Tim Berners-Lee, CERN

The WWW Book

An attempt to describe most aspects of W3

1. World-Wide Web

Documentation on the World Wide Web is normally picked up by browsing with a W3 browser. If you need a printed copy, it is also available in "laTeX" and "Postscript" formats by anonymous FTP from node info.cern.ch.

This document introduces the "World Wide Web Book", a paper document derived from the hypertext about the project. The book contains

- General informaion about the project, people and history;
- A list of things to be done, including how YOU can put data onto the web;
- A technical discussion of the design issues in projects such as WWW;
- Actual details of the implementation of the WWW project
- Such low-level details as software architecures and coding standards

The text of the book has been automatically generated from the hypertext, so it may seem strange in places due to links in the hypertext which are not there in the printed copy.

The authors of the material are general members of the W3 team at CERN, except where otherwise noted.

1.1 WorldWideWeb - Summary

The WWW project merges the techniques of information retrieval and hypertext to make an easy but powerful global information system.

The project is based on the philosophy that much academic information should be easily available anywhere. It aims to allow information sharing within internationally dispersed teams, and the dissemination of information by support groups. Originally aimed at the High Energy Physics community, it has spread to other areas and attracted much interest in user support, resource discovery and collaborative work areas. It is currently the most advanced information system deployed on the Internet.

Clients and server for many platforms exist and are undercontinual development. Much more information about all aspects of the web is available online –– so skip to "Getting started" if you have an internet connection.

Reader view

The WWW world consists of documents, and links. Indexes are special documents which, rather than being read, may be searched. The result of such a search is another ("virtual") document containing links to the documents found. A simple protocol (" HTTP ") is used to allow a browser program to request a keyword search by a remote information server.

The web contains documents in many formats. Those documents which are hypertext, (real or virtual) contain links to other documents, or places within documents. All documents, whether real, virtual or indexes, look similar to the reader and are contained within the same addressing scheme.

To follow a link, a reader clicks with a mouse (or types in a number if he or she has no mouse). To search and index, a reader gives keywords (or other search criteria). These are the only operations necessary to access the entire world of data.

Information provider view

The WWW browsers can access many existing data systems via existing protocols (FTP, NNTP) or via HTTP and a gateway. In this way, the critical mass of data is quickly exceeded, and the increasing use of the system by readers and information suppliers encourage each other.

Providing information is as simple as running the W3 server and pointing it at an existing directory structure. The server automatically generates the a hypertext view of your files to guide the user around.

To personalize it, you can write a few SGML hypertext files to give an even more friendly view. Also, any file available by anonymous FTP, or any internet newsgroup can be immediately linked into the web. The very small start-up effort is designed to allow small contributions. At the other end of the scale, large information providers may provide an HTTP server with full text or keyword indexing. This may allow access to a large existing database without changing the way that database is managed. Such gateways have already been made into Oracle(tm), WAIS, and Digital's VMS/Help systems, to name but a few.

The WWW model gets over the frustrating incompatibilities of data format between suppliers and reader by allowing negotiation of format between a smart browser and a smart server. This should provide a basis for extension into multimedia, and allow those who share application standards to make full use of them across the web.

This summary does not describe the many exciting possibilities opened up by the WWW project, such as efficient document caching. the reduction of redundant out-of-date copies, and the use of knowledge daemons. There is more information in the online project documentation, including some background on hypertext and many technical notes.

Getting Started

You can try the simple line mode browser by telnetting to info.cern.ch (no user or password. From UK JANET, use the Gateway). You can try a full screen interface "Lynx" by telnetting to ukanaix.cc.ukans.edu, login in as "www". You can also find out more about WWW in this way. These are the least sophisticated browsers –– remember that the window-oriented ones are much smarter! It is much more efficient to install a browser on your own machine, and you have many more facilities.

If you have an X-windows based workstation, just FTP to FTP.NCSA.UIUC.EDU and get the binary of NCSA's "Mosaic" browser in directory /Web/xmosaic. Download it, uncompress it, set it executable, and run it. It will tell you all you need to know.

If you have an MSDOS machine with Windows, get the "Cello" browser from FATTY.LAW.CORNELL.EDU.

If you have a Macintosh, pick up the MacWWW browser from info.cern.ch, in /pub/www/bin/mac.

The line mode browser is currently available in source form by anonymous FTP from node info.cern.ch [currently 128.141.201.74]if you take both files

```
/pub/www/src/WWWLibrary_v.vv.tar.Z.
/pub/www/src/WWWLineMode_v.vv.tar.Z.
```

(v.vv is the version number - take the latest.)

Also available is a hypertext editor for the NeXT (in /pub/www/bin/next), and many other browers, servers, and W3-related tools. You must read the online documentation, including the latest list of software available , to be up to date.

Printable (postscript) documentation and articles are in /pub/www/doc. Tim BL

1.2 WWW people

This is a list of some of those who have contributed to the WWW project , and whose work is linked into this web. Unless otherwise stated they are at CERN, Phone +41(22)767 plus the extension given below or look them up in the phone book . Address: 1211 Geneva 23, Switzerland. See also: Wizards at SLAC .

1.2.1 Marc Andreesen

National Center for Supercomputing Applications (NCSA), Urbana Champagne, IL, USA. Design lead and co-developper of XMosaic . <marca@ncsa.uiuc.edu>. (more)

1.2.2 Eelco van Asperen

Ported the line-mode browser the PC under PC-NFS; developed a curses version. Email: evas@cs.few.eur.nl.

1.2.3 Carl Barker

Carl was at CERN for a six month period during his degree course at Brunel University, UK. Carl worked on the server side, on client authentication and multiple format handling.

1.2.4 Eric Bina

<img src="People/eric_bina.gif">Worked on NCSA Mosaic and the HTMLWidget. (more)

1.2.5 Tim Berners-Lee

<img src="People/tim.gif">Currently in CN division. Before coming to CERN, Tim worked on, among other things, document production and text processing. He developed his first hypertext system, "Enquire", in 1980 for his own use (although unaware of the existence of the term HyperText). With a background in text processing, real-time software and communications, Tim decided that high energy physics needed a networked hypertext system and CERN was an ideal site for the development of wide-area hypertext ideas. Tim started the WorldWideWeb project at CERN in 1989. He wrote the application on the NeXT along with most of the communications software. Phone: 3755, Email: timbl@info.cern.ch. See bio , disclaimer

1.2.6 Thomas R Bruce

Formerly a staff member in charge of computer operations at the Cornell Law School, Tom is now a research associate working on a variety of projects involving the dissemination of legal information on the Internet. He is the author Cello , an all-singing, all-dancing WWW browser for Microsoft Windows. E-mail:tom@law.mail.cornell.edu.

1.2.7 Robert Cailliau

Formerly in programming language design and compiler construction, Robert has been interested in document production since 1975, when he designed and implemented a widely used document markup and formatting system. He ran CERN's Office Computing Systems group from 87 to 89. He is a long-time user of Hypercard, which he used to such diverse ends as writing trip reports, games, bookkeeping software, and budget preparation forms. Robert is mainly supporting physics experiments with WWW. The rest of the time, when he is not doing WWW's public relations, he is contributing browser software for the Macintosh platform. Phone: +41 (22) 767 50 05, Email: cailliau@www1.cern.ch. Be aware of his diary .

1.2.8 Dan Connolly

An early follower of the project, Dan wrote a private X-Windows editor for his company, and encouraged the use of proper SGML and MIME in the future. He wrote a DTD for HTML and an HTML legalizer for old files. The "SGML cop" himself, Dan has put a lot of work in on the HTML specs. Dan is now at Atrium Technologies and cannot give support for his work with W3. Email: connolly@atrium.com.

1.2.9 Peter Dobberstein

While at the DESY lab in Hamburg (DE), Peter did the port of the line-mode browser onto MVS and, indirectly, VM/CMS. These were the most difficult of the ports to date. He also overcame many incidental problems in making a large amount of information in the DESY database available.

1.2.10 "Erwise" team

Kim Nyberg, Teemu Rantanen, Kati Suominen and Kari Syd{nmaanlakka ('{' is 'a' with two dots above it.. we must get some character set description into HTML!) (under the supervision of Ari Lemmke) are "Erwise". At Helsinki Technical University, they are writing a Motif-based WWW browser (editor? we can hope...) for their undergraduate final year project. The team can be reached as erwise@cs.hut.fi and Ari as arl@cs.hut.fi.

1.2.11 Alain Favre

Alain is an undergraduate working with ECP/PT on a browser for Windows on PCs. Phone: 8265, no email yet. In CERN mostly in the afternoons.

1.2.12 David Foster

With wide experience in networking, and a current conviction information systems and PC/Windows being the way of the future, Dave is having a go at a MS-Windows browser/editor. Dave also has a strong interest in server technology and intelligent information retrieval algorithms.

1.2.13 Jean-Francois Groff

During his stay at CERN as "cooperant", J-F joined the project in September 1991. He wrote the gateway to the VMS Help system , worked on a new modular browser architecure, and helped support and present WWW at all levels. He later as consultant ported the communications code to DECnet in order to set up servers for physics experiments., and helped the Danish Technical Library set up their W3 server. JF also worked for NeXT Europe. He now is a consultant in networked information systems (Contact) jfg@infodesign.ch

1.2.14 Tony Johnson

<img src="People/tony_johnson.gif">Tel: (415) 926 2278, TONYJ@scs.slac.stanford.edu.

Designer of MidasWWW . Boston University, collaborating with SLAC, SSC, etc. A SLAC server expert and a WWWizard .

1.2.15 Paul Kunz

Paul took the W3 word across to SLAC, installed the clients and inspired the setting up of servers by the WWWizards . Paul spreads enthusiasm for all sort of good ideas such as OO programming, NeXTs, etc...

1.2.16 Willem van Leeuwen

at NIKHEF, WIllem put up many servers and has provided much useful feedback about the w3 browser code.

1.2.17 Ari Luotonen

A Technical Student at CERN, in the project from July 1993 to June 1994. He will complete his MSc degree at Tampere University of Technology, Finland, in May 1995. Ari is now working in the Programming Techniques Group of ECP on the WWW Access Authorization (e-mail: luotonen@dxcern.cern.ch).

