On the Internet Information Services

Technical Report No. 94-008

Masayuki Ida*
Computer Science Research Laboratory
Information Science Research Center

Aoyama Gakuin University
Address: 4-4-25 Shibuya, Shibuya-ku, Tokyo Japan 150

Feb. 24, 1995

*ida@csrl.aoyama.ac.jp

Preface

This report is a status quo report of the Internet Information services by the author. Since the Internet is rapidly growing, and itself is providing the latest information always, this report cannot be a sort of final summary of such tools.

The motivation to write this report is from the following idea:there are so many articles and books describing the related technology and the author is not confident about judging which information is up-to-date. So did it myself.

Most of the contents are NOT describing the author's original work, though the author is proud of the arrangement of this report emerged from the reading of many documents and internet resources. The author would like express his gratitude to the developers and the administrators of the tools described in this report. Actually some texts are extracted from the public resources.

Masayuki Ida
Feburary, 24th, 1995

— No Migration but Co-existence —

Contents

1 Introduction 1

1.1 The Internet Information Services and TCP/IP technology 1
1.2 Server-Client Model 2
1.3 Categoralization of the Network Services 2

2 Fundamental tools 5

2.1 NFS 5
2.2 NIS 5
2.3 Anonymous FTP 6
2.4 finger and whois 6

3 Archie: Directory Service for Locating Information 9

3.1 Archie Overview 9
3.2 Archie Client 10
3.3 Telnet to an archie server 12
3.4 ls-lR Files and Archie Support 13

4 WAIS: Distributed Information Retrieval System 15

4.1 WAIS: The Wide Area Information Server 15
4.2 WAIS protocol and WAIS server 16
4.3 WAIS Client 17
4.4 Short Summary of WAIS mechanism 18
4.5 Waissearh: An easy command for clients 18
4.6 Where you can get the software 18

5 Gopher: Distributed Document Search System 19

5.1 The Internet Gopher: Menu Based Document Search System 19
5.2 Gopher Architecture, Gopher server and Gopher protocol 20
5.3 Gopher Client 20
5.4 Where you can get the software 23

6 WWW: Merging Information Discovery and Hypertext 25

6.1 The World-Wide Web Overview 25
6.2 HTTP and HTML 26

6.3 Uniform Resource Locators . . . 27
6.4 Navigating the Web . . . 28
6.5 Server-Client Model . . . 28
6.6 WWW Clients such as Mosaic . . . 29
6.7 HTML and HTML+: How to write Hyptertext for WWW . . . 29
6.7.1 Writing HTML documents. . . . 30
6.7.2 HTML Editors and Converters . . . 30
6.7.3 Hints for HTML Documents Writing . . . 31
6.8 Setting up the WWW . . . 32
6.9 Where are the software for HTTP server and browser . . . 33
6.9.1 Unix Servers . . . 34
6.9.2 Macintosh Server . . . 34
6.9.3 Windows and NT Servers . . . 34
6.9.4 OS/2 Server . . . 35
6.9.5 DOS Server . . . 35
6.9.6 VMS Servers . . . 35
6.9.7 Amiga Server . . . 35
6.9.8 Commercial Servers . . . 36
6.9.9 Browsers . . . 36
6.10 HTML Documents . . . 36
6.11 Lisp based WWW tools . . . 37
6.11.1 Report Generation language . . . 37
6.11.2 Scheme based scripting . . . 39
6.11.3 Common Lisp HTTP server . . . 41
6.11.4 Scheme based Indexer . . . 41
6.11.5 w3.el . . . 42
6.12 How does the WWW compare to Gopher and WAIS? . . . 42
6.12.1 Limitations . . . 43
6.12.2 Popularity . . . 43
6.13 Line Mode WWW Client . . . 44
6.14 References . . . 46

7 Network News **47**
7.1 Usenet (NetNews) and NNTP . . . 47
7.2 RFC-977: Network News Transfer Protocol . . . 50

8 Network-wide File System Sharing **57**
8.1 Prospero . . . 57
8.2 Alex . . . 58
8.3 Ange-FTP feature on Emacs . . . 58

9 HTML Beginner's Guide **61**

10 CGI and URL Guide **79**
10.1 The Common Gateway Interface Overview 79
10.1.1 CGI 79
10.1.2 What's a gateway used for? 79
10.1.3 What language can I write these gateways in? 80
10.1.4 Documentation on the Interface Itself 80
10.1.5 Some examples of the uses of CGI: 80
10.1.6 Examples of CGI behavior and programs 80
10.1.7 Who came up with it? 81
10.2 Primer 81
10.2.1 How do I get information from the server? 81
10.2.2 How do I send my document back to the client? 82
10.3 Specification 83
10.3.1 Environment Variables 83
10.3.2 Command Line 85
10.3.3 Standard Input 85
10.3.4 Standard Output 85
10.4 Upgrading 87
10.5 A Beginner's Guide to URLs 88

Chapter 1

Introduction

1.1 The Internet Information Services and TCP/IP technology

Among the various types of information services, this report describes the server-client model oriented information services on the Internet.

As the worldwide academic computer network grows and expands far beyond its previous confines, so the resources and services available on the network evolve and multiply at a dizzying rate. The typical user is hardpressed to keep up with this explosive growth. Fortunately, a number of tools are avilable to facilitate the task of locating and retrieving network resorces, so that users anywhere can utilize texts, data software and information for public access. Facilities to explore public domain software repositories, to consult mailing list archives and databases, to retrieve directory information and to participate in global group discussions are now available to all.

The exponentially growing global Internet is a virtual infinite fountain of information. Today, the Internet has the potential of being the researcher's main information source, however, the Internet's size and complexity is a considerable obstacle.

Recently, a number of tools have been developed to assist users in finding information of interest. These "resource access" tools specialize in browsing, searching, and organizing information distributed throughout the Internet. Browsing tools allow users to navigate through the available information space, and find information of interest during this navigation process. Query-based search tools automatically locate relevant data for the user based on information about the user's interest. Independent of the approach used, Access services can also provide users with the ability to organize information once found, so that in the future they can refere to it without having to repeat the entire accesses process.

This paper presents an overview of resource access tools currently available on the Internet.

TCP/IP is the protocol for the Internet. And most of the technology for the Internet has been developed with TCP/IP. This report assume the basic knowledge of TCP/IP and the Internet. In other words, the knowledge for the following keywords are assumed:

Telnet, FTP, Mail, Internet address and Domain Name Service, RIP

(DNS: DNS maps IP adresses to the names assigned to network devices. RIP: Routing is central to the way TCP/IP works. RIP is used by network devices to exchange routing information.)

1.2 Server-Client Model

The key to exploiting these resources on the Internet is a **server**, special software on a computer somewhere in the network which accepts requests (or queries or commands) and sends a response automatically. The requestor does not have to be working on the same computer (or even in the same part of teh world) in order to use the server. Many servers accept requests via electronic amil, so that often the requestor need not even be on the same computer netowork as the server. In many cases, servers are interconnected so that once you have established contact with one server, you can easily communicate with other servers as well.

Today, many users have powerful computers on the desktop, with advanced graphical, audio and storage capabilities, which are connected to the network. This fact has given rise to what is known as the **server-client** model. Users can have special software on their local computer called **client** which can utilize the capabilities of that computer and can also communicate with a server on the network. These clients provide an easy-to-use, intuitive user interface, allow use of pointing devices such as a ouse, and exploit other local features. The client sends the user's requests to a server using a standardised format (called a *protocol*) and the server sends its response in a condensed format which the client displays to the user in a more readable way.

1.3 Categoralization of the Network Services

The functionalities described in this report have been divided into two major functional areas; Fundamental tools and Information Resource Access Services.

Fundamental tools literary means the basic tools which are used by various applications and services, such as NFS, NIS, anonymous FTP, finger, whois and so on.

Information Resource Access services and tools can be categoralized into six areas. Then, the following is summerizing the classification.

1. Fundamental tools, such as NFS, NIS, anonymous FTP
2. Information Resource Access services and tools
   (a) Directory service for locating information: Archie
   (b) Distributed information retrieval system: WAIS
   (c) Distributed menu based document search system: Gopher
   (d) Distributed hypertext based information system: WWW
   (e) Networked News reading tools, based on NNTP
   (f) Wide area File System Sharing tools: Ange-FTP, WWFS, Prospero, Alex, Transparent-FTP for Hurd, Andrew

This report describes the above in this sequence.

Chapter 2

Fundamental tools

2.1 NFS

NFS is a network file system developed by Sun. NFS is a protocol which allows files to be shared by various hosts on the network. NFS uses a server-client model. NFS client requests server to *mount* the file system.

NFS is run by several daemons. **nfsd** runs on servers. **biod** is the block I/O daemon runs on clients. **rpc.lockd** is the lock daemon which handles file lock requests. Both clients and servers run the lock daemon. Clients request file locks, and servers grant them. **rpc.statd** is the network status monitor daemon, which is required by rpc.lockd to provide monitoring services. Both clients and servers run the statd daemon. **rpc.mountd** is the NFS mount daemon, which processes the clients' mount requests. Servers run the mount daemon.

2.2 NIS

NIS is a network directory service developed by Sun. Originally it is called YP (Yellow Pages).

It provides central control and automatic dissemination of important administrative files. NIS can be used in conjunction with DNS. NIS provides service for LAN, unlike DNS. In other words, NIS is not intended as a service for the whole Internet.

NIS converts several standard (UNIX) files into databases that can be queried over the network. The databases are called *NIS maps.* The important NIS maps are created from the password file and the group file, and provides the information for NIS clients. Other NIS maps are from, */etc/ethers* for RARP support, */etc/hosts* for various host names lookup, */etc/networks* for network name lookup, */etc/netmasks* for subnet masks, */etc/protocols* for active protocol names, */etc/services* for network service names, */etc/aliases* for e-mail aliases and others, and *netgroup.*

The name of NIS server process is *ypserv*, and client process is *ypbind*.

2.3 Anonymous FTP

Anonymous FTP is a special provision to use FTP function. Anonymous FTP gives anonymous access to anonymous FTP servers. Anyone on the Internet can access the contents as far as the server is allowing, by placing the name as 'ftp' or 'anonymous' and 'your name' as password. Some anonymous FTP servers have incoming directory to accept files, but most servers are read-only.

2.4 finger and whois

Finger is a Unix command to find the name and related information. Finger asks finger daemon to get the information.

By default, finger displays information about each logged-in user, including his or her: login name, full name, terminal name (prepended with a '*' if write-permission is denied), idle time, login time, and location (comment field in /etc/ttytab for users logged in locally, hostname for users logged in remotely) if known. Idle time is minutes if it is a single integer, hours and minutes if a ':' is present, or days and hours if a d is present. Furthermore, the user's home directory and login shell, the time they logged in if they are currently logged in, or the time they last logged in if they are not, as well as the terminal or host from which they logged in and, if a terminal, the comment field in /etc/ttytab for that terminal, the last time they received mail, and the last time they read their mail, any plan contained in the file .plan in the user's home directory and any project on which they are working described in the file .project (also in that directory).

Whois is also a Unix command to get network information by giving a keyword. Some tools like Netfind are made upon such basic tools.

WHOIS provides directory service to network users. This service is a way to finding e-mail addresses, postal addresses and telephone numbers. It may also deliver information about networks, networking organizations, domains and sites.

The main database of networking-related names (organizations, sites, networks, people, etc.) is maintained by the Internet Registration Service (InterNIC). Actually, names in this database, the names of the administrative and technical contacts for registered domains are automatically entered into the database when domain or IP number applications are processed by the Internet coordination authority. Each entry of the database has a *handle* (a unique identifier), a name, a record type, and various other fields depending on the type of record. This database will be used as an example in the descriptions below.

Before April 1, 1993, the Network Information Center (NIC) of the Defense Data Network (DDN) was the Internet coordination authority and, therefore, maintained the database (known as the NIC database). The NIC database is now restricted to information bout the *.mil* domain. Many documents still refere to these names.

Many academic sites maintain their own database to offer information about their staff members and students.

WHOIS is available to users on the international TCP/IP network. A WHOIS server is accessible across the network from a user program running on local machines or via an interactive Telnet session to the site which hosts the server.

There are many WHOIS servers throughout the network and a comprehensive list would be too long to be included here. A WHOIS server offers information about the organization to which it belongs: it doesn't share a common directory with other WHOIS servers and doesn't know either where to find information about other institutions.

Unix computers have a native whois command. On non-UNIX machines, ask your system administrator whether your computer has it or not. This command searches the database on the specified site for entry which contains identifier. The format is:

```
whois    <-h sitename> identifier
```

where: **sitename** the domain address of the site which hosts the database you want to query (eg, whois.internic.net). On some installations, the default value is set to the NIC database site (nic.ddn.mil). **identifier** a name (person, host, domain or network), an IP number or a *handle.*

Special characters may be used to specify the search:

. before the identifier will cause a name-only search.

! before the identifier will cause a *handle*-only search.

... **or** . after the indentifier will cause a partial search: everything starting with identifier will match.

@ in the identifier will cause a search on the e-mail addresses.

* before the identifier will return the entire membership list of the entry that match identifier (egm, shows a site and its registered users).

% before the indentifier will return only the entire membership list of the entry that match identifer (eg, shows the reigstered users of a site).

Examples: whois MI27, whois ida@csrl..., whois aoyama.ac.jp

```
ida@chris>whois -h rs.internic.net MI27
Ida, Masayuki (MI27)            ida@CSRL.AOYAMA.AC.JP
   Associate Professor
   Aoyama Gakuin University
   Computer Science Research Lab.
   4-4-25, Shibuya, Shibuya-ku
   Tokyo 150, JAPAN

   +81 3 3409 8111 ext. 2185 (FAX) +81 3 3498 4870

   Record last updated on 03-Jun-92.

The InterNIC Registration Services Host contains ONLY Internet Information
(Networks, ASN's, Domains, and POC's).
Please use the whois server at nic.ddn.mil for MILNET Information.
```

Netfind is a white pages directory tool which given an Internet user's name and organization, tries to locate available information about the user. A successful netfind query returns information such as the user's e-mail address, and telephone number. Netfind was born at the resource discovery project at the University of Colorado-Boulder.

Netfind builds its indexing database, called the *seed database*, based on data scattered across multiple existing sources, such as network news messages, the DNS, the SMTP, and the *finger* utility. The seed database keeps organization names, city names, and corresponding host names gathered from news message headers overtime.

Chapter 3

Archie: Directory Service for Locating Information

3.1 Archie Overview

Archie specifically addresses the file discovery and retrieval problem in large internetworks. Archie provides an electronic directory service for locating information on the Internet. Archie servers centralize indexing information on file name data distributed throughout Internet archive sites.

The best known use of archie is for scanning a database of the contents of more than 1000 anonymous FTP sites around the world. Currently, this database contains more than 2,100,000 file names from anonymous FTP sites. This database is known as the archie database.

The files made available at anonymous FTP sites are software packages for various systems (MSDOS, Windows, Mac. Unix, etc.), utilities, information or documentation files, mailing list or usenet group discussion archives. At most FTP sites, the resources are organized hierarchically in directories and subdirectories. The database tracks both the directory path and the file names.

The archie database is automatically updated, therby ensuring that the information is accurate. Using this database, users can easily find the location of files they need without loggin onto several machines.

There are three ways to access the archie database: via a local client, interactive Telnet session or electronic mail.

Users on any network can access the archie database by electronic mail. Through the e-mail interface, users can submit a query to archie by sending an e-mail message to an archie server, which sends a message back to the user with the query results.

The telnet interface to archie consists of connecting to an archie server through the `telnet`

command, and then submitting queries. The telnet interface has proven to be very resource consuming, since each `telnet` session requires a significant amount of server resources. As an alternative, the Prospero interface has been made available to users. It consists of having a Prospero server as a front-end to the archie server. The Prospero server allows Prospero users to access archie's databases without the need to log onto the archie server directly. The Prospero interface has also caused the development of archie clients using the Prospero protocol. Because of Prospero's UDP based protocol and the Prospero server scheduling and caching techniques, the Prospero interface to archie is less resource consuming and presents higher throughput than the telnet access.

You are requested to respect a few basic rules when you request information from an archie server. Among them, one of the most important issue is to use the archie server closest to you and, in particularly, don't overload the transatlantic and transpaciffic lines.

The archie database is maintaind in 15 different locations.

```
archie.au (Australia), archie.funet.fi (Finland),
archie.th-darmstadt.de (Germany), archie.doc.ic.ac.uk,
archie.cs.huji.ac.il (Israel), archie.wide.ad.jp,
archie.kuis.kyoto-u.ac.jp, archie.sogang.ac.kr, archie.nz,
archie.luth.se (Sweden), archie.ncu.edu.tw (Taiwan),
archie.ans.net (USA), archie.rutgers.edu, archie.sura.net, archie.unl.net
```

There are more archie servers for local communities with limited Internet access. Consistency among the archie servers is maintained by having them copy the site listing information kept at the main archie server in Montreal, Canada.

3.2 Archie Client

The archie client is a command with parameters that you enter on your local machine. With most versions of the archie client, if you type **archie** with no parametes, you will get a list of the possible parameters and a short description of each.

The format of the command is:

```
archie <-options> string | pattern
```

where the options are:

o specifies an output file name to store the results (not available with all clients).

l lists the result one match per line. This form is suitable for parsing by programs.

t sorts the result inverted by date

m# specifies maximum number of matches to return (# within the range 0 to 1000).

h **archie_server** specifies an archie server to send the query to; if this parameter is not given, then the query will be sent to the default archie server, if one is defined.

L lists known servers and current default

The following group of options determines the kind of search performed on the database. They are mutually exclusive.

s a match occurs if the file/directoryu name contains *string*. The search is case insensitive.

c as above, but the search is case sensitive.

e *string* must EXACTLY match (including case) the file/directory name in the database. This is the DEFAULT search method.

r searches the database using *pattern*. It contains special characters which must be interpreted before performing the search.

There may be some slight differences in the options available with different clients on different platforms.

The result is a list of FTP site addresses with entries matching the argument, the size of the resouce, its last modification data and its directory. By default, the list is sorted by host address.

