgi-script .cgi`

You are more than half way there. Keep editing!

The final configuration requires you to edit access.conf. First cp access.conf-dist access.conf.

The access.conf file is made up of `<Directory></Directory>` pairs. By default the file defines access configurations for your HTML document root and your `cgi-bin` directory. You will want to change the values already in the file to the same values you specified for the `DocumentRoot` and `ScriptAlias` above. Examples: `<Directory /usr/local/apache/htdocs>` and <Directory /usr/local/apache/cgi-bin>.

If you compiled your httpd binary with `status_module`, then uncomment all the lines in the `<Location /server-status>` directive and edit the allow from line to include your domain. If you do this, then once your server is running you will be able to enter a URL like http://www.lib.univ.edu/server-status and see how your server is running.

Windows Configuration

Installing and configuring the Windows95/NT version of Apache includes two options. First, you can download the source code and compile it. This requires a Microsoft C++ compiler. Alternatively, you can download a pre-compiled version of the application. If you own the compiler, then you don't need any instructions. Otherwise, download the pre-compiled version.

Whether you have downloaded the source code or the pre-compiled version, you will want to edit the `*.conf` files just as above.

Starting Up

It is now time to actually start up your server. Move up one directory and run the httpd application with `./httpd -f path`, where `path` is the fullpath name to your `httpd.conf` file. For examaple, `httpd -f /usr/local/apache/conf/httpd.conf`. (If you are running the Windows version, then you will want to run the application from the command line and specify the fullpath of the `httpd.conf` file as an argument: `c:\apache\apache.exe -f c:\apache\conf\httpd.conf`.) If an error is returned, then read it carefully and try to resolve the problem. If no errors occured, then use your browser to open a connection to your server. Remember to specify the Port in your URL as you defined above. If for some reason you do not get the "It worked!" message, then check the contents for your `logs/error_log` file for clues to what went wrong.

At this point your are either overjoyed or overwhelmed. Those of you who are overjoyed should now go back to your configurations to make more specific modifications. Those of you who are overwhelmed are encouraged to visit news:comp.infosystems.www.servers.unix and try to get some assistance there. Carefully worded questions will get responses.

Quid Pro Quo

This describes how to bring up the Macintosh-based HTTP server called Quid Pro Quo.

Quid Pro Quo may not be the most popular Macintosh-based HTTP available, but it has just about every feature of the leader and the price certainly is right. Free. Chris Hawk, the developer, has certainly created a fine piece of softare. Furthermore, after describing how to bring up this server on our Macintosh, you will wonder at the complexity of Apache.

Begin my acquiring the application from its download page at www.socialeng.com. Be sure to download Quid Pro Quo 2.x since it is the free version. The other more feature rich versions are available for free 30-day trials and require a serial number to activate.

If your WWW browser is configured correctly, then the downloaded file should uncompress and result in a self-extracting archive. Launch the self-extracting archive and tell it to save the compressed file any place on your hard disk.

Launch Quid Pro Quo and you will be asked a few configuration questions. For the most part you can accept the defaults. The only thing you should modify is the setting for the root directory. Select the Examples directory that came with the distribution. After running through the configuration section Quid Pro Quo will complete its startup functions and you can open up a connection to yourself through your WWW browser.

This simplicity does not preclude configuration options. The configuration options are available by selecting Server Settings... from the Control menu. After you become experienced, you will discover that the default settings are just fine. The most useful piece of information is found in the Default Files section. This is also where you define the name of the file to return when clients select directories. The default is default.html. You might want to change this to index.html. You might also want to look in the Error Files section. Here you will learn what informational HTML files are returned to clients when errors occurs (file not found, permission denied, etc).

Website

This describes how to bring up a WebSite server.

WebSite, just like Apache, began its life as a port of the NCSA httpd server. Robert Deny, the orignal developer, is now working with O'Reilly & Associates, Inc. and has made quite a number of improvements since the original distribution.

Bringing up a WebSite server is painless. It involves downloading the archive, uncompressing it, and doing the tiniest bit of configuration. Begin by acquiring the distribution from the download page at website.ora.com. After the archive is downloaded, uncompress it.

The uncompressed distribution will result in a self-extracting installer named "Ws11e" or something similar. Launch the installer and be prepared to answer:

1. where will you be installing WebSite on our computer
2. what is the IP address of your computer
3. what is the email address of the WebSite administrator

After answering these quesetions, the installer will uncompress more files and save various system extensions in their proper locations. It will then add itself to your Start menu.

After restarting your computer, all your icons will be updated correctly, and all you have to do is select WebSite Server from your Start menu to begin.

Access Control

This section describes how to limit access to your server based on IP addresses and/or passwords.

There are sets of data and information that some people are allowed to view and there are some sets of data and information that some people are not allowed to view. Despite what the professional ethics of librarianshp may say, which can be debated forever and a day, this is a reality in today's world. At the same time, there are some sets of data and information that are not so much secret as they are just nobody's business but your own. For these reasons, methods can be put in place for restricting access to parts of all of your HTTP server.

In a nutshell, there are two ways to restrict access. The first is based on the client's IP address or domain name. The other is with usernames and passwords. All of the servers described previously support both of these access control methods. Here is how to implement them.

Apache

As you may expect, restricting access to your Apache server is more complicated when compared to the desktop servers. The simplest and most direct way to restrict access requires you to create .htaccess files within the restricted directories. These .htaccess files are exactly like the instructions in your access.conf file except they do not require the `<Directory></Directory>` nor `<Location></Location>` directive tags.

IP and domain name restrictions

Below is a simple sample .htaccess file restricting access based on addresses. The first line specifies that all `GET` and `POST` queries will be restricted. Next, the restrictions will be processed in the `deny-allow` order since later configurations override earlier ones. Then the hosts and/or IP address that are denied and allowed are specified. Finally, the file is closed.

```
# this is an .htaccess file for IP addresses

# limit the types of access
<Limit GET POST>

# define how restriction will be processed
order deny,allow

# define who to deny and allow
deny from all
allow from .ncsu.edu
```

```
# close the directive
</Limit>
```

Usernames and passwords

Restricting access based on usernames and passwords first requires you to define users and sets of users called groups. Then you create .htaccess files specifying these users and/or groups.

If it hasn't been done before, you will have to compile the `htpasswd` program. It is located in the `support` directory of your original distribution. To compile the program change directories to the `support` directory and enter `make`. This should result in the creation of a number of utilities one of which is `htpasswd`.

For the `.htaccess` technique to work, you must edit your `access.conf` file and specify parent directories of your restricted directories with the `AllowOverride Limit` option.

Next, using `htpasswd` you will create new users. The command takes the following form:

```
htpasswd [-c] filename username
```

where:

`-c`
This is used only the first time htpasswd is used. Its purpose is to create the password file

filename
This is the full path name to your password file. It can be any path or name, but make sure it is not in your server's document directory structure. Example: `/usr/local/apache/conf/passwd`

`username`
This is the name of the user you are creating. Example: `eric`

After this, you have to create groups. Groups are ASCII text files with the following form:

```
group: member member member
```

where:

`group`
This is the name of a group. Example: `users`

`member`
This is the name of somebody previously defined by the htpasswd program. Example: `eric`

Now you are finally ready to create your `.htaccess` file. Specify the realm using `AuthName`. The value of `AuthName` will appear in the message asking for username and password. Specify the authorization type with `AuthType`. (The majority of the time this will be Basic.) Echo the location

of your password and group files next. Define the restricted methods of access. Specify what groups and users have access to this directory. Last, close the directive.

Here is a example `.htaccess` file limiting users to passwords.

```
# this is an .htaccess file for passwords

# define the realm
AuthName The Super Secret Space

# define the authentication type
AuthType Basic

# where are the password and group file
AuthUserFile  /usr/local/users/Eric/apache/conf/passwd
AuthGroupFile /usr/local/users/Eric/apache/conf/group

# limit the types of access
<Limit GET POST>

# say exactly who can access
require group users

# close the directive
</Limit>
```

Once this sort of .htaccess file is saved in a directory, the first time a user tries to access the directory they will be asked to enter their username and password.

Quid Pro Quo

IP and domain name restrictions

Using Quid Pro Quo, restrictions based on IP addresses or domain names are called Allow/Deny. This server only permits you to "allow or deny" sets of IP address or domain names for your entire server, not parts of it. It is unfortunately an all or nothing deal.

To set up this sort of restriction, select Server Settings... from the Control menu. Then select Allow/Deny from the resulting Configure Quid Pro Quo dialog box. Next you will want to add a new item and enter either IP addresses or domain names. Like the other servers, you do not have to specify the entire address or name, just enough to make it meaningful. Your configuration takes place as soon as you close the configuration dialog box.

Usernames and passwords

Restrictions to parts of your server using usernames and passwords are called realms. This method is more secure than the first method. Implementing it is a two step process. First, using the Configure dialog box, select Realms. Now you can enter any name you want for a realm and then a

string of characters that will be used to match parts of client requests. Any URL containing the string of characters will be restricted by usernames and passwords.

The second step is to choose Passwords from the Configuration dialog box. From here you can create new user names and passwords, and then associate them with realms created in the first step. Like IP/domain name restrictions, these configurations take place as soon as you close the Configuration dialog box.

Website

Like Quid Pro Quo, WebSite provides the means of access control through a series of dialog boxes. Its features are more robust than Quid Pro Quo's and at least on par with Apache's. Its only technical drawback is it inability to edit settings once they have been created. Many times, to make changes you must delete the old settings and recreate them with your new edits.

IP and domain name restrictions

The first step to limiting access by IP addresses and domain names is to open the WebSite Server Properties window and select Users. Next, create an authentication realm by clicking the New... button. Any label for your realm will do, just make it meaningful to yourself. Third, select Access Control and create a new URL Path. Enter the full path from your server's root to the directory you want to restrict. Select a realm too. Finally, specify what IP addresses and/or domain names will be allowed access to your realm using the Class Restrictions panel. If you want to restrict access to particular hosts, then select the Deny, then Allow button and enter the IP addresses or domain names you are allowing access. Conversely, if you want to deny access to particular hosts (people who might be malicious for example), then select the Allow, then Deny button and enter the hosts to deny. Close the WebSite Server Properties window and when you hear a system beep you will know the configuration has taken place.

Usernames and passwords

To limit access to some or all of your site, begin again with the WebSite Server Propertites window and the Users section. This time select or create a new Authentication Realm and create a new user using the New... button of the User panel. Second, go to the Access Control section and select or create a new URL Path for password protection. Remember to specify the full path name of the password protected directory begining with the root of your WebSite server. Finally, select the users allowed to access the directory by using the Add... button of the Authorized Users and Groups panel. Using the Add... button shoul result in a list of users you previously created. Again, changes won't take effect until you close the WebSite Server Properties window and you hear the system beep.

Four Essential Qualities of Information Systems

> *Useful information systems need to be readable, browsable, searchable, and provide interactive assistance.*

In order to be useful, any information must be readable, browsable, and searchable. With increasing size and complexity of today's information systems, interactive user assistance is becoming a necessary feature as well. These sections outlines these qualities so you, as an information system manager, can incorporate them into your HTTP services.

An information system, in the present context, is any organized collection of information. In our culture, information systems abound. The dash board of our cars are information systems. Maps are information systems. World Wide Web servers are no exception. While World Wide Web servers are primarily intended to be an electronic publishing medium they are also information systems. In order to be most effective, all but the smallest of information systems must be readable, browsable, as well as searchable.

All of these qualities (readability, browsability, and searchability) do not have to be equally represented in every information system. As your collection of information increases, different aspects of these qualities take on greater significance. Thus, the amount of readability, browsability, and searchability your information system exhibits depends on the type and quality of your collected data, as well as the information needs of your clientele.

Readability Means Good Page Layout

> *Readability connotes an appealing graphic design and page layout.*

All information systems, no matter how small must incorporate principles of good graphic design. You and your information system are competing with a myriad of other information systems. If your data is not presented in a visually appealing, easy-to-read manner, then your chances of retaining the attention of your intended audience are significantly reduced.

Guidelines

1. Use a consistent layout
2. White space is good
3. Visually organize the page; employ horizontal rules
4. Keep pages short
5. Include elements of contrast
6. Use all stylistic elements in moderation

Use a consistent layout

Your documents reflect you, your organization, and your information. By consistently using the same layout you are creating a unified whole, an identity. With the use of a consistent layout it is easier for the reader to know when they are reading your text and not someone else's. The creation of a template file can be helpful here. The template file would consist of your standard headers, footers, logo, signature, last data updated, as well as any other stylistic features you may incorporate.