1.2.18 Jon Mittelhauser

<img src="People/jon_mittelhauser.gif">Works on NCSA Mosaic for MS Windows. (more)

1.2.19 Lou Montulli

Before WWW: <img src="People/lou_montulli2.gif"> After WWW: <img src="People/lou_montulli.gif"> montulli@ukanaix.cc.ukans.edu Lou is the author of "Lynx", a curses based hypertext browser, and Lynx 2.0 which is a WWW browser. He is a student/employee of the University of Kansas and is actively spreading the WWW word to whoever will listen. Picture .

1.2.20 Nicola Pellow

With the project from November 1990 to August 1991, and October 1992 to ??. A graduate of Leicester Polytechnic, UK, Nicola wrote the original line mode browser . (More). Nicola is now (Oct 92) working on the Mac browser .

1.2.21 Bernd Pollermann

Bernd is responsible for the "XFIND" indexes on the CERNVM node, for their operation and, largely, their contents. He is also the editor of the Computer Newsletter (CNL), and has experience in managing large databases of information. Bernd is in the AS group of CN division. He has contributed code for the FIND server which allows hypertext access to this large store of information. Phone: 2407, Office: 513-1-16, Email: bernd@cernvm.cern.ch

1.2.22 Steve Putz

<img src="People/steve_putz.gif">Created custom gateway servers to other information sources. (more)

1.2.23 Dave Ragget

dsr@hplb.hpl.hp.com Dave is the editor of the HTML+ document type, currently (July 93) in discussion for more sophisticated documents than HTML can handle. Dave has written his own WWW browser, and is working on authentication and opensubnet gateways. And that is all just what he does in his spare time!

1.2.24 Tony Sanders

<img src="People/tony_sanders.gif">Member of Technical Staff at Berkeley Software Design, Inc. currently doing software development, customer support and maintaining the BSDI WWW server (with ambitions of online manuals and technical support via the Web). Developed the Plexus HTTP server (based on the server from cs.indiana.edu), there are some demos available online. Network connectivity is via a 56K link to Alternet in Austin, Texas. Can be reached by Phone: 1-512-251-1937 or Email: . See hyplan .

1.2.25 Arthur Secret

A student at CERN during August and September 1992, Arthur wrote the first W3-Oracle gateway . Arthur is back at CERN from 1 March to 31 October 1993, working on graphics formats, scanning and conversion techniques, the www code library, and answering user requests.

1.2.26 Chuck Shotton

Assistant Director, Academic Computing, U. of Texas Health Science Center Houston <cshotton@oac.hsc.uth.tmc.edu> (713) 794-5650. Author of the MacHTTP , the W3 server for the Macintosh.

1.2.27 Jonathan Streets

Online Support group, FNAL. Jonathan put up a VMS server using DCL and later C. He helped debug the Mac browser.

1.2.28 Nathan Torkington

"Gnat" has put up all kinds of useful thing son the web, and contributed such things as an HTML to TeX converter. (More)

1.2.29 Aleksander Totic

<img src="People/aleksander_totic.gif">Develops Mac Mosaic. (more)

1.2.30 Pei Wei

<img src="People/pei_wei.gif">Pei is the author of " Viola", a hypertext browser, and the ViolaWWW variant which is a WWW browser. He was at the University of California at Berkeley, Experimental Computing Facility, now full time with O'Reilly and Associates, Sebastopol, CA, USA. Email: wei@xcf.berkeley.edu

1.2.31 Bebo White

one of the WWWizards at SLAC, Bebo enthusiastically spreads the word. During a short stay at CERN in summer '92, Bebo put up a number of servers for information from the Aleph experiment.

1.2.32 James Whitescarver

New Jersey Institute of Technology. Author of the curses based W3 client, and of a number of server tools. Email: jim@eies2.njit.edu

1.2.33 Chris Wilson

<img src="People/chris_wilson.gif">Chris works on NCSA Mosaic and Windows NT Mosaic. (more)

1.3 Policy

This outlines the policy of the W3 project at CERN. Whilst not legally binding, this attempts to explain my understanding of the CERN rules and the desires of the team at CERN.

1.3.1 Aim

The basic aim of the project is to promote communication and information availability for the High Energy Physics (HEP) community. The project is based at CERN, whose budget is provided by contributions of taxpayer's money from the European member states. It is in the interests of HEP, CERN, and the project itself that it should interwork with systems and information in many other fields, and so active collaboration with other groups is essential. To produce an information system isolating HEP from the rest of the world would be counter-productive, so the aim can be seen as furthering a global web of information.

The WWW team are all enthusiastic that information of all types should be available as widely as possible.

1.3.2 Collaboration

We encourage collaboration by academic or commercial parties. There are always many things to be done, ports to be made to different environments, new browsers to be written, and additional data to be incorporated into the "web". There have already been many contributions in these terms, and also with hardware support from manufacturers.

If you may be interested in extending the web or the software, please mail or phone us.

1.3.3 Code distribution

Code written at CERN is covered by the CERN copyright except where explicitly put into the Public Domain. The basic WWW code is in the Public Domain, the rest is covered by the conditions of distribution . In practice the interpretation of this in the case of the W3 project is that the programs are freely available to academic bodies of CERN member states and to the world-wide High-Energy Physics community. To commercial organizations who are not reselling it, but are using it to participate in global information exchange, the charge is generally waived in order to cut administrative costs. Code is of course shared freely with all collaborators. Commercial organizations wishing to sell software based on W3 code should contact CERN.

We are in the process of getting agreement to release certain parts of the WWW project code into the Public Domain.

Where CERN code is included in otherwise public domain code, that CERN code becomes also public domain.

Code not originating at CERN is of course covered by terms set by the copyright holder involved.

1.3.4 Protocols and Data Formats

The definition of protocols such as HTTP and data formats such as HTML are in the public domain and may be freely used by anyone. Tim BL

1.3.5 WorldWideWeb CERN-distributed code

See the CERN copyright . This is the README file which you get when you unwrap one of our tar files. These files contain information about hypertext, hypertext systems, and the WorldWideWeb project. If you have taken this with a .tar file, you will have only a subset of the files.

THIS FILE IS A VERY ABRIDGED VERSION OF THE INFORMATION AVAILABLE ON THE WEB. IF IN DOUBT, READ THE WEB DIRECTLY. If you have not got ANY browser installed yet, do this by telnet to info.cern.ch (no username or password).

Files from info.cern.ch are also mirrored on ftp.ripe.net.

Archive Directory structure

Under /pub/www , besides this README file, you'll find bin , src and doc directories. The main archives are as follows:

bin/xxx/bbbb Executable binaries of program bbbb for system xxx. Check what's there before you bother compiling. (Note HP700/8800 series is "snake")

bin/next/WorldWideWeb_v.vv.tar.Z The Hypertext Browser/editor for the NeXT –– binary.

src/WWWLibrary_v.vv.tar.Z The W3 Library. All source, and Makefiles for selected systems.

src/WWWLineMode_v.vv.tar.Z The Line mode browser - all source, and Makefiles for selected systems. Requires the Library .

src/WWWDaemon_v.vv.tar.Z The HTTP daemon, and WWW-WAIS gateway programs. Source. Requires the Library.

src/WWWMailRobot_v.vv.tar.Z The Mail Robot.

doc/WWWBook.tar.Z A snapshot of our internal documentation - we prefer you to access this on line –– see warnings below.

Basic WWW software installation from source

This applies to the line mode client and the server. Below, $prod means LineMode or Daemon depending on which you are building.

Generated Directory structure The tar files are all designed to be unwrapped in the same (this) directory. They create different parts of a common directory tree under that directory. There may be some duplication. They also generate a few files in this directory: README.*, Copyright.*, and some installation instructions (.txt).

The directory structure is, for product $prod and machine $WWW_MACH

WWW/$prod/Implementation Source files for a given product

WWW/$prod/Implementation/CommonMakefile The machine-independent parts of the Makefile for this product

WWW/$prod/$WWW_MACH/ Area for compiling for a given system

WWW/All/$WWW_MACH/Makefile.include The machine-dependent parts of the makefile for any product

WWW/All/Implementation/Makefile.product A makefile which includes both parts above and so can be used from any product, any machine.

Compilation on already supported platforms You must get the WWWLibrary tar file as well as the products you want and unwrap them all from the same directory.

You must define the environmant variable WWW_MACH to be the architecure of your machine (sun4, decstation, rs6000, sgi, snake, etc)

In directory WWW, type BUILD.

Compilation on new platforms If your machine is not on the list:

- Make up a new subdirectory of that name under WWW/$prod and WWW/All, copying the contents of a basically similar architecture's directory.
- Check the WWW/All/$WWW_MACH/Makefile.include for suitable directory and flag definitions.
- Check the file tcp.h for the system-specific include file coordinates, etc.
- Send any changes you have to make back to www-request@info.cern.ch for inclusion into future releases.
- Once you have this set up, type BUILD.

NeXTStep Browser/Editor

The browser for the NeXT is those files contained in the application directory WWW/Next/Implementation/WorldWideWeb.app and is compiled. When you install the app, you may want to configure the default page, WorldWideWeb.app/default.html. These must point to some useful information! You should keep it up to date with pointers to info on your site and elsewhere. If you use the CERN home page note there is a link at the bottom to the master copy on our server. You should set up the address of your local news server with

```
dwrite WorldWideWeb NewsHost  news
```

replacing the last word with the actual address of your news host. See Installation instructions .

Line Mode browser

Binaries of this for some systems are available in /pub/www/bin/ . The binaries can be picked up, set executable, and run immediately.

If there is no binary, see "Installation from source" above.

(See Installation notes). Do the same thing (in the same directory) to the WWWLibrary_v.cc.tar.Z file to get the common library.

You will have an ASCII printable manual in the file WWW/LineMode/Defaults/line-mode-guide.txt which you can print out at this stage. This is a frozen copy of some of the online documentation.

Whe you install the browser, you may configure a default page. This is /usr/local/lib/WWW/default.html for the line mode browser. This must point to some useful information! You should keep it up to date with pointers to info on your site and elsewhere. If you use the CERN home page note there is a link at the bottom to the master copy on our server.