The following is a log of sample query.

```
ida@kepa>archie mule

Host etlport.etl.go.jp

    Location: /pub/ccipr
      DIRECTORY drwxrwxr-x        512  Oct 29 1992  mule
    Location: /pub
      DIRECTORY drwxrwxr-x       1024  Aug  8 10:13  mule

Host ftp.ae.keio.ac.jp

    Location: /pub/.x1/FreeBSD/binaries-1.1.5.1
      DIRECTORY drwxrwxr-x        512  Sep  9 22:40  mule
    Location: /pub/msdos/386
      DIRECTORY drwxrwxr-x        512  May  3 1994  mule
```

```
Host ftp.cs.titech.ac.jp

    Location: /pub/gnu-rel
      DIRECTORY drwxr-xr-x        512  Aug 16 21:40  mule

Host ftp.csce.kyushu-u.ac.jp

    Location: /pub/GNU/etc
      DIRECTORY drwxr-xr-x        512  Apr  7 1994  mule

Host ftp.csrl.aoyama.ac.jp

    Location: /gnu
      DIRECTORY drwxr-xr-x        512  Aug  9 12:02  mule

Host ftp.dit.co.jp

    Location: /pub/GNU
      DIRECTORY drwxr-xr-x       2560  Aug 31 13:46  mule

...
```

3.3 Telnet to an archie server

To access an archie server interactively, telnet to one of the existing servers. At the *login:* prompt enter `archie`, the login procedure ends leaving the user at a `archie>` prompt. The server is ready for the user requests and the following commands are available:

exit, quit, bye exits archie.

list <pattern> provides a list of the sites in the database and the time at which they were last updated. The optional parameter limits the list to specific sites. The result is a list of site names, sites IP address and data of the last update in the database. The command `list` with no *pattern* will list ALL SITES in the database (more than 1000 sites).

site site-name lists the directories and, recursively, the subdirectories, of site-name in the database. The result may be very long.

whatis string searches the database of software package descriptions for *string*. The search is case-insensitive.

find <expression> searches the database for string or pattern created by *expression* which represents the name of the resource to be found in the database. Searches may be performed in a number of different ways specified in the variable *search* which also decides the interpretation of the parameter. The result is a list of FTP site addresses with matching entires, the size of the resource, its last modification data and the directory to find it.

mail <email1><,email2...> sends the result of the last command in a mail message to the specifed e-mail address(es). If issued with no argument, the result is sent to the address specified in the variable *mailto.*

show variable displays the value of the gien variable name. If issued with no argument, it displays all variables.

Variables are, `mailto` (the e-mail address(es) to mail the result), `maxhits` (the maximum number of matches prog will generate), `search` (determines the kind of search performed. values are `sub` for a partial and case insensitive search, `exact` for the exact matching, `regex` for pattern matching, `sortby {hostname, time, size, filename, none}` for how to sort the result of prog), `term` (terminal type).

3.4 ls-lR Files and Archie Support

In Japan at least, the directory information of anonymous FTP server which is kept at each registered anonymous FTP site are periodically gathered into files at several NOCs. The data is referred and used by Japan domestic archie servers, such as archie.foretune.co.jp, archie.kyoto-u.ac.jp, and others.

For anonymous FTP administrators and their community network administrator, creation of ls-lR by each and gathering them into one is a easy task and give huge benefit to public.

Chapter 4

WAIS: Distributed Information Retrieval System

4.1 WAIS: The Wide Area Information Server

WAIS, the Wide Area Information Server project, resulted from the combined effort of four companies with complementary interests in the information discovery and retrieval problem. One of the project's primary goals is to provide users with a uniform, easy-to-use, localtion-transparent mechanism to access information. WAIS was originally developed at Thinking Machines, working with Dow Jones, Apple Computer, and KPMG Peat Marwick. Thinking Machines made their UNIX implementation of the WAIS software freely available. The core of the software is an indexer, used to create full-text indexes of files, and a server that can use those indexes to search for keywords or whole English expressions among the files indexed. The software was such a hit on the Internet that Brewster Kahle, the leader, founded WAIS Inc. to produce a commercial version of WAIS.

WAIS is a full text information retrieval architecture whose clients and servers communicate through an extension of the Z39-50 protocol standard, an ANSI standard.

> Clifford A. Lynch. The Z39-50 information retrieval protocol: An overview and status report. ACM Computer Communication Review, 21(1):58-70, 1991

WAIS, Wide Area Information Server, is a distributed information retrieval system. The databases (called sources) are mostly collections of text-based documents, but they may also contain sound, pictures or video as well. More than 400 databases on topics ranging from *Agriculture* to *Social Science* can be searched with WAIS.

The databases may be organized in differennt ways, usign various database systems, but the user isn't required to learn the query languages of the different databases. WAIS uses natural language queries to find relevant documents. The result of the query is a set of

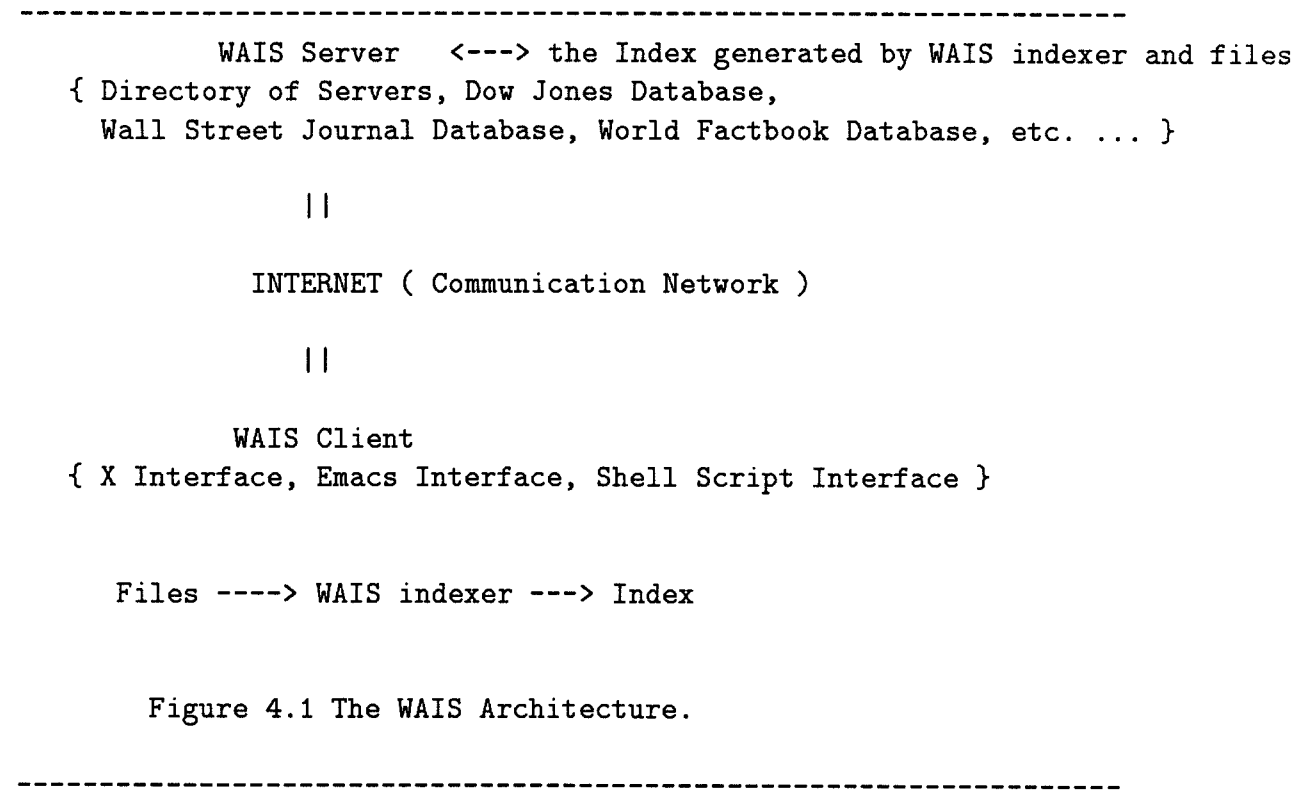

Figure 4.1 The WAIS Architecture.

documents which contain the words of the query: no semantic information is extracted from the query.

WAIS uses the client-server model to provide access to databases. You must be on the international TCP/IP network (the Intenet) in order to use WAIS.

One server is distinguished as the directory of services. Figure 4.1 shows a schematic view of the WAIS architecture.

4.2 WAIS protocol and WAIS server

The WAIS client translates user queries into the WAIS protocol, and can query the WAIS directory of servers for relevant servers. The request is then transmitted to an appropriatye set of servers. WAIS servers keep complete inverted indices on the contents of documents they store and execute full-text searches on them. In response to a query, a WAIS server returns a list of relevant object descriptors. These descriptors correspond to documents that contain words and phrases specified in the user query. The WAIS client displays the results of the query to the user and retrieves the selected documents form the corresponding servers. WAIS clients support relevance feedback to help users refine their queries.

Originally, users were supposed to keep track of the available WAIS servers by storing their addresses, a description of their contents, and if applicable, their accessing costs. However, as participating servers proliferate, it became unreasonable to expect that every user would keep track of all the available servers. A directory of servers service was made available. This 'yellow page' service resides in a well-known address and is queried the same way as the regular WAIS servers. Instead of reference to documents, it replies with references to participating WAIS servers. When a new information provider wants to join WAIS, it must submit its location, description, and other relevant information to the directory server.

4.3 WAIS Client

A query from WAIS client is first submitted to the directory of services which responds with descriptors of pertinent servers. The user can then select servers from this set and submit the query to them. By looking at their descriptions answered, the user can identify relevant documents to be retrieved. If the user wishes to refine the search, one or more of the returned documents can be selected and placed in the `Similar to:` box. When the query is re-run, the results are updated to include documents similar to the selected ones. Similarity is ranked in terms of number of common words.

Typical steps to use WAIS are:

- Step 1: The user selectes a set of databases to be searched from among the available databases.
- Step 2: The user formulates a query by giving keywords to be searched for.
- Step 3: When the query is run, WAIS asks for information from each selected database.
- Step 4: Headlines of documents satisfying the query are displayed. The selected documents contain the requested words and phrases. Selected documents are ranked according to the number of matches.
- Step 5: To retrieve a document, the user simply selects it form the resulting list.
- Step 6: If the response is incomplete, the user can state the question differently or feed back to the system any one or more of the selcted documents he finds relevant.
- Step 7: When the search is run again, the results will be updated to include documents which are *similar* to the ones selected, meaning documents which share a large number of common words.

4.4 Short Summary of WAIS mechanism

WAIS is divided into three main components: an indexer, a server, and a client. The indexer is responsible for taking documents fo various formats in a database and digesting them into a number of different tables and indexes for efficient searching. The server uses the indexes to determine whether the words a user is looking for occur in the database, and if so, which documents they occur in and where. The clients simply build queries for the server in the appropriate Z39.50 format, display search results to users, and allow users to retrieve documents form the server.

The directory of servers is a special server, which keeps track of the current WAIS servers' contents, so that clients can utilize each WAIS server easily.

4.5 Waissearh: An easy command for clients

Once WAIS is established, WAIS database can be accessed through WWW, or using a simple command. `waissearch` is it.

```
waissearch -h host -d directory -p port keyword
```

4.6 Where you can get the software

WAIS server and client are available via anonymous FTP from the University of North Carolina (SunSITE.unc.edu) in the directory /pub/wais as file *wais-sources.tar.Z*. A complete listing of the advertised databases and the clients software are available from such site.

Public domain clients for accessing WAIS are available for: Macintosh, MS/DOS, Windows, Unix, X-Window etc.

There is a newsgroup named comp.infosystems.wais.

The most up-to-date information for the availability is as follows: To get freeWAIS, anonymous FTP from ftp.cnidr.org as /pub/NIDR.tools/freeWAIS-0.3.tar.Z as far as the author knows as of '95 Feb.

Chapter 5

Gopher: Distributed Document Search System

5.1 The Internet Gopher: Menu Based Document Search System

The Internet Gopher (R.Alberti, F. Anklesaria, P. Lindner, M. McCahill, and D. Torrey. The Internet Gopher protocol: a distributed document search and retrieval protocol. On-line documentation, Spring 1992.) allows user to search and browse distributed information, residing on different locations in a seamless fashion. Gopher organizes information into a hierarchy (actually, the Gopher information space is a generalized directed graph, since it allows cycles.) where intermediate nodes are directories or indices, and leaf nodes are documents.

When browsing it, the information appears to the user as a series of nested memus. This kind of menu structure resembles the organization of a directory with many subdirectories and files. The subdirectories and the files may be located either on the local server site or on remote sites served by other Gopher servers. From the user point of view, all information items presented on the menus appear to come from the same place.

The information can be a text or binary file, directory information (loosely called *phone book*), image or sound. In addition, Gopher offers gateways to other information systems (WWW, WAIS, archie, WHOIS) and network services (Telnet, FTP). Gopher is often a more convenient way to navigate in a FTP directory and to download files.

The Internet Gopher is developed by the Computer and Information Services Dept. of the University of Minnesota. Bug reports, comments, etc. should be mailed to teh Gopher development team at: `gopher@boombox.micro.umn.edu`.

5.2 Gopher Architecture, Gopher server and Gopher protocol

The Gopher architecture is composed of clients and servers communicating through the Gopher protocol, which is implemented on top of TCP-IP. A Gopher server holds the information and handles the users' queries. In addition, links to other Gopher servers create a network wide cooperation to form the global Gopher web (*Gopher-space*).

The root of Gopher's heirarchy is stored on host *rawBits.micro.umn.edu* at the University of Minnesota. This is the default directory retrieved by the Gopher hierarchy. The Gopher root server knows about all top-level services, so that it can advertise their existence to users. In the Gopher architecture, there is essentially one *top-level* services, so that it can advetise their existence to users. In the Gopher architecture, there is essentially one *top-level server* per participating organization, such as a university campus, a private corporate institution, or a government agency. *Lower-level servers* may be linked to the corresponding top-level server, so that once users find the appropriate top-level server, they can nagivate through the hierarchy by following the links to the lower-level servers. For example, university campuses running Gopher servers may register a central top-level server with the Gopher root server. Each university's central Gopher server may have links to existing departmental servers, which in turn may have links to lower-level servers.

Gopher objects are identified by their type, user-visible name, server's host name and port number, and the object's absolute path name within the server's file system. The user selects an object based on its user-visible name, and the Gopher client retrieves it by constructing a handle from the server's host name and port number, and the object's path name. Users can then navigate through the available information base containing full-text document objects, which are stored as files in the corresponding servers, and directory objects that may be distributed across multiple servers. Full-text search operations can also be performed. Gopher's*search servers* maintain full-text inverted indices of subsets of the documnets stored in a Gopher server. Search servers can be configured to index more than one server. For instance, there is an Internet Request for Comments (RFC) full-text search server returns to the client handles to documents that match a Boolean search pattern. Gopher clients can also retrieve objects from WAIS, archie, and FTP servers.

5.3 Gopher Client

Gopher uses the client-server model to provide access to the Gopher web. You must be on the international TCP/IP network (the Internet) in order to use a client on your computer

to access Gopher.

Users explore the Gopher menus using various local clients or accessing a remote client via an interactive Telnet session.

When using such a remote client, at the login prompt, type **gopher** (unless specifed otherwise) and the top-level Gopher menu for that site will be displayed. Users are requested to use the site closest to them.

When issuing the gopher command, you will be connected automatically to the default Gopher server specified at the installation. The format of the command is:

```
gopher <hostname>
```

where *hostname* is an optional alternative Gopher server you want to talk to.

When connected to a Gopher server, it is still possible to access another server by exploring the *Other Gopher servers in the rest of the world* branch. To locate them more easily, the Gopher servers are distributed in geographical regions.

Access to a Gopher server is identical whether using a local or a remote client: a simple menu driven interface which doesn't require any special training or knowledge from the user.

Here is a sample menu:

```
------------------------------------------------------------------------

Internet Gopher Information Client v1.1
Information About Gopher

1. About Gopher.
2. Search Gopher News <?>
3. Gopher News Archieve/
4. comp.infosystems.gopher (USENET newsgroup)/
5. Gopher Software Distribution/
6. Gopher Protocol Information/
7. Umniversity of Minnesota Gopher software licensing policy.
8. Frequently Asked Questions about Gopher.
9. gopher93/
10. Gopher| example server/
11. How to get your information into Gopher.
    --> 12. New Stuff in Gopher.
13. Reporting Problems or Feedback.
14. big Ann Arbor gopher conference picture.gif <Picture>

Press ? for Help, q to Quit, u to go up a menu            Page: 1/1
------------------------------------------------------------------------
```

In the example above, any item can be selected by typing its line number or by moving the cursor ($->$) next to it.

An item could be:

1. a subdirectory, its contents are displayed. To go up one level, use the up command.
2. a text file, the file is displayed. Then you can browse it, search for a particular string, print it on a local printer or copy (save) it onto your local disk space in a user-specified file (the last 2 functions may not be available to you).
3. a binary file, the remote file is simple copied onto your local disk space in a user-specified file. Binary files are binhexed Macintosh files, archives (.zip, .tar, ...), compressed files, programs, etc.
4. a sound file, the remote file is played through your local audio device if it exists, as well as the appropriate utility. Only one sound file can be active at a time;
5. an image file, the remote file is displayed on your computer screen if an image viewer exists on your computer.
6. a phone book (directory information), you are prompted for a search string to look up people information through the selected phone book. Since different institutions have different directory services, the queries are not performed in the same fashion.
7. an index-search, you are prompted for a search string which may be one or more words, plus the special operators **and, or** and **not**. The search is case-insensitive. Usually, an index is created to help users locate the information in a set of documents quickly. Eg: `terminal and setting or tset` will find all documents which contain both the words *terminal* and *setting*, or the word *tset*. **or** is non-exclusive so the documents may contain all of the words. The result of the index search looks like any Gopher menu, but each menu item is a file that contains the specified search string.
8. a Telnet session, Telnet sessions are normally text-based information services, for example, access to library catalogs.