White space is good

White space is the empty areas of a page. It adds contrast and provides a place for you eyes to rest. White space is not wasted space. For this same reason, stay away from all capital letters. Capital letters are usually the same height and width. This creates a block effect reducing the white (negative) space around letters. Instead, use a combination of lower and upper case letters because it increases the amount of white space around the letters.

Visually organize your pages

In other words, group similar concepts on your page together. Employ proximity. Don't put our email address at the top of the page and the URL to your personal home page at the bottom. Both items are electronic pointers relating to you. Group them together. When there is more than one type of information on a page, delimit the page with white space or horizontal rules.

Keep your pages short

In general, people do not like reading text from a computer screen. Using the popular vertical scroll bar of graphical WWW browsers it is very easy to get lost in a document. In general keep you pages shorter than two or three screens in length. By keeping your pages short and to the point, the attention span of your readership will increase. Put another way, break up long pages into shorter ones.

Include elements of contrast

A boring looking page is completely filled with text. It contains no white space, no change in fonts or font sizes, no lists, no pictures. It is boring. Elements of contrast breakup the monotony and make our page more dynamic. Elements of contrast include emphasizing some text with styles like `<em></em>` or `<strong></strong>`. Other examples include the use of very heavy horizontal rulers or very large headers. With the current version of MacWeb and MacMosaic you as a

information provider can distribute preference files defining the size of style of fonts you want your readers to use. This gives you the ability to select different font families for different parts of your documents. Unfortunately, this is an extremely uncommon practice and difficult to actually implement. Another method for accomplishing the same goal is the use of cascading style sheets and described in Version 3.2 of the HTML standard.

Use all stylistic elements in moderation

Be a stoic Greek, "All things in moderation." The use of too many headers gets old and the reader feels like you are shouting. Too many emphasized elements loose their distinction. To many graphics take too long to download no matter how fast your computer or network connection is.

Examples

- Consumer Information Center
- New York Times on the Web
- HotWired

See Also

1. Dave Raggett, "Introducting HTML 3.2" - "HTML 3.2 adds widely deployed features such as tables, applets and text flow around images, superscripts and subscripts while providing backwards compatibility with the existing standard HTML 2.0." <URL:http://www.w3.org/pub/WWW/MarkUp/Wilbur/>
2. Dave Raggett, "HyperText Markup Language (HTML) " - This is a useful guide to other HTML pages. <URL:http://www.w3.org/pub/WWW/MarkUp/>
3. HTML Writers Guild, "Advice for HTML Authors" - "This is a list of advice for HTML authors, aimed at helping people produce quality HTML. It is intended to educate HTML authors to the elements of good and bad HTML style, focusing on some common problems with current HTML on the Web. It does not seek to ``control" Guild members, but rather to encourage them to adopt these practices in their everyday HTML construction." <URL:http://ugweb.cs.ualberta.ca/~gerald/guild/style.html>
4. HTML Writers Guild, "HTML Writers Guide Website" - "Welcome to The HTML Writers Guild Website, the first international organization of World Wide Web page authors and Internet Publishing professionals. Guild members have access to resources including: HTML and Web business mailing lists, information repositories, and interaction with their peers." <URL:http://www.hwg.org/>
5. James "Eric" Tilton, "Composing Good HTML" - "This document attempts to address stylistic points of HTML composition, both at the document and the web level." <URL:http://www.cs.cmu.edu/~tilt/cgh/>
6. Jan V. White, Graphic Design for the Electronic Age, (Watson-Guptill : New York 1988)
7. Kevin Werbach, "Bare Bones Guide to HTML" - "The Guide lists every tag in the official HTML 3.2 specification, plus the Netscape extensions, in a concise, organized format." <URL:http://werbach.com/barebones/>
8. Microsoft, "Microsoft Site Builder Workshop: Authoring" - This set of pages outline how to take advantage of HTML extentions with Microsoft's Internet Explorer. <URL:http://www.microsoft.com/workshop/author/>
9. Microsoft, "Microsoft Site Builder Workshop: Design/Creative" - Here you will find examples of page layout possibilities for HTML and Internet Explorer. <URL:http://www.microsoft.com/workshop/design/>

---

10. Mike Sendall, "HTML converters" - Here is a list of applications converting documents into HTML. <URL:http://www.w3.org/pub/WWW/Tools/Filters.html>
11. NCSA, "Beginner's Guide to HTML" - "The guide is used by many to start to understand the hypertext markup language (HTML) used on the World Wide Web. It is an introduction and does not pretend to offer instructions on every aspect of HTML. Links to additional Web-based resources about HTML and other related aspects of preparing files are provided at the end of the guide." <URL:http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html>
12. Netscape Communications Corporation, "Creating Net Sites" - This is a page to Netscape HTML extensions. <URL:http://home.netscape.com/home/how-to-create-web-services.html>
13. Robin Williams, The Non-Designer's Design Book (Peach Pit Press: Berkeley CA 1994)
14. Roy Paul Nelson, Publication Design, 5th ed. (Wm. C Brown: Debuque IA 1991)
15. Tim Berners-Lee, "Style Guide for online hypertext" - "This guide is designed to help you create a WWW hypertext database that effectively communicates your knowledge to the reader." <URL:http://www.w3.org/pub/WWW/Provider/Style/Overview.html>
16. W3, "HyperTetxt Design Issues" - "This lists decisions to be made in the design or selection of a hypermedia information system. It assumes familiarity with the concept of hypertext. A summary of the uses of hypertext systems is followed by a list of features which may or may not be available. Some of the points appear in the Comms ACM July 88 articles on various hypertext systems. Some points were discussed also at ECHT90 . Tentative answers to some design decisions from the CERN perspective are included." <URL:http://www.w3.org/pub/WWW/DesignIssues/Overview.html>
17. Yahoo!, "HTML Editors (Yahoo!)" - This is a list of HTML editors and guides to other lists of editors. <URL:http://www.yahoo.com/Computers_and_Internet/Software/Internet/World_Wide_Web/HTML_Editors/>
18. Yale Center for Advanced Instructional Media, "Yale C/AIM WWW Style Manual" - This is one of the more scholarly treatments of the subject. <URL:http://info.med.yale.edu/caim/StyleManual_Top.HTML>

Browsability Means Logically Classifying Your Data and Information

Organizing your content.

As the size of your information system grows, so does the need to logically organize your data. This implies grouping conceptual sets of data with similar conceptual sets of data. Browsability becomes apparent when it is coupled with hypertext and logical groupings of information.

Advantages

A browsable information system has a number of advantages.

1. Readers see entire system at a glance
2. Knowledge of a vocabulary is not necessary
3. Like items are grouped together
4. Easy to navigate
5. Fosters serendipity
6. Stimulates thinking

Disadvantages

But a solely browsable system is not without its disadvantages as well.

1. Easy to get "lost"
2. Classification system may be foreign to reader
3. Classification breaks down as the quantity of information increases
4. Classification changes over time

An easily browsable system is logically organized by topics. Effective organization of your information is critical to the success of your server. This point cannot be overstated and bears repeating. Effective organization of your information is critical to the success of your server. If you want your server to be an effective information tool and you want people to use your server more than once, then it must be organized.

Philosophy of Classification

Classifying knowledge and bringing like things together have been fundamental aspects of at least Western culture since the before the Golden Age of Greece when philosophers systematized their ways of thinking. This classification process was their way of creating an intellectual cosmos from the apparent chaos of their experience. The process brought a sense of order to their disordered society. It provided a common ground for others to work from and to use as a basis for further discovery. Without organization intellectual anarchy rushes in to fill the void. Given enough time, eventually, no one is speaking the same intellectual language and communication breaks down. Stagnation sets in.

"Sour milk"

To make matters worse, "One person's cheese is another person's sour milk." In other words, one person's view of the world may not adequately represent the next person's view. Interpretations of the night sky represent an excellent example. Everybody in the northern or southern hemispheres have exactly the same data in which to make interpretations, the stars. Yet every culture creates their own distinct constellations and explanations about what they mean. The interpretations these people make reflect their culture. War-like cultures see warriors. Fishing cultures see boats. Wandering peoples see migrating animals.

A cultural example

For better or for worse, classification systems of knowledge ultimately break down. This is because cultural revolutions and technological change occur. For example, the Medices of the late Middle Ages used their newly found wealth to hire artisans and craftsman. These creative individuals explored new ways of looking at the human form and rekindled an interest in humanity. These cultural perception shifts were then picked up by others throughout Europe. At the same time, technologies like the telescope offered people like Galileo a new perspective of the skies. For example, he noticed Venus moves through phases just like our moon. These observations lead Galileo to believe Venus did not revolve around the Earth as previously thought but around the Sun instead. While these two phenomenon (the rekindled interest in humanity and observations of Venus) did not by themselves alter a system of knowledge, these phenomenon represented the beginning of major intellectual shift in Western thought. Specifically, these two phenomenon contributed to the Renaissance and Reformation where Western civilization's entire intellectual basis where shaken. Therefore, old ways of thinking always seem to give way to new ways of

thinking and the process begins anew. Put another way, there is no perfect intellectual organization of knowledge or information just as there is no perfect circle. Similarly, as the habits and technologies of societies change so does their interpretation of the "cosmos" they live in.

Back to the real world

The point is, despite the dynamic nature of intellectual constructs, the organization of information and knowledge seem to be a necessary part of human existence. Since the primary purpose of information servers is to disseminate knowledge, facts, and ideas, it then follows the information they disseminate must be organized in some reasonable fashion.

Guidelines

1. Know your audience
2. Provide an "about" text
3. Use the vocabulary of your intended audience
4. Create a hierarchial system of ideas
5. Create a system that is both flexible and exhaustive
6. Classify by format last

Know your audience

If your intended readers cannot make sense of your server's organizational scheme, then they will only use it as a last-resort information resource. Thus follows the first and foremost guideline for a useful organizational scheme. The organizational scheme must be comprehensible to your intended audience. Think about the people who will be using your server. What are their backgrounds? What do they want? What specialized terminology do they use? In general, how do they think? Incorporate the answers to these questions into the structure of your server. To paraphrase a respected librarian, "Servers are for use" and, in order for this to happen, your organizational scheme must be understandable by the majority of your intended clientele. A thesaurus listing the vocabulary of a discipline may be indispensable in this regard, especially for those of you who are creating collections of Internet resources.

Provide "about" texts

Embed "about" texts, texts describing your organizational system, its intended audience, and how it can be used, within as much of your system as possible. You can not include too much explanatory information as long as it stays within the guidelines for good readability.

Use the vocabulary of your readership

Once you have identified your intended audience, use their terminology. Thus your readership will identify with your information system and be more likely to use it again and again.

Create a hierarchial system of ideas

By definition, a hierarchial system of ideas begins with broad terms and is subdivided into narrower terms. There is no perfect hierarchial system of ideas, but your intended audience will bring to your system some preconceived ideas on how information on their topic should be organized. These people will have similar, but not exact preconceptions. Biologist think similarly, just as computer scientists think similarly. Create a hierarchial system fitting the preconceptions of as much of your intended audience as possible. This is called "literary warrant." At this point it may be helpful to employ the use of a specialized thesaurus or handbook written for your audience. This thesaurus or handbook will contain definitions of the discipline's terminology. These tools will help clarify and provide a structure for your hierarchial system.

Create a system that is both flexible and exhaustive

Make sure the system is exhaustive as well as flexible. In other words, create a structure striving to be both enumerative and synthetic.

> "Enumerative classification attempts to assign designations for (to enumerate) all the single and composite subject concepts required in the system. . . . Synthetic classifications are more likely to confine their explicit lists of designations to single, unsubdivided concepts, giving the local classifier generalized rules with which to construct headings of composite subject." [Wynar]

Classify by format last

Organize materials based on format as a last resort. People usually don't care what format the data is in just as long as the answer to their query can be found. This means when you have a collection of various Internet resources group them by subject first and Internet protocol last; do not put all the telnet sessions in one section, FTP sites in another section, gopher sites in a third, WAIS sites in a fifth, and so on. Organizing your information by topic rather than form brings like things together and end-users will not have to navigate throughout your server for the information they need.

Examples

- Consumer Information Center
- Yahoo
- A2Z

See Also

1. Aslib, Proceedings of the International Study Conference on Classification for Information Retrieval (London: Aslib, 1957)
2. Bohdan S. Wynar, Introduction to Cataloging and Classification (Libraries Unlimited: Littleton CO, 1980) pg. 394
3. Derek Langridge, Approach to Classification for Students of Librarianship (Hamden, Connecticut: Linnet Books, 1973)

Searchability Means Direct Information Access

Provide an index.

The largest of information systems must include search features.

Advantages

These features help overcome the disadvantages of the purely browsable system. They have a number of distinct advantages.