Some basic documentation on the browser is delivered with the home page in the directory WWW/LineMode/Defaults. A separate tar file of that directory (WWWLineModeDefaults.tar.Z) is available if you just want to update that.

The rest of the documentation is in hypertext, and so wil be readable most easily with a browser. We suggest that after installing the browser, you browse through the basic documentation so that you are aware of the options and customisation possibilities for example.

Server

The server can be run very simply under the internet daemon, to export a file directory tree as a browsable hypertext tree. Binaries are avilable for some platofrms, otherwise follow instructions above for compiling and then go on to " Installing the basic W3 server ".

XMosaic

XMosaic is an X11/Motif W3 browser.

The sources and binaries are distributed separately from FTP.NCSA.UIUC.EDU , in /Web/xmosaic . Binaries are available for some platforms. If you have to build from source, check the README in the distribution.

The binaries can be picked up, uncompressed, set "executable" and run immediately.

Viola browser for X11

Viola is an X11 application for reading global hypertext. If a binary is available from your machine, in /pub/www/bin/.../viola*, then take that and also the Viola "apps" tar file which contains the scripts you will need.

To generate this from source, you will need both the W3 library and the Viola source files. There is an Imakefile with the viola source directory. You will need to generate the XPA and XPM libraries and the W3 library befere you make viola itself.

Documentation

In the /pub/www/doc directory are a number articles, preprints and guides on the web.

See the online WWW bibliography for a list of these and other articles, books, etc. and also the list of WWW Manuals available in text and postscript form.

General

Your comments will of course be most appreciated, on code, or information on the web which is out of date or misleading. If you write your own hypertext and make it available by anonymous ftp or using a server, tell us and we'll put some pointers to it in ours. Thus spreads the web... Tim Berners-Lee

WorldWideWeb project

CERN, 1211 Geneva 23, Switzerland

Tel: +41 22 767 3755; Fax: +41 22 767 7155; email: timbl@info.cern.ch

1.3.6 Copyright CERN 1990-1993

Tim Berners-Lee

CERN

1211 Geneva 23, Switzerland Tel +41(22)767 3755, Fax +41(22)767 7155, Email: tbl@cernvax.cern.ch

1.3.7 World-Wide Web Printed Manuals

These manuals are available by anonymmous FTP from info.cern.ch. They all are generated automatically from the online hypertext, and so may read a little strangely at times . Also, they are of course always out of date compared with the online documents which are continuously being updated. To get them, follow the links or, if you don't yet have www software, your FTP session may look something like:

```
ftp info.cern.ch
> login: anonymous
> password: your@mail.address
> cd /pub/www/doc
> ls
> get line-mode-guide.txt
> quit
```

Each manual is available in Postscript (filename suffix .ps), plain ASCII text (.txt) and sometimes TeX (.tex). The plain text versions do not have a table of contents.

If you are reading this in hypertext, you can pick up the plain text or postscript by following links below. The titles are linked to the hypertext versions.

Line Mode Browser user guide A non-hypertext version of the line mode browser hypertext user guide . About 13 pages.

See the introductory page for a contents overview.Filename: line-mode-guide.txt (plain text) or line-mode-guide.ps (postscript)

The WWW Server Guide The manual for information providers. Instructions on how to install and configure and if necessary debug the basic W3 server. Examples of simple shell script servers. Lists of tools available to help construct servers and generate hypertext. Descriptions of how to make a server for an existing database. Lists of existing gateways servers.

Filename: www-server-guide.txt (plain text) or ww-server-guide.ps (postscript)

The HTML Specification A complete description of the Hypertext Markup language generated by W3 servers. Includes desription of each tag and its significance, lists of special characters, the SGML DTD for HTML, and an explanation of the relationships between HTML and SGML and MIME. About 28 pages.

Filename: html-spec.txt (plain text), html-spec.ps (postscript)

The HTTP Specification A description of the protocol used by new W3 clients and servers to communicate.

Filename: http-spec.txt (plain text) , http-spec.ps (postscript)

OUT OF PRINT

The "W3 Book" If you want technical details, we recommend browsing the web, for the latest versions of all our thoughts we have had time to type in. If you want to take it on the plane, then we occasionally dump a part of the hypertext onto paper. This is the "World-Wide Web Book". Some of it is mere philosophical wonderins, some of it is proposal, some is explanation.

NO LONGER AVAILABLE IN PAPER FORM.

See the introductory page for contents overview.

Around 75 pages.

2. Frequently Asked Questions on W3

An FAQ list is really a cop-out from managed information. You should be able to find everything you want to know by browsing from the WWW project page, as everything should be arranged in a logical way. Here though are things which maybe didn't fit into the structure, with pointers to the answers which maybe did. Its an experiment, started May 92. The questioners are anonymous.

- I am just starting: how do I find out more?
- How does www keep track of the available servers?
- How does W3 compare with WAIS and Gopher ?
- How do I create my own server ?
- Can I get W3 documents if I'm not on the internet ?
- How can I access WWW though a internet firewall ?

See also Nathan Torkington's FAQ list posted every now and again to comp.infosystems.www , in hypertext form .,WWW Primer Tim BL

2.1 Getting Started

2.1.1 Question

So where can I find information about W3?

2.1.2 Getting Started using telnet

All the information about W3 is on ther web. So you do you get started? A number of ways. You can browse through all that information by just telnetting to one of the addreses below, or you can pick up information using anonymous FTP. If you want to use telnet, try some of the following. (Log in as www if asked for a user name)

telnet info.cern.ch	(or telnet 128.141.201.74) The simplest line mode browser. This server is in Geneva, Switzerland.
telnet ukanaix.cc.ukans.edu	A full screen browser "Lynx" which requires a vt100 terminal. Log in as www. University of Kansas
telnet www.njit.edu	Log in as www. A full-screen browser in New Jersey Institute of Technology. USA.
telnet vms.huji.ac.il	(or telnet 128.139.4.3). A dual-language Hebrew/English database, with links to the rest of the world. The line mode browser, plus extra features. Log in as www. Hebrew Uiversity of Jerusalem, Israel.
telnet sun.uakom.cs	Slovakia. Has a slow link, use from nearby.
telnet fserv.kfki.hu	Hungary. Has slow link, use from nearby. Login is as www.
telnet info.funet.fi	(or telnet 128.214.6.100) (FINLAND)
Cornel Law school	(address?)

2.1.3 Using FTP

Alternatively, you can pick up some information in plain text or postscript form from the anonymous FTP archive on info.cern.ch. Just FTP to info.cern.ch (or 128.141.201.74) and log in as "anonymous" with for password your mail address user@host.

Change directory (cd) to pub/www/doc, and see what there is (ls command).

If you have an X-windows workstation, pick up the binary (preferebly) or else the source of NCSA's Mosaic for X from ftp.ncsa.uiuc.edu, in directory /Web/xmosaic. Just uncompress it, set it executable, and run it.

See also: the W3 bibliography , about W3 distributed code , using FTP . Tim BL

2.2 How does www keep track of the available servers?

2.2.1 Q

How does www keep track of the available servers? How does a user know where to go to get a specific piece of information? According to the description of the http protocol, when a user wants to do a search, the corresponding UDI specifies, among other things, the server's address. How does the user find out about the server's address? Or from the server's perspective, how does a server announce its existence?

2.2.2 The resource discovery problem

14 May 1992 This is what people seem to call this problem in general.

As a physical sever can serve many different types of information from different servers, we talk about finding documents and indexes, as that is what the user sees. To the reader, the web is a continuum. When a new server appears, it may serve many databases of data from different sources and on different subjects. The new data must be incorporated into the web. This means putting links to data on the new server (especially to a general overview document for the server if there is one) from existing documents which interested readers might be reading, or putting it into an index which people might search.

The person publishing the data must go through the same process as the person searching for it. When (s)he has found an overview page which (s)he feels ought to refer to the new data, (s)he can ask the author of that document (who ought to have signed it with a link to his or her mail address) to put in a link. There may be several links from different documents: there is not one master list. Of course, some servers are put up for internal use only, and links are only made from local documents. I only find out about these servers by word of mouth, but they exist.

Currently, there are three parallel trees in the web for finding data starting from scratch. The most interesting one is a classification by subject. I've got an "Other subjects" link from Cern's home page to a master page of information by subject . From that I have links to individual servers of all kinds (W3, WAIS and Gopher), and in cases where there are a lot like physics and biology, a link to a page about one specific subject. In this way you can browse the web by subject like a library. I am looking for people in other disciplines to take over the subtrees for those disciplines as the load gets heavier (I may have candidates for some). The tree tends to be ought of date, and its authors rely on feedback to put in things which are missing.

The other trees are by organization and by server type. The list by server type is easy, because the people responsible for each protocol keep a list of the servers using it. That is, there is a tree of gophers, and there is an index of WAIS indexes. There is the W3/WAIS/Archie server for FTP sites. This tree isn't so useful unless you know what sort of a server you are looking for, but it tends to be more up-to-date than the subject index. It also has things in which aren't just about subjects. The third tree was going to be a geographic tree of organizations, but that isn't at all up-to-date.

By the way, it would be easy in principle for a third party to run over these trees and make indexes of what they find. Its just that noone has done it as far as I know because there isn't yet an indexer which runs over the web directly.

As you can see, the web is sufficiently flexible to allow a number of ways of finding infomation. In the end, I think a typical resource discovery session will involve someone starting on their "home" document, following one to two links to an index, then doing a search, and following several links from what they have found. In some cases, there will be more than one index search involved, such as at first for an organization, and having found that, a search within it for a person or document. We need to keep this flexibility, as the available information in diffferent places has such different characteristics.

In the long term, when there is a really large mass of data out there, with deep interconnections, then there is some really exciting work to be done on automatic algorithms to make multi-level searches. Tim BL

2.3 W3 vs WAIS and Gopher

2.3.1 Question

What's the difference betwen W3 and WAIS? What's the difference between W3 and Gopher? Why invent yet another system? Which one should I use?