Items are displayed with an identifying symbol next to them. In the example above, "<?>" means a full text index search, "/" means a subdirectory, "<Picture>" means an image file and no symbol means a text file.

Some Gopher clients are not able to handle certain file types (eq, sound files). Some clients display only files of types they can handle or files they suppose you are interested in. Others display all types of files.

Most Gopher clients allow you to create, view and select *bookmarks*. A bookmark keeps track of the exact location of a Gopher item, regardless of where it resides. It is useful

when you often need to reach a file or a service located far from the top-level directory. A collection of bookmarks is like a customized Gopher menu.

Some capabilities of a local Gopher client are bound to the capabilities of your own computer. In fact, for sound files, image files and Telnet sessions, the Gopher client looks for the appropriate software on your computer and prasses control to it to perform the requested task. When the task in completed, control is returned to the Gopher client.

At any time, it is possible to terminate the session (quit command), to cancel the current processing or to get the on-line help (help command).

5.4 Where you can get the software

Some sites allow public access via Telnet to a client, for those who have no local clients yet. `tolten.puc.cl (Columbia), ecnet.ec (Ecuador), consultant.micro.umn.edu (USA) etc.`

The implementations of the Gopher clients on various platforms are slightly different to take advantage of the platforms' (mouse, graphic functions, X-Window server) and to offer the popular look and feel. Even with different implementations, the same set of functions and commands is available.

There is a system named Veronica, which is to be a solution to the problem of resource discovery in the rapidly expanding Gopher web, providing a keyword search of more than 500 Gopher menus. Veronica helps you find Gopher-based information witout doing a menu-by-menu, site-by-site search. Veronica was developed at the University of Nevada.

Currently as of Feb., 1995, Three major Gopher servers are available for UNIX:

1. Gopher 1.13 from University of Minnesota. This was the last unlicensed version they did. It's still widely used, although development stopped in mid-1993.

2. Gopher 2.0.16 from University of Minnesota. This is the current version under development. It requires licensing fees for commercial use. This version is also known as **Gopher+**.

3. *gn* server, which is distributed under the GNU Public License, allowing free use. This server provides Gopher 1.x level function and Web server function.

The clients are available for anonymous FTP from many FTP sites (eq, boom-box.micro.umn.edu in the directory /pub/gopher).

Clients for UNIX are *gopher* for line mode text client, *xgopher* for X window, *moog* for X with simple motif interface, and Emacs-client (Emacs Gopher mode).

Machintosh clients are *TurboGopher* availabe from
`boombox.micro.umn.edu:/pub/gopher/Macintosh` by anonymous FTP, and *GopherApp* from
`ftp.bio.indiana.edu:/util/gopher/gopherapp/gopherapp++.hqx`.

DOS/Windows clients are *PC Gopher III* from the Univ of Minnesota, *PC Gopher for LAN Workplace for DOS*, *BCG* (A Gopher from Boston College), and *WSG* (Winsock Gopher). Some commercial software are supporting gopher as a part, such as Chameleon.

Chapter 6

WWW: Merging Information Discovery and Hypertext

6.1 The World-Wide Web Overview

The World-Wide Web, or WWW merges the techniques of information discovery and hypertext. WWW organizes data into a distributed hypertext, where nodes are either full-text objects, directory objects called *cover pages*, or indices. WWW also supports full-text searches over documents stored at a particular WWW server.

With hypertext links to other textual documents or files, users can click on a highlighted word or words in the text to provide additional information about the selected word(s). Users can also access graphic pictures, images, audio clips, or even full-motion video through hypermedia, an extension of hypertext.

The World-Wide Web project was started by CERN (European Center for Nuclear Research) in an attempt to build a distributed hypermedia system. Users access WWW through a "browser program" that can read and fetch documents locally as well as from sites around the world via the Internet. Browsers access files using FTP, NNTP, Gopher, and several other Internet-based protocols. Browser clients can also search remote documents and databases if the associated server at that site has built-in search capabilities.

The WWW architecture is also based on the client-server model. Besides its native HyperText Transfer Protocol (HTTP), WWW clients understand FTP, the Networks News Transfer Protcol (NNTP), Gopher and WAIS. FTP is used for acessing file archives on the Internet, where file directories are browsed as hypertext objects. NNTP allows access to Internet news groups and news articles. News articles may contain references to other articles or news groups, which are represented as hypertext links.

Information accessible through WWW can be seen through three discovery trees. One tree is a classification by subject. An entryin the WWW root directory links to the current

subject classification tree. This information is spread across all kinds of servers, including WAIS, Gopher, and WWW. Because WWW was created for the high-energy physics community, a special entry in the root directory links to a cover page with the existing HTTPservers specialized in the subject. As they become available, indices to other disciplines wil be added. The other WWW discovery tree is a classification by server type. The cover page corresponding tothis classification lists all the servers available through WWW. This includes entries for WAIS, Gopher, NNTP and WWW servers. There is even an entry for anonymous FTP sites that are searched through archie. The third tree is a classification by organization and is not very populated.

The WWW discovery trees correspond to the different ways information is organized and can be discovered. Discovery sessions involve users starting on their home cover page, following a link to an index, then executing a search, and following the resulting links. As users find interesting information, they build their personalized web by linking to nodes in the global web. Currently, the dafult cover page, which resides on host *info.cern.ch* and represents the root of the WWW information space, is the one retrieved by the WWW client when first invoked. However, users can customize their home cover page so that they can start anywhere in the WWW information space.

Making information available through WWW may involve a similar discovery task. The information publisher must try to find the appropriate cover page that should reference the new data. Then the publisher should contact the person responsible for that cover page so that a link to the new data is added. Another possiblity is for the publisher to run a new server. This last option requires that the new server's administrator contacts the WWW administrators to add the new server to the list of existing servers.

6.2 HTTP and HTML

HTTP is the protocol employed between the server and client. It requires only a reliable connection-oriented transport service, typically TCP/IP. The client establishes a connection with the server and sends a request containing the word GET, a space, and the partial URL of the node to be retrieved, terminated by the carriage return and line feed characters. The server responds with the node contents, which consist of HTML text documents. The end of the contents is signaled by the server closing the connection.

HTTP allows document retrieval and full-text search operations. HTTP runs on top of TCP and maps each request to a TCP connection. HTTP objects are identified by the HTTP protocol type, the corresponding server's name, and the path name to the file where the objects' contents reside. Parts of documents can also be specified. If a search operation is requested, the HTTP object identifier carries the set of specified keywords,

instead of a path name. Future implementations of HTTP protocol will include data format negotiation between client and server. Currently, only plain text and simple hypertext formats (HyperText Markup Language; HTML) are implemented.

To distribute information via the WWW, you first need to set up an HTTP server. The HTTP protocol is stateless, lightweight, and extremely fast, and it provides capabilities that are not found in earlier protocols, such as FTP.

6.3 Uniform Resource Locators

A Uniform Resource Locator (URL) referes to the format used by WWW documents to locate other files and is currently a draft standard for specifying an object or resource on the Internet. A URL gives the type of resource being accessed (for example, Gopher or WAIS) and the path of the file. The format used is:

```
scheme://host.domain[;port]/path/filename.
```

The first part of the URL, the scheme, specifies the access method. Several keywords can be specified for the scheme:

- FTP : retrieves a file on your local system or on an anonymous FTP server,
- HTTP : retrieves a file on a WWW server (the native WWW protocol),
- Gopher : retrieves a file on a Gopher server,
- WAIS : retrieves a file on a WAIS server,
- News : reads the latest Usenet news, or
- Telnet : uses the telnet protocol to connect to some site.

The part of the URL after the colon is interpreted with regard to the specific access method. In general, two slashes after the colon indicate a machine name or, optionally, a port (For additonal information, see "A Beginner's Guide to URLs."
`http://www.ncsa.uiuc.edu/demoweb/url-primer.html`.)

The last part of the URL describes the location-specific path and filename of the desired document. Some example URLs are:

```
ftp://ftp.uu.net/info/README
http://info.cern.ch;80/default.html
news:alt.hypertext
telnet://dra.com
```

6.4 Navigating the Web

World-Wide Web (also called WWW or W3) is an information system based on hypertext, which offers a means of moving from document to document (usualy called to *navigate*) within a network information.

Hypertext documents are linked to each other through a selected set of words. For example, when a new word, or a new concept, is introduced in a text, hypertext makes it possible to point to another document which gives more details about it. The reader can open the second document by selecting the unknown word or concept and the relevant section is displayed. The second document may also contain links to further details. The reader need not know where the referenced document is, and there is no need to type a command to display it. or to browse it to find the right paragraph. Cross-references may be defined in the same document. A collection of documents is a database.

Also, special documents (indexes) in the WWW information space can be search for given keyword(s). The result is a document which contains links to the documents found.

6.5 Server-Client Model

WWW uses the server-client model to provide access to the information universe. You must be on the international TCP/IP network (the Internet) in order to use a client on your computer to access WWW.

Users access the World-Wide Web facilities via a client called a *browser*. This interface provides transparent access to the WWW servers.If a local WWW client is not avilable on your computer, you may use a client at a remote site. Thus, an easy way to start with WWW is to access a remote client.

To access a remote WWW client, telnet to one of the server sites. At the *login:* prompt enter **www**, no password is needed. There are many WWW servers already available throught the network, and the number is growing, so it is not possible to include a comprehensive list here.

CERN (European Particle Physics Laboratory) is the entry point to find information about WWW itself and to have an overview of the Web with a catalogue of the databases sorted by subject.

When using a graphical interface, you access the WWW function by pressing mouse buttons. In particular, references are highlighted or underlined words. To follow a link, double click on the associated reference.

6.6 WWW Clients such as Mosaic

Browsers can be line-oriented (to work with traditional terminals) or graphics oriented (to work with more modern graphics display terminals). One of the most popular graphics oriented browsers is Mosaic, which was developed at the National Center for Supercomputing Applications (NCSA) as a way to graphically navigate the WWW. The first version of NCSA's Web browser, a WWW client for Unix with X Windows called XMosaic, was made available to the Internet community in 1993. Due primarily to its easy-to-use graphical user interface (GUI), XMosaic soon became the most popular interface to the Web.

Mosaic is a system for wide-area distributed asynchronous collaboration and hypermedia-based information discovery and retrieval. Mosaic browsers are currently available for Unix workstations running X Windows, PCs running MS Windows, and Macintosh computers. Mosaic can access data in WWW servers, WAIS, Gopher servers, Archie servers, and several others.

Many commercial and government organizations and universities are migrating to Mosaic because of its many useful features and capabilities. It has a consistent mouse-driven graphical interface and can display electronic data in a variety of fonts and styles. Mosaic reads SGML (Standard Generalized Markup Language) and HTML (Hyper Text Markup Language) text, enabling information to be presented in an attractive way. Mosaic also supports a wide variety of basic forms elements, such as fields, check boxes, and radio buttons, as well as characters from different languages (as defined by the ISO 8859 standard), including French, German, and Hawaiian. Finally, Mosaic supports graphics of up to 256 colors (in GIF or X bitmaps format) as well as digital audio and video.

Users can extend Mosaic's functionality by creating custom servers and by letting other applications control its display remotely. Screen contents can be broadcast to a network of users running multiplatform groupware, such as NCSA's Collage. Mosaic users can traverse hypermedia links to provide basic network services (for example, FTP, Gopher, Telnet, NNTP, and WAIS) and store those links in "hotlists" for quick reference. And with the additional features available in HTML+, Mosaic users can "check out" Web documents for revision or for local or remote annotation.

6.7 HTML and HTML+: How to write Hyptertext for WWW

The Web uses HTML for creating and recognizing hypermedia documents. This is a simple markup system, related to the Standard Generalized Markup Language (SGML), used to

format hypertext documents. (On the Web, these documents usually carry the ".html" suffix.) HTML code describes how textual elements (like paragraphs, lists, numbered and bulleted lists, descriptive lists, and quoted paragraphs) will be displayed. The current HTML standard supports basic hypermedia document creation and layout, but this capability is still very limitted. The HTML+ extension supports interactive forms, defined "hotspots" (hyperlinks) in images, formatted tables, and more versatile document layout, formatting options, and styles.

6.7.1 Writing HTML documents.

Most HTML documents are written using a standard text editor – for example, *vi* or *emacs* in a Unix environment – in a plain text format. Although HTML code is a tedious language to write, some authoring tools can now convert plain text to HTML automatically.

One way to learn how to write HTML documents is to look at how others have created their documents. You can do this by using the "source" button (available on most Mosaic browsers) to view the HTML code for a document or page you find particularly interesting. Most HTML statements are fairly straightforward. You may check the Mosaic screen and the corresponding HTML source code to have more understanding. If you intend to write HTML code, we recommend that you read the on-line Mosaic document "A beginner's guide to HTML," which is available on the Web (see the subsection on HTML editors and converters), and is attached to this report.

6.7.2 HTML Editors and Converters

HTML documents can be created with WYSIWYG editors or with editors that simply assist you by allowing you to select the desired markup tags from a menu. HTML editors are easier to use and interactively display the resulting HTML document on-screen. For example, current versions of the Emacs editor have an "HTML mode" to assist users with writing statements. There is also an editor for MS Windows called HTML Assistant. For X users, tkWWW supports WYSIWYG HTML editing; and since it is a browser, you can try out links immediately after creating them. HotMetal is a professional HTML Plus editor for X and MS Windows. For Macintosh users, the BBEdit HTML extensions allow the BBEdit and BBEdit Lite text editors to conveniently edit HTML documents.

Existing files (for example, a LaTex document) can be converted to HTML automatically by using a set of special software tools. Here is the list of some editors and converters.

BBEditHTML

A Macintosh editor for HTML documents obtained by accessing
`http://www.uji.es/bbedit-html-extensions.html`. You can also obtain the extensions package by anonuymous FTP from
`sumex-aim.stanford.edu in file /info-mac/text/bbedit-html-ext-b4.hqx`.

Emacs editor

Current version Emacs have an HTML mode to assist users with writing HTML statments (`ftp://ftp.ncsa.uiuc.edu/Web/elisp/html-mode.el`).

HTML Assistant

A MS Windows editor with features to assit in the creation of HTML documents can be obtained by anonymous FTP from `ftp.cs.dal.ca:/htmlasst`.

LaTex2html

Converts LaTex documents to HTML (`http://cbl.leeds.ac.uk/nikos/tex2html.html`)

Mail2HTML

Converts electronic mailboxes to HTML documents
`ftp://info.cern.ch/pub/www/dev`

NCSA's list of filters and editors

This document mentions several editors, including two for MS Windows
`http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/faq-software.html#editors`

RTF2HTML

Converts Rich Text Format (RTF) to HTML
`ftp://oac.hsc.uth.tmc.edu/public/unix/WWW`

6.7.3 Hints for HTML Documents Writing

Remember that most users accessing information over the Internet are browsing and are therefore unwilling to wait long for a document to appear on their screen. Keep the size of images on the homepage small. Wait until the user selects the icon to incorporate high-quality (and large) images into your documents. This will let users access the home page quickly and will ease server and network load. Although each image takes time to process and slows the display, using a particular image multiple times in a document causes very little performance degradation compared to using it once.

Be sure to test the document's remote access time to ensure that it is reasonable, before making any documet public.

Always provide an estimate of the number of bytes that will be sent when a user selects a given "hotword" (especially for high-resolution images and digital audio and video). This helps users decide whether to select a given hotword.

Do not mix a large number of different fonts, colors, and so forth on a single display screen. Use good graphic arts design principles.

Do not say "Click Here." It is redundant and only clutters the screen. Hotwords are normally a different color or font, so it is obvious (especially to frequent users) that "clicking here" will cause a hyperlink to be traversed.

Put a reference (for example, an electronic mail address) at the bottom of every page telling whom to contact about the document. This can be accomplished by using the HTML Address Tag construct.

Do not make your pages too long. Rather, create a hierarchical set of menus that group items into meaningful categories. That way only those items of interest will have to be selected.

6.8 Setting up the WWW

To access the Web, you first need an account on a machine that is connected to the Internet. All WWW features can be accessed with client/browser software from this machine. To obtain a browser for your machine, refer to the discussion below. Many of the browsers discussed here require that you have SLIP, PPP, or other TCP/IP networking to your computer. SLIP or PPP can be accomplished over phone lines, but only with the active cooperation of your network provider or educational institution. If you only have normal dial-up access, your best option is to run a browser on the system you call directly or from a system that can be connected to.

Here is the steps to establish your WWW.

Step 0: Browsers accessible via Telnet.

Several browsers are available over the Internet through Telnet. An up-to-date list of these is available on the Web as `http://info.cern.ch/hypertext/WWW/FAQ/Bootstrap.html`. For example,

- info.cern.ch : No password is required. This is in Switzerland, so continental US users might be better off using a closer browser.

- ukanaix.cc.ukans.edu : A full-screen browser "Lynx" that requres VT1000 terminal emulation. Log in as www.
- www.njit.edu : A full-screen browser in NerJersey Institute of Technology.
- www.vms.huji.ac.il : A dual-language Hebew/English database at Hebrew University of Jerusalem in Israel, with links to the rest of the world. Log in as www.
- sun.uakom.cs : A machine in Slovakia. Log in as www.
- fserv.kfji.hu : A machine in Hungary. Log in as www.

STEP 1: Obtaining your own browser

The preferred method of accessing the Web is to run a browser, in other words WWW client, yourself. Browsers are available for many platforms, both in source and executable forms.

STEP 2: Setting up your own server

Browsers can obtain data from programs supplied by information providers. These programs can either be WWW servers that understand the HyperText Transfer Protocol, "gateway" programs that convert ans existing information format to hypertext, or a non-HTTP server that WWW browsers can access – anonymous FTP or Gopher, for example.

STEP 3: Creating a public home page

There are several things you can do to publicize your new HTML servedr or other offering:

Submit it to the NCSA "What's New Page"

at `http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/whatsnew.html`.

Post it to the newsgroup comp.infosystems.announce. Please read the newsgroup first to get a feel for the contents. You can also crosspost to comp.infosystems.www.