- Creates alternative logical classifications
- Simplifies location of known items
- Works independently of collection size

As described in the previous section, your conception of the information universe is not necessarily the same as your reader's. While you try to group things in the most logical manner, your reader's "logic" will be different than yours. Searchability can help over come this discrepancy by allowing the reader to create their own set of logically similar items.

Searchability readily lends itself to locating known items rather than making the reader browse down a number of menus to get what they want. Similarly, a reader may have used an item from you before and not put it into their hotlist. If you have searching mechanisms in place, then it may be easier for the reader to find the item again.

Searchability works independently of your collection's size. Browsable systems begin to break down after the collection becomes too large. The effectiveness of searching an information system is not directly impeded by the size of the system.

Disadvantages

Yet, purely searchable systems are not perfect either. Users:

- Must know searching syntax
- Must have a preconceived idea, phrase, or term
- Must know the structure of the data

In order to effectively search an information system, the reader must know the query language of the search engine. This may include Boolean logic or Unix regular expressions. They may have to know the meaning of right-hand truncation and the symbol for its use. While this sort of knowledge is necessary to use the system, it is irrelevant to the information itself and is seen by the reader as an impediment.

The ability to search an information system assumes your readership has a preconceived idea describing what they need. This is a notorious example of the chick and egg problem. How are you suppose to find information about a particular topic if you don't know about that topic in the first place. In other words, the reader must come to the information with some terms or phrases describing want they want to find. Many times those terms or phrases will not be found in the collection, but synonyms will be found. It is difficult to think of many synonyms and it is difficult to "guess" at the controlled vocabulary used by the collection.

Finally, totally searchable systems require the searcher to know the data structure of the indexed collection. Is the data divided into fields? If so, what are those fields and how do they specify them in their query?

Guidelines

What to do? Here are some guidelines for creating searchable systems.

- Include help texts
- Map located items to similar items
- Provide simple as well as "power user" search mechanisms

Include help texts

Just as "about" texts are necessary in a browsable system, help texts are necessary for searchable systems. Help texts describe the features and limitations of the system. They list system's data structure including fields available for searching and the contents of those fields. Help texts also list plenty of example searches and provide explanations on what the end-user should do if they encounter too many or too few results.

Map located items to similar items

After items are located with the search mechanism, there should be links to similar items. This answers the perennial question, "Can you find me more items like this one?" These links should go directly back to your browsable collection where the end-user can freely "wander". From there the end-user will have the ability to see terms that can be applied to more searches. This is where you provide the end-user with the vocabulary terms of your system, in case they are unfamiliar with your system of information organization.

Provide simple as well as "power user" search mechanisms

Simple search mechanisms will be most useful for the first-time or casual end-user. Unfortunately, these same mechanisms often return too many or too few "hits". Providing power user search mechanisms like field searching, truncation, Boolean qualifiers, and number-of-term limitations can compensate for the simple searches. Unfortunately, the cost of these services is effectively describing the more powerful searching mechanisms to the end-user. Again, readability comes into play.

Software

Readability is by achieved by exploiting HTML. Browsability is most effectively and efficiently via datbase applications. Searchability is acquired through indexing HTML files or directly searching the contents of a database. Below is a non-exhaustive list of software to help accomplish the goals of browsability and searchability:

Unix
- database - Mini SQL with W3-mSQL
- indexing - Harvest or freewais-sf

Windows
• database - FileMaker Pro
• indexing - WebIndex, a port of SWISH that comes with WebSite

Macintosh
• database - FileMaker Pro with or without Lasso Lite
• indexing - Apple e.g. (if you can find it) or Search Server by Social Engineering

See Also

1. "Web Server Search for Windows" - "WSS is a CGI back-end for Windows based Web servers that allows your clients to conduct simple queries on html files in an unlimited number of directories. The output is a listing of links containing the title, heading, or file name of files that contain the search string. You simply modify the search.ini file for the directories you want users to search, and insert a form into your page that includes the number of directories to search, a reference to these directores and a submit button. WSS takes care of the rest." <URL:http://wgg.com/wgg/best/search.htm>
2. Chuck Shotton, "Using FileMaker Pro with MacHTTP" - An archive with sample forms and CGI that shows how to hook MacHTTP to FMPro. <URL:http://www.biap.com/machttp/examples/fmpro.sit.hqx>
3. Chuck Shotton, "Writing Search Engines for MacHTTP " - This points to an archive containing C source code for a sample application that performs searches in conjunction with MacHTTP using the "srch" AppleEvent. <URL:http://www.biap.com/machttp/ftp/search_ex.sit.hqx>
4. Glimpse Working Group, "Glimpse" - "Glimpse is a very powerful indexing and query system that allows you to search through all your files very quickly. It can be used by individuals for their personal file systems as well as by organizations for large data collections. Glimpse is the default search engine in Harvest." <URL:http://glimpse.cs.arizona.edu/>
5. Kevin Hughes, "SWISH Documentation" - "SWISH stands for Simple Web Indexing System for Humans. With it, you can index directories of files and search the generated indexes. For an example of swish can do, try searching for the words "office and map" at EIT. All of the search databases you see there were indexed by swish. When you do a search, it's the swish program that's doing the actual searching." <URL:http://www.eit.com/software/swish/swish.html>
6. Mic Bowman, et al., "Harvest Information Discovery and Access System" - "Harvest is an integrated set of tools to gather, extract, organize, search, cache, and replicate relevant information across the Internet. With modest effort users can tailor Harvest to digest information in many different formats from many different machines, and offer custom search services on the web." <URL:http://harvest.transarc.com/>
7. Yahoo!, "Gateways (Yahoo!)" - A collection on searching gateway scripts, as well as a number of CGI examples are found here. <URL:http://www.yahoo.com/Computers/World_Wide_Web/Gateways/>

Assistance

*Provide interactive help.**

The number of information systems we encounter in our "Knowledge Worker" lives is steadily increasing. Even for the people who deal with vasts amounts of information on a daily basis, the ability to understand and make sense of all these new information systems is dubious.

Consequently, even if your information system exemplifies the best of readability, browsability, and searchability, there will always be the people who still need help. Even the people who believe they can use the system(s) to their fullest potential may sometimes need assistance to achieve the highest ratio of precision to recall from their information gathering sessions. This is why the biggest and largest of information systems will require some sort of interactive assistance built into them.

This is nothing new. Libraries, huge information systems, have always had librarians. Shopping malls have "information desks." Even though you have a telephone book for your local area, how many times have you dialed directory assistance because the book was out-of-date?

The larges of information systems will require some sort of extra, interactive assistance in order for their services to be most effective. This interactive assistance could simply be a person at the other end of a telephone or an Internet Relay Chat. It could be an "expert system." In either case, there will need to be some sort of procedure where you can ask questions and/or the system can ask question of you. Based on the answers, other questions would be asked. At the end of the question-and-answer process some sort of game plan or alternative method for using the information system would be provided.

Interactive assistance can be proactive or reactive. Proactive interactive assistance queries users for the information needs. It analyses the answers to the queries and either formulates possible solutions to the information need or continues the query process. A reactive assistance model would only provide possible solutions after being asked questions by the users. The difference between these two models is similar to the difference between browsability and searchability. Both browsability and proactive assistance layout ready-made solutions or information paths. Searchability and reactive assistance require the users to articulate their needs and translate them into language of the system. Like browsability and searchability, ideally, elements of both proactive and reactive assistance are desirable in the implementation of any large, automated information system.

With the advent of the Internet, providing interactive assistance manifested itself through the use of e-mail. This has almost become a ubiquitous means for providing reference. Another, more innovative solution, was the creation of MOOs and MUDs. These text-based, virtual realities were populated by people with common needs and desires. MOOs and MUDs always seemed popular with the gaming crowd, but because of its text-based orientation it never seemed to cause very much of a stir.

The big thing these days is the World Wide Web. Everybody's individual WWW server represents an information system and the larger they get the more they necessitate interactive assistance to make them more useful. The first hints of interactive assistance efforts include relevance ranking and meta-search engines. While these applications hint at interactive assistance, they have a way to go before they provide anything remotely like "proactive" assistance.

Ask Alcuin, proto-typed by myself in 1995, is a Perl script whose purpose is to put into practice the concept of interactive assistance. Using a question-and-answer process similar to a reference interview, Ask Alcuin tries to identify a user's information need and translate that need into the query language of remote and local databases. The program is functional in that it executes correctly, but the method it uses convert information needs into queries needs to be updated and refined. Enhancements could include the log of user needs so repeat users do not have to re-build a profile with the application. Furthermore, the program needs to be updated in terms of the search engines it queries and how it queries them.

Meta-search engines are those Internet applications allowing you to input queries into a field, select various databases, and submit your query. The main strength of these applications is their ability to translate your query into something many databases understand. Examples include SavvySearch, All4one, Cyber411, Inference Find!, and MetaFind. These applications all work with

varying degrees of success. The better ones collate the results, remove the duplicates, and list the results is some sort of order. None of these engines help you refine your queries.

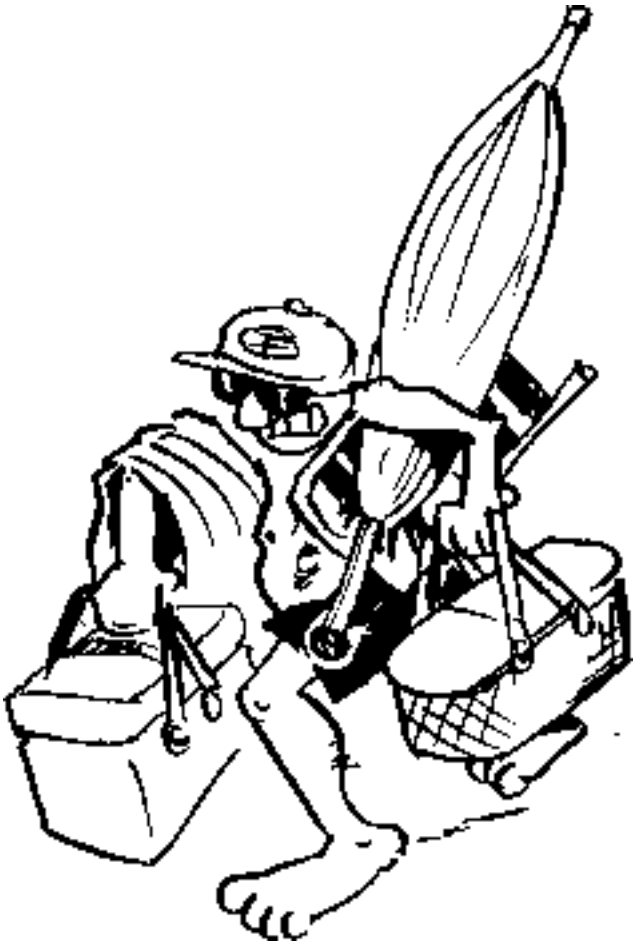
In summary, if librarians want to participate in providing information services electronically, then librarians must create useful electronic information systems. These systems, depending on their size, ought to exemplify varying degrees of readability, browsability, searchability, and interactive assistance. Interactive assistance, as specialized help for specialized situations, can take many forms: telephone conversations, video conferencing, or expert systems. The implementation of these electronic services will be a challenge for the profession, especially considering the current "do more with less" mind set, but with perseverance and determination we can make it happen.

This last element of Web page design is the most difficult to implement and quite possibly the most controversial, yet this element has always existed in previous information systems. Why not information systems on the Internet?

* This section borrows heavily from Eric Lease Morgan, "Creating User-Friendly Electronic Information Systems." *Computers in Libraries* v17n8 (September 1997) pgs. 31-32.

Examples

- SavvySearch
- Ask Alcuin



WWW Scripting

These sections provide information about CGI scripting, using Perl as an example.

One of the most powerful features of HTTP servers lies in their ability to run programs behind the scenes and return the results of these programs to the client. This is known as common gateway interface (CGI) scripting. Basic CGI scripts include the ability to display the current time or the number of users who have accessed a server. More advanced and useful CGI scripts allow readers to search databases, complete survey forms and have them sent to an email address, implement imagemaps, or take pictures of remote places and have the pictures returned.

CGI scripts are made available to a Web browser through the use of simple links, specialized URLs containing a question marks (?), `ISINDEX` HTML tags, or HTML+ FORMs. After the user submits an HTML document containing one of these elements, a query is passed to the HTTP server, which is passed on to the CGI script itself. The script processes the input, formats the output into HTTP codes and/or an HTML document, and returns these items back to the HTTP server. The server then passes it along to the client application.