2.3.2 The data model

W3 is comparable to both WAIS and Gopher , in that it is a client-server information system running over the internet. There is a difference in the data models. The W3 model is that everything (document, menu, index etc) is represented to the user as a hypertext (hypermedia) object. There two navigation operations are available to the user: to follow a link or to send a query to a server. Only certain documents are flagged as having a search facility, and not all documents have links, but some documents have both. That's a pretty simple model, and results in a pretty simple user interface.

Two neat things fall out of this model. One is that it turns out that almost all other information systems can be represented in terms of W3 documents. A W3 user can interrogate WAIS indexes (example) and Gopher servers (example). This comes from the flexibility of the W3 model to describe other structures. A WAIS database is a searchable document. The hit-list returned by a WAIS server (or any other query engine) is a hypertext document with links to the documents found. Gopher menus (or any other hierarchical menu system, including a file system) are represented as lists of items linked to other objects. The W3 system has an open addressing scheme allowing links to be made to any objects on W3, WAIS, Gopher, FTP, NFS, or Network News servers.

Therefore, the Web is the SUPERSET of the FTP, WAIS, Gopher and HTTP spaces.

This flexibility has allowed lots of different kinds of data to be put on-line by writing a simple script to generate a hypertext "view" of the database.

The hypertext model, then, is flexible. It is also powerful as a communications medium. To author a document in hypertext is to communicate better. It allows one to put in a link whenever the reader might need background information .

2.3.3 WAIS lacks links

You miss the links in WAIS in two ways. One is when you are looking for an index. You can't follow links from an overview page to "browse" through different indexes. You can only use a master index (the directory of sources) to find indexes. The other way is that when you have retrieved something, whether part of the FORTRAN manual or part of a mail discussion, you get it in isolation. You can't follow links from that document to related documents.

2.3.4 Gopher menus lack text

A Gopher menu is a dry list of items. Each line has 80 characters in which to describe an option. In practice, to communicate with the reader, one needs the full power of text formatting in a number of styles. A plain list turns out to be relatively infrequently used when the author or the program generating the document has a choice. Note that the "Panda" project adds some plain text to Gopher menus, but this is only a small step toward the flexible blending of links and text which is hypertext.

2.3.5 Group work

The second big difference is that W3 is designed to include collaborative authoring (CSCW) so that groups can share information, rather than simply individuals disseminate it. We only have a first stab at this on the NeXT platform, as we were overtaken by the web's success in dissemination mode. XMosaic is bringing this further along.

2.3.6 Deployment levels

The W3 software was not (in May 92) as deeply deployed as WAIS and Gopher software. This is basically because it takes more time to write a hypertext client than a menu or query client. (Also, because the initial W3 instigators are paid to work for the world of High-Energy Physics primarily). Updating this in June 93, we see W3's own "http" protocol move ahead of the WAIS protocol in the NSF Backbone packet count statistics. The number of W3 servers is now similar to the number of WAIS servers (around 100) while smaller than the number of Gopher servers (around 1000).

The W3 world is growing very fast. Between May 91 and May 93, load on CERN's W3 server doubled every four months or less. There is widespread recognition that hypertext is essential for the next generation. It is planned to merge the W3 and Gopher systems, and there is no reason (apart from server simplicity and, perhaps, response time ... both strong issues in the market) why both of these systems could not use the WAIS protocol when it settles down. However these distinctions are largely practical details for the web, which in using a number of protocols, allows technology to advance without anyone having to suddenly change everything.

2.3.7 The Choice

Bear in mind:

- A W3 client can read data from any other system.
- If you run a W3 server you can upgrade certain parts of the documentation to hypertext later.
- Hypertext is neat for representing existing data easily. This you can only try for yourself.

So install W3 clients, and W3 servers. If you want to install a Gopher or WAIS server, fine: the W3 clients will access it. If you install a WAIS server, then you could install the W3-WAIS gateway locally to save bandwidth. Tim BL

2.4 How to create a W3 server

2.5 (Warning: page under construction)

2.5.1 Questions

What's the difference between a server and a gateway ? Which one do I need ? What's the easiest way to create a server ? How do I customize the distributed software to my needs ? How can I write an index server ?

2.5.2 Basics

Since this FAQ came up, a lot more documentation about starting servers has come onto the web, so it might pay you to browse a bit more. There are so many ways, that it might seem complicated, but in fact most of the methods are very easy.

The server that we distribute as WWWDaemon is in two parts: a common HTDaemon program taking care of the communications, and a variable HTRetrieve function called by HTDaemon with the significant part of the address (i.e. after host) split as argument and keywords (look at the code). The default HTRetrieve function, located in HTRetrieve.c, in fact runs the www common code library functions (yes, the same fucntions as in a client) to retrieve the information

2.5.3 Index server

An index server allows searches as well as retrieval. To make an index server, you need an HTRetrieve that knows how to query a database, or otherwise find information, from keywords supplied after the '?' in the address. It can be as simple as a 'grep' in a series of files.

We provide 3 example index servers built this way: VMSHelpGate, which gives access to VMS help, FindGate which gives access to a mainframe search engine called XFind, and WAISGate, which speaks the WAIS protocol to contact their search engines. You can read about all this in the Web...

Oh yes, the ISINDEX tag should be returned by the server to notify the client that it accepts searches.

Page under construction. Tim BL

2.6 No internet connection

You are not on the internet? You know you aren't because none of the useual commands are available to you like telnet and ftp. Don't worry: all is not lost. There are two possibilities.

2.6.1 If you are on DECnet

If you are on a DECnet, then you can use DECnet versions of the W3 software. You can get infromation from outside the DECnet you are on so long as somebody somewheer runs a gateway into the internet ona a machine which is connected to both.

2.6.2 By mail

If you have electronic mail, then it possible (though slow!) to get W3 information by mail. try sending a mail to listserv@info.cern.ch with a line in it saying just

HELP

to get back instructions . Your mail system must have a mail gataeway onto internet mail, but hat is quite likely. You might have to take the internet address above and ask your friendly system manager how to convert it into the equivalent mail address on the system you are using. See more about the robot . Tim BL

2.7 Getting Through Fire Walls

2.7.1 Question

My company (organisation, etc) has an internal internet but general access to the external internet is not allowed. How can I run a www client to access offf-site things?

2.7.2 TCP tunneling through firewalls

There are two ways. If you are simply a user with no sympathy from the guys running the gateway, then you have to use an existing gateway. Typically, there will be a telnet gateway to which you can telnet which then allows you to ask politely to be connected to a given remote machine.

It is possible to change way that www clients to a TCP connect to do all of this for every connection. The code in in HTTP.c and HTTCP.c in the libwww library. This has been done successfully at for example Xerox PARC ands other places.

2.7.3 Running a WWW Gateway

You can do it this way if you can run a new gateway on the firewall machine. You will need to persuade your management that this is safe and necessary. (Try showing them a good www client and then see what they say!)

WWW clients can be set to redirect requests to a gateway using

```
setenv WWW_http_GATEWAY  http://gw.here.com/
setenv WWW_wais_GATEWAY  http://gw.here.com/
setenv WWW_gopher_GATEWAY http://gw.here.com/
```

setting it separately for each type of URL. They then use HTTP protocol to go to the gateway, which returns whatever document/search it was they wanted. See also: running a WWW gateway.

The client can in fact run with a rule file instead of the environment variables, which allows more complicated selections to be set up.

The CERN WWW server (httpd) will run as a gateway just by being configured correctly. The rule file needs lines like for example

```
pass http:*
pass wais:*
pass gopher:*
fail news:alt.*
pass news:*
```

The gateway can at the same time be a server for files, by putting in lines like

```
pass http://gw.here.com/* file:/pub/*
pass /* file:/pub/*
```

There are lots of alternatives. Clearly you can be quite specific about what you do and do not want to allow through. But censorship may only get you into trouble with your users. Tim BL

3. How can I help?

There are lots of ways you can help if you are interested in seeing the web grow and be even more useful...

Put up some data There are many ways of doing this. The web needs both raw data —— fresh hypertext or old plain text files, or smart servers giving views of existing databases. See more details , etiquette , style guide.

Suggest someone else does Maybe you know a system or some information which you would like to see on the web. Suggest to the person involved that they put up a W3 server.

Manage a subject area If you know something of what's going on in a particular field, organization or country, would you like to keep up-to-date an overview of online data?

Write some software We have a big list of things to be done. Help yourself —— all contributions gatefully received! see the list .

Send us suggestions We love to get mail... www-bug@info.cern.ch

Tell your friends Install/get installed the client software on your site. Quote things by their W3 address to allow w3 users to pick them straight up.

Tim BL

3.1 W3 developments

Thisis a sort of sign-up sheet for W3-related devlopments.

If you have a moment, take your pick! Some of these would make good projects, or would be a way your organization could contribute to the building of the web. This list is only a few ideas —— more are welcome!

If you are interested in a particular area, mail www-request@info.cern.ch to have your name put against this list, with a statement of whether your interest.

There are also special lists of little things to do for each existing project such as the line mode browser , the NeXT browser , the common library , and the server .

3.1.1 Client side

More clients Clients exist for many platforms, but not all. Editors only exist on the NeXT, but will be really useful for sourcing info and group work. (Group editor?)

Search engines Now the web of data and indexes exists, some really smart intelligent algorithms ("know-bots?") could run on it. Recursive index and link tracing, Just think...

Text from hypertext We need a quick way to print a book from the web. (A simple method using TeX exists)

Slide show Timed sequences of presentations. Scripting language for Multimedia stuff. References: Apple Quicktime, IBM/PC's AVI, HyTime. Make a new MIME content type for such a 4-D montage of other documents. Interest: jcasey@maths.tcd.ie Aug 93.

3.1.2 Server side

Server upgrade Easier to install, port. Run shell scripts embedded in the directory for virtual documents and searches. This is being tackled: improvements to the CERN server, the NCSA server by Rob McCool, the Mac server, and CNIDR are tackling a Windows/NT (MSDOS?) server. Additional feature patches welcomed!

More Servers The list of information we have thought of or been pointed to which could be put into the web.

WAIS integration WAIS protocol extensions to allow hypertext; HTML data type, docids to be conforming UDIs. WAIS has been integrated into the client now, using freeWAIS code. Maybe that code could all be simplified/speeded up? Z39.50 latest version added?