Submit it to the maintainers of various catablogs, such as the WWW Virtual Library `http://info.cern.ch/hypertext/DataSources/bySubject/Overview.html`.

6.9 Where are the software for HTTP server and browser

One of the first steps in setting up a server is getting server software. HTTP server software is available from various WWW sources for Unix, Macintosh, Windows, and VMS systems (see http://info.cern.ch/hypertext/WWW/Daemon/Overview.html for more information on writing servers and gateways in general).

A more extensive listing of available browsers can be found at `http://info.cern.ch/hypertext/WWW/Clients.html`.

To learn more about WWW servers, you can consult a WWW server primer available at http://www.muw.ac.nz/who/Nathan.Torkington/ideas/www-servers.html.
See http://info.cern.ch/hypertext/WWW/Daemon/Overview.html for more information on writing servers and gateways.

6.9.1 Unix Servers

- NCSA httpd. This is available at the URL (universal resource locator) `ftp://ftp.ncsa.uiuc.edu/Web/ncsa_httpd`.

- CERN httpd. CERN's server is available for anonymous FTP from info.cern.ch (the URL is http://info.cern.ch/hypertext/WWW/Daemon/Status.html). It's also available from many other sources; use your local copy of Archie to search for "www" to find anear-by site.

- GN Gopher/HTTP server. The GN server is unique in that it can serve WWW and Gopher clients (in their native modes). This is a good server for those migrating from Gopher to WWW, although it does not have the server-side script capabilities of the NCSA and CEERN servers. See the URL http://hopf.math.nwu.edu/.

- Perl server. This is a server written in the Perl scripting language, called Plexus, for which documentation is available at the URL http://bsdi.com/server/doc/plexus.html.

6.9.2 Macintosh Server

MacHTTP is available at the URL `http://www.uth.tmc.edu/mac_info/machftp_info.html`.

6.9.3 Windows and NT Servers

- HTTPS. A server for Intel- and Alpha-based Windows NT systems. It's available via anonymous FTP from emwac.ed.ac.uk in the directory pub/https (the URL is ftp://emwac.ed.ac.uk/pub/https). Make sure to download the version that's appropriate for your particular processor. You can either read a detailed announcement at the FTP site or use the URL ftp://emwac.ed.ac.uk/pub/https.txt.

- NCSA httpd for Windows. This server includes most of the features of the Unix version, including scripts (which generate pages on the fly, bassed on user input). This server is available by anonymous FTP from ftp.ncsa.uiuc.edu in the Web/ncsa_http/contrib directory as the file whtpl lab6.zip, or at the URL `ftp://ftp.ncsa.uiuc.edu/Web/ncsa_httpd/contrib/whtpllab6.zip`.

- SerWeb. A simple, effective server for Windows. It's available by anonymous FTP from winftp.cica.indiana.edu (or one of its mirror sites, such as nic.switch.ch), as the file serweb03.zip, in the directory /pub/pc/win3/winsock. There is also a Windows NT version of SerWeb, available by anonymous FTP from emwac.ed.ac.uk as the file `/pub/serweb/serweb_i.zip`.

- WEB4HAM. This server is available via anonymous FTP from ftp.informatik.unihamburg.de as the file /pub/net/winsock/web4ham.zip.

6.9.4 OS/2 Server

OS2HTTPD. For information, see the home page for details (the URL is ftp://ftp.netcom.com/pub/kfan/overview.html) or fetch the package by anonymous FTP from ftp.netcom.com in the directory pub/kfan.

6.9.5 DOS Server

KA9QNOS (nosllc.exe) is an Internet server package for DOS that includes HTTP and Gopher servers. It can be obtained via anonymous FTP from inorganic5.chem.ufl.edu at the directory /pub/nos, or biochemistry.cwru.edu.

6.9.6 VMS Servers

- CERN HTTP for VMS. This is a port of the CERN server. It's available at the URL http://delonline.cern.ch/disk_user/duns/doc/vms/distribution.html.

- Region 6 Threaded HTTP Server. A native VMS server that uses DECthreads. This offers a performance advantage, because VMS has a high overhead for each process, which is a problem for the frequently forking NCSA and CERN servers that began life under Unix. A multithreaded server avoids this overhead. It's available at the URL http://kegll.eng.ohio-state.edu/www/doc/serverinfo.html.

6.9.7 Amiga Server

NCSA's Unix server. This Unix server has been ported to the Amiga and is bundled with the AMosaic browser. See the URL http://insti.physics.sunysb.edu/AMosaic/home.html for details.

6.9.8 Commercial Servers

Currently, Netsite from Netscape Communications Corp. is a good one. The price is $5000 (list, $1495 for introductory) Mosaic Netsite Communications Server, and $25000 for Mosaic Netsite Commercie Server. (Netscape Communications Corp. 650 Castro St., Suite 500, Mt.View, CA 94041 (415)254-1900, FAX: (415)254-2601) Netscape Communications is a company formed from most of the original Mosaic team and under the business tutelege of Jim Clark, founder of Silicon Graphics.

6.9.9 Browsers

6.9.9.1 Windows

Cello (ftp.law.cornell.edu:/pub/LII/cello) and
Mosaic for Windows (ftp.ncsa.uiuc.edu:/PC/Mosaic)

6.9.9.2 Macintosh

Mosaic for Mac (ftp.ncsa.uiuc.edu:/Mac/Mosaic) and Samba info.cern.ch:/pub/www/bin/mac

6.9.9.3 X/Dec-windows/Motif

Mosaic for X (ftp.ncsa.uiuc.edu:/Web),
tkWWW Browser (harbor.ecn.purdue.edu:/pub/tcl/extensions/tkwww),
Viola for X (ftp.ora.com:/pub/www/viola,
Viola has two versions; one using Motif, one using Xlib.), Chimera (ftp.cs.unlv.edu:/pub/chimera,
Chimera is using Athena. doesn't require Motif)

6.9.9.4 Text-mode Unix

Line Mode Browser (info.cern.ch:/pub/www/src for dumb terminal),
Lynx full-screen (ftp2.cc.ukans.edu:/pub/WWW/lynx for VT100),
perlWWW (archive.cis.ohio-state.edu:/pub/w3browser)

6.10 HTML Documents

HTML An introductory reference, "A Beginner's Guide to HTML," is available at `http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html`.

HTML+ To learn more about HTML+, you can examine the ASCII text of a draft specification for it at
`ftp://ds.internic.net/internet-drafts/draft-raggett-www-html-00.txt.`; or a PostScript version of the same at
`ftp://ds.internic.net/internet-drafts/draft-raggett-www-html-00.ps`

6.11 Lisp based WWW tools

6.11.1 Report Generation language

Lisp serves well enough as a Report Generation Language (RPG) for creating Hyper Text Markup Language (HTML).

Note: 3.1 release of SIOD will have a '-v0' flag so that it can be used as a 'completely silent' scripting language, such as in this use as a common-gateway-interface (CGI) underneath a Hyper Text Transfer Protocol Daemon (HTTP) such as from NCSA, CERN or OSU. SIOD wins over TCL and PERL here in terms of smaller startup size and faster execution speed. (Its moved to /ftp.village.com/pub/gjc).

The script /cgi-protected/topical would be:
siod -v0 -i/usr/local/httpd/topical.scm "-e(serve-topical-db)"
and topical.scm is then:

```
(defun serve-topical-db ()
  (cgi-start)
  (let ((path (or (getenv "WWW_PATH_INFO") ""))
(args nil))
    (cond ((eqv? 0 (strcmp path ""))
   (html-start "TOPICAL NEWS")
   (let ((l (select-category-info-list)))
     (rollback)
     (while l
       (html "<H1><A HREF=\".../by-rco/"
     (encode-url-component (cat-symbol (car l)))
     "\">"
     (cat-title (car l))
     "</A>\n"
     "<A HREF=\".../by-title/"
     (encode-url-component (cat-symbol (car l)))
     "\"> (by title)."
     "</A></H1>\n")
       (setq l (cdr l)))))
  ((memq (intern (car (setq args (decode-url-components path))))
 '(by-rco by-title))
```

```
   (html-start (or (car (select-category-info (cadr args))) path))
   (let ((titles (if (eq (intern (car args)) 'by-rco)
     (nreverse (select-titles-mdates-mdate-order
(cadr args)))
   (select-titles-mdates (cadr args))))
 (n nil))
     (rollback)
     (setq n (length titles))
     (html (number->string n)
   " article"
   (if (eqv? n 1) "" "s")
   "<P>\n")
     (while titles
       (html "<H4>"
     (format-date (cadr (car titles)))
     " "
     "<A HREF=\".../article/"
     (encode-url-component (cadr args))
     "/"
     (encode-url-component (car (car titles)))
     "\">"
     " "
     (car (car titles))
     "</A></H4>\n")
       (setq titles (cdr titles)))))
  ((eq (intern (car args)) 'article)
   (let ((info (select-article-info (cadr args)
    (caddr args))))
     (rollback)
     (html-start (caddr args))
     (cond ((null info)
    (html "Sorry, could not find article in database."))
   ('else
    (html "<h4>received "
  (format-date (car info))
  " ")
    (cond ((not (eqv? 0 (strcmp (car info) (cadr info))))
   (html "updated "
 (format-date (cadr info))
 " ")))
    (cond ((not (eqv? 0 (strcmp (cadr info)
(car (cdddr info)))))
   (html "current "
 (format-date (car (cdddr info)))
 " ")))
    (html "</h4><P>\n<PRE>\n")
```

```
    (fwrite (caddr info) *html-out*)
    (html "</PRE>\n")))))
  ('else
   (html-start path)
   (html path)))
    (html-end)))

;; Supporting SQL procedures:

 (defsql select-category-info-list (CATEGORY TITLE PURGE_HOURS)
    "select category,title,purge_hours from topical_categories"
    ()
    *)

(defsql select-category-info (CATEGORY)
  "select title,purge_hours from topical_categories where category = ?"
  (TITLE PURGE_HOURS))

(defsql select-titles-mdates (CATEGORY)
  "select title,mdate from topical where category = ?"
  (TITLE MDATE)
  *)

(defsql select-titles-mdates-mdate-order (CATEGORY)
  "select title,mdate from topical where category = ? order by mdate"
  (TITLE MDATE)
  *)

(defsql select-article-info (CATEGORY TITLE)
  "select article,cdate,mdate,ldate from topical where
   category = ? and title = ?"
  (CDATE MDATE ARTICLE LDATE))
```

6.11.2 Scheme based scripting

(The following sections are extracted from MIT AI Lab using
http://www.ai.mit.edu/projects/su/su.html.)

There are several Scheme based projects at AI Lab. The following is the description of the MIT Scheme Underground project.

6.11.2.1 The Scheme Underground

The Scheme Underground is a new effort starting between LCS and the AI Lab to develop useful software packages in Scheme for use by research projects and for distribution on the net.

We want to take over the world. The internet badly needs a public domain software environment that allows the rapid construction of software tools using a modern programming language. Our goal is to build such a system using Scheme 48, an ultra-portable Scheme implementation which is easily interfaced to existing software written in other languages.

MIT Undergrads: want to hack Scheme for the summer?

Are you interested in hacking advanced Scheme systems at MIT? We are looking for several UROPS to do design and implementation work on these packages. These packages include Unix shells, World Wide Web systems, graphics, text editors, and base systems tools for a new Scheme implementation developed at MIT, Scheme 48.

A major emphasis of this effort will to create a hacker culture that teaches and encourages elegant coding style. People working on this project will be expected to be mature enough to allow other people to constructively critique their code. They should also be mature enought to critique others' code in a constructive and professional manner. This project will give you the opportunity to learn good coding style from highly experienced Scheme programmers, many of whom helped define the language.

UROP projects

Here is a partial list of projects for which we are willing to take summer UROPS. Many of these projects interweave and overlap. Any of them, if well-executed, would make a real impact both inside MIT and on the Internet.

Scsh, a Scheme-based shell for Unix.
World Wide Web projects
Graphics projects
Systems programming projects

We are looking for motivated hackers, with good programming taste, who like Scheme, are looking for interesting and fun projects, and are willing to continue into the fall semester. If any of these projects appeal to you, and you think you fit the bill, get in touch with us.

Olin Shivers (shivers@lcs.mit.edu)
Ian Horswill (ian@ai.mit.edu)
Jonathan Rees (jar@altdorf.ai.mit.edu)
Hal Abelson (hal@ai.mit.edu)
Franklyn Turbak (lyn@ai.mit.edu)

Ian Horswill — ian@ai.mit.edu

6.11.2.2 World Wide Web Projects of the Scheme Underground

Web agents

Suppose World-Wide Web servers had Scheme interpreters linked in with their executables. Then small Scheme programs could transfer themselves across the network from server to server to carry out tasks for their masters, perhaps coming back to a home machine to report on their results in the end. Scheme is a good choice for such a language because Scheme implementations are "safe" – it is easy to guarantee that an arbitrary Scheme program received by a server can't damage the server or compromise its security. With proper cryptographic authentication, an agent can access or commit the user's resources: buy movie tickets, check his bank account, and so forth.

TeXinfo to html converter

TeXinfo is a hypertext markup language for writing documentation used by the Gnu project. Many of the Gnu project's tools, such as Gnu emacs, gawk, bison, and gcc, are documented using the TeXinfo system. Documents written in TeXinfo can be either converted into on-line hypertext documentation, or converted into TeX and then typeset into paper manuals.

HTML is the the hypertext markup language for the World-Wide Web. A text processor that converts documents from TeXinfo to HTML would allow authors to write a single document, in TeXinfo, and provide it in all three formats.

6.11.3 Common Lisp HTTP server

John C. Mallery's Common Lisp HTTP server at http://www.ai.mit.edu/projects/iiip/doc/cl-http/server.html. It includes HTML authoring macros that make writing scripts a real snap.

6.11.4 Scheme based Indexer

Here is the information for the subject.

```
From: cyber_surfer@wildcard.demon.co.uk (Cyber Surfer)
Subject: Re: Using Lisp as an HTML scripting language.
Organization: The Wildcard Killer Butterfly Breeding Ground
Date: Wed, 4 Jan 1995 11:46:36 +0000
Message-ID: <789219996snz@wildcard.demon.co.uk>

I've been using Scheme to do exact that. I compile my own HTML
index (currently) for 480+ URLs, and 100+ pages. I find that
```

```
symbolic expressions are ideally suited to describing heirarchies
of (gopher-like) index pages.

It's still at the alpha stage, but when it's more complete, and
properly tested, I may put it on an FTP site. Right now, I'm
also writing a compiler for my index compiler. (For recursion,
see recursion.) I'd like to make it a lot faster, coz I have
a slow machine. Plus, the Lisp compiler will have other uses.
--
CommUnity: http://www.demon.co.uk/community/index.html
CommUnity: ftp://ftp.demon.co.uk/pub/archives/community
Me: http://cyber.sfgate.com/examiner/people/surfer.html
```

6.11.5 w3.el

Try William Perry's w3.el package: it's a very comprehensive WWW browser (still in development).... you can get it by anonymous ftp from:

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

It seems to working quite well with mule these days, and the new "display engine" is real quick.

If you are a beginner I hope you can work out how to install the package: there are instructions included.

6.12 How does the WWW compare to Gopher and WAIS?

While all three of these information presentation systems are client-server based, they differ in terms of their data models. In Gopher, data is either a menu, a document, an index, or a Telnet connection. In WAIS, everything is an index, and everything that is returned from the index is a document. In WWW, everything is a hypertext document, which may be searchable if it contains text. This means that WWW can represent Gopher and WAIS data models as well as provide extra functionality. Only WWW has hyperlinks.

Although text is supported by all three systems, WWW provides facilities for displaying "richer" text (for example, headings, lists, and emphasized text) in a standardized way. Image, audio, and video data are all supported by external views (for exampke, MPEGplay). There is little direct support for application-specific data in any of the three systems, and all use different methods for expressing type and encoding. Also, all three systems have

little support for controlling the presentation of nontext data. For example, backdrops and synchronizatioin in time are not supported, and buttons are only supported in a limited way. (See Adie at `ftp://ftp.edinburgh.ac.uk/pub/mmaccess` for a more comprehensive evaluation.)

6.12.1 Limitations

Perhaps the greatest limitation of current distributed hypermedia systems is that there is no simple way to find out what information has changed, what information is new, or even what information is out there on the Internet. Mosaic users commonly complain about difficulties finding resources on a particular topic or subject. Unless the user has extensive knowledge of the Internet, it is hard to locate resources of interest on the WWW.

Another problem is that there are simply too many links, as Hall has pointed out in a recent article. Users of distributed hypermedia systems are often overwhelmed by the large number of possible links and become dis-oriented when moving between different application packages and information servers.

Steps are being taken to address some of these problems in Mosaic. Serveral resources now provide information on new and established Mosaic servers by topic. For example, The WWW Virtual Library (The WWW Virtual Library,
`http://info.cern.ch/hypertext/Data-Sources/bySubject/Overview.html` June 1994) is a good place to find resources on a particular subject, and What's New With NCSA Mosaic (`http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/Docs/whatsnew.html` June 1994) carries announcements of new servers on the Web.

More recently, several software tools have been developed that create automatic indexes you can search. Two such tools are WebCrawler
(`http://www.bio-tech.washington.edu/WebQuery.html`)
and WWWWorm
(`http://www.cs.colorado.edu/home/mcbryan/WWWW.html`).

WebCrawler indexes the contents of the documents, whereas WWWWorm builds its index based on page titles and URL contents. Although tools that automatically traverse the Web are useful, tools that recursively search for information may not notice self-referencing links and may begin to return an infinite number of indexes.