CGI scripts can be written in almost any language. Common languages include C, Perl, AppleScript, VisualBasic, and Unix shell scripts. This section outlines how to write simple CGI scripts using Perl. Perl is freely distributed, runs on just about any computing platform, includes functions for doing TCP communications, and has complete string manipulation functions. Most importantly, Perl is has one of the best support groups available anywhere, the Internet.

Keep in mind that Perl may not be the best CGI scripting language for your particular need. It is always a good idea to use the best tool for your particular job and other scripting languages may provide more useful hooks to your server's operating system.

See Also

1. "Overview of CGI" - "This page contains pointers to information and resources on the Common Gateway Interface, a standard for the interface between external gateway programs and information servers." <URL:http:// www.w3.org/hypertext/WWW/CGI/Overview.html>
2. "FastCGI" - FastCGI is a new, open extension to CGI that provides high performance for all Internet applications without any of the limitations of existing Web server APIs. <URL:http:// www.fastcgi.com/>
3. "Perl Language Home Page" - This is the offical home page for Perl <URL:http:// www.perl.com/perl/>
4. "[Perl Ports]" - This will take you to a randomly chosen FTP site hosting the Macintosh and Windows-based ported versions of Perl. <URL:http:// www.perl.com/CPAN/ports/>
5. O'Reilly & Associates, Inc., "Software Library - Extras" - Here is a set of CGI resources specicically for WebSite. <URL:http:// software.ora.com/techsupport/software/extras.html>
6. Chuck Shotton, "Using FileMaker Pro with MacHTTP" - An archive with sample forms and CGI that shows how to hook MacHTTP to FMPro. <URL:http:// www.biap.com/machttp/examples/fmpro.sit.hqx>
7. Chuck Shotton, "Writing Search Engines for MacHTTP " - This points to an archive containing C source code for a sample application that performs searches in conjunction with MacHTTP using the "srch" AppleEvent. <URL:http:// www.biap.com/machttp/ftp/search_ex.sit.hqx>
8. Danny Goodman, Complete AppleScript Handbook (Random House: New York, 1994)
9. Dave Winer, "Frontier Community Center" - "Frontier is a scripting system for the Macintosh. Lots of features, lots of verbs. It used to be a commercial product, but now it's free. Why? Because I want Frontier to have a shot at becoming a standard. I think it'll be fun!" <URL:http:// www.scripting.com/frontier/>
10. Derrick Schneider, Tao of AppleScript (Hayden Books: Carmel, IN, 1993)
11. Gisle Aas, "LIBWWW-PERL-5" - "The libwww-perl distribution is a collection of Perl modules which provides a simple and consistent programming interface (API) to the World-Wide Web. The main focus of the library is to provide classes and functions that allow you to write WWW clients, thus libwww-perl said to be a WWW client library. The library also contain modules that are of more general use." <URL:http:// www.sn.no/libwww-perl/>
12. John G. Cope, "Win-httpd CGI-DOS" - Here is a wrapper for Perl scripts written for DOS/Windows machines. <URL:http:// www.achilles.net/~john/cgi-dos/>
13. Lincoln D. Stein, "CGI.pm" - This is Perl 5 module for creating and processing HTML+ forms. <URL:http:// www.genome.wi.mit.edu/ftp/pub/software/WWW/cgi_docs.html>
14. Lincoln D. Stein, "CGI::* Modules for Perl5" - Here is a collection of Perl libraries for creating Perl-based CGI scripts. <URL:http:// www.genome.wi.mit.edu/WWW/tools/scripting/CGIperl/>
15. Matt Wright, "Matt's Script Archive" - A collection of Perl scripts as well as lists of other Perl script collections. <URL:http:// worldwidemart.com/scripts/>

16. Matthias Neeracher, "MacPerl and PCGI" - This points to the FTP archive for MacPerl and PCGI, a script inserting the necessary resources into a MacPerl script so it can be executed as a CGI script. <URL:ftp://err.ethz.ch/pub/neeri/MacPerl/>
17. Meng Weng Wong, "Index of Perl/HTML archives" - "This is a list of Perl scripts and archives involving HTML." <URL:http:// www.seas.upenn.edu/~mengwong/perlhtml.html>
18. NCSA, "Common Gateway Interface" - This is the official specification for CGI scripting. <URL:http:// hoohoo.ncsa.uiuc.edu/cgi/intro.html>
19. NCSA, "Mosaic for X version 2.0 Fill-Out Form Support" - This is the original specification for what was then called HTML+ FORMS. <URL:http:// www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/fill-out-forms/overview.html>
20. Robert Godwin-Jones, "Guide to Web Forms and CGI Scripts for Language Learning" <URL:http:// www.fln.vcu.edu/cgi/interact.html>
21. Roy Fielding, "WWW Protocol Library for Perl" - "libwww-perl is a library of Perl packages/modules which provides a simple and consistent programming interface to the World Wide Web. This library is being developed as a collaborative effort to assist the further development of useful WWW clients and tools." <URL:http:// www.ics.uci.edu/pub/websoft/libwww-perl/>
22. Sandra Silcot, "MacPerl Primer" - "This Primer is intended to assist new users get started with Macintosh Perl, and to point out salient differences for experienced Unix Perlers. This Primer is not a language reference manual, nor does it replace Matthias's documentation or Hal Wine's Frequently Asked Questions (FAQ) about MacPerl. The primer assumes you have already obtained and installed MacPerl, and that you have read the MacPerl FAQ." <URL:http:// www.unimelb.edu.au/~ssilcot/macperl-primer/home.html>
23. Selena Sol, "Selena Sol's Public Domain CGI Script Archive and Resource List" - "n the following pages we have included both working examples of our scripts as well as the text of the code so that you can have one window open with the code and the other with the working script. Hopefully this design will help you figure out how we did what we did, so that you can take the ideas and run with them for your own needs." <URL:http:// www.eff.org/~erict/Scripts/>
24. Selena Sol, "Selena Sol's Public Domain CGI Script Archive and Resource Library" - This is a very useful collection of free Perl scripts and libraries for use on our HTTP server. <URL:http:// www.eff.org/~erict/Scripts/>
25. StarNINE, "Extending WebSTAR" - This is an extensive list of scripts and "plug-ins" for WebSTAR. <URL:http:// www.starnine.com/development/extendingwebstar.html>
26. Steven E. Brenner, "cgi-lib.pl Home Page" - Here is a long list of Perl instruction books as well as documentation for cgi-lib.pl, a very popular Perl library for processing the input of HTML+ forms. <URL:http:// www.bio.cam.ac.uk/cgi-lib/>
27. Yahoo!, "Gateways (Yahoo!)" - A collection on searching gateway scripts, as well as a number of CGI examples are found here. <URL:http:// www.yahoo.com/Computers/World_Wide_Web/Gateways/>
28. Yahoo!, "CGI - Common Gateway Interface (Yahoo!)" - A large collection of CGI scripts. <URL:http:// www.yahoo.com/Computers_and_Internet/Internet/World_Wide_Web/ CGI___Common_Gateway_Interface/>

Hello, World!

The simplest of scripts.

It is a tradition to begin any programing demonstration with a "Hello, World!" example; an example that displays a simple text message. That is exactly what our first example does here. Give it a try:

Hello, World!

The script itself is only two lines long and saved as `01-helloWorld.cgi`:

```
#!/usr/local/bin/perl
print "content-type: text/html\n\nHello, World!";
```

Filenames

The first thing to keep in mind is the name of the script, specifically its extention (.cgi). In this case .cgi denotes an executable script to the HTTP server as defined in its MIME types table. If your script is not saved with a defined extenstion, then, by default, the content of the script will be returned to the client and not the result of the script's execution.

Wrappers

The first line of the script is a comment. All lines begining with a hash mark (#) are considered comments. On Unix platforms, if the first line of a script is a comment and followed by an exclamation point (! and sometimes called a "bang"), then the text following the exclamation point is the considered to be the application used to interpret the balance of the text. In this case, the application "`/usr/local/bin/perl`" is called. This convention is called a "wrapper." The Macintosh and Windows platforms also need wrappers and are available from the pages describing their respective Perl ports.

Output

The second line of the script represents the real guts of the demonstration:

1. The print statement outputs the content of everything between the quote markes and until the end of the line (`;`)
2. The `content-type: text/html\n` is the "magic line" necessary for your WWW browser to know how to handle the incoming text. ("Remember the HTTP headers the Introduction?").
3. A single carriage return (`\n`) delimits the HTTP header from its data just like Internet email (SMTP) messages.
4. Finally, the text Hello, World! is printed.

Creating Valid HTML

Writing polite code.

The previous example was not very polite. In fact, it out and out lied about its content saying its data was HTML when it was plain text. The second example demonstrates how to conform a bit more to HTTP and HTML standards by including more information about the script's content and correctly formating the output. Try the following script, `02-validCoding.cgi`:

Hello, World! #2

The script (below) has the exact functionality as the first example, but this one is more truthful about its output's content and it formats the results. View the source code of the output of Hello, World! and Hello, World! #2 to see the difference.

The Code

Here is the script's code:

```
#!/usr/local/bin/perl

# 02-validCoding.cgi
# This script demonstrates valid HTTP/HTML coding output.

# Eric Lease Morgan
# from Becoming a World Wide Web Server Expert
# http://sunsite.berkeley.edu/~emorgan/waves/
# 04/01/97 - renamed file as a .cgi
# 01/20/97 - Martin Luther King Day

# declare the MIME standard
$header  = "MIME-Version: 1.0\n";

# describe the MIME type
$header .= "Content-type: text/html\n";

# terminate the HTTP header
$header .= "\n";

# create an HTML file
$html    = "<html>\n";
$html   .= "<head>\n";
$html   .= "<title>\n";
$html   .= "Example #2 - Hello, World!\n";
$html   .= "</title>\n";
$html   .= "</head>\n";
$html   .= "<body>\n";
$html   .= "Hello, World!\n";
$html   .= "</body>\n";
$html   .= "</html>\n";

# output the header and html content
print "$header$html";

# exit gracefully
exit;
```

How It Works

Here is how the script works:

1. The first few lines of the script are comments. The first line is the script's wrapper. The next few lines provide some documentation.
2. The next few lines build the HTTP header (`$header`) describing the MIME version and type that is being returned.
3. Next, the variable `$html` is built including valid HTML tags.
4. Then the variables representing the HTTP header and HTML data are returned with a print statement.
5. Finally, the script quits with the `exit` statement.

Using Environment Variables

Getting and using rudimentary input.

Unless your scripts are intened to only display something like the date and time, you will be wanting to get some input from your users in order to create dynamic pages or do some other sort of processing.

As outlined in the Introduction, a proper WWW browser sends requests for data to an HTTP server. These requests also include information about the browser's computing environment. These things might include the Internet name and IP address of the browser's computer, what sort of data the browser can accept, the name of the browser, and the URL the browser is currently displaying. To Perl, this set of information is known as environment variables and can be quite useful in CGI scripting.

Example #1

The following example (`03-environment.cgi`) extracts the data sent by the WWW browser and then creates some simple output based in this data:

"Show me my environment"

As the script demonstrates, quite a lot of useful information is available to CGI scripts through the environment variables. Based on this information alone a person could create simple authentication scripts, or dynamic HTML pages based on the client's operating system or the preferred image format.

Example #2

By appending a question mark (?) and some text to the script's URL, an HTML author can supply more input for the exact same script as demonstrated below. (Notice how the value of query string changes from Example #1 and Example #2.)