Relational Database Gateway Flexible tools for generating hypertext views of relational databases.

FTP server distribution Persuade the newer FTP server implementations to include a HTTP server in the distributed code, to allow more efficient access

3.1.3 Other software

Transport level Gateways JANET and DECnet for example. Real need.

HTTP enhancements Format conversion (done), authorization (Ari working on it) , better logging information for statistics.

Mail manager A hypertext view of a mail archive with lists of messages by author, topic, with links between messages. A friendly face on a mail archive would be a great project management tool.

HTTP enhancements Format conversion, authorization, better logging information for statistics.

Form processing If you can edit hypertext, you edit a hypertext form and return it. To be able to submit a form back to the server would allow special search patterns, administrative processing, electronic voting, ...

Graphic overview Display the web from any document in a graphical form.

HyperGraphics Why only text? Perhaps a generic format for putting an overlay of sensitive areas over any image (or video) format. See Tony Sander's ISMAP work.

Phone-line protocol There is a need for a point-point low bandwidth protocol designed for beating the heck out of a phone line. The protocol will keep the phone line occpied in a very inteligent way with look-ahead fetches of related documents and lists or parts of them so that a home user with a big disk can explore with optimised ease when he is paying by the minute.

MOO Integration WWW meets MOO, IRC, etc. MOO users explore the web, WWW users find MOO rooms. See MOOs on the web .

3.1.4 Documentation

Tutorials Hypertext leading a new or prospective user through W3.

Canned demos A hypertext path through some good represenattive places to visit. Maybe a script to make a timed presentation (with sound?).

Videos Would save us giving so many talks!

Policy documents Statements on NIR: collect from institutes, provide background for organizations making NIR policy decisions.

Tim BL

3.2 Information Provider

There are many ways of making your new or existing data available on the "web" . The best method depends on what sort of data you have. (If you have any questions, mail the www team at www-bug@info.cern.ch.). See also: Web etiquette . How can I help ?

3.2.1 Running a server

You can set up a basic W3 server and configure it to serve data in certain directory trees.

There are several servers available: pick one suitable forthe platform on which you wish to run.

If you like, to save the trouble of writing hypertext, you can use the -dy option to allow readers to browse through the directory structure: directories will appear as hypertext documents. Any README files can (optionally) be automatically included at the top or bottom of the directory listsings. This is the simplest way of serving data. See more about the HTTP server .

If you have some plain text files then you can easily write, or generate using a script, a small hypertext file which points to them. To make them accessible you can use either anonymous FTP , or the HTTP daemon .

3.2.2 Editing hypertext

You can use our prototype hypertext editor to create a web of hypertext, linking it to existing files. This is not YET available for X11 workstations –– you need a NeXT. This is a fast way of making online documentation, as well as performing the hyper-librarian job of making sure all your information can be found.

If you don't yet have a editor for directly editing a document as you see it ("wysiwyg"), then you can still edit the HTML markup with an ordinary text editor. This requires a knowledge of the HTML language .

Whatever way you do it, you might like to read a Style Guide for online hypertext.

3.2.3 Indexing your files

If you want to generate a full-text index, then you could use the public domain WAIS software - your data will then be accessible (as plain text, not hypertext) through the WAIS gateway .

3.2.4 You have an existing information base

If you have a maintained base of information, don't rush into changing the way you manage it. A "gateway" W3 server can run on top of your existing system, making the information in it available to the world. This is how it works:

- Menus map onto sets of hypertext links
- Different search options map onto different "index" document addresses (even if they use the same index underneath in your system).
- Procedures used by those who contribute and manage information stay unaltered.

An W3 server is such a simple thing that a simple shell script will often suffice. This is great for bits of information available locally through other programs, which you would like to publish. See more details on writing servers using shell scripts under unix , or in DCL under VMS .

If your database is WAIS, VMS/HELP, XFIND, or Hyper-G, a gateway exists already. These gateway servers did not take long to write. You can pick up a skeleton server in C from our distribution .

For more information, see:

- W3 Server software
- A case study of one system .
- Making a W3 server for existing data .
- Allowing multiple selections

3.2.5 Setting up a telnet service

A telnet service allows people to telnet to your machine and get information from the web. You can set things up so that the moment they telnet to your server, they are in a www browser. See:

- How to set up a telnet server

3.2.6 Professional Help

If you would like advice on methods designing information systems, and setting up clients and customized servers, there are professional services who will be pleased to discuss your situation. (See list) Tim BL

3.3 Etiquette

There are a few conventions which will make for a more useable, less confusing, web. As a server administrator you should make sure this applies to your data. The Style Guide for Online Hypertext gives more ideas for all information providers.

- Signing your work –– especially the root page.
- Giving its status

Your server needs these things set up once per server:

3.3.1 A welcome page for outsiders

You don't have to have any particular structure to the data you publish: you can let it evolve as you think best. However, it is neat to have a document on each host which others can use to get a quick idea (with pointers) of what information is available there. You should put a "pass" line into your daemon rule file to map the document name "/" onto such a document. As well as a summary of what is available at your host, pointers to related hosts are a good idea.

3.3.2 An alias for your server

If you have a serious server then it may last longer than the machine on which it runs. Ask your internet domain name manager to make an alias for it so that you can refer to it, instead of as "mysun12.dom.edu" as "info.dom.edu" for example, or "www.dom.edu". This will mean that when you change machines, you move the alias, and people's links to your data will still work.

3.3.3 An alias for yourself

You should make a mail alias "webmaster" on the server machine so that people who have problems with your server can mail you about it easily. This is similar to the "postmaster" alias for people who have mail problems with your machine. Tim BL

3.4 Using Anonymous FTP

Anonymous FTP access means using the FTP protocol with username "anonymous" and a password which is the reader's mail address. It is a conventional way of publishing material on the internet.

It allows anyone on the internet (or gatewayed network) to access a SUBSET of your files. On a unix system, this involves setting up a user "ftp" in whse home directory the public files will be. An external user sees a file system whose root is that home directory. Look at the man page for ftpd on your system.

Normally, the public files are kept in /pub or a subdirectory of that.

Any file with which is accesible using anonymous ftp is accesible as a W3 document with address

```
file://node.domain/path/path/path
```

for (imaginary) example

```
file://info.cern.ch/pub/README.txt
```

Tim BL

4. Technical details

(See also: W3 project)

How to provide data	How can I make my own data available on the web?
Developments	Things to be done, thing speople are doing.

4.1 Specs:

The HTTP protocol	WWW's own protocol. WWW clients all speak many other protocols.
HTML format	A description of the markup language used for some documents and for search hit-lists.
Access Authorization	Document protection in W3.
Addressing (URLs)	The syntax of W3 document addresses.

4.2 Discussion:

Design Issues	Discussions of decisions to be made when designing or selecting a hypertext/IR system. See also related products .
Working notes	Work in progress at the drawing board, notes of meetings etc. Additions welcome: send URLs
News	Some internet/usenet newsgroups of possible interest to the WorldWideWeb project.

4.3 Other:

Coding standards	A basic style guide for W3 code contributors. If you write code, read this!
Using CVS in WWW	A guide to setting up the code management system for for those hacking the code.
Test data	A collection of data for testing

5. Design Issues

This lists decisions to be made in the design or selection of a hypermedia information system. It assumes familiarity with the concept of hypertext. A summary of the uses of hypertext systems is followed by a list of features which may or may not be available. Some of the points appear in the Comms ACM July 88 articles on various hypertext systems. Some points were discussed also at ECHT90 . Tentative answers to some design decisions from the CERN perspective are included.

Here are the criteria and features to be considered:

- Intended uses of the system.
- Availability on which platforms?
- Navigational techniques and tools: browsing, indexing, maps, resource discovery, etc
- Keeping track of previous versions of nodes and their relationships
- Multiuser access: protection, editing and locking, annotation.
- Notifying readers of new material available
- The topology of the web of links
- The types of links which can express different relationships between nodes

These are the three important issues which require agreement betwen systems which can work together

- Naming and Addressing of documents
- Protocols
- The format in which node content is stored and transferred
- Implementation and optimisation - Caching , smart browsers, knowbots etc., format conversion , gateways.

5.1 Intended Uses

Here are some of the many areas in which hypertext is used. Each area has its specific requirements in the way of features required.

- General reference data - encyclopaedia, etc.
- Completely centralized publishing - online help, documentation, tutorial etc
- More or less centralized dissemination of news which has a limited life
- Collaborative authoring
- Collaborative design of something other than the hypertext itself
- Personal notebook

The CERN requirement has a mixture of many of these uses, except that there is not a requirement for distribution of fixed hypertext on hard media such as optical disk. Evidently, the system will have to be networked, though databases may start life at least as personal notebooks.

For looking up data bases, the user should be able to refer to already prepared complex queries by simple UDIs. A moe advanced user should also be able to prepare a complex query himself, store it (interpreted language!) in his local filing space, and use it through a simple UDI.

[The (paper) document "HyperText and CERN" describes the problem to be solved at CERN, and the requirements of a system which solves them.

5.2 Availability on various platforms

The system is to be available (at CERN) on many sorts of machine, but priorities must be decided. A list comprises:

- A unix or VMS workstation with X-windows
- An 80 character terminal attached to a unix or VMS machine, or an MSDOS PC
- An 80 character terminal attached to an IBM mainframe running VM/CMS
- A Macintosh
- A unix workstation with NextStep
- An MS-DOS/Windows PC

The order above does not imply a priority. It may be that the implementation on one system will lead more easily to an implementation on one of the others, and this would in practice change the order of porting. The requirement for 80 column terminals to be useable (emphasized by M. Goossens) follows from low budgets of many of our users.

The order of implementation of special browsers at CERN is a function

5.3 Navigational Techniques and Tools

TBL There are a number of ways of accessing the data one is looking for. Navigational access (i.e., following links) is the essence of hypertext, but this can be enhanced with a number of facilities to make life more efficient and less confusing.

5.3.1 Defined structure

It is sometimes nice for a reader to be able to reference a document structure built specifically to enhance his understanding, by the document author. This is especially important when the structure is part of the information the author wishes to convery.

See a separate discussion of this point .

5.3.2 Graphic Overview

A Graphic overview is useful and could be built automatically. Should it be made by the author, server, browser or an independent daemon?