6.12.2 Popularity

The WWW is gaining popularity as evidenced by the amount of network traffic (in bytes) across the National Science Foundation's North American Network (NSFNet). In fact,

WWW traffic recently surpassed Gopher traffic, according to statistics produced by monitoring traffic on the Internet backbone. (Since WWW browsers can also access Gopher servers, the Gopher traffic may be somewhat inflated.) From January to August 1993, the amount of traffic attributed to Web use multiplied by 414 times, and usage of the server at CERN doubled every four months – twice the rate of Internet expansion. The Web is now ranked eighth of all network services in terms of sheer byte traffic as of May 1994 (Gopher is ranked tenth). (World-Wide Web Growth, ftp:/nic.merit.edu, 1994)

6.13 Line Mode WWW Client

To access WWW with the line mode browser, type **www**. The default first document will appear on your screen. From this point, you should be able to navigate through the WWW universse by reading the text and following the instructions at the bottom of the screen. If you want to start with a first document other than the default, or if you want to change some other aspect of the usual interaction, there are a number of command line parameters and options available. The full format of the WWW command to invoke the line mode browser is:

```
www  <options> <docaddress <keyword>>
```

where, **docaddress** is the hypertext address of the document at which you want to start browsing. **keyword** queries the index specified by docaddress with the supplied keyword(s). A list of matching entries is displayed. Multiple keywords are separated by blanks. **options** are to control the session.

There is a line mode browser which provides a simple user-interface: references are numbers in square brackets next to words. Type the number and hit the RETURN key to follow a reference. For example, here is the beginning of the *subject Catalogue* you get on the CERN server:

```
------------------------------------------------------------------------
The World-Wide Web Virtual Library: Subject Catalogue
Information by subject

see also arrangements by organization[1] or by service type[2].
Mail www-request@info.cern.ch if you know of online information
not in these lists....

Aeronautics Mailing list archive index[3].

Astronomy and Astrophysics
Abstract Indexes[4] (unavailable). Astrophsics work at FNAL[5]
```

```
Bio Sciences[6] separate list.

Computing[7] Separate list.

Geography CIA World Fact Book[8], India: Miscellaneous
Information[9], Thai-Yunnan: Davis collection[10].

Law US copyright law[11]

Libraries[12] Lists of online catalogues etc.

Literature & Art [13] separate list.

Mathematics CIRM library[14] (french). The International
Journal of Analytical and Experimental Model
1-30, Bank, <RETURN> for more, Quit, or Help:

-------------------------------------------------------------------------
```

The following commands are available within WWW. Some are disabled when not applicable (eg, FIND is enabled only when the current document is an index). Angle brackets (<>) indicate an optional parameter.

Help gives a list of available commands depending on the context, and the hypertext address of the current document.

Manual displays the on-line manual.

Quit exits www.

Up, Down scrolls up or down one page in the current document.

Top, Bottom goes to the top or the bottom of the current document.

Back boes back to the document you were reading before.

Home goes back to the first document you were reading.

Next, Previous goes to the next or previous document in the list of pointers from the document that led to the current one.

List gives a numbered list of the links from the current document. To follow a link, type in the number.

Recall < *number* > if number is omitted, gives a numbered list of the documents you have visited. To display one specific document, re-issue the command with number.

< *Find* >**keyword** queries the current index with the supplied keyword(s). A list of matching entries is displayed with possibly links to further details. *Find* can be omitted if the first keyword does not conflict with another WWW command. Multiple keywords are separated by blanks.

Go address goes to the document represented by the given hypertext address, which is interpreted relatively to the current document.

Extra command available on Unix versions only:

Print prints the current document, without the numbereed document refereces. The default print command is lpr, but it may be defined in your local working environment by the variable `WWW_PRINT_COMMAND`.

6.14 References

- K.Hughes, Entering the WWW: A Guide to Cyberspace, ftp://taurus.cs.nps.navy.mil:/pub/mbmy/world-wide-web-guide.ps.Z 1994
- T. Berners-Lee, The HTTP Protocol as Implemented in W3, ftp://info.cern.ch:/pub/www/doc/http-spec.txt.Z Jan.1992
- R. Brandwein and M.Sendall, HTML Converters, http://info.cern.ch/hypertext/WWW/Tools/Filters.html 1994
- Intelliigent Agents, special issue, CACM Vol 37 No.7 July 1994
- Ronald J. Vetter, Chris Spell and Charles Ward: Mosaic and the World-Wide Web, Computer IEEECS, Vol27, No.10 pp 49-57, Oct. 1994.

Chapter 7

Network News

7.1 Usenet (NetNews) and NNTP

Usenet, or NetNews, is a collection of *newsgroups* distributed electronically around the world. Usenet provides a means for local users to communicate with each other, and with other users nationally and internationally.

Usenet was developed for Unix systems in 1979 by tow graduate students at Duke University. Within a year, fifty Unix sites, including Bell Labs, were participating. Now, there are thousands of sites running a number of perating systems on a variety of hardware platforms communicating via Usenet around the globe. Site administrators control their own istes. No one has any real control over any site but his own.

Sites are not entirely without influence on their neighbors, however. There is a vague notion of *upstream* and *downstream* related to the direction of high-volume news flow. To the extent that *upstream* sites decide what traffic they will carry for their *downstream* neighbors, those *upstream* sites ahve some influence on their neighbors' paricipation in Usenet.

There are many misconceptions about Usenet. Despite the myths: Usenet is not an organization,. Usenet is not a democracy. There is no person or group i charge of Usenet as a whole. Usenet is not an academic network. Although many Usenet sites are universities, research labs or other academic institutions, by plain count, most Usenet sites are commercial entities. Usenet is not the Internet. The Internet is only one of the various networks carrying Usenet traffic. Usenet is not a UUCP network. UUCP is a protocol for sending data over point-to-point connections, typically using dialup modems. Sites use UUCP to carry many kinds of traffic, of which Usenet is only one. Usenet is not a UNIX network. Usenet can be found on many operating systems. Usenet is not software. There are dozens of software packages used at various sites to transport and read Usenet articles. So no one program or package can be called *the Usenet software.*

If your site provides Usenet access, then there are a large number of software packages

available for reading news. These packages either access a local news spool, or use the NNTP protocol to access the news spool on some other computer in the network. Most of these packages provide the same basic abilities: o *subscription* to news groups, so that you can choose to read the postings of groups that interest you quickly and easily. o keeping records of which postings you have already read. o *threads* of discussion, so that you can follow groups of postings that deal with the same subject. o posting to news groups, so that you can easily participate in group discussions.

The following excerpted mostly from the article **USENET Software: History and Sources**, by Gene Spafford, is not an exhaustive list of the exsisting news packages:

For Unix Several popular screen-oriented news reading interfaces have been developed in the last few years to replace the traditinal **readnews** interface. The first of these was **vnews** and it was written by Kenneth Almquist. **vnews** provides a readnews-like command interface, but displays articles using direct screen positioning. It appears to have been inspired, to some extent, by the **notes** system. Vnews is currently distributed with the standard 2.11 news source.

A second, more versatile interface, **rn**, was develooped by Larry Wall (the author of Perl) and released in 1984. This interface also uses full-screen display with direct positioning, but it includes many other useful features and is very popular with many regular net readers. The interface includes reading, discarding, and/or processing of articles based on user-definable partterns and the ability of the user to develop customized macros for display and keyboard interaction. **rn** is currently at realse 4.4. It is being maintained by Stan Barber of the Baylor College of Medicice. **rn** is not provided with the standard news software release, but is very widely available because of its popularity. The software can be obtained from its official archive site, lib.tmc.edu, using FTP, and vial mail for archive-server@bcm.tmc.edu.

A variant of rn is **trn** by Wayne Davison. Trn adds the ability to follow threads of discussions in newsgroups; its latest version 2.2 is based on rn 4.4. It uses a Reference-line database to allow the user to take advantage of the discussion tree formed by an article and its replies. This results in a true reply-ordered reading of the articles, complete with a small ascii representation of the current article's position in the discussion tree. Trn can be obtained from ftp.coe.montana.edu in the /pub/trn directory, from uunet in the news subdirectory, and from many other archive servers world-wide.

There are two popular macro packages named **GNUS** and "GNews" that can be used with the GNU Emacs text editor. These allow reading, replying, and posting interaction with the news from inside the Emacs text editor. Client code exists to get the articles using NNTP ratgher than from a local disk. Copies can be found on most archive sites that carry the GNU archives.

nn is yet anotgher reader interface, developed by Kim F. Storm of Texas Instruments

A/S, Denmark, and released in 1989. **nn** differs from the traditional readnews and vnews by presenting a menu of article subject and sender-name lines, allowing you to preselect articles to read. **nn** is also a very fast newsreader, as it keeps a database of article headers on-line. (i.e. it trades space for time. A good rule of thumb is that the **nn** database size is 5%-10% of your news spool. So up to 110% of your news spool is the amount of space news and the **nn** database will take.) The current version of **nn** is 6.4.16. **nn** can be obtained via anonymous FTP from dkung.dk, uop.uop.edu or various other sites.

Yet another newsreader is the **tin** reader. It operates with threads, has different article organization methods, and is full-screen oriented. **tin** works on a local news spool or over an NNTP connection. It has been posted to alt.sources, and further information is available from Iain Lea. The current release of **tin** is 1.1PL5.

xrn is an X-11based interface to NNTP that was written by Rick Spickelmier and Ellen Sentovich (UC Berkeley). The current version is 6.17. **xrn** supports manuy features, including sorting by subject, user-settable key bindings, graceful handling of NNTP server crashes, and many of the features of **rn** (including KILL files and key bindings similar to rn). xrn is actively supported by the authors with bug fixing and feature addition support from many of the users. xrn can be retrieved from most of the popular FTP sites. (gatekeeper.dec.com, ftp.uu.net, export.lcs.mit.edu).

Another X11-based newsreader is **xvnews**. This is a news reader designed primarily for Sun workstations running OpenWindows. It runs with NNTP and is compatible with rn style commands. It is available from export.lcs.mit.edu in the contrib directory.

To read News in Japanese, use GNUS or **Mnews**.

There are many NNTP based newsreaders for various platforms. **ANU-NEWS** is for VMS, which is available from ftpvms.ira.uka.de. **PSU NetNews** is for IBM VM/CMS and has NNTP support. **NNMVS** is for TSO/ISPF, which is available at ftp.uni-stutgart.de under the directory /soft/kommunikation/news/beginner/software/nnmvs. A NNTP newsreader for Macintosh is called **News**. It is implemented as a Hyper-Card stack and depends on MacTCP. It is available from manuy Mac archives, including ftp.apple.comn and sumex-aim.stanford.edu. **Trumpet** is an NNTP newsreader for MSDOS. It requires the use of a packet driver. It is available as shareware from tasman.cc.utas.edu.au.

RFC 977 specifies NNTP, the Network News Transfer Protocol. RFC 1036 specifies the format of Usenet articles.

Most terrible recent problem on news reader is, the effect of explosion of news groups. In other words, there are too many newsgroups now. `.newsrc` file becomes bigger and almost 2M bytes for my case. Some news readers cannot handle this size of `.newsrc` by exceeding their capability.

7.2 RFC-977: Network News Transfer Protocol

(excerpted from RFC-977)

Network Working Group Brian Kantor (U.C. San Diego) Request for Comments: 977 Phil Lapsley (U.C. Berkeley) February 1986

Network News Transfer Protocol

A Proposed Standard for the Stream-Based Transmission of News

Status of This Memo

NNTP specifies a protocol for the distribution, inquiry, retrieval, and posting of news articles using a reliable stream-based transmission of news among the ARPA-Internet community. NNTP is designed so that news articles are stored in a central database allowing a subscriber to select only those items he wishes to read. Indexing, cross-referencing, and expiration of aged messages are also provided. This RFC suggests a proposed protocol for the ARPA-Internet community, and requests discussion and suggestions for improvements. Distribution of this memo is unlimited.

1. Introduction

For many years, the ARPA-Internet community has supported the distribution of bulletins, information, and data in a timely fashion to thousands of participants. We collectively refer to such items of information as "news". Such news provides for the rapid dissemination of items of interest such as software bug fixes, new product reviews, technical tips, and programming pointers, as well as rapid-fire discussions of matters of concern to the working computer professional. News is very popular among its readers.

There are popularly two methods of distributing such news: the Internet method of direct mailing, and the USENET news system.

1.1. Internet Mailing Lists

The Internet community distributes news by the use of mailing lists. These are lists of subscriber's mailbox addresses and remailing sublists of all intended recipients. These mailing lists operate by remailing a copy of the information to be distributed to each subscriber on the mailing list. Such remailing is inefficient when a mailing list grows beyond a dozen or so people, since sending a separate copy to each of the subscribers occupies large quantities of network bandwidth, CPU resources, and significant amounts of disk storage at the destination host. There is also a significant problem in maintenance of the list itself: as subscribers move from one job to another; as new subscribers join and old ones leave; and as hosts come in and out of service.

1.2. The USENET News System

Clearly, a worthwhile reduction of the amount of these resources used can be achieved if articles are stored in a central database on the receiving host instead of in each subscriber's mailbox. The USENET news system provides a method of doing just this. There is a central repository of the news articles in one place (customarily a spool directory of some sort), and a set of programs that allow a subscriber to select those items he wishes to read. Indexing, cross-referencing, and expiration of aged messages are also provided.

1.3. Central Storage of News

For clusters of hosts connected together by fast local area networks (such as Ethernet), it makes even more sense to consolidate news distribution onto one (or a very few) hosts, and to allow access to these news articles using a server and client model. Subscribers may then request only the articles they wish to see, without having to wastefully duplicate the storage of a copy of each item on each host.

1.4. A Central News Server

A way to achieve these economies is to have a central computer system that can provide news service to the other systems on the local area network. Such a server would manage the collection of news articles and index files, with each person who desires to read news bulletins doing so over the LAN. For a large cluster of computer systems, the savings in total disk space is clearly worthwhile. Also, this allows workstations with limited disk storage space to participate in the news without incoming items consuming oppressive amounts of the workstation's disk storage.

We have heard rumors of somewhat successful attempts to provide centralized news service using IBIS and other shared or distributed file systems. While it is possible that such a distributed file system implementation might work well with a group of similar computers running nearly identical operating systems, such a scheme is not general enough to offer service to a wide range of client systems, especially when many diverse operating systems may be in use among a group of clients. There are few (if any) shared or networked file systems that can offer the generality of service that stream connections using Internet TCP provide, particularly when a wide range of host hardware and operating systems are considered.

NNTP specifies a protocol for the distribution, inquiry, retrieval, and posting of news articles using a reliable stream (such as TCP) server-client model. NNTP is designed so that news articles need only be stored on one (presumably central) host, and subscribers on other hosts attached to the LAN may read news articles using stream connections to the news host.

NNTP is modelled upon the news article specifications in RFC 850, which describes the USENET news system. However, NNTP makes few demands upon the structure, content, or storage of news articles, and thus we believe it easily can be adapted to other non-USENET

news systems.

Typically, the NNTP server runs as a background process on one host, and would accept connections from other hosts on the LAN. This works well when there are a number of small computer systems (such as workstations, with only one or at most a few users each), and a large central server.

1.5. Intermediate News Servers

For clusters of machines with many users (as might be the case in a university or large industrial environment), an intermediate server might be used. This intermediate or "slave" server runs on each computer system, and is responsible for mediating news reading requests and performing local caching of recently-retrieved news articles.

Typically, a client attempting to obtain news service would first attempt to connect to the news service port on the local machine. If this attempt were unsuccessful, indicating a failed server, an installation might choose to either deny news access, or to permit connection to the central "master" news server.

For workstations or other small systems, direct connection to the master server would probably be the normal manner of operation.

This specification does not cover the operation of slave NNTP servers. We merely suggest that slave servers are a logical addition to NNTP server usage which would enhance operation on large local area networks.

1.6. News Distribution

NNTP has commands which provide a straightforward method of exchanging articles between cooperating hosts. Hosts which are well connected on a local area or other fast network and who wish to actually obtain copies of news articles for local storage might well find NNTP to be a more efficient way to distribute news than more traditional transfer methods (such as UUCP).

In the traditional method of distributing news articles, news is propagated from host to host by flooding - that is, each host will send all its new news articles on to each host that it feeds. These hosts will then in turn send these new articles on to other hosts that they feed. Clearly, sending articles that a host already has obtained a copy of from another feed (many hosts that receive news are redundantly fed) again is a waste of time and communications resources, but for transport mechanisms that are single-transaction based rather than interactive (such as UUCP in the UNIX-world), distribution time is diminished by sending all articles and having the receiving host simply discard the duplicates. This is an especially true when communications sessions are limited to once a day.

Using NNTP, hosts exchanging news articles have an interactive mechanism for deciding which articles are to be transmitted. A host desiring new news, or which has new news to send, will typically contact one or more of its neighbors using NNTP. First it will inquire if

any new news groups have been created on the serving host by means of the NEWGROUPS command. If so, and those are appropriate or desired (as established by local site-dependent rules), those new newsgroups can be created.

The client host will then inquire as to which new articles have arrived in all or some of the newsgroups that it desires to receive, using the NEWNEWS command. It will receive a list of new articles from the server, and can request transmission of those articles that it desires and does not already have.

Finally, the client can advise the server of those new articles which the client has recently received. The server will indicate those articles that it has already obtained copies of, and which articles should be sent to add to its collection.

In this manner, only those articles which are not duplicates and which are desired are transferred.

2. The NNTP Specification

2.1. Overview

The news server specified by this document uses a stream connection (such as TCP) and SMTP-like commands and responses. It is designed to accept connections from hosts, and to provide a simple interface to the news database.

This server is only an interface between programs and the news databases. It does not perform any user interaction or presentation- level functions. These "user-friendly" functions are better left to the client programs, which have a better understanding of the environment in which they are operating.

When used via Internet TCP, the contact port assigned for this service is 119.

2.2. Character Codes

Commands and replies are composed of characters from the ASCII character set. When the transport service provides an 8-bit byte (octet) transmission channel, each 7-bit character is transmitted right justified in an octet with the high order bit cleared to zero.

2.3. Commands

Commands consist of a command word, which in some cases may be followed by a parameter. Commands with parameters must separate the parameters from each other and from the command by one or more space or tab characters. Command lines must be complete with all required parameters, and may not contain more than one command.

Commands and command parameters are not case sensitive. That is, a command or parameter word may be upper case, lower case, or any mixture of upper and lower case.

Each command line must be terminated by a CR-LF (Carriage Return - Line Feed) pair.