Rudimentary HTML input

The Code

```
#!/usr/local/bin/perl

# 03-environment.cgi
# This script outputs the environment variables.

# Eric Lease Morgan
# from Becoming a World Wide Web Server Expert
# http://sunsite.berkeley.edu/~emorgan/waves/
# 04/01/97 - renamed file as a .cgi
# 01/20/97 - Martin Luther King Day

# create the http header
$header  = "MIME-Version: 1.0\n";
$header .= "Content-type: text/html\n";
$header .= "\n";

# initalize the html output
$html  = "<html>\n";
$html .= "<head>\n";
$html .= "<title>\n";
$html .= "Example #3 - Environment variables\n";
$html .= "</title>\n";
$html .= "</head>\n";
$html .= "<body>\n";

# using brute force, extract each environment variable
$html .= "Environment variables\n";
$html .= "<ol>\n";
$html .= "<li><b>server software</b>:    $ENV{SERVER_SOFTWARE}\n";
$html .= "<li><b>gateway interface</b>:$ENV{GATEWAY_INTERFACE}\n";
$html .= "<li><b>server protocol</b>:    $ENV{SERVER_PROTOCOL}\n";
$html .= "<li><b>server name</b>:        $ENV{SERVER_NAME}\n";
$html .= "<li><b>server port</b>:        $ENV{SERVER_PORT}\n";
$html .= "<li><b>authorization type</b>: $ENV{AUTH_TYPE}\n";
$html .= "<li><b>remote user</b>:        $ENV{REMOTE_USER}\n";
$html .= "<li><b>remote address</b>:     $ENV{REMOTE_ADDR}\n";
$html .= "<li><b>remote host</b>:        $ENV{REMOTE_HOST}\n";
$html .= "<li><b>remote identity</b>:    $ENV{REMOTE_IDENT}\n";
$html .= "<li><b>request method</b>:     $ENV{REQUEST_METHOD}\n";
$html .= "<li><b>script name</b>:        $ENV{SCRIPT_NAME}\n";
$html .= "<li><b>path</b>:               $ENV{PATH_INFO}\n";
$html .= "<li><b>path translation</b>:   $ENV{PATH_TRANSLATED}\n";
$html .= "<li><b>query string</b>:       $ENV{QUERY_STRING}\n";
$html .= "<li><b>content type</b>:       $ENV{CONTENT_TYPE}\n";
$html .= "<li><b>content length</b>:     $ENV{CONTENT_LENGTH}\n";
$html .= "<li><b>http accept</b>:        $ENV{HTTP_ACCEPT}\n";
$html .= "<li><b>http user agent</b>:    $ENV{HTTP_USER_AGENT}\n";
$html .= "<li><b>http referer</b>:       $ENV{HTTP_REFERER}\n";
```

```
$html .= "<li><b>http cookie</b>:        $ENV{HTTP_COOKIE}\n";
$html .= "</ol>\n";

# add a cute line demonstrating the use of one such variable
$html .= "Hello there, $ENV{REMOTE_HOST} It's nice to meet
you!\n";

# finish the html
$html .= "</body>\n";
$html .= "</html>\n";

# output the header and html content
print "$header$html";

# exit gracefully
exit;
```

How It Works

The operation of this script is not much different from the operations of the previous examples:

1. Like the previous examples, the script begins with a wrapper and some documentation.
2. It then builds the HTTP header
3. The creation of the HTML is divided into four parts
   i. Like before, the HTML is initialized with standard codes
   ii. Next each value in the array `%ENV` is extracted and added to the HTML
   iii. A simple line is generated including the value of one of the environment variables
   iv. The HTML codes is closed
4. The HTTP header and HTML data returned to the server
5. The script quits

Getting Input From Forms

FORMs allow for user-friendly input.

In the "old days," there was the `<ISINDEX>` HTML tag. By inserting this tag into the HEAD of your HTML documents, a text field appeared on your page allowing you to supply input for a script. The technique still works but is limiting. The developers at NCSA created the CGI specification providing the means for getting a wider variety of input called FORMs. At the time, this new development was called HTML+.

FORMs provide multiple types of user input including:

- check boxes
- hidden variables
- pop-up menus
- radio buttons
- scrolling lists where one or more items can be selected
- single and multiple line text fields

Each of these input elements assign data to programer-defined variables in the FORM. When the FORM is "submitted", these variables and their contents are sent to your CGI script for processing.

Each FORM must have a begining and ending FORM tag. The FORM tag must have an ACTION attribute. The ACTION attritute is a URL pointing to your CGI script. The FORM tag optionally includes a METHOD attribute whose value is either GET, POST, or PUT. The default value is GET.

The differences between GET and PUT are subtle. FORMs using GET can send a limited amount of data to the remote script and these FORMs display their contents as a URL in your browser's location field. FORMs using POST can send much more data to the remote script and do not display their contents as URLs. The PUT method is not widely implemented by HTTP servers yet. It is used to copy files from your computer to the host.

Examples #1 and #2 (below) illustrate some of the subtle differences between GET and POST. Both examples have an ACTION attribute pointing to the previous script in the "Using Environment Variables" section. Notice the differences in output.

Examples #3 and #4 use the same GET and POST techniques as #1 and #2, but these scripts produce output that begins to be useful.

Example #1

This FORM's ACTION attribute points to the environment variable script from the "Using Environment Variables" section and its METHOD attribute is GET.

Name?
Address?

Example #2

This FORM's ACTION attribute points to the environment variable script from the "Using Environment Variables" section and its METHOD attribute is POST.

Name?
Address?

Example #3

This FORM's ACTION attribute points to a script that takes the form's input and does some simple processing. It's METHOD attribute is GET.

Name?
Address?

Example #4

This FORM's ACTION attribute does the same processing as the example above. It's METHOD attribute is POST.

Name?
Address?

The Code

```
#!/usr/local/bin/perl

# 04-gettingInput.cgi
# This script handles input via GET and PUT from forms.
# It also does some simple processing
# on the REMOTE_HOST environment variable.

# Eric Lease Morgan
# from Becoming a World Wide Web Server Expert
# http://sunsite.berkeley.edu/~emorgan/waves/
# 04/01/97 - renamed file as a .cgi
# 01/20/97 - Martin Luther King Day

# include a cgi processing library.
# consider also cgi.pm
require "cgi-lib.pl";

# extract the input from the form
# and put it into an array, @input
&ReadParse (*input);
$n = $input {'name'};
$a = $input {'address'};

# determine whether or not the user is from NCSU
$ncsu = 0;
$h = $ENV{REMOTE_HOST};
if ($h =~ /\.ncsu\./i) {$ncsu = 1}
if ($h =~ /^152\./)    {$ncsu = 1}

# create the http header
$header  = "MIME-Version: 1.0\n";
$header .= "Content-type: text/html\n";
$header .= "\n";

# start the html
$html  = "<html>\n";
$html .= "<head>\n";
$html .= "<title>\n";
$html .= "Example #4 - Getting input from FORMs\n";
$html .= "</title>\n";
$html .= "</head>\n";
$html .= "<body>\n";
$html .= "<p>You said your name was <b>$n</b> and your email was
<b>$a</b>.</p>\n";
```

```
$html .= "\n";

# echo whether or not they are from NCSU
if ($ncsu) {
  $html .= "<p>According to your computer's Internet name or IP
address, ";
  $html .= "you <b>are</b> a student, staff, or faculty member of
NCSU.</p>\n";
  }
else {
  $html .= "<p>According to your computer's Internet name or IP
address, ";
  $html .= "you <b>are not</b> a student, staff, or faculty member
of NCSU.</p>\n";
  }

# finish the html
$html .= "</body>\n";
$html .= "</html>\n";

# output the header and html
print "$header$html";

# quit gracefully
exit;
```

How It Works

This example builds on the previous examples:

1. The script is initialized and documented.
2. A perl library (`cgi-lib.pl` and described below) is required.
3. The contents of the form are parsed (`&ReadParse (*input);`) by the library and the results are assigned values (`$n = $input {'name'}; and $a = $input {'address'};`).
4. The client's host name is determined (`$h = $ENV{REMOTE_HOST};`).
5. Using regular expressions, the host name is evaluated for a specific strings (`if ($h =~ /\.ncsu\./i) {$ncsu = 1} and if ($h =~ /^152\./) {$ncsu = 1}`).
6. The HTML data is initialized and built as in the previous examples.
7. The HTTP header and HTML data are returned to the server.
8. The script exits.

Obviously, this is script is much more complicated than the previous examples, but this is the first script that approaches real-world applicability. The most important difference between this script and the previous examples is the inclusion of the Perl library, cgi-lib.pl. This library, written by Steve Brenner, removes much of the complexity of CGI scripting by decoding and parsing the input of forms. CGI.PM, another more robust and full-featured CGI scripting library by Lincoln Stein, is an alternative to cgi-lib.pl and takes advantage of Perl5's object oriented nature. Either library is an indespensible tool for your CGI scripting needs.

Server Maintenance

Here methods for maintaining URL integrity and logfile analysis are discussed.

Believe it or not, the easy part of HTTP servers is bringing them up in the first place. The hard part is making them run smoothly after the initial installation. This is akin to maintaining your OPAC's database, weeding your collection, refining your bibliographic instruction techniques, and generating reports on usage. Because of this, truely useful HTTP servers are sometimes few and far between. Because of this it takes a commitment by your institution to not only purchase any necessary hardware, but more importantly, commit time to the server's upkeep.

See Also

1. Boutell.Com, Inc., "Wusage" - "Wusage is a statistics system that helps you determine the true impact of your web server. By measuring the popularity of your documents, as well as identifying the sites that access your server most often, wusage provides valuable marketing information. Practically all organizations, whether commercial or educational or nonprofit, need solid numbers to make credible claims about the World Wide Web. Wusage fills that need." <URL:http://www.boutell.com/wusage/>
2. Gisle Aas, "LIBWWW-PERL-5" - "The libwww-perl distribution is a collection of Perl modules which provides a simple and consistent programming interface (API) to the World-Wide Web. The main focus of the library is to provide classes and functions that allow you to write WWW clients, thus libwww-perl said to be a WWW client library. The library also contain modules that are of more general use." <URL:http://www.sn.no/libwww-perl/>
3. Roy Fielding, "wwwstat and splitlog" - "The wwwstat program will process a sequence of HTTPd common logfile format (CLF) access_log files and output a log summary in HTML format suitable for publishing on a website. The splitlog program will process a sequence of CLF (or CLF with a prefix) access_log files and split the entries into separate files according to the requested URL and/or vhost prefix." <URL:http://www.ics.uci.edu/WebSoft/wwwstat/>
4. Roy Fielding, "MOMSpider: Mulit-owner Maintenance Spider" - "MOMspider is a web-

roaming robot that specializes in the maintenance of distributed hypertext infostructures (i.e. wide-area webs). The program is written in Perl and, once customized for your site, should work on any UNIX-based system with Perl 4.036."
<URL:http://www.ics.uci.edu/pub/websoft/MOMspider/>

URL Integrity

This section describes how to use a link checker named MOMspider.

There is nothing more frustrating to an Internet surfer than error "404", file not found. The dynamic nature of the Internet make the elimination of this error a challenge, to say the least. This is why it is imperitive for you to constantly check the validity of your links. This is especially true if your site collects pointers to other sites.

There are a number of free and fee link checkers available on the Internet. One of the very first and still quite useful is MOMspider. Written by Roy Fielding in 1994, MOMSpider or "Multi-Owner Maintenance spider", transverses your WWW site reporting on broken and redirected HTTP-based links. It is written in Perl, and therefore an application available for just about any operating system.

To get MOMSpider up and running, you must:

1. Install Perl
2. Install libwww-perl (or LWP), a library Perl-based Internet routines
3. Install MOMSpider
4. Write MOMSpider instruction files

Installing Perl is something you may have already have on your computer. Installing libwww-perl is merely a matter of downloading the archive and running the make command. Similarly, installing MOMspider is just as easy. Just download the archive.

The most difficult thing in using MOMspider is the creation of instruction files. Instruction files are files describing what sets of HTML pages should be checked and how to report on what it finds. Below is an simple instruction file to check the validity of the URL at <URL:http://sunsite.berkeley.edu/~emorgan/morganagus/serial-list.html>

Example

```
# This is a simple MOMspider instruction file
# intended to check for broken links in Index Morganagus

# Eric Lease Morgan
# 02/16/97 - first cut

AvoidFile   /home/emorgan/.momspider-avoid
SitesFile   /home/emorgan/.momspider-sites
<Tree
  Name    Index Morganagus
  TopURL
```

```
http://sunsite.berkeley.edu/~emorgan/morganagus/serial-list.html
  IndexURL
http://sunsite.berkeley.edu/~emorgan/morganagus/spider-report.html
  IndexFile     /home/emorgan/public_html/morganagus/spider-
report.html
  EmailAddress  eric_morgan@ncsu.edu
  EmailBroken   eric_morgan@ncsu.edu
  EmailRedirected eric_morgan@ncsu.edu
>
```

Explanation

In a nutshell, this instruction file tells MOMspider to:

1. Avoid transfering anything found in the .momspider-avoid file
2. Keep track of visited sites in the .momspider-sites file
3. Begin checking links in http:// sunsite.berkeley.edu/~emorgan/morganagus/serial-list.html
4. Create a report named http:// sunsite.berkeley.edu/~emorgan/morganagus/spider-report.html whose local file name is /home/emorgan/public_html/morganagus/spider-report.html
5. Send everything to eric_morgan@ncsu.edu

Once run, MOMspider creates a report and sends summary information to the specified email address looking something like this:

```
This message was automatically generated by MOMspider/1.00 after a
web traversal on Sun, 16 Feb 1997 20:08:30

The following parts of the Index infostructure may need
inspection:

Broken Links:
<http://timon.sir.arizona.edu/pubs/olive.html>

For more information, see the index at
<http://sunsite.berkeley.edu/~emorgan/morganagus/spider-
report.html>

Examining the data at <URL:http://
sunsite.berkeley.edu/~emorgan/morganagus/spider-report.html> then
provides more detailed information.
```

MOMspider works; it does exactly what it was designed to do. Run regularly, it can help sigificantly with the integrity of your WWW server.