Can one provide an overview with less granularity than the basic web by grouping nodes in some way? The user could select from link types used to imply the tree structure. (JFG)

I think this depends on how long it will take. It might be interesting to experiment with daemons which will independently make and update maps of the web. This is not essential for a first pilot model.

5.3.3 History mechanism

This allows users to retrace their steps. Typical functions provided can be interpreted in a hypertext web as follows:

Home	Go to initial node
Back	Go to the node visited before this one in chronological order. Modify the history to remove the current node.
Next	When the current node is one of several nodes linked to the “back” node, go to the next of those nodes. Leave the “Back” node unchanged. Modify the history to remove the current node and replace it with the "next" (new current) node.

Previous When the current node is one of several nodes linked to the "back" node, go to the preceding one of those nodes.

In many hypertext systems, a tree structure is forcibly imposed on the data, and these functions are interpreted only with respect to the links in the tree. However, the reader as he browses defines a tree, and it may be more relevant to him to use that tree as a basis for these functions. I would therefore suggest that an explicit tree structure not be enforced.

(If a default tree is needed by the system for some reason, then we can always use the creation order: when a node is created it is always created with a link to an existing node. Such links, whatever their type, may be used to define a tree. If they are deleted, an alternative link must be chosen to become a tree link.)

If authors want to write a tree structure into their documents, then the words "after", "before" and "above" could be used to mean a static structure.

5.3.4 Intelligent navigation

See A. Secret's discussion of intelligently navigation techniques .

5.3.5 Index

An Index helps new readers of a large database quickly find an obscure node. Keyword schemes I include in the general topic of indexes. The index must, like a graphic overview, be built either by the author, or automatically by one of the server, browser, or a daemon . The index entries may be taken from the titles, a keyword list, or the node content or a combination of these. Note that keywords, if they are specifically created rather than random words, map onto hypertext "concept" nodes, or nodes of special type "keyword". It is interesting to establish an identity relationship between keywords in two different databases –– this may lead a searcher from one database into another.

Index schemes are important but indexes or keywords should look like normal hypertext nodes. The particular special operation one can do with a good keyword index system which one can't do with a normal hypertext system is to do a fast search on multiple keywords. This must to be provided as an extension to the hypertext navigation scheme. However, it is in fact analogous to a trace starting with more than one node, which is a valid hypertext tracing operation. The difference is that the tracing would normally be done by a browser, but the indexed search done by the server.

When many nodes in a web represent different indexes, then a query search can chain between them (See " Web of indexes "). Nat Torington's musings .

See also: HyperText and Information Retrieval

5.3.6 Node Names

These allow faster access if one knows the name. They allow people to give references to hypertext nodes in other documents, over the telephone, etc. This is very useful. However, in Notecards, where the naming of nodes was enforced, it was found that thinking up names for nodes was a bore for users. KMS thought that being able to jump to a named node was important. The node name allows a command line interface to be used to add new nodes.

I think that naming a node should be optional: perhaps by default the system could provide a number which can be used instead of a name.The system should certainly support the naming of nodes, and access by name.

5.3.7 Menu of links

Regular linkwise navigation may be done with "hotspots" (highlighted anchors) or may be done with a menu. It may be useful to have a menu of all the links from a given node as an alternative way of navigating. Enquire, for example, offers a menu of references as the only way of navigating.

5.3.8 Web of Indexes

In WWW , an index is a document like any other. An index may be built to cover a certain domain of information. For example, at CERN there is a CERN computer center document index . There is a separate functional telephone book index . Indexes may be built by the original information provider, or by a third party as a value-added service.

Indexes may point to other indexes. An index search on one index may turn up another index in the result hit list. In this case, the following algorithm seems appropriate.

Index context

Most index searches nowadays, though some look like intelligent semantically aware searches, are basically associative keyword searches. That is, a document matches a search if there is a large correlation (with or without boolean operations) between the set of words it or its abstract contains and the set of words specified in the search. Let us consider extending these searches to linked indexes.

Each index has a certain context. This may be represented by a set of keywords which may be considered to apply implicitly to everything indexed. For example, in the CERN computer center documentation index, one may imagine that everything in it will be considered as pertaining to the CERN computer center. We might represent the context by the keyword list "CERN computer center documentation physics support".

Context narrowing

Suppose we search a general physics index with the keywords "CERN NEWSLETTER". That index may contain an entry with keyword "CERN" pointing to the CERN index. Therefore, a search on the first index will turn up the CERN index. We should then search the CERN index, but looking only for the keyword "NEWSLETTER". The keyword "CERN" is discarded, as it is assumed by the new context. In this simple model, we can assume that the contextwords could be used directly as the keywords for the index itself.

A simple algorithm, then, would be for the server to discard from a search list any keywords matching the index's context – – but is this really what we want to do? Perhaps those keywords have a more refined meaning within the context. For example, if I am looking for documents about document storage schemes at CERN, I might search the index with the keyword "documents". I don't want this to be discarded because it is in the context: I am looking for documents about documents. It is understood that we are already within the context of computer center documentation, so to ask about documentation in this context implies more than that I am looking for a document.

A more refined approach would therefore be to strip from the search those keywords which were used in order to find the index. The keyword list for the entry of one index within anotherthen reflects the change in context.

Context Broadening

We have discussed here only a narrowing of context, not a broadening. One can imagine also a reference to a broader context index. In this case, perhaps one should add to the search some keywords which come from the original context but were not expressed. This would be dangerous, and people would not like it as they often feel that they are expressing their request in absolute terms even when they are not. Also, they may have been trying to escape from too restricing a context.

One should also consider a search which traces hypertext links as well as using indexes.

See also: Navigational techniques , Hypertext and IR ,

Tim BL

5.4 Tracing Links

A form of search in a hypertext base involves tracing the links between given nodes. For example, to find a module suitable for connecting a decstation to SCSI, one might try finding paths between a document on decstations and a document on SCSI. This is similar to relevance feedback in index searching.

Tracing is made more powerful by using typed links. In that case, one could perform semantic searches for all document written by people who were part of the same organisation as the author of this one, for example. This can use node typing as well.

When using link tracing, documents take over from keywords.

See Scott Preece's vision.

Tim BL

5.4.1 Intelligent Navigation

Rating documents

We could take into account:

The author's view The author could rate the interest of his documents (optional).

Most looked at Each time a document is looked at, it could increment a "popularity" value, otherwise decreasing as time goes by, e.g.

```
pop = exp(-alpha*(t - tp)),
```

tp being equal to ln(pop)/alpha + t_old, t_old being the last time someone looked at the file or through the link, and t the current time.

The reader's view Any reader might rate the interest of the document, like "boring" or "interesting" (optional).

The reader should be allowed at any time to select the weight of each rating, with a default value that he could set in a default file.

The best-rated link, as well as any link over a default interest value (e.g. the average value for the whole text, or a constant), should be colored in a special way.

Search in the Web where no index is provided

The problem dealt with here is: I am in an html file, I know what I am looking for, I know keywords for it, and I want to see if there is anything about it available FROM the current file through its links.

Where to search ? A breadth-first traversal searching seems to be the only way, if we don't want our grand-children to get the answer for us...

The search might detect if a file has already been looked through, and save the results for it.

For best results, after a study of each link of a given file, the search should study the links of the best-rated file, and then the links of the next best-rated file so far, wherever it is.

How to rate the interest of a file ? We could take into account:

The text How many times the keywords are used in the text.

The titles Each time one of the keywords is used in a title, this shall increase very much the interest of a given file.

Its own links The file should be given some feedback about the interest rates of its own linked files, which might have themselves been corrected if their own linked files show sufficient interest, and so on.

General interest general information such as quoted above would also be used: the author's rating, the readers' average rating, the number of readers having looked through it per unit of time ...

How long ? This is of course the most important. Given an infinite time, a search can be quite accurate, but will be very inefficient for the reader.

There are many ways to stop the search:

File found A file that seems enough related to the keywords is found, and the reader wants to get right down to it. The reader would have to define the limit between "enough related" and "not enough related" somehow, or use a default value.

Depth reached The reader could set a depth as a limit (e.g. the search should not follow more than three consecutive links).

Time over The reader could set a maximum time for search, whatever other limitations he may use. He should also be allowed to stop the search at any time and get the best result so far.

Then what ? Once the search is over, the reader could have two choices:

- To get the best-rated file, and then by decreasing interest the others. The best-rated file isn't necessarily directly linked to the reader's current file current file.
- To get the best-rated path, which means that the best link he should use would be highlighted in some way at each stage.

The reader should have the possibility to keep searching, while starting to read the documents found.

We now see better the difference between his two choices:

When he only wants the best-rated files, the reader will have access to files that won't be much related one to each other. When he takes the best-rated path, the user will follow links that have been created by a human being in an order that we may suppose to be logical.

When the reader wants some detail on a well-known field, he could take the first search method; when he needs a somewhat more logical information on an unknown field, he could take the second search method.

Increased Speed

Depending on its own possibilities, the client could dedicate part of its memory to guess what file(s) might be asked next by the reader, and memorize it/them while the reader would be reading its document. This should depend upon what amount of memory is available for it, what size the documents are, how difficult is the guess, how blocked is the network...