Command lines shall not exceed 512 characters in length, counting all characters including spaces, separators, punctuation, and the trailing CR-LF (thus there are 510 characters maximum allowed for the command and its parameters). There is no provision for continuation

command lines.

2.4. Responses

Responses are of two kinds, textual and status.

2.4.1. Text Responses

Text is sent only after a numeric status response line has been sent that indicates that text will follow. Text is sent as a series of successive lines of textual matter, each terminated with CR-LF pair. A single line containing only a period (.) is sent to indicate the end of the text (i.e., the server will send a CR-LF pair at the end of the last line of text, a period, and another CR-LF pair).

If the text contained a period as the first character of the text line in the original, that first period is doubled. Therefore, the client must examine the first character of each line received, and for those beginning with a period, determine either that this is the end of the text or whether to collapse the doubled period to a single one.

The intention is that text messages will usually be displayed on the user's terminal whereas command/status responses will be interpreted by the client program before any possible display is done.

2.4.2. Status Responses

These are status reports from the server and indicate the response to the last command received from the client.

Status response lines begin with a 3 digit numeric code which is sufficient to distinguish all responses. Some of these may herald the subsequent transmission of text.

The first digit of the response broadly indicates the success, failure, or progress of the previous command.

1xx - Informative message
2xx - Command ok
3xx - Command ok so far, send the rest of it.
4xx - Command was correct, but couldn't be performed for some reason.
5xx - Command unimplemented, or incorrect, or a serious program error occurred.

The next digit in the code indicates the function response category.

x0x - Connection, setup, and miscellaneous messages
x1x - Newsgroup selection
x2x - Article selection
x3x - Distribution functions
x4x - Posting

x8x - Nonstandard (private implementation) extensions
x9x - Debugging output

The exact response codes that should be expected from each command are detailed in the description of that command. In addition, below is listed a general set of response codes that may be received at any time.

Certain status responses contain parameters such as numbers and names. The number and type of such parameters is fixed for each response code to simplify interpretation of the response.

Parameters are separated from the numeric response code and from each other by a single space. All numeric parameters are decimal, and may have leading zeros. All string parameters begin after the separating space, and end before the following separating space or the CR-LF pair at the end of the line. (String parameters may not, therefore, contain spaces.) All text, if any, in the response which is not a parameter of the response must follow and be separated from the last parameter by a space. Also, note that the text following a response number may vary in different implementations of the server. The 3-digit numeric code should be used to determine what response was sent.

Response codes not specified in this standard may be used for any installation-specific additional commands also not specified. These should be chosen to fit the pattern of x8x specified above. (Note that debugging is provided for explicitly in the x9x response codes.) The use of unspecified response codes for standard commands is prohibited.

We have provided a response pattern x9x for debugging. Since much debugging output may be classed as "informative messages", we would expect, therefore, that responses 190 through 199 would be used for various debugging outputs. There is no requirement in this specification for debugging output, but if such is provided over the connected stream, it must use these response codes. If appropriate to a specific implementation, other x9x codes may be used for debugging. (An example might be to use e.g., 290 to acknowledge a remote debugging request.)

2.4.3. General Responses

The following is a list of general response codes that may be sent by the NNTP server. These are not specific to any one command, but may be returned as the result of a connection, a failure, or some unusual condition.

In general, 1xx codes may be ignored or displayed as desired; code 200 or 201 is sent upon initial connection to the NNTP server depending upon posting permission; code 400 will be sent when the NNTP server discontinues service (by operator request, for example); and 5xx codes indicate that the command could not be performed for some unusual reason.

100 help text
190 through 199 debug output

200 server ready - posting allowed
201 server ready - no posting allowed

400 service discontinued

500 command not recognized
501 command syntax error
502 access restriction or permission denied
503 program fault - command not performed

Chapter 8

Network-wide File System Sharing

Most of the tools are based on anonymous FTP feature.

8.1 Prospero

The Prospero File System is a tool for organizing distributed information. Prospero lets users build customized views, or *virtual systems*, of directories distributed throughout the Internet. One of the applications is Archie.

The Prospero name space forms a generalized directed graph, where intermediate nodes are directories, leaves are files, and edges are Prospero links. Just like traditional distributed file systems, subtrees of the Prospero name space can be stored by different Prospero servers. A user's name space corresponds to the subgraph starting at a particular node, which is the root of the user's name space.

Users organize their name space hierarchiecally by building views, which are essentially directories composed from various sources uncluding the user's own views and imported views from other users. These directories may reside in different Prospero servers. One special type of view is an index, which returns a directory of objects that satisfy some query. This allows Prospero users to access other search engines. For example, the Prosper-archie interface lets users build views containing directories with objects resulting from archie queries.

Users can find information by navigating through available views. The Prospero client provides users with navigational tools that are analogous to the ones provided by tranditional file systems. One command allows a user to change his current virtual directory to the one specified in the command. Another command displays the name of the current virtual directory and describes its physical location. Users can also list the contents of a virtual directory.

A user starts by using the `vfsetup` command which places the user at a specific point in

the Prospero name space. Then through `vcd`, `vwd`, and `vls` the user can change his current virtual directory, display the name of the current virtual directory, and list its contentts, respectively. To submit an archie request to find all file names that match the string *wais*, the user changes directory to */databases/archie/regex/wais*, and lists the results by using the `vls` command.

When a user finds an interesting object, which may be a simple file or a view, the user can include the object in his view by linking to it. The Prospero client provides commands to add and delete links from the current node in the user's name space to a target node or leaf. A Prospero link specifies the name of the host wherer the object is stored and the local name of the object on that host. If the target of the link is a directory, the link provides information to resolve a name in that directory by querying the corresponding server. For files, the associated links contact the appropriate server to provide access mode information. Currently, Prospero supports Sun's Network File System, the Andrew File System, and anonymous FTP.

8.2 Alex

Alex is a file system that provides users with transparent read access to files in Internet anonymous FTP sites. Through Alex, users see the collection of Internet anonymous FTP sites and their corresponding directories and files as a hierarchical file system, where intermediate nodes are Internet domains, hosts, or directories within hosts, and leaves are files. Using the standard filesystem commands, users can browse through this hierarchy and retrieve files of interest. To get reasonable performance, Alex caches information such as machine names, directory inforemation, and the contents or remote files. Alex implements a soft consistency mechanism which guarantees that only updates that occureed in the last 5% of the reported age of the file might not yet been reflected locally. Alex is currently implemented as an NFS server, and already integrates access to archie. WAIS servers that index README files and computer science technical reports available through Alex have also been built. (Vincent Cate. Alex-a global filesystem. On-line documentation, April 1992.)

8.3 Ange-FTP feature on Emacs

In Emacs 19 an dsome earlier version of Emacs with this feature, you can refer to files on other machines using a special file name syntax:

/HOST:FILENAME /USER@HOST:FILENAME

When you do this, Emacs uses the FTP program to read and write files on the specified host. It logs in through FTP using your user name or the name USER. It may ask you for a password from time to time; this is used for logging in on HOST.

'$' in a file name is used to substitute environment variables. For example, if you have used the shell command 'export FOO=rms/hacks' to set up an environment variable named 'FOO', then you can use '/u/$FOO/test.c' or '/u/$FOO/test.c' as an abbreviation for '/u/rms/hacks/test.c'. The environment variable name consists of all the alphanumeric characters after the '$'; alternatively, it may be enclosed in braces after the '$'. Note that shell commands to set environment variables affect Emacs only if done before Emacs is started.

To access a file with '$' in its name, type '$$'. This pair is converted to a single '$' at the same time as variable substitution is performed for single '$'. The Lisp function that performs the substitution is called 'substitute-in-file-name'. The substitution is performed only on file names read as such using the minibuffer.

Chapter 9

HTML Beginner's Guide

(The contents of this chapter is entirely from the server at ncsa.uiuc.edu).

A Beginner's Guide to HTML

This is a primer for producing documents in HTML, the markup language used by the World Wide Web.

Acronym Expansion
What This Primer Doesn't Cover
Creating HTML Documents
   The Minimal HTML Document
   Basic Markup Tags
      Titles
      Headings
      Paragraphs
   Linking to Other Documents
      Relative Links Versus Absolute Pathnames
      Uniform Resource Locator
      Anchors to Specific Sections in Other Documents
      Anchors to Specific Sections Within the Current Document
Additional Markup Tags
   Lists
      Unnumbered Lists
      Numbered Lists
      Definition Lists
      Nested Lists
   Preformatted Text
   Extended Quotes
   Addresses
Character Formatting
   Physical Versus Logical: Use Logical Tags When Possible

Logical Styles
Physical Styles
Using Character Tags
Special Characters
Escape Sequences
Forced Line Breaks
Horizontal Rules
In-line Images
Alternate Text for Viewers That Can't Display Images
External Images, Sounds, and Animations
Troubleshooting
Avoid Overlapping Tags
Embed Anchors and Character Tags, But Not Anything Else
Check Your Links
A Longer Example
For More Information
Fill-out Forms
Style Guides
Other Introductory Documents
Additional References

Acronym Expansion

WWW World Wide Web (or Web, for short).

SGML Standard Generalized Markup Language – this is a standard for describing markup languages.

DTD Document Type Definition – this is a specific markup language, written using SGML.

HTML HyperText Markup Language – HTML is a SGML DTD. In practical terms, HTML is a collection of styles (indicated by markup tags) that define the various components of a World Wide Web document.

What This Primer Doesn't Cover

This primer assumes that you have: at least a passing knowledge of how to use NCSA Mosaic or some other Web browser

a general understanding of how Web servers and client browsers work access to a Web server for which you would like to produce HTML documents, or that you wish to produce HTML documents for personal use

Creating HTML Documents

HTML documents are in plain (also known as ASCII) text format and can be created using any text editor (e.g., Emacs or vi on UNIX machines). A couple of Web browsers (tkWWW for X Window System machines and CERN's Web browser for NeXT computers) include rudimentary HTML editors in a WYSIWYG environment. There are also some WYSIWIG editors available now (e.g. HotMetal for Sun Sparcstations, HTML Edit for Macintoshes). You may wish to try one of them first before delving into the details of HTML.

You can preview a document in progress with NCSA Mosaic (and some other Web browsers). Open it with the Open Local command under the File menu.

After you edit the source HTML file, save the changes. Return to NCSA Mosaic and Reload the document. The changes are reflected in the on-screen display.

The Minimal HTML Document

Here is a bare-bones example of HTML:

```
<TITLE>The simplest HTML example</TITLE>
<H1>This is a level-one heading</H1>
Welcome to the world of HTML.
This is one paragraph.<P>
And this is a second.<P>
```

Click here to see the formatted version of the example.

HTML uses markup tags to tell the Web browser how to display the text. The above example uses:

```
the <TITLE> tag (and corresponding </TITLE> tag), which specifies the
title of the document
the <H1> header tag (and corresponding </H1>)
the <P> paragraph-separator tag
```

HTML tags consist of a left angle bracket (<), (a "less than" symbol to mathematicians), followed by name of the tag and closed by a right angular bracket (>). Tags are usually paired, e.g. <H1> and </H1>. The ending tag looks just like the starting tag except a slash (/) precedes the text within the brackets. In the example, <H1> tells the Web browser to start formatting a level-one heading; </H1> tells the browser that the heading is complete.

The primary exception to the pairing rule is the <P> tag. There is no such thing as </P>.

NOTE: HTML is not case sensitive. <title> is equivalent to <TITLE> or <TiTlE>.

Not all tags are supported by all World Wide Web browsers. If a browser does not support a tag, it just ignores it.

Basic Markup Tags

Title

Every HTML document should have a title. A title is generally displayed separately from the document and is used primarily for document identification in other contexts (e.g., a WAIS search). Choose about half a dozen words that describe the document's purpose.

In the X Window System and Microsoft Windows versions of NCSA Mosaic, the Document Title field is at the top of the screen just below the pulldown menus. In NCSA Mosaic for Macintosh, text tagged as <TITLE> appears as the window title.

Headings

HTML has six levels of headings, numbered 1 through 6, with 1 being the most prominent. Headings are displayed in larger and/or bolder fonts than normal body text. The first heading in each document should be tagged <H1>. The syntax of the heading tag is:

```
<Hy>Text of heading </Hy >
```

where y is a number between 1 and 6 specifying the level of the heading.

For example, the coding for the "Headings" section heading above is

```
<H3>Headings</H3>
```

Title versus first heading

In many documents, the first heading is identical to the title. For multipart documents, the text of the first heading should be suitable for a reader who is already browsing related information (e.g., a chapter title), while the title tag should identify the document in a wider context (e.g., include both the book title and the chapter title, although this can sometimes become overly long).

Paragraphs

Unlike documents in most word processors, carriage returns in HTML files aren't significant. Word wrapping can occur at any point in your source file, and multiple spaces are collapsed into a single space. (There are couple of exceptions; space following a <P> or <Hy> tag, for example, is ignored.) Notice that in the bare-bones example, the first paragraph is coded as

```
Welcome to HTML.
This is the first paragraph. <P>
```

In the source file, there is a line break between the sentences. A Web browser ignores this line break and starts a new paragraph only when it reaches a <P> tag.

Important: You must separate paragraphs with <P>. The browser ignores any indentations or blank lines in the source text. HTML relies almost entirely on the tags for formatting instructions, and without the <P> tags, the document becomes one large paragraph. (The exception is text tagged as "preformatted," which is explained below.) For instance, the following would produce identical output as the first bare-bones HTML example:

```
<TITLE>The simplest HTML example</TITLE><H1>This is a level
one heading</H1>Welcome to the world of HTML. This is one
paragraph.<P>And this is a second.<P>
```

However, to preserve readability in HTML files, headings should be on separate lines, and paragraphs should be separated by blank lines (in addition to the <P>tags).

NCSA Mosaic handles <P> by ending the current paragraph and inserting a blank line.

In HTML+, a successor to HTML currently in development, <P> becomes a "container" of text, just as the text of a level-one heading is "contained" within <H1> ... </H1>:

```
<P>
This is a paragraph in HTML+.
</P>
```

The difference is that the </P> closing tag can always be omitted. (That is, if a browser sees a <P>, it knows that there must be an implied </P> to end the previous paragraph.) In other words, in HTML+, <P> is a beginning-of-paragraph marker.

The advantage of this change is that you will be able to specify formatting options for a paragraph. For example, in HTML+, you will be able to center a paragraph by coding

```
<P ALIGN=CENTER>
This is a centered paragraph. This is HTML+, so you can't do it yet.
```

This change won't effect any documents you write now, and they will continue to look just the same with HTML+ browsers.

Linking to Other Documents

The chief power of HTML comes from its ability to link regions of text (and also images) to another document. The browser highlights these regions (usually with color and/or underlines) to indicate that they are hypertext links (often shortened to hyperlinks or simply links).

HTML's single hypertext-related tag is <A>, which stands for anchor. To include an anchor in your document:

1. Start the anchor with <A . (There's a space after the A.)
2. Specify the document that's being pointed to by entering the parameter HREF="filename" followed by a closing right angle bracket:>
3. Enter the text that will serve as the hypertext link in the current document.
4. Enter the ending anchor tag: </A>.

Here is an sample hypertext reference:

```
<A HREF="MaineStats.html">Maine</A>
```

This entry makes the word "Maine" the hyperlink to the document MaineStats.html, which is in the same directory as the first document. You can link to documents in other directories by specifying the relative path from the current document to the linked document. For example, a link to a file NJStats.html located in the subdirectory AtlanticStates would be:

```
<A HREF="AtlanticStates/NJStats.html">New Jersey</A>
```

These are called relative links. You can also use the absolute pathname of the file if you wish. Pathnames use the standard UNIX syntax.

Relative Links Versus Absolute Pathnames

In general, you should use relative links, because

1. You have less to type.
2. It's easier to move a group of documents to another location, because the relative path names will still be valid.

However, use absolute pathnames when linking to documents that are not directly related. For example, consider a group of documents that comprise a user manual. Links within this group should be relative links. Links to other documents (perhaps a reference to related software) should use full path names. This way, if you move the user manual to a different directory, none of the links would have to be updated.

Uniform Resource Locator

The World Wide Web uses Uniform Resource Locators (URLs) to specify the location of files on other servers. A URL includes the type of resource being accessed (e.g., gopher, WAIS), the address of the server, and the location of the file. The syntax is:

```
scheme://host.domain[:port]/path/filename
```

where scheme is one of

file (a file on your local system, or a file on an anonymous FTP server), http (a file on a World Wide Web server), gopher (a file on a Gopher server), WAIS (a file on a WAIS server), news (an Usenet newsgroup), telnet (a connection to a Telnet-based service)

The port number can generally be omitted. (That means unless someone tells you otherwise, leave it out.)

For example, to include a link to this primer in your document, you would use

```
<A HREF = "http://www.ncsa.uiuc.edu/General/Internet/WWW/HTMLPrimer.html">
NCSA's Beginner's Guide to HTML</A>
```

This would make the text "NCSA's Beginner's Guide to HTML" a hyperlink to this document.

For more information on URLs, look at

WWW Names and Addresses, URIs, URLs, URNs, written by people at CERN A Beginner's Guide to URLs, located on the NCSA Mosaic Help menu

Links to Specific Sections in Other Documents

Anchors can also be used to move to a particular section in a document. Suppose you wish to set a link from document A and a specific section in document B. (Call this file documentB.html.) First you need to set up a named anchor in document B. For example, to set up an anchor named "Jabberwocky" to document B, enter

```
Here's <A NAME = "Jabberwocky">some text</a>
```

Now when you create the link in document A, include not only the filename, but also the named anchor, separated by a hash mark (#).

```
This is my <A HREF = "documentB.html#Jabberwocky">link</A> to document B.
```

Now clicking on the word "link" in document A sends the reader directly to the words "some text" in document B.

Links to Specific Sections Within the Current Document

The technique is exactly the same except the filename is omitted.

For example, to link to the Jabberwocky anchor from within the same file (Document B), use

```
This is <A HREF = "#Jabberwocky">Jabberwocky link</A> from within Document B.
```

Additional Markup Tags

The preceding is sufficient to produce simple HTML documents. For more complex documents, HTML has tags for several types of lists, preformatted sections, extended quotations, character formatting, and other items.