Another alternative is the installation of a PURL (Persistant URL) server. See <URL:http://purl.oclc.org>. The PURL server, written and freely distributed by OCLC, is an HTTP server mapping real URLs to virtual URLs. It works much like the Internet names assigned to computers allowing you to keep your URLs (PURLs) constant and only updating your database mapping of your virtual URLs to real URLs.

Analyzing Log Files

This section outlines methods for doing rudimentary logfile analysis.

HTTP servers generate a lot of logfiles. Depending on your server, it may generate lists of what was accessed, who accessed it, and with what browser. In a nutshell, there are basically two ways to analyse this information. The first and most popular is to apply some sort of analysis tool to your logfiles. Some of these tools include Wusage, Getstats, and wwwstat. But the most popular seems to be Analog.

The second approach is to import your log files into a database and then query the database to create reports. This second approach is less popular, more difficult to implement, but may give you more exact information concerning the use of your server. To make the job a bit easier, you may want to try tabulate, a perl script that outputs tab-delimited text from "common log format" logfiles. If you use Apache, then you can configure it to create tab-delimited files automtically.

ANALOG

Analog is an application that can analyse your log files. It runs on Unix, Windows, and Macintosh computers. It can generate HTML or plain text output. All of its options are compiled into the application, can be overridden through a configuration file, or even the command line. Its fast and its free.

The most common structure of HTTP server logfiles is the "common logfile format." This format has the following structure:

```
remotehost rfc931 authuser [date] "request" status bytes
```

Where:

- `remotehost` is the name of the computer accessing your server
- `rfc931` is the name of the remote user (usually blank)
- `authuser` is the authenticated user
- `[date]` is... the date
- `"request"` is the URL requested from the server
- `status` is the error code generated from the request
- `bytes` is the size (in bytes) of the data returned

Analog can read the "common logfile format" (as well as others) and generate reports accordingly.

To use Analog, first you must download and uncompress the archive from the Analog home page or any of its many mirror sites.

If you are using a Unix computer, you will have to edit the analog.h file to define the application's defaults. You must then make (compile) the application. Don't fret. Its easy. The Windows and Macintosh version come pre-compiled and require little extra configuration.

The next step in using Analog is the editing of its configuration file, `analog.cfg`. This file directs Analog how to process your logfiles. The most important option is `LOGFILE` telling Analog the exact location of the file(s) to analyse. The next most important option is `OUTFILE` telling Analog

where to save its output. You will also want to edit `HOSTNAME`, `HOSTURL`, and `BASEURL` so your resulting reports make sense.

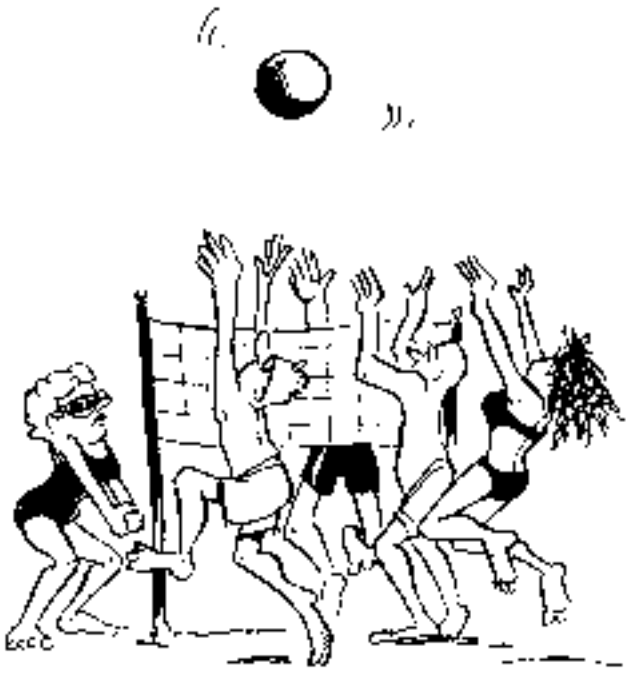
Next, running Analog will examine your logfile, politely report any errors, create your report, and exit. Furthermore, it will do this quickly!

After playing with Analog for a little while, you may want to explore fine tuning some of its miriade of options thus customizing your reports to your needs. Such options include dates, times, host name exclusion, text/HTML graphic/text output, and browser types, etc.

Analog is worth much more than what you will pay for it.

See Also

1. Boutell.Com, Inc., "Wusage" - "Wusage is a statistics system that helps you determine the true impact of your web server. By measuring the popularity of your documents, as well as identifying the sites that access your server most often, wusage provides valuable marketing information. Practically all organizations, whether commercial or educational or nonprofit, need solid numbers to make credible claims about the World Wide Web. Wusage fills that need." <URL:http:// www.boutell.com/wusage/>
2. Eric Lease Morgan, "tabulate" - A rudimentary Perl script that takes "common log format" logfiles and outputs tab-delimited text. <URL:http:// www.lib.ncsu.edu/staff/morgan/tabulate.txt>
3. Kevin Hughes, "Getstats" - "Getstats (formerly called getsites) is a versatile World-Wide Web server log analyzer. It takes the log file from your CERN, NCSA, Plexus, GN, MacHTTP, or UNIX Gopher server and spits back all sorts of statistics." <URL:http:// www.eit.com/software/getstats/getstats.html>
4. Roy Fielding, "wwwstat and splitlog" - "The wwwstat program will process a sequence of HTTPd common logfile format (CLF) access_log files and output a log summary in HTML format suitable for publishing on a website. The splitlog program will process a sequence of CLF (or CLF with a prefix) access_log files and split the entries into separate files according to the requested URL and/or vhost prefix." <URL:http:// www.ics.uci.edu/WebSoft/wwwstat/>
5. Stephen Turner, "Analog: A WWW Server Logfile Analysis Program" - Analog seems to be the most popular logfile analysis program. It is available for Unix, Windows, and Macintosh computers. Its fast and flexible, but just a tiny bit difficult to configure. <URL:http:// www.statslab.cam.ac.uk/~sret1/analog/>
6. W3, "Logging in W3C httpd" - This page describes the format of log files for the WC3 server, and specifically the "common log format." <URL:http:// www.w3.org/pub/WWW/Daemon/User/Config/Logging.html #common-logfile-format>
7. Yahoo!, "Log Analysis Tools (Yahoo!)" - Here is a collection of logfile applications and utilities. <URL:http:// www.yahoo.com/Computers_and_Internet/Software/Internet/ World_Wide_Web/Servers/Log_Analysis_Tools/>



People Connection

None of this happens without people and for people.

Technology does not exist in a vacuum. The purpose of bringing up HTTP services is to benefit your constituents (people). Additionally, your services won't go very far unless you have people to staff them. The following sections outline how you can staff your HTTP services and how you can use your OPAC and your HTTP services to the benefit of your patrons.

Staffing

*Dedicate staff resources as well as computer resources to your HTTP server initiatives.**

As stated previously, bringing up an HTTP server is easy. The hard part is maintaining it. This requires staff. If you plan to provide HTTP services, then plan for staff to manage them.

The World Wide Web is primarily a communications medium. For it to be most effective, it requires the skills of various professions. These skills include:

- computer networking and administration
- copy editing
- graphic design and illustration
- information collection and organization
- programing
- writing

Individuals possesing expert skills in more than one of these areas are few and far between. People possessing all of these skills are practially non-existant. Consequently, the staffing for robust HTTP services requires multiple personnel.

The ideal solution is to create a new department (a "Web Publishing Unit") and hire experts in each of the areas above to provide your HTTP services. A more practical approach would be to pull staff from existing departments who posses the needed skills and have these people work as a team.

At the very least, you will want your team to include:

- 1 graphic artist who understands HTML and illustration
- 1 editor who creates and modifies content
- 1 computer programmer/administrator who keep the machine running smoothly

If your organization is heirarchial, then you will want to consider adding a manager to the Web Publishing Unit to do supervision and maintain your institution's "vision."

HTTP services do not exist in a vacuum. It will be paramount for your Web Publishing Unit to communicate with other staff and its constituency. Therefore you might want to have a "Web Board" with liasons from each of your institution's departments. These people will bring to the table issues for implementation as well as content for your server.

Disadvantages

This model is not without its problems. First, you might not have the monetary resources to create a free standing unit responsible for server maintenance.

Second, this model would need to be created from scratch and it would not necessarily fit neatly into your current organizational structure. It would mean another hierarchy of some sort for staff to traverse.

Advantages

There are a number of advantages to this model. First, a freestanding unit like this one with several levels of expertise would assure the consistency and quality of the content while distributing the functions across several staff each with important and different roles.

Second, having a centralized unit would allow for the efficient purchase and use of highly specialized software: authoring tools, analysis software, graphic support equipment, link checkers, etc. Other staff throughout the library could concentrate on developing Web content themselves, discovering and helping disseminate other resources, connecting with faculty or other partners for collaborative opportunities and more, all without having to learn the tools in question. This is not unlike your current organizational structure which leaves certain tasks to a cadre of specialized staff with whom others interact in a known workflow.

Third, to say HTTP services represent a powerful communication medium is an understatement. The ever-increasing importance of the HTTP services to your institution's mission requires full-time commitment for the identical reasons that units such as reference, circulation, cataloging, and acquisitions do. Staying current with Web technologies, supervising staff assistants, interacting with staff and users, and pursuing the your institution vision are not elements to be passed on to either a committee or made partial responsibilities in an already existing job.

* This section borrows heavily from the NCSU Libraries internal document Eric Morgan, Keith Morgan, and Doris Sigl, "Taming the Web: Structured Web Management in the NCSU Libraries" June 1996.

Your OPAC

Your OPAC can form the founation for your HTTP services.

For decades card catalogs were the heart of library services. With the advent of computers the card catalogs were turned into online public access catalogs (OPACs). In most cases, these databases are still of fundamental importance to library services.

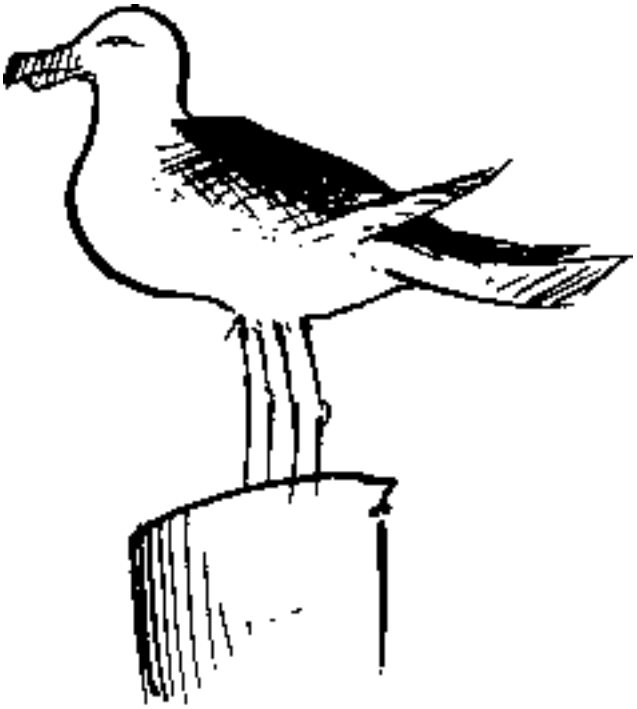
As you know, machine readable cataloging (MARC) records make up the content of your OPAC. In 1994 a new MARC field was defined, the 856 field for Electronic Location and Access. (See <URL:http:// www.oclc.org/oclc/bib/856.htm>.) This field describes:

> The information required to locate and retrieve an electronic item. Use to provide information that identifies the electronic location containing the items or from which the resource is available. In addition field 856 may be used for linking to an electronic finding aid.

One of the most useful subfields of 856 is subfield u. Subfield u is as place holder for the URL of an electronic resource. The existance of the 856 field and subfield u provide the means for cataloging Internet resources.

The development of CGI interfaces to OPACs provide the means to literally link your users with fulltext items from your catalog. One of the very first such interfaces was written by Tim Kambitsch for the DRA system. (See <URL:http:// dmcpl.dayton.lib.oh.us/~kambitsch/niso/lotf.html>.) Since then a number of vendors have created CGI interfaces to thier databases. An incomplete list is available at <URL:http:// www.lib.ncsu.edu/staff/morgan/alcuin/wwwed-catalogs.html>.