The smartest might be to ask for a transfer of the first page only, so that the rest of the file could be transferred while the reader would read the beginning of it. AS

```
Date: Tue, 13 Jul 1993 11:20:37 -0800
To: www-talk@nxoc01.cern.ch
From: kevin@scic.intel.com (Kevin Altis)
```

5.4.2 Link To Living

The document is frozen

```
For "frozen text documents" it might be wise to ignore white space, line
delimiters, and other control characters since the same document (a mail
message or news article) may appear to  be the same on different platforms
(Unix, Mac, DOS/Windows), but different line endings will be used, tabs
could be converted to spaces, etc. so it might be "safer" to refer to
character offsets that ignore white space. This would still allow a link to
a particular word, sentence, paragraph, etc.
```

```
On the other hand, link references to frozen documents could be made such
as "word 2 to 4 of sentence 5 of paragraph six" which would be the same on
all platforms if the terms "word", "line", "sentence", and "paragraph" are
defined. Apple uses this model in their text object model today. Many
modern scripting environments: AppleScript, HyperCard, ToolBook, MetaCard,
etc. also support this type of referencing.

Finally, references might be made to a word(s) or phrase(s) contained in a
document, so that the actual physical location of the link isn't determined
until the document is retrieved. Under this scenario, a browser also needs
to be able to find the next occurance within the document so that the user
sees all references, not just the first. A link reference by search
"phrase" works for formatted documents as well as straight text, so the
search method will work for Microsoft Word, RTF, WordPerfect, FrameMaker,
TeX, plain text, etc. versions of the same raw information, the search just
ignores formatting information. It doesn't matter if the document changes,
since no exact location offset references are made; the worst case is that
the search phrase is removed from the document so that the link is
effectively gone. An example of this kind of lookup is done by the On
Location software for the Macintosh; it maintains an index of all text on a
drive, but the index refers only to the document, not an exact location
within a document. On Location allows you to search for text "as is" as
well as the root of a word, so a link to "link" would match "links" or
"linking" if you wanted. I think this kind of link would fit in well with
the HTML+ specification.

I like the last approach, since it appears to work better with frozen
documents as well as documents with different versions. By referencing a
lookup word, phrase, etc. a browser or server can easily make a index of a
document as well as look for other documents that might be applicable. This
is one of those rare cases where the solution works well for the user and
the machine.
```

ka

5.4.3 MOOs and WWW

Who's interested in getting MOOs and MUDDs into the web and vice-versa? Here are a few pointers. Laso see

- MOOs on the web

marcus@x4u.desy.de (Marcus Speh)

Hi,
do you know anyone who's working on a MOO-WWW? Amy Bruckman from MIT told me maybe Joe Wang would, but I couldnt reach him –– I thought this would be an interesting possibility –– I am just about to enter the MediaMOO at MIT.

Tim:

Larry Masinter.parc@xerox.com is interested in that sort of area, and has made a Gopher object within a MOO.

Later, Marcus:

things have developed faster than this...in the meantime, I have entered the board of Usenet University and work with Joseph. Also, there is a WWW prototype object at MediaMOO, and we're thinking about hooking tkWWW into it. If anyone is interested: there is a mailing list discussing these things (technology for a UU MOO): contact creilly@maths.tcd.ie to be included.

5.5 Versioning

Definition: The storage and management of previous copies of a piece of information, for security, diagnostics, and interest.

Do you want version control?

Can you reference a version only?

If you refer to a particular place in a node, how does one follow it in a new version, if that place ceases to exist?

("Peter Aiken is the expert in this area" - Tim Oren, Apple)

Yes, at CERN we will want versioning. Very often one wants to correct a news item, even one of limited life, without reissuing it. This is a problem with VAX/NOTES for example. I would suggest that the text for the current version is stored, and separately those modifications necessary to backtrack to previous versions. I would expect previous versions to be regenerated only on the fly, as needed. (Apparaently SCCS stores the original file and the differences. This system does allow you to ditsribute the differences when updating copies.)

If full differences (deltas) are kept, the first version is just the first delta from a null document. The latest version is not available without regenerating it from all the deltas. For speed, it is obviously useful to keep a copy of the latest version (see caching).

Versioning is necessary for accountability (−−David Durand, dgd@cs.bu.edu). If an author is to be accountable for information published, it should be possible to demonstrate later what he wrote, even if he has later changed it.

A WWW server may provide versioning, by allowing links between a document version and its previou and succesive versions. This would be a good use of link typing .

5.6 Multiuser considerations

Multiuser access is made easier with a client/server model.We obviously want this. We also need simultaneous reading and writing of the same database. This is done by locking parts or all of the database while they are updated. One has to decide on the unit of data to be locked. I (TBL) imagine that it would be a node, not a database.

There is a specific problem which all distributed hypertext systems have had to tackle, in linking to living documents .

5.6.1 Annotation

Annotation is the linking of a new commentary node to someone else's existing node. It is the essence of a collaborative hypertext. An annotation does not modify the text necessarily: one can separate protection against writing and annotation.

5.6.2 Protection

Protection against unauthorized reading and writing is provided by servers. We use the word "domain" to describe a set of data which has the same protection. Life is simple if the domain is the database, or all the data administered by a given server. One can also add author-based protection to the contents of a node, or links, which have author information stored about them.

There is a problem illustrated by the following example. One might want to make a private annotation to something which is visible world-wide but unwritable. The annotation would be invisible to another reader: it would be stored in a private domain. The node itself is visible everywhere: it is stored in a public domain. This is a general problem of links being in a different domain to nodes.

5.6.3 Private overlaid web

A possible solution to this is to have, in the private domain, a partial copy of the public web, so that link information can be added to it. The copy of the net could also be used to tag on local cached copies of the contents of the remote nodes.

The writer would have to be aware of the domain into which he was writing. One could use a server per domain, but could imagine the need for more than one server per domain, or more than one domain per server.

See also: Generic Linking

5.6.4 Locking and modifying

Modification of text in a multiuser environment requires in principle some sort of atomic locking feature, so that two users do not update the same text at the same time. In fact some systems do not have this and still survive quite well: it depends a lot on the human environment.

Practically, the HTTP protocol must contain a lock/unlock command, and some way of recovering from a lock left on by a vanished user. The actual implementation will depend on the server or gateway. In the case of files, then a number of possibilitie exist:

- One can write-protect the file temporarily. This unfortunately levaes no clue as to who has locked it, when and why. It is also indistinguishable from a genuine protection to a document which should not be modified
- One can create a lock file containing information about who/when/why, whose name is derived from the name of the file in question.

5.6.5 Annotation

Annotation is the linking of a new commentary node to someone else's existing node. It is the essence of a collaborative hypertext. (See Multiuser considerations)

There is a problem when the web includes more than one authentication domain. See "protection".

5.7 Notification of new material

Does one need to bring it to a reader's attention when new unread material is added?

- Asynchronously (e.g. by mail) when the update is made?
- Synchronously when he browses or starts the application?
- Under the control of the modifying author? (i.e. can I say whether my change is a notifiable change? - Yes)

How do you express interest - in a domain, in a node, in things near a node, in anything you have read already, etc? A separate web which is stored locally, and logically overlay the public web?

There are two ways to make the connection between the modified material, and an interested person. One is, at the time of modification, to trace the interested parties. The other is, at some later time, for a daemon program (or a browser) to make a search for new things of interest to a given reader.

This is an essential feature. I suspect that a mixture of the two techniques might be necessary. Efficient dating of nodes, and date-based searches provided by the server, could make it easier for the browser to find interesting things which are new. It should also be possible to create a mailing list of people interested in a given topic, and use it to mail announcements of change.

This requirement is addressed by the "Interested" relationship of HTML, along with the POST method of HTTP for a generic notification semantics.

5.8 Topology

Here are a few questions about the underlying connectivity of a hypertext web.

5.8.1 Are links two- or multi-ended?

The term "link" normally indeicates with two ends. Variations of this are liks with multiple sources and/or multiple destinations, and constructs which relate more than two anchors. The latter map onto logic description systems, predicate calculus, etc. See the "Aquanet" system from Xerox PARC - paper at HT91). This is a natural step from hypertext whose the links are typed with semantic content. For example, the relation "Document A is a basis for document B given argument C". From now on however, let us restrict ourselves to links in the conventional sense, that is, with two ends.

5.8.2 Should the links be monodirectional or bidirectional?

If they are bidirectional, a link always exists in the reverse direction. A disadvantage of this being enforced is that it might constrain the author of a hypertext - he might want to constrain the reader. However, an advantage is that often, when a link is made between two nodes, it is made in one direction in the mind of its author, but another reader may be more interested in the reverse link. Put another way, bidirectional linking allows the system to deduce the inverse relationship, that if A includes B, for example, that B is part of A. This effectively adds information for free. This is important when a critical parameter of the system is how long it takes someone to create a link.

KMS and hypercard have one-way links; Enquire has two-way links.

There is a question of how one can make a two-way link to a protected database. The automatic addition of the reverse link is very useful for enhancing the information content of the database. See also: Private overlaid web , Generic Links .

It may be useful to have bidirectional links from the point of view of managing data. For example: if a document is destroyed or moved, one is aware of what dangling links will be created, and can possibly fix them.

A compromise that links be one-way in the data model, but that a reverse link is created when any link is made, so long as this can be done without infringing protection. An alternative is for the reverse links to be gathered by a background process operating on a basically monodirectionally linked web. See Building Back-links.

5.8.3 Should anchors have more than one link?

There is a design issue in whether one anchor may lead to many links, and/or on link have many anchors. It seems reasonable for many anchors to lead to the same reference. If one source anchor leads to more than one destination anchor, then there will be ambiguity if the anchor is clicked on with a mouse. This could be resolved by providing a menu to the user, but I feel this would complicate it too much. I therefore suggest a many-to-one mapping. JFG disagrees and would like to see a small menu presented to the user if the link was ambiguous. Microcosm does this.

5.8.4 Should links be typed?

A typed link carries some semantic information, which allows the system to manage data more efficiently on behalf of the user. A default type ("untyped") normally exists in some form when types are implemented. See also a list of some types . (Should a link be allowed to have many types? (- JFG) I don't think so: that should be represented by more than one link.(- TBL))

Link typing helps with the generation of graphical overviews , and with automatic tracing .

5.8.5 Should links contain ancillary information?

Does the system allow dating, versioning, authorship, comment text on a link? If so, how is it displayed and accessed? This sort of information complicates the issue, in that readable information is no longer carried within node contents only. Pretty soon, following this path leads to a link becoming a node in itself, annotatable and all. This perverts the data model significantly, and I cannot see that that is a good idea. Information about the link can always be put in the source node, or in an intermediate node, for example an annotation. However, this makes tracing more difficult. It is certainly nice to be able to put a comment on a link. Perhaps one should make a link annotatable. I think not.

5.8.6 Should a link contain Preview information?

This is information stored at the source to allow the reader to check whether he wants to follow a link before he goes. I feel that the system may cache some data (such as the target node title), or the writer of the node may include some descriptive material in the highlighted spot, but it is not necessary to include preview information just because access may be slow. Caching should be done instead of corrupting the user interface. If you have a fast graphic overview , this could

5.9 Link Types

See discussion of whether links should be typed .