Lists

HTML supports unnumbered, numbered, and definition lists.

Unnumbered Lists

To make an unnumbered list,

1. Start with an opening list <UL> tag. 2. Enter the <LI> tag followed by the individual item. (No closing </LI> tag is needed.) 3. End with a closing list </UL> tag.

Below an example two-item list:

```
<UL>
<LI> apples
<LI> bananas
</UL>
```

The output is:

- apples
- bananas

The <LI> items can contain multiple paragraphs. Just separate the paragraphs with the <P> paragraph tags.

Numbered Lists

A numbered list (also called an ordered list, from which the tag name derives) is identical to an unnumbered list, except it uses <OL> instead of <UL>. The items are tagged using the same <LI> tag. The following HTML code

```
<OL>
<LI> oranges
<LI> peaches
<LI> grapes
</OL>
```

produces this formatted output:

1. oranges
2. peaches
3. grapes

Definition Lists

A definition list usually consists of alternating a term (abbreviated as DT) and a definition (abbreviated as DD). Web browsers generally format the definition on a new line.

The following is an example of a definition list:

```
<DL>
<DT> NCSA
<DD> NCSA, the National Center for Supercomputing Applications,
     is located on the campus of the University of Illinois
     at Urbana-Champaign. NCSA is one of the participants in the
     National MetaCenter for Computational Science and Engineering.
<DT> Cornell Theory Center
<DD> CTC is located on the campus of Cornell University in Ithaca,
     New York. CTC is another participant in the National MetaCenter
     for Computational Science and Engineering.
</DL>
```

The output looks like:

```
NCSA
   NCSA, the National Center for Supercomputing Applications, is located on
   the campus of the University of Illinois at Urbana-Champaign. NCSA is one
   of the participants in the National MetaCenter for Computational Science and
   Engineering.
Cornell Theory Center
   CTC is located on the campus of Cornell University in Ithaca, New York.
   CTC is another participant in the National MetaCenter for Computational
   Science and Engineering.
```

The <DT> and <DD> entries can contain multiple paragraphs (separated by <P> paragraph tags), lists, or other definition information.

Nested Lists

Lists can be arbitrarily nested, although in practice you probably should limit the nesting to three levels. You can also have a number of paragraphs, each containing a nested list, in a single list item.

An example nested list:

```
<UL>
<LI> A few New England states:
    <UL>
    <LI> Vermont
    <LI> New Hampshire
    </UL>
<LI> One Midwestern state:
    <UL>
    <LI> Michigan
    </UL>
</UL>
```

The nested list is displayed as

```
A few New England states:
   Vermont
   New Hampshire
One Midwestern state:
   Michigan
```

Preformatted Text

Use the <PRE> tag (which stands for "preformatted") to generate text in a fixed-width font and cause spaces, new lines, and tabs to be significant. (That is, multiple spaces are displayed as multiple spaces, and lines break in the same locations as in the source HTML file.) This is useful for program listings. For example, the following lines

```
<PRE>
  #!/bin/csh
  cd $SCR
  cfs get mysrc.f:mycfsdir/mysrc.f
  cfs get myinfile:mycfsdir/myinfile
  fc -02 -o mya.out mysrc.f
  mya.out
  cfs save myoutfile:mycfsdir/myoutfile
  rm *
</PRE>
```

display as

```
#!/bin/csh
cd $SCR
cfs get mysrc.f:mycfsdir/mysrc.f
cfs get myinfile:mycfsdir/myinfile
fc -02 -o mya.out mysrc.f
mya.out
cfs save myoutfile:mycfsdir/myoutfile
rm *
```

Hyperlinks can be used within <PRE> sections. You should avoid using other HTML tags within <PRE> sections, however.

Note that because <, >, and & have special meaning in HTML, you have to use their escape sequences (<, >, and &, respectively) to enter these characters. See the section Special Characters for more information.

Extended Quotations

Use the <BLOCKQUOTE> tag to include quotations in a separate block on the screen. Most browsers generally indent to separate it from surrounding text.

An example:

```
<BLOCKQUOTE>
I still have a dream. It is a dream deeply rooted in the
American dream. <P>
I have a dream that one day this nation will rise up and
live out the true meaning of its creed. We hold these truths
to be self-evident that all men are created equal. <P>
</BLOCKQUOTE>
```

The result is:

I still have a dream. It is a dream deeply rooted in the American dream.

I have a dream that one day this nation will rise up and live out the true meaning of its creed. We hold these truths to be self-evident that all men are created equal.

Addresses

The <ADDRESS> tag is generally used to specify the author of a document and a means of contacting the author (e.g., an email address). This is usually the last item in a file.

For example, the last line of the online version of this guide is

```
<ADDRESS>
A Beginner's Guide to HTML / NCSA / pubs@ncsa.uiuc.edu
</ADDRESS>
```

The result is

A Beginner's Guide to HTML / NCSA / pubs@ncsa.uiuc.edu

NOTE: <ADDRESS> is not used for postal addresses. See "Forced Line Breaks" on page 10 to see how to format postal addresses.

Character Formatting

You can code individual words or sentences with special styles. There are two types of styles: logical and physical. Logical styles tag text according to its meaning, while physical styles specify the specific appearance of a section. For example, in the preceding sentence, the words "logical styles" was tagged as a "definition." The same effect (formatting those words in italics), could have been achieved via a different tag that specifies merely "put these words in italics."

Physical Versus Logical: Use Logical Styles When Possible

If physical and logical styles produce the same result on the screen, why are there both? We devolve, for a couple of paragraphs, into the philosophy of SGML, which can be summed in a Zen-like mantra: "Trust your browser."

In the ideal SGML universe, content is divorced from presentation. Thus, SGML tags a level-one heading as a level-one heading, but does not specify that the level-one heading should be, for instance, 24-point bold Times centered on the top of a page. The advantage of this approach (it's similar in concept to style sheets in many word processors) is that if you decide to change level-one headings to be 20-point left-justified Helvetica, all you have to do is change the definition of the level-one heading in the presentation device (i.e., your World Wide Web browser).

The other advantage of logical tags is that they help enforce consistency in your documents. It's easier to tag something as <H1> than to remember that level-one headings are 24-point bold Times or whatever. The same is true for character styles. For example, consider the <STRONG> tag. Most browsers render it in bold text. However, it is possible that a reader would prefer that these sections be displayed in red instead. Logical styles offer this flexibility.

Logical Styles

<DFN> for a word being defined. Typically displayed in italics. (NCSA Mosaic is a World Wide Web browser.)

<EM> for emphasis. Typically displayed in italics. (Watch out for pickpockets.)

<CITE> for titles of books, films, etc. Typically displayed in italics. (A Beginner's Guide to HTML)

<CODE> for snippets of computer code. Displayed in a fixed-width font. (The <stdio.h> header file)

<KBD> for user keyboard entry. Should be displayed in a bold fixed-width font, but many browsers render it in the plain fixed-width font. (Enter passwd to change your password.)

<SAMP> for computer status messages. Displayed in a fixed-width font. (Segmentation fault: Core dumped.)

<STRONG> for strong emphasis. Typically displayed in bold. (Important)

<VAR> for a "metasyntactic" variable, where the user is to replace the variable with a specific instance. Typically displayed in italics. (rm filename deletes the file.)

Physical Styles

<B> bold text

<I> italic text

<TT> typewriter text, e.g. fixed-width font.

Using Character Tags

To apply a character style,

1. Start with <tag>, where tag is the desired character formatting tag, to indicate the beginning of the tagged text.
2. Enter the tagged text.
3. End the passage with </tag>.

Special Characters

Escape Sequences

Four characters of the ASCII character set – the left angle bracket (<), the right angle bracket (>), the ampersand (&) and the double quote (") – have special meaning within HTML and therefore cannot be used "as is" in text. (The angle brackets are used to indicate the beginning and end of HTML tags, and the ampersand is used to indicate the beginning of an escape sequence.)

To use one of these characters in an HTML document, you must enter its escape sequence instead:

< the escape sequence for <

> the escape sequence for >

& the escape sequence for &

" the escape sequence for "

Additional escape sequences support accented characters. For example:

ö the escape sequence for a lowercase o with an umlaut: Av

ñ the escape sequence for a lowercase n with an tilde: Aq

È the escape sequence for an uppercase E with a grave accent: AH

A full list of supported characters can be found at CERN.

NOTE: Unlike the rest of HTML, the escape sequences are case sensitive. You cannot, for instance, use < instead of <.

Forced Line Breaks

The
 tag forces a line break with no extra space between lines. (By contrast, most browsers format the <P> paragraph tag with an additional blank line to more clearly indicate the beginning the new paragraph.)

One use of
 is in formatting addresses:

```
National Center for Supercomputing Applications<BR>
605 East Springfield Avenue<BR>
Champaign, Illinois 61820-5518<BR>
```

Horizontal Rules

The <HR> tag produces a horizontal line the width of the browser window.

In-line Images

Most Web browsers can display in-line images (that is, images next to text) that are in X Bitmap (XBM) or GIF format. Each image takes time to process and slows down the initial display of the document, so generally you should not include too many or overly large images.

To include an in-line image, use

```
<IMG SRC=image\_URL>
```

where image_URL is the URL of the image file. The syntax for IMG SRC URLs is identical to that used in an anchor HREF. If the image file is a GIF file, then the filename part of image_URL must end with .gif. Filenames of X Bitmap images must end with .xbm.

By default the bottom of an image is aligned with the text as shown in this paragraph.

Add the ALIGN=TOP option if you want the browser to align adjacent text with the top of the image as shown in this paragraph. The full in-line image tag with the top alignment is:

```
<IMG ALIGN=top SRC=image\_URL>
```

ALIGN=MIDDLE aligns the text with the center of the image.

Alternate Text for Browsers That Can't Display Images

Some World Wide Web browsers, primarily those that run on VT100 terminals, cannot display images. The ALT option allows you to specify text to be displayed when an image cannot be. For example:

```
<IMG SRC = "UpArrow.gif" ALT = "Up">
```

where UpArrow.gif is the picture of an upward pointing arrow. With NCSA Mosaic and other graphics-capable viewers, the user sees the up arrow graphic. With a VT100 browser, such as lynx, the user sees the word "Up."

External Images, Sounds, and Animations

You may want to have an image open as a separate document when a user activates a link on either a word or a smaller, in-line version of the image included in your document. This is considered an external image and is useful if you do not wish to slow down the loading of the main document with large in-line images.

To include a reference to an external image, use

```
<A HREF = image\_URL>link anchor</A>
```

Use the same syntax is for links to external animations and sounds. The only difference is the file extension of the linked file. For example,

```
<A HREF = "QuickTimeMovie.mov">link anchor</A>
```

specifies a link to a QuickTime movie. Some common file types and their extensions are:

File Type Extension

Plain text .txt

HTML document .html

GIF image .gif

TIFF image .tiff

XBM bitmap image .xbm

JPEG image .jpg or .jpeg

PostScript file .ps

AIFF sound .aiff

AU sound .au

QuickTime movie .mov

MPEG movie .mpeg or .mpg

Make sure your intended audience has the necessary viewers. Most UNIX workstations, for instance, cannot view QuickTime movies.

Troubleshooting

Avoid Overlapping Tags

Consider this snippet of HTML:

```
<B>This is an example of <DFN>overlapping</B> HTML tags.</DFN>
```

The word "overlapping" is contained within both the <B> and <DFN> tags. How does the browser format it? You won't know until you look, and different browsers will likely react differently. In general, avoid overlapping tags.

Embed Anchors and Character Tags, But Nothing Else

It is acceptable to embed anchors within another HTML element:

```
<H1><A HREF = "Destination.html">My heading</A></H1>
```

Do not embed a heading or another HTML element within an anchor:

```
<A HREF = "Destination.html">
<H1>My heading</H1>
</A>
```

Although most browsers currently handle this example, it is forbidden by the official HTML and HTML+ specifications, and will not work with future browsers.

Character tags modify the appearance of other tags:

```
<UL><LI><B>A bold list item</B>
    <UL>
    <LI><I>An italic list item</I>
</UL>
```

However, avoid embedding other types of HTML element tags. For example, it is tempting to embed a heading within a list, in order to make the font size larger:

```
<UL><LI><H1>A large heading</H1>
    <UL>
    <LI><H2>Something slightly smaller</H2>
</UL>
```

Although some browsers, such as NCSA Mosaic for the X Window System, format this construct quite nicely, it is unpredictable (because it is undefined) what other browsers will do. For compatibility with all browsers, avoid these kinds of constructs.

What's the difference between embedding a <B> within a <LI> tag as opposed to embedding a <H1> within a <LI>? This is again a question of SGML. The semantic meaning of <H1> is that it's the main heading of a document and that it should be followed by the content of the document.Thus it doesn't make sense to find a <H1> within a list.

Character formatting tags also are generally not additive. You might expect that

```
<B><I>some text</I></B>
```

would produce bold-italic text. On some browsers it does; other browsers interpret only the innermost tag (here, the italics).

Check Your Links

When an <IMG> tag points at an image that does not exist, a dummy image is substituted. When this happens, make sure that the referenced image does in fact exist, that the hyperlink has the correct information in the URL, and that the file permission is set appropriately (world-readable).

A Longer Example

Here is a longer example of an HTML document:

```
<HEAD>
<TITLE>A Longer Example</TITLE>
</HEAD>
<BODY>
<H1>A Longer Example</H1>
This is a simple HTML document. This is the first
paragraph. <P>
This is the second paragraph, which shows special effects.  This is a
word in <I>italics</I>.  This is a word in <B>bold</B>.
Here is an in-lined GIF image: <IMG SRC = "myimage.gif">.
<P>
This is the third paragraph, which demonstrates links.  Here is
a hypertext link from the word <A HREF = "subdir/myfile.html">foo</A>
to a document called "subdir/myfile.html". (If you
try to follow this link, you will get an error screen.) <P>
<H2>A second-level header</H2>
Here is a section of text that should display as a
fixed-width font: <P>
<PRE>
    On the stiff twig up there
    Hunches a wet black rook
    Arranging and rearranging its feathers in the rain ...
</PRE>
This is a unordered list with two items: <P>
<UL>
<LI> cranberries
<LI> blueberries
</UL>
This is the end of my example document. <P>
<ADDRESS>Me (me@mycomputer.univ.edu)</ADDRESS>
</BODY>
```

Click here to see the formatted version.

In addition to tags already discussed, this example also uses the <HEAD> ... </HEAD> and <BODY> ... </BODY> tags, which separate the document into introductory information about the document and the main text of the document. These tags don't change the appearance of the formatted document at all, but are useful for several purposes (for example, NCSA Mosaic for Macintosh 2.0, for example, allows you to browse just the header portion of document before deciding whether to download the rest), and it is recommended that you use these tags.

For More Information

This guide is only an introduction to HTML and not a comprehensive reference. Below are additional sources of information.

Fill-out Forms

One major feature not discussed here is fill-out forms, which allows users to return information to the World Wide Web server. For information on fill-out forms, look at this Fill-out Forms Overview

Style Guides

The following offer advice on how to write "good" HTML:

```
Composing Good HTML
CERN's style guide for online hypert
```

Other Introductory Documents

These cover similar information as this guide:

```
How to Write HTML Files
Introduction to HTML
```

Additional References

```
The HTML Quick Reference Guide, which provides a comprehensive listing of
HTML codes
The official HTML specification
A description of SGML, the Standard Generalized Markup Language
Dan Connolly's HTML Design Notebook. Dan Connolly is one of the
originators of HTML.
```

National Center for Supercomputing Applications / pubs@ncsa.uiuc.edu

Chapter 10

CGI and URL Guide

(The entire contents is from the ftp server.)

10.1 The Common Gateway Interface Overview

You can obatin the http connection with

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

first.

10.1.1 CGI

The Common Gateway Interface, or CGI, is a standard for external gateway programs to interface with information servers such as HTTP servers. The Common Gateway Interface, or CGI, is an interface for running external programs, or gateways, under an information server. Currently, the supported information servers are HTTP servers.

The current version is CGI/1.1. The differences between 1.1 and 1.0 are minor, see the list.

10.1.2 What's a gateway used for?

What we refer to as gateways are really programs which handle information requests and return the appropriate document or generate a document on the fly. With CGI, your server can serve information which is not in a form readable by the client (such as an SQL database), and act as a gateway between the two to produce something which clients can use.

Gateways can be used for a variety of purposes, the most common being the handling of ISINDEX and FORM requests for HTTP.

Gateway programs, or scripts, are executable programs which can be run by themselves (but you wouldn't want to). They have been made external programs in order to allow them to run under various (possibly very different) information servers interchangably.

10.1.3 What language can I write these gateways in?

Gateways conforming to this specification can be written in any language which produces an executable file. Some of the more popular languages to use include:

```
C/C++
PERL
TCL
The Bourne Shell
The C Shell
```

There are many others.

10.1.4 Documentation on the Interface Itself

You should first read this introduction to CGI to find out what it is, how it affects you, what you can use it for, and where it came from.

If you are a veteran of NCSA httpd or some versions of the CERN httpd, and remember the /htbin stuff, you should read this guide to upgrading your scripts to CGI compliance. It's not as painful as you might think.

Once you have a basic idea of what CGI is and what you can use it for, you should read this primer which will help you get started writing your own gateways.

If you are interested in handling the output of HTML forms with your CGI program, you will want to read this guide to handling forms with CGI programs.

Security is a crucial issue when writing CGI programs. Please read these tips on how to write CGI programs which do not allow malicious users to abuse them.

When you get more advanced, you should read the interface specification which will help you utilize CGI to the fullest extent. If you are a server software author, it will help you add CGI compliance to your information server.

10.1.5 Some examples of the uses of CGI:

Converting your system's manual pages into HTML on the fly and sending the HTML result to the client. Interfacing with WAIS and archie databases, converting the results to HTML and sending the result to the client. Allowing user feedback about your server through an HTML form and an accompanying CGI decoder.

10.1.6 Examples of CGI behavior and programs

You may wish to look at this page of examples which demonstrate how the client URL affects the interface variables.