The combination of these standards and technologies allow you to take the "card catalog" where it hasn't been before. Now, more then ever it can become a finding tool as opposed to inventory list. Taking the process one step further, CGI scripts could be placed in 856 fields to provide access to more specialized resources. For example, a library could collect electronic texts, mark them up with SGML, catalog them, and provide access to the texts as well as searching mechanisms completely through the OPAC. Another option is to create a CGI script that download a MARC record in communications format from the OPAC to a librarian's desktop. This would allow libraries to share their MARC record without going through a bibliographic utility. If the MARC records in question described Internet resources, then this might even encourage more libraries to catalog electronic items.



Webliography

Browsability

1. Aslib, Proceedings of the International Study Conference on Classification for Information Retrieval (London: Aslib, 1957)
2. Bohdan S. Wynar, Introduction to Cataloging and Classification (Libraries Unlimited: Littleton CO, 1980) pg. 394
3. Derek Langridge, Approach to Classification for Students of Librarianship (Hamden, Connecticut: Linnet Books, 1973)

CGI Scripting

1. "Overview of CGI" - "This page contains pointers to information and resources on the Common Gateway Interface, a standard for the interface between external gateway programs and information servers." <URL:http:// www.w3.org/hypertext/WWW/CGI/Overview.html>
2. "FastCGI" - FastCGI is a new, open extension to CGI that provides high performance for all Internet applications without any of the limitations of existing Web server APIs. <URL:http:// www.fastcgi.com/>
3. "Perl Language Home Page" - This is the offical home page for Perl <URL:http:// www.perl.com/perl/>
4. "[Perl Ports]" - This will take you to a randomly chosen FTP site hosting the Macintosh and Windows-based ported versions of Perl. <URL:http:// www.perl.com/CPAN/ports/>
5. O'Reilly & Associates, Inc., "Software Library - Extras" - Here is a set of CGI resources

specicically for WebSite. <URL:http:// software.ora.com/techsupport/software/extras.html>
6. Chuck Shotton, "Using FileMaker Pro with MacHTTP" - An archive with sample forms and CGI that shows how to hook MacHTTP to FMPro. <URL:http:// www.biap.com/machttp/examples/fmpro.sit.hqx>
7. Chuck Shotton, "Writing Search Engines for MacHTTP " - This points to an archive containing C source code for a sample application that performs searches in conjunction with MacHTTP using the "srch" AppleEvent. <URL:http:// www.biap.com/machttp/ftp/search_ex.sit.hqx>
8. Danny Goodman, Complete AppleScript Handbook (Random House: New York, 1994)
9. Dave Winer, "Frontier Community Center" - "Frontier is a scripting system for the Macintosh. Lots of features, lots of verbs. It used to be a commercial product, but now it's free. Why? Because I want Frontier to have a shot at becoming a standard. I think it'll be fun!" <URL:http:// www.scripting.com/frontier/>
10. Derrick Schneider, Tao of AppleScript (Hayden Books: Carmel, IN, 1993)
11. Gisle Aas, "LIBWWW-PERL-5" - "The libwww-perl distribution is a collection of Perl modules which provides a simple and consistent programming interface (API) to the World-Wide Web. The main focus of the library is to provide classes and functions that allow you to write WWW clients, thus libwww-perl said to be a WWW client library. The library also contain modules that are of more general use." <URL:http:// www.sn.no/libwww-perl/>
12. John G. Cope, "Win-httpd CGI-DOS" - Here is a wrapper for Perl scripts written for DOS/Windows machines. <URL:http:// www.achilles.net/~john/cgi-dos/>
13. Lincoln D. Stein, "CGI.pm" - This is Perl 5 module for creating and processing HTML+ forms. <URL:http:// www.genome.wi.mit.edu/ftp/pub/software/WWW/cgi_docs.html>
14. Lincoln D. Stein, "CGI::* Modules for Perl5" - Here is a collection of Perl libraries for creating Perl-based CGI scripts. <URL:http:// www.genome.wi.mit.edu/WWW/tools/scripting/CGIperl/>
15. Matt Wright, "Matt's Script Archive" - A collection of Perl scripts as well as lists of other Perl script collections. <URL:http:// worldwidemart.com/scripts/>
16. Matthias Neeracher, "MacPerl and PCGI" - This points to the FTP archive for MacPerl and PCGI, a script inserting the necessary resources into a MacPerl script so it can be executed as a CGI script. <URL:ftp:// err.ethz.ch/pub/neeri/MacPerl/>
17. Meng Weng Wong, "Index of Perl/HTML archives" - "This is a list of Perl scripts and archives involving HTML." <URL:http:// www.seas.upenn.edu/~mengwong/perlhtml.html>
18. NCSA, "Common Gateway Interface" - This is the official specification for CGI scripting. <URL:http:// hoohoo.ncsa.uiuc.edu/cgi/intro.html>
19. NCSA, "Mosaic for X version 2.0 Fill-Out Form Support" - This is the original specification for what was then called HTML+ FORMS. <URL:http:// www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/fill-out-forms/overview.html>
20. Robert Godwin-Jones, "Guide to Web Forms and CGI Scripts for Language Learning" <URL:http:// www.fln.vcu.edu/cgi/interact.html>
21. Roy Fielding, "WWW Protocol Library for Perl" - "libwww-perl is a library of Perl packages/modules which provides a simple and consistent programming interface to the World Wide Web. This library is being developed as a collaborative effort to assist the further development of useful WWW clients and tools." <URL:http:// www.ics.uci.edu/pub/websoft/libwww-perl/>
22. Sandra Silcot, "MacPerl Primer" - "This Primer is intended to assist new users get started with Macintosh Perl, and to point out salient differences for experienced Unix Perlers. This Primer is not a language reference manual, nor does it replace Matthias's documentation or Hal Wine's Frequently Asked Questions (FAQ) about MacPerl. The primer assumes you have already obtained and installed MacPerl, and that you have read the MacPerl FAQ." <URL:http:// www.unimelb.edu.au/~ssilcot/macperl-primer/home.html>
23. Selena Sol, "Selena Sol's Public Domain CGI Script Archive and Resource List" - "n the

following pages we have included both working examples of our scripts as well as the text of the code so that you can have one window open with the code and the other with the working script. Hopefully this design will help you figure out how we did what we did, so that you can take the ideas and run with them for your own needs." <URL:http:// www.eff.org/~erict/Scripts/>

24. Selena Sol, "Selena Sol's Public Domain CGI Script Archive and Resource Library" - This is a very useful collection of free Perl scripts and libraries for use on our HTTP server. <URL:http:// www.eff.org/~erict/Scripts/>
25. StarNINE, "Extending WebSTAR" - This is an extensive list of scripts and "plug-ins" for WebSTAR. <URL:http:// www.starnine.com/development/extendingwebstar.html>
26. Steven E. Brenner, "cgi-lib.pl Home Page" - Here is a long list of Perl instruction books as well as documentation for cgi-lib.pl, a very popular Perl library for processing the input of HTML+ forms. <URL:http:// www.bio.cam.ac.uk/cgi-lib/>
27. Yahoo!, "Gateways (Yahoo!)" - A collection on searching gateway scripts, as well as a number of CGI examples are found here. <URL:http:// www.yahoo.com/Computers/World_Wide_Web/Gateways/>
28. Yahoo!, "CGI - Common Gateway Interface (Yahoo!)" - A large collection of CGI scripts. <URL:http:// www.yahoo.com/Computers_and_Internet/Internet/World_Wide_Web/CGI___Common_Gateway_Interface/>

Maintenance

1. Boutell.Com, Inc., "Wusage" - "Wusage is a statistics system that helps you determine the true impact of your web server. By measuring the popularity of your documents, as well as identifying the sites that access your server most often, wusage provides valuable marketing information. Practically all organizations, whether commercial or educational or nonprofit, need solid numbers to make credible claims about the World Wide Web. Wusage fills that need." <URL:http:// www.boutell.com/wusage/>
2. Gisle Aas, "LIBWWW-PERL-5" - "The libwww-perl distribution is a collection of Perl modules which provides a simple and consistent programming interface (API) to the World-Wide Web. The main focus of the library is to provide classes and functions that allow you to write WWW clients, thus libwww-perl said to be a WWW client library. The library also contain modules that are of more general use." <URL:http:// www.sn.no/libwww-perl/>
3. Roy Fielding, "wwwstat and splitlog" - "The wwwstat program will process a sequence of HTTPd common logfile format (CLF) access_log files and output a log summary in HTML format suitable for publishing on a website. The splitlog program will process a sequence of CLF (or CLF with a prefix) access_log files and split the entries into separate files according to the requested URL and/or vhost prefix." <URL:http:// www.ics.uci.edu/WebSoft/wwwstat/>
4. Roy Fielding, "MOMSpider: Mulit-owner Maintenance Spider" - "MOMspider is a web-roaming robot that specializes in the maintenance of distributed hypertext infostructures (i.e. wide-area webs). The program is written in Perl and, once customized for your site, should work on any UNIX-based system with Perl 4.036." <URL:http:// www.ics.uci.edu/pub/websoft/MOMspider/>

Readability

1. Dave Raggett, "Introducting HTML 3.2" - "HTML 3.2 adds widely deployed features such as tables, applets and text flow around images, superscripts and subscripts while providing backwards compatibility with the existing standard HTML 2.0." <URL:http://

www.w3.org/pub/WWW/MarkUp/Wilbur/>
2. Dave Raggett, "HyperText Markup Language (HTML) " - This is a useful guide to other HTML pages. <URL:http:// www.w3.org/pub/WWW/MarkUp/>
3. HTML Writers Guild, "Advice for HTML Authors" - "This is a list of advice for HTML authors, aimed at helping people produce quality HTML. It is intended to educate HTML authors to the elements of good and bad HTML style, focusing on some common problems with current HTML on the Web. It does not seek to ``control" Guild members, but rather to encourage them to adopt these practices in their everyday HTML construction." <URL:http:// ugweb.cs.ualberta.ca/~gerald/guild/style.html>
4. HTML Writers Guild, "HTML Writers Guide Website" - "Welcome to The HTML Writers Guild Website, the first international organization of World Wide Web page authors and Internet Publishing professionals. Guild members have access to resources including: HTML and Web business mailing lists, information repositories, and interaction with their peers." <URL:http:// www.hwg.org/>
5. James "Eric" Tilton, "Composing Good HTML" - "This document attempts to address stylistic points of HTML composition, both at the document and the web level." <URL:http:// www.cs.cmu.edu/~tilt/cgh/>
6. Jan V. White, Graphic Design for the Electronic Age, (Watson-Guptill : New York 1988)
7. Kevin Werbach, "Bare Bones Guide to HTML" - "The Guide lists every tag in the official HTML 3.2 specification, plus the Netscape extensions, in a concise, organized format." <URL:http:// werbach.com/barebones/>
8. Microsoft, "Microsoft Site Builder Workshop: Authoring" - This set of pages outline how to take advantage of HTML extentions with Microsoft's Internet Explorer. <URL:http:// www.microsoft.com/workshop/author/>
9. Microsoft, "Microsoft Site Builder Workshop: Design/Creative" - Here you will find examples of page layout possibilities for HTML and Internet Explorer. <URL:http:// www.microsoft.com/workshop/design/>
10. Mike Sendall, "HTML converters" - Here is a list of applications converting documents into HTML. <URL:http:// www.w3.org/pub/WWW/Tools/Filters.html>
11. NCSA, "Beginner's Guide to HTML" - "The guide is used by many to start to understand the hypertext markup language (HTML) used on the World Wide Web. It is an introduction and does not pretend to offer instructions on every aspect of HTML. Links to additional Web-based resources about HTML and other related aspects of preparing files are provided at the end of the guide." <URL:http:// www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html>
12. Netscape Communications Corporation, "Creating Net Sites" - This is a page to Netscape HTML extensions. <URL:http:// home.netscape.com/home/how-to-create-web-services.html>
13. Robin Williams, The Non-Designer's Design Book (Peach Pit Press: Berkeley CA 1994)
14. Roy Paul Nelson, Publication Design, 5th ed. (Wm. C Brown: Debuque IA 1991)
15. Tim Berners-Lee, "Style Guide for online hypertext" - "This guide is designed to help you create a WWW hypertext database that effectively communicates your knowledge to the reader." <URL:http:// www.w3.org/pub/WWW/Provider/Style/Overview.html>
16. W3, "HyperTetxt Design Issues" - "This lists decisions to be made in the design or selection of a hypermedia information system. It assumes familiarity with the concept of hypertext. A summary of the uses of hypertext systems is followed by a list of features which may or may not be available. Some of the points appear in the Comms ACM July 88 articles on various hypertext systems. Some points were discussed also at ECHT90 . Tentative answers to some design decisions from the CERN perspective are included." <URL:http:// www.w3.org/pub/WWW/DesignIssues/Overview.html>
17. Yahoo!, "HTML Editors (Yahoo!)" - This is a list of HTML editors and guides to other lists of editors. <URL:http:// www.yahoo.com/Computers_and_Internet/Software/Internet/World_Wide_Web/HTML_Editors/>

18. Yale Center for Advanced Instructional Media, "Yale C/AIM WWW Style Manual" - This is one of the more scholarly treatments of the subject. <URL:http:// info.med.yale.edu/caim/StyleManual_Top.HTML>

Searchability

1. "Web Server Search for Windows" - "WSS is a CGI back-end for Windows based Web servers that allows your clients to conduct simple queries on html files in an unlimited number of directories. The output is a listing of links containing the title, heading, or file name of files that contain the search string. You simply modify the search.ini file for the directories you want users to search, and insert a form into your page that includes the number of directories to search, a reference to these directores and a submit button. WSS takes care of the rest." <URL:http:// wgg.com/wgg/best/search.htm>
2. Chuck Shotton, "Using FileMaker Pro with MacHTTP" - An archive with sample forms and CGI that shows how to hook MacHTTP to FMPro. <URL:http:// www.biap.com/machttp/examples/fmpro.sit.hqx>
3. Chuck Shotton, "Writing Search Engines for MacHTTP " - This points to an archive containing C source code for a sample application that performs searches in conjunction with MacHTTP using the "srch" AppleEvent. <URL:http:// www.biap.com/machttp/ftp/search_ex.sit.hqx>
4. Glimpse Working Group, "Glimpse" - "Glimpse is a very powerful indexing and query system that allows you to search through all your files very quickly. It can be used by individuals for their personal file systems as well as by organizations for large data collections. Glimpse is the default search engine in Harvest." <URL:http:// glimpse.cs.arizona.edu/>
5. Kevin Hughes, "SWISH Documentation" - "SWISH stands for Simple Web Indexing System for Humans. With it, you can index directories of files and search the generated indexes. For an example of swish can do, try searching for the words "office and map" at EIT. All of the search databases you see there were indexed by swish. When you do a search, it's the swish program that's doing the actual searching." <URL:http:// www.eit.com/software/swish/swish.html>
6. Mic Bowman, et al., "Harvest Information Discovery and Access System" - "Harvest is an integrated set of tools to gather, extract, organize, search, cache, and replicate relevant information across the Internet. With modest effort users can tailor Harvest to digest information in many different formats from many different machines, and offer custom search services on the web." <URL:http:// harvest.transarc.com/>
7. Yahoo!, "Gateways (Yahoo!)" - A collection on searching gateway scripts, as well as a number of CGI examples are found here. <URL:http:// www.yahoo.com/Computers/World_Wide_Web/Gateways/>

Security

1. A.L. Digital Ltd., "Apache-SSL" - "Apache-SSL is a secure Webserver, based on Apache and SSLeay. It is licensed under a BSD-style licence, which means, in short, that you are free to use it for commercial or non-commercial purposes, so long as you retain the copyright notices. This is the same licence as used by Apache from version 0.8.15." <URL:http:// www.algroup.co.uk/Apache-SSL/>
2. Brigitte Jellinek, "bjellis perl scripts" - This page hosts a few scripts used to modify username/password combinations for basic HTTP authentication. <URL:http:// www.cosy.sbg.ac.at/www-doku/tools/bjscripts.html>
3. CERT, "COPS" - "COPS is a unix security toolkit that analyzes your system security."

<URL:ftp:// ftp.cert.org/pub/tools/cops/>
4. Lincoln D. Stein, "World Wide Web Security FAQ" - "It attempts to answer some of the most frequently asked questions relating to the security implications of running a Web server. There is also a short section on Web security from the browser's perspective." <URL:http:// www.genome.wi.mit.edu/WWW/faqs/www-security-faq.html>
5. Rutgers University Network Services www-security team, "World Wide Web Security" - "This document indexes information on security for the World Wide Web, HTTP, HTML, and related software/protocols." <URL:http:// www-ns.rutgers.edu/www-security/>
6. W3, "Access Authorization in WWW" - "This is the documentation of WWW telnet-level Access Authorization as implemented in October 1993 (Basic) scheme, part of the WWW Common Library). Contains also proposals for encryption level protection (Pubkey scheme proposal and RIPEM based proposal)." <URL:http:// www.w3.org/pub/WWW/AccessAuthorization/Overview.html>

Servers

1. "Apache HTTP Server Project" - "The Apache project has been organized in an attempt to answer some of the concerns regarding active development of a public domain HTTP server for UNIX. The goal of this project is to provide a secure, efficient and extensible server which provides HTTP services in sync with the current HTTP standards." <URL:http:// www.apache.org/>
2. "WebSite Central" - This is the official home page of WebSite. <URL:http:// website.ora.com/>
3. Brian Behlendorf, et al., "Running a Perfect Web Site with Apache" (Indianapolis, IN: Que, 1996) - "This book is designed for those who are new to setting up a Web server on a UNIX platform. The featured Web server is Apache, though many of the subjects covered are applicable to other Web servers."
4. CERN, "[Summary of HTTP Error Codes]" <URL:http:// www.w3.org/pub/WWW/Protocols/HTTP/HTRESP.html>
5. David Strom, "WebCompare" - "[T]he leading site for in-depth information on server software for the World Wide Web." <URL:http:// webcompare.iworld.com/>
6. NCSA, "NCSA HTTPd Overview" - These pages document the NCSA HTTPd server, the server WebSite is based upon. <URL:http:// hoohoo.ncsa.uiuc.edu/docs/Overview.html>
7. Roy Fielding, "WWW Protocol Library for Perl" - "libwww-perl is a library of Perl packages/modules which provides a simple and consistent programming interface to the World Wide Web. This library is being developed as a collaborative effort to assist the further development of useful WWW clients and tools." <URL:http:// www.ics.uci.edu/pub/websoft/libwww-perl/>
8. StarNINE, "WebSTAR Product Information" - "WebSTAR(TM) is the industry standard for transforming your Mac into a powerful Web server. WebSTAR can serve millions of connections per day, and is fully extensible through WebSTAR plug-ins." <URL:http:// www.starnine.com/webstar/webstar.html>
9. StarNine, "WebSTAR" - Based on Chuck Shotton's MacHTTP, WebSTAR(TM) helps you publish hypertext documents to millions of Web users around the world, right from your Macintosh. You can also use WebSTAR to put any Macintosh file on the Web, including GIF and JPEG images and even QuickTime(TM) movies. And yet, using WebSTAR is as easy as AppleShare(r). Plus, it's faster than many Web servers running on UNIX. <URL:http:// www.starnine.com/webstar/webstar.html>
10. Stephen Turner, "Analog" - "Fast, professional WWW logfile analysis for Unix, DOS, NT, Mac and VMS." <URL:http:// www.statslab.cam.ac.uk/~sret1/analog/>

World Wide Web

1. "World Wide Web" - This URL will take you to a terminal-based WWW browser. <URL:telnet://telnet.w3.org/>
2. "World Wide Web Consortium [W3C]" - The Consortium provides a number of public services: 1) A repository of information about the World Wide Web for developers and users, especially specifications about the Web; 2) A reference code implementation to embody and promote standards 3) Various prototype and sample applications to demonstrate use of new technology. <URL:http:// www.w3.org/pub/WWW/>
3. Alan Richmond, "WWW Development" <URL:http:// www.charm.net/~web/Vlib/>
4. Bob Alberti, et al., "Internet Gopher protocol" <URL:ftp:// ftp.lib.ncsu.edu/pub/stacks/finding/protocol.txt>
5. CERN European Laboratory for Particle Physics , "CERN Welcome" - CERN is one of the world's largest scientific laboratories and an outstanding example of international collaboration of its many member states. (The acronym CERN comes from the earlier French title: "Conseil Europeen pour la Recherche Nucleaire") <URL:http:// www.cern.ch/>
6. CNIDR, "freewais Page" <URL:http:// cnidr.org/cnidr_projects/freewais.html>
7. Daniel W. Connolly, "WWW Names and Addresses, URIs, URLs, URNs, URCs" - "Addressing is one of the fundamental technologies in the web. URLs, or Uniform Resouce Locators, are the technology for addressing documents on the web. It is an extensible technology: there are a number of existing addressing schemes, and more may be incorporated over time." <URL:http:// www.w3.org/hypertext/WWW/Addressing/Addressing.html>
8. Distributed Computing Group within Academic Computing Services of The University of Kansas, "About Lynx" <URL:http:// kufacts.cc.ukans.edu/about_lynx/about_lynx.html>
9. Internet Engineering Task Force (IETF), "HTTP: A protocol for networked information" - HTTP is a protocol with the lightness and speed necessary for a distributed collaborative hypermedia information system. It is a generic stateless object-oriented protocol, which may be used for many similar tasks such as name servers, and distributed object-oriented systems, by extending the commands, or "methods", used. A feature if HTTP is the negotiation of data representation, allowing systems to be built independently of the development of new advanced representations. <URL:http:// www.w3.org/hypertext/WWW/Protocols/HTTP/HTTP2.html>
10. Karen MacArthur, "World Wide Web Initiative: The Project" - [This site hosts many standard concerning the World Wide Web in general.] <URL:http:// www.w3.org/>
11. Mary Ann Pike, et al., Special Edition Using the Internet with Your Mac (Que: Indianapolis, IN 1995)
12. N. Borenstein, "MIME (Multipurpose Internet Mail Extensions)" - "This document is designed to provide facilities to include multiple objects in a single message, to represent body text in character sets other than US-ASCII, to represent formatted multi- font text messages, to represent non-textual material such as images and audio fragments, and generally to facilitate later extensions defining new types of Internet mail for use by cooperating mail agents. <URL:http:// www.oac.uci.edu/indiv/ehood/MIME/1521/rfc1521ToC.html>
13. National Center for Supercomputing Applications, "A Beginner's Guide to URLs" <URL:http:// www.ncsa.uiuc.edu/demoweb/url-primer.html>
14. NCSA, "NCSA Home Page" <URL:http:// www.ncsa.uiuc.edu/>
15. NCSA, "NCSA Mosaic Home Page" <URL:http:// www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/help-about.html>
16. NCSA, "NCSA Mosaic for the Macintosh Home Page" <URL:http:// www.ncsa.uiuc.edu/SDG/Software/MacMosaic/MacMosaicHome.html>
17. NCSA, "NCSA Mosaic for Microsoft Windows Home Page" <URL:http:// www.ncsa.uiuc.edu/SDG/Software/WinMosaic/HomePage.html>
18. NCSA HTTPd Development Team, "NCSA HTTPd Overview" <URL:http://

hoohoo.ncsa.uiuc.edu/docs/Overview.html>
19. Software Development Group (SDG) at the National Center for Supercomputing Applications, "SDG Introduction" <URL:http:// www.ncsa.uiuc.edu/SDG/SDGIntro.html>
20. Thomas Boutell, "World Wide Web FAQ" - "The World Wide Web Frequently Asked Questions (FAQ) is intended to answer the most common questions about the web." <URL:http:// www.boutell.com/faq/>
21. Tim Berners-Lee, Roy T. Fielding, and Henrik Frystyk Nielsen, "Hypertext Transfer Protocol" - "The Hypertext Transfer Protocol (HTTP) has been in use by the World-Wide Web global information initiative since 1990. HTTP is an application-level protocol with the lightness and speed necessary for distributed, collaborative, hyper media information systems. It is a generic, stateless, object-oriented protocol which can be used for many tasks, such as name servers and distributed object management systems, through extension of its request methods (commands). A feature of HTTP is the typing and negotiation of data representation, allowing systems to be built independently of the data being transferred." <URL:http:// www.w3.org/hypertext/WWW/Protocols/Overview.html>
22. Ulrich Pfeifer, "FreeWAIS-sf" <URL:http:// ls6-www.informatik.uni-dortmund.de/freeWAIS-sf/>
23. University of Kansas, "KUfact Online Information System" <URL:http:// kufacts.cc.ukans.edu/cwis/kufacts_start.html>
24. University of Minnesota Computer & Information Services Gopher Consultant service, "Information about gopher" <URL:gopher:// gopher.tc.umn.edu/11/Information%20About%20Gopher>
25. URI working group of the Internet Engineering Task Force, "Uniform Resource Locators" <URL:http:// www.w3.org/hypertext/WWW/Addressing/URL/URL_TOC.html>
26. Vannevar Bush, "As We May Think" Atlantic Monthly 176 (July 1945): 101-108 <URL:http:// www.isg.sfu.ca/~duchier/misc/vbush/>
27. WAIS, Inc., "WAIS, Inc." <URL:http:// www.wais.com/>