We have created an archive of CGI programs on our FTP server. These programs were written by various people around the world in a variety of programming languages. Some of the entries are libraries which may make writing your CGI program easier.

If you would like to submit one of your CGI programs to the archive, you should first package it with any documentation, copyright notices, etc. Then, upload it to ftp.ncsa.uiuc.edu into the directory /incoming/cgi and send mail to httpd@ncsa.uiuc.edu with a short description of what the file is.

10.1.7 Who came up with it?

The specification was discussed between the main HTTP server authors. Credits go to:

Tony Sanders sanders@bsdi.com
Ari Luotonen luotonen@ptsun00.cern.ch
George Phillips phillips@cs.ubc.ca
John Franks john@math.nwu.edu

as well as countless others.
Return to the overview
Rob McCool robm@ncsa.uiuc.edu

10.2 Primer

This section is intended to ease you into the idea of writing your own CGI programs with simplified examples and explanation which is less technically oriented than the interface specification itself.

10.2.1 How do I get information from the server?

Each time a client requests the URL corresponding to your gateway, the server will execute your program. The output of your program will go more or less directly to the client.

A common misconception about CGI is that you can send command-line switches to your program, such as foobar -qa blorf. CGI uses the command line for other purposes and thus this is not directly possible. Instead, CGI uses environment variables to send your program its parameters. The two major environment variables you will use for this purpose are:

QUERY_STRING

QUERY_STRING is defined as anything which follows the first ? in the URL used to access your gateway. This information could be added either by an HTML ISINDEX document, or by an HTML form (with the GET action). It could also be manually embedded in an HTML anchor which references your gateway. This string will usually be an information query, i.e. what the user wants to search for in the archie databases, or perhaps the encoded results of your feedback GET form.

This string is encoded in the standard URL format of changing spaces to +, and encoding special characters with need to decode it in order to use it.

If your gateway is not decoding results from a FORM, you will also get the query string decoded for you onto the command line. This means that each word of the query string will be in a different section of ARGV. For example, the query string "forms rule" would be given to your program with argv[1]="forms" and argv[2]="rule". If you choose to use this, you do not need to do any processing on the data before using it.

PATH_INFO

Much of the time, you will want to send data to your gateways which the client shouldn't muck with. Such information could be the name of the FORM which generated the results they are sending.

CGI allows for extra information to be embedded in the URL for your gateway which can be used to transmit extra context-specific information to the scripts. This information is usually made available as "extra" information after the path of your gateway in the URL. This information is not encoded by the server in any way.

To illustrate this, let's say I have a CGI program on my server called /scripts/foobar. When I access foobar from a particular document, I want to tell foobar that I'm currently in the English language directory, not the Pig Latin directory. In this case, I could access my script in an HTML document as:

```
<A
HREF="/scripts/foobar/language=english">foobar</A>
```

When the server executes foobar, it will give me PATH_INFO of /language=english, and my program can decode this and act accordingly.

10.2.2 How do I send my document back to the client?

I have found that the most common error in beginners' CGI programs is not properly formatting the output so the server can understand it.

CGI programs can return a myriad of document types. They can send back an image to the client, and HTML document, a plaintext document, or perhaps even an audio clip of

your bodily functions. They can also return references to other documents. The client must know what kind of document you're sending it so it can present it accordingly. In order for the client to know this, your CGI program must tell the server what type of document it is returning.

In order to tell the server what kind of document you are sending back, whether it be an document or a reference to one, CGI requires you to place a short header on your output. This header is ASCII text, consisting of lines separated by either linefeeds or carraige returns followed by linefeeds. Your program must output at least two such lines before its data will be sent directly back to the client.

The first line will be different depending on whether your program is returning a full document or a reference to one:

A full document with a corresponding MIME type

In this case, you must know the MIME type of your output. Common MIME types are things such as text/html for HTML, and text/plain for straight ASCII text.

In order to tell the server your output's content type, the first line of your output should read:

Content-type: type/subtype

type/subtype is the MIME type for your output.

A reference to another document

Let's say you want to send a file already available on your information server to the client, or perhaps you want them to retrieve a document from another server altogether.

If you want to reference another file on your own server, you should output a partial URL, such as the following:

Location: /dir1/dir2/myfile.html

In this case, the server will act as if the client had not requested your script, but instead requested http://yourserver/dir1/dir2/myfile.html. It will take care of all access control, determining the file's type, and all sorts of that ugly stuff that servers do.

However, let's say you want to reference a file on your Gopher server. In this case, you should know the full URL of what you want to reference and output something like:

Location: gopher://httprules.foobar.org/0

In this case, the client will interpret your reply, and fetch the URL for the client automatically.

Next, you have to send the second line. With the current specification, THE SECOND LINE SHOULD BE BLANK. This means that it should have nothing on it except a linefeed. Once the server retrieves this line, it knows that you're finished telling the server about your output and will now begin the actual output. If you skip this line, the server will attempt to parse your output trying to find further information about your request and you will

become very unhappy.

Advanced usage: If you would like to output headers such as Expires or Content-encoding, you can if your server is compatible with CGI/1.1. Just output them along with Location or Content-type and they will be sent back to the client.

10.3 Specification

This is the specification for CGI version 1.0, or CGI/1.0. Further revisions of this protocol are guaranteed to be backward compatible.

The server and the CGI script communicate in four major ways. Each of the following is a hotlink to graphic detail.

Environment variables The command line Standard input Standard output

10.3.1 Environment Variables

In order to pass data about the information request from the server to the script, the server uses command line arguments as well as environment variables. These environment variables are set when the server executes the gateway program.

Specification

The following environment variables are not request-specific and are set for all requests:

`SERVER_SOFTWARE`

The name and version of the information server software answering the request (and running the gateway). Format: name/version

`SERVER_NAME`

The server's hostname, DNS alias, or IP address as it would appear in self-referencing URLs.

`GATEWAY_INTERFACE`

The revision of the CGI specification to which this server complies. Format: CGI/revision

The following environment variables are specific to the request being fulfilled by the gateway program:

`SERVER_PROTOCOL`

The name and revision of the information protcol this request came in with. Format: protocol/revision

`SERVER_PORT`

The port number to which the request was sent.

`REQUEST_METHOD`

The method with which the request was made. For HTTP, this is "GET", "HEAD", "POST", etc.

`PATH_INFO`

The extra path information, as given by the client. In other words, scripts can be accessed by their virtual pathname, followed by extra information at the end of this path. The extra information is sent as PATH_INFO. This information should be decoded by the server if it comes from a URL before it is passed to the CGI script.

`PATH_TRANSLATED`

The server provides a translated version of PATH_INFO, which takes the path and does any virtual-to-physical mapping to it.

`SCRIPT_NAME`

A virtual path to the script being executed, used for self-referencing URLs.

`QUERY_STRING`

The information which follows the ? in the URL which referenced this script. This is the query information. It should not be decoded in any fashion. This variable should always be set when there is query information, regardless of command line decoding.

`REMOTE_HOST`

The hostname making the request. If the server does not have this information, it should set REMOTE_ADDR and leave this unset.

`REMOTE_ADDR`

The IP address of the remote host making the request.

`AUTH_TYPE`

If the server supports user authentication, and the script is protects, this is the protocol-specific authentication method used to validate the user.

`REMOTE_USER`

If the server supports user authentication, and the script is protected, this is the username they have authenticated as.

`REMOTE_IDENT`

If the HTTP server supports RFC 931 identification, then this variable will be set to the remote user name retrieved from the server. Usage of this variable should be limited to logging only.

`CONTENT_TYPE`

For queries which have attached information, such as HTTP POST and PUT, this is the content type of the data.

`CONTENT_LENGTH`

The length of the said content as given by the client.

In addition to these, the header lines recieved from the client, if any, are placed into the environment with the prefix HTTP_ followed by the header name. Any - characters in the header name are changed to _ characters. The server may exclude any headers which it has already processed, such as Authorization, Content-type, and Content-length. If necessary, the server may choose to exclude any or all of these headers if including them would exceed any system environment limits.

An example of this is the HTTP_ACCEPT variable which was defined in CGI/1.0. Another example is the header User-Agent.

`HTTP_ACCEPT`

The MIME types which the client will accept, as given by HTTP headers. Other protocols may need to get this information from elsewhere. Each item in this list should be separated by commas as per the HTTP spec.

Format: type/subtype, type/subtype

`HTTP_USER_AGENT`

The browser the client is using to send the request. General format: software/version library/version.

Examples

Examples of the setting of environment variables are really much better demonstrated than explained.

10.3.2 Command Line

CGI Command line options

Specification

The command line is only used in the case of an ISINDEX query. It is not used in the case of an HTML form or any as yet undefined query type. The server should search the query information for a non-encoded = character to determine if the command line is to be used, if it finds one, the command line is not to be used. This trusts the clients to encode the = sign in ISINDEX queries, a practice which was considered safe at the time of the design of this specification.

If the server finds one, it will decode the query information by first splitting it on the pluses given in the URL. It will then perform the additional decoding before placing the resulting words on argv[1....].

If the server finds that it cannot send the string due to internal limitations (such as exec() or /bin/sh command line restrictions) the server should include NO command line information and provide the non-decoded query information in the environment variable QUERY_STRING.

Examples

Examples of the command line usage are much better demonstrated than explained. For these examples, pay close attention to the script output which says what argc and argv are.

10.3.3 Standard Input

CGI Script Input

Specification

For requests which have information attached after the header, such as HTTP POST or PUT, the information will be sent to the script on stdin.

The server will send CONTENT_LENGTH bytes on this file descriptor. Remember that it will give the CONTENT_TYPE of the data as well. The server is in no way obligated to send end-of-file after the script reads CONTENT_LENGTH bytes.

Example

Let's take a form with METHOD="POST" as an example. Let's say the form results are 7 bytes encoded, and look like a=b&b=c.

In this case, the server will set CONTENT_LENGTH to 7 and CONTENT_TYPE to application/x-www-form-urlencoded. The first byte on the script's standard input will be "a", followed by the rest of the encoded string.

10.3.4 Standard Output

CGI Script Output

Script output

The script sends its output to stdout. This output can either be a document generated by the script, or instructions to the server for retrieving the desired output.

Script naming conventions

Normally, scripts produce output which is interpreted and sent back to the client. An advantage of this is that the scripts do not need to send a full HTTP/1.0 header for every request.

Some scripts may want to avoid the extra overhead of the server parsing their output, and talk directly to the client. In order to distinguish these scripts from the other scripts, CGI requires that the script name begins with nph- if a script does not want the server to parse its header. In this case, it is the script's responsibility to return a valid HTTP/1.0 (or HTTP/0.9) response to the client.

Parsed headers

The output of scripts begins with a small header. This header consists of text lines, in the same format as an HTTP header, terminated by a blank line (a line with only a linefeed or

CR/LF).

Any headers which are not server directives are sent directly back to the client. Currently, this specification defines three server directives:

Content-type

This is the MIME type of the document you are returning.

Location

This is used to specify to the server that you are returning a reference to a document rather than an actual document.

If the argument to this is a URL, the server will issue a redirect to the client.

If the argument to this is a virtual path, the server will retrieve the document specified as if the client had requested that document originally. ? directives will work in here, but # directives must be redirected back to the client.

Status

This is used to give the server an HTTP/1.0 status line to send to the client. The format is nnn xxxxx, where nnn is the 3-digit status code, and xxxxx is the reason string, such as "Forbidden".

Examples

Let's say I have a fromgratz to HTML convertor. When my convertor is finished with its work, it will output the following on stdout (note that the lines beginning and ending with — are just for illustration and would not be output):

```
--- start of output ---
Content-type: text/html

--- end of output ---
```

Note the blank line after Content-type.

Now, let's say I have a script which, in certain instances, wants to return the document /path/doc.txt from this server just as if the user had actually requested http://server:port/path/doc.txt to begin with. In this case, the script would output:

```
--- start of output ---
Location: /path/doc.txt

--- end of output ---
```

The server would then perform the request and send it to the client.

Let's say that I have a script which wants to reference our gopher server. In this case, if the script wanted to refer the user to gopher://gopher.ncsa.uiuc.edu/, it would output:

```
--- start of output ---
Location: gopher://gopher.ncsa.uiuc.edu/

--- end of output ---
```

Finally, I have a script which wants to talk to the client directly. In this case, if the script is referenced with SERVER_PROTOCOL of HTTP/1.0, the script would output the following HTTP/1.0 response:

```
--- start of output ---
HTTP/1.0 200 OK
Server: NCSA/1.0a6
Content-type: text/plain

This is a plaintext document generated on the fly just for you.

--- end of output ---
```

10.4 Upgrading

Changing your htbin scripts to CGI scripts

If you have upgraded your server from NCSA httpd 1.0a5 to NCSA httpd 1.0, you have probably discovered that your old scripts don't work in their ScriptAlias locations. This is because NCSA httpd 1.0 implements the CGI specification.

CGI stands for the Common Gateway Interface. It is the next generation of the script interface, called Common because many of the major HTTP servers will be implementing it.

To use your scripts with CGI, you will have to change a few minor semantic things to reflect the fact that CGI passes information in different locations than htbin did.

Method

Whether your script is a GET script or a POST script, you should check that your script is being accessed with the proper method. This can be done with the environment variable REQUEST_METHOD. This variable will be either GET, HEAD, or POST depending on which method was used to access your script.

The command line

For GET scripts, argv[1] used to be path information, and argv[2] used to be the query string. This information can now be found in the environment variables PATH_INFO and QUERY_STRING respectively.

In addition, for non-form GET requests, the query string will be decoded and placed on the command line. If you are writing a script to handle forms, you will not get this information on the command line.

For POST scripts, argv[1] used to contain the content length. This has been moved into the environment variable CONTENT_LENGTH.

DocumentRoot and ServerRoot

Because the environment variables DOCUMENT_ROOT and SERVER_ROOT may have no equivalent in server packages besides NCSA httpd, these have been removed from the CGI specification. You should not have need of this data unless you are looking for the unescape support program.

The Location: output header

If you used the Location: output header for scripts, you'll note that URL specifications still work, but file paths are no longer absolute filesystem paths. They are now virtual paths. This means the server will perform Alias and DocumentRoot translation on this path before retrieving the document.

There are many more features in the specification which you will want to take advantage of. The amount of information now given to the script is much greater than it was.

Rob McCool, robm@ncsa.uiuc.edu

10.5 A Beginner's Guide to URLs

What's a URL? A URL is a Uniform Resource Locator. Think of it as a networked extension of the standard filename concept: not only can you point to a file in a directory, but that file and that directory can exist on any machine on the network, can be served via any of several different methods, and might not even be something as simple as a file: URLs can also point to queries, documents stored deep within databases, the results of a finger or archie command, or whatever.

Since the URL concept is really pretty simple ("if it's out there, we can point at it"), this beginner's guide is just a quick walk through some of the more common URL types and should allow you to be creating and understanding URLs in a variety of contexts very quickly.

File URLs

Suppose there is a document called "foobar.txt"; it sits on an anonymous ftp server called "ftp.yoyodyne.com" in directory "/pub/files". The URL for this file is then:

```
file://ftp.yoyodyne.com/pub/files/foobar.txt
```

The toplevel directory of this FTP server is simply:

```
file://ftp.yoyodyne.com/
```

The "pub" directory of this FTP server is then:

```
file://ftp.yoyodyne.com/pub
```

That's all there is to it.

Gopher URLs

Gopher URLs are a little more complicated than file URLs, since Gopher servers are a little tricker to deal with than FTP servers. To visit a particular gopher server (say, the gopher server on gopher.yoyodyne.com), use this URL:

```
gopher://gopher.yoyodyne.com/
```

Some gopher servers may reside on unusual network ports on their host machines. (The default gopher port number is 70.) If you know that the gopher server on the machine "gopher.banzai.edu" is on port 1234 instead of port 70, then the corresponding URL would be:

```
gopher://gopher.banzai.edu:1234/
```

News URLs

To point to a Usenet newsgroup (say, "rec.gardening"), the URL is simply:

```
news:rec.gardening
```

Currently, network clients like NCSA Mosaic don't allow you to specify a news server like you would normally expect (e.g., news://news.yoyodyne.com/rec.gardening); this may be coming down the road but in the meantime you will have to specify your local news server via some other method. The most common method is to set the environment variable NNTPSERVER to the name of your news server before you start Mosaic.

HTTP URLs

HTTP stands for HyperText Transport Protocol. HTTP servers are commonly used for serving hypertext documents, as HTTP is an extremely low-overhead protocol that capitalizes on the fact that navigation information can be embedded in such documents directly and thus the protocol itself doesn't have to support full navigation features like the FTP and Gopher protocols do.

A file called "foobar.html" on HTTP server "www.yoyodyne.com" in directory "/pub/files" corresponds to this URL:

```
http://www.yoyodyne.com/pub/files/foobar.html
```

The default HTTP network port is 80; if a HTTP server resides on a different network port (say, port 1234 on www.yoyodyne.com), then the URL becomes:

```
http://www.yoyodyne.com:1234/pub/files/foobar.html
```

Partial URLs

Once you are viewing a document located somewhere on the network (say, the document http://www.yoyodyne.com/pub/afile.html), you can use a partial, or relative, URL to point to another file in the same directory, on the same machine, being served by the same server software. For example, if another file exists in that same directory called "anotherfile.html", then anotherfile.html is a valid partial URL at that point.

This provides an easy way to build sets of hypertext documents. If a set of hypertext documents are sitting in a common directory, they can refer to one another (i.e., be hyperlinked) by just their filenames – however a reader got to one of the documents, a jump can be made to any other document in the same directory by merely using the other document's filename as the partial URL at that point. The additional information (access method, hostname, port number, directory name, etc.) will be assumed based on the URL used to reach the first document.

Other URLs

Many other URLs are possible, but we've covered the most common ones you might have to construct by hand. At the top of each Mosaic document viewing window is a text field called "Document URL"; if you watch the contents of that as you navigate through information on the network, you'll get to observe how URLs are put together for many different types of information.

The current IETF URL spec is here; more information on URLs can be found here.

mosaic@ncsa.uiuc.edu