Descriptive (normal) link types are mainly for the benefit of users and tracing, and graphics representation algorithms. Some link types for example express relationships between the things described by two nodes.

A Is part of B / B includes A

A Made B / B is made by A

A Uses B / B is used by A

A refers to B / B is referred to by A

5.9.1 Magic link types

These have a significance known to the system, and may be treated in special ways. Many of these relate whole nodes, rather than particular anchors within them. (See also multiended links and predicate logic) Suggestions:

UseIndex

The destination is the related index for a search by a user reading this document who asks for an index search function.

A document may have any number of index links, causing several indexes top be searched in a client-defined manner.

UseGlossary

The destination of the link is an index which should be used to resiolve glossary queries in the document. (Typically, a double-clik on a word which is not within an anchor).

A document may have any number of glossary links.

Annotation

The information in the destination node is additional to that in the source node, and may be viewed at the same time. It may be filtered out (as a function of author?).

Annotation is used by one person to write the equivalent of "margin notes" or other criticism on another's document, for example.

Tracing may ignore annotations when generating trees or sequences.

Next, Previous, Up

These terms may be applied to the tree the user creates in her browsing, but if the author puts links in, then a tree structure may be proposed by the author. This is very natural with hypertext versiins of books, etc.

Embedded information

If this link is followed, the node at the end of it is embedded into the display of the source node. This is supported by Guide, but not many other systems. It is used, in effect, by those systems (VAX/notes under Decwindows, Microsoft Word) which allow "Outlining" −− expanding a tree bit by bit.
The browser has a more difficult job to do if this is supported.

person described by node A is author of node B

This information can be used for protection, and informing authors of interest, for sending mail to authors, etc.

person described by node A is interested in node B

This information can be used for informing readers of changes.

Node A is in fact a previous version of node B

Node A is in fact a set of differences between B and its previous

version. This information will probably not be stored as nodes, but be generated from regular diff files. or some other delta method.

5.10 Document Naming

This is probably the most crucial aspect of design and standardization in an open hypertext system. It concerns the syntax of a name by which a document or part of a document (an anchor) is referenced from anywhere else in the world.
As many protocols are currently used for information retrieval, the address must be capable of encompassing many protocols, access methods or, indeed, naming schemes.
The WWW scheme uses a prefix to give the addressing sub-scheme, and then a syntax dependent on the prefix used, in order to be open to any new naming systems.

5.10.1 Name or Address, or Identifier?

Conventionally, a "name" has tended to mean a logical way of referring to an object in some abstract name space, while the term "address" has been used for something which specifies the physical location. The term "unique identifier" generally referred to a name which was guaranteed to be unique but had little significance as regards the logical name or physical address. A name server was used to convert names or unique identifiers into addresses.

With wide-area distributed systems, this distinction blurs. Locally, things which at first look like physical addresses develop more and more levels of translation, so that they cease to give the actual location of the object. At the same time, a logical name or a unique identifier must contain some information which allows the name server to know where to start looking. In a global context, for example "1237159242346244234232342342423468762342368" might well be unique, but it contains insufficient (apparent) structure for a name server to look it up. The name "info.cern.ch" has a structure which allows a search to be made in several stages. In fact, practical systems using unique identifiers generally hide within them some clues for the name server, such as a node name.

A hypertext link to a document ought to be specified using the most logical name as opposed to a physical address. This is (almost) the only way of getting over the problem of documents being physically moved. As the naming scheme becomes more abstract, resolving the name becomes less of a simple look-up and more of a search.

One expects in practice the translation of a document name taking several stages as the name becomes less abstract and more physical.

5.10.2 Hints

Some document reference formats contain "hints" to the reader about the document, such as server availability, copyright status, last known physical address and data formats. It is very important not to confuse these with the document's name, as they have a shorter lifetime than the document.

5.10.3 X500

The X500 directory service protocol defines an abstract name space which is hierarchical. It allows objects such as organizations, people, and documents to be arranged in a tree. Whereas the hierarchical structure might make it difficult to decide in which of two locations to put an object (it's not hypertext), this does allow a unique name to be given for anything in the tree. X500 functionally seems to meet the needs of the logical name space in a wide-area hypertext system. Implementations are somewhat rare at the moment of writing, so it cannot be assumed as a general infrastructure.

If this direction is chosen for naming, it still leaves open the question of the format of the address into which a document name will be translated. This must also be left as open-ended as the set of protocols. Tim BL

5.11 Document formats

The question of the format of the contents of a node is independent of the format of all the management information (except for the format of the anchor position within the node content). Therefore, the hypertext system can be largely defined without specifying the node format. However, agreement must be reached between client and server about how they exchange content information. Many hypertext systems qualify as "hypermedia" systems because they handle media other than plain text. Examples are graphics, video and sound clips, object-oriented graphics definitions, marked-up text, etc.

5.11.1 Format negotiation

Most hypermedia systems on the market today have the same application program responsible for the hypertext navigation and for the browsing. It would be safer to separate these features as much as possible: otherwise, in defining a universal hypertext system, one is burdened with defining a universal multimedia browser. This would certainly not stand the test of time. Node content must be left free to evolve. This implies that format conversion facilities must be available to allow simple browsers to access data which is stored in a sophisticated format. Such conversion facilities tend to exist in many applications, though not, in general, in hypertext applications.

The format of the content of a node should be as flexible as possible. Having more than one format is not useful from the user's point of view –– only from the point of view of an evolving system. I suggest the following rules:

5.11.2 1. Basic formats

There is a set of formats which every client must be able to handle. These include 80-column text and basic hypertext (HTML).

5.11.3 2. Conversion

A server providing a format which is not in the basic set of formats required for a client must have the possibility of generating some sort of conversion of the text (even if necessary an apology for non-conversion in the case of graphics to text) for a client which cannot handle it. This ensures universal readability world over.

5.11.4 3. Negotiation

For every format, there must be a set of other possible formats which the server can convert it into, and the most desirable format is selected by negotiation between the two parties. The negotiation must take into account:

- the expected translation time, including current load factors
- the expected data degradation
- the expected transmission time (?!!)

The times one could assume will be roughly proportional to the length of the document, or at least linear in it.

Application-specific node formats (e.g. physics event) would allow specialized browsers to perform local processing. This is a natural extension of the hierarchy of node formats. I would suggest one stick to the rule that a server providing such a type of data must provide some default conversion to a standardized view.

An index or a keyword could be a specific node format which would be manageable by a browser.

5.11.5 Examples

Examples of rich text formats which exist already at CERN are as follows, with, in brackets after each, other formats into which it might be convertible:

- SGML (Tex , Postscript, plain text)
- Bookmaster (Postscript, I3812, plain text)
- TeX (DVI, plain text)
- DVI
- Microsoft RTF (postscript, text, Next “WriteNow”) - See Specs
- Postscript, Editable Postscript (IBM 3812 bitmap)
- plain text

When a server (or browser) is obliged to perform a conversion from one format to another, one imagines that the result would be cached so that, if the same conversion were needed later, it would be available more rapidly. Format conversion, like notification of new material, is something which can be triggered either by the writer or by the browser. In many cases, a conversion from, say, SGML into Postscript or plain text would be made immediately on entry of the new material, and kept until the source has been updated (See caching , design issues).

5.12 Document caching

Three operations in the retrieval of a document may take significant time:

- Format conversion by the server, including version regeneration
- Data transmission across the network
- Format conversion by the browser

At each stage, the server (in the first case) or browser (in the other cases) may decide to keep a temporary copy of the result. This copy should ideally be common to many browsers.

Automatic caching relieves the user of having to explicitly save things which may be referred to again. It also relieves the system of keeping multiple copies (one for each user who has read the document). It allows local disk space to used optimally. Cache management takes into account such factors as

- expiry date
- file size
- time taken to get the file
- frequency of access
- time since access

5.12.1 Expiry date

As a guide to help a cache program optimise the data it caches, it is useful if a document is transmitted with an estimate by the server of the lengt of time the data may be kept for. This allows fast changing documents to be flushed from the system, preventing readers from being mislead. (I would not propose any notification of document changes to be distributed to cache managers automatically). For example, an RFC may be cached for years, while the state of the alarm system may be marked as valid for only one minute.

Window-oriented browsers effectively cache documents when they keep several at a time in memory, in different windows. In this case, for very volatile data, it may be useful to have the browser automatically refresh the window when its data expires.

(design issues)

5.13 Scott Preece on retrieval

3 Oct 91 (See tracing, Navigation)

My own "vision" of information retrieval models the whole database as

a network of objects. Some of the objects are words, some of them are

index terms (from a controlled vocabulary), some of them are documents some of them are pieces of documents, some of them are authors, etc. There are also typed links between nodes in the graph –– words are connected to documents by occurrence links, words are tied to words by dictionary links, document pieces are tied to documents by "is section of" links, etc. Searching then becomes a process of turning some of the nodes on, then turning on the nodes attached to them by certain kinds of links, and so forth.

So a dictionary expansion of the query works by activating a set of terms and then following all the dictionary links from those terms to other terms; a "search" works by activating a set of terms, then following all the occurrence links to the documents they appear in; relevance feedback works by starting with a set of activated documents and following the links back to the terms that occur in them.

If you use appropriate rules for calculating the level of activation of a node you can implement many of the similarity functions that have been reported in the literature and do a pretty effective job of seaching. For instance, suppose you have a term node which is activated with a

weight of 1. Suppose the spreading rule is that the weight is split among all the occurrence links leading from it to documents and the

combining rule is that all weights coming into a node are summed. Then after one spreading cycle each active document will have a weight equal to the sum of the inverse frequency of the terms in contains, which is a pretty reasonable search strategy. One enhancement is to have each link also weighted –– for term occurrence links it makes sense for that weight to be the number of occurrences of the term in the document.

It is true that doing this effectively requires doubly inverting the database, so that each document points to all its terms as well as vice versa, although you can finesse that by encoding the document as a list of terms rather than as Ascii text, with a slightly higher cost of

rebuilding the text when you need to display the document.

[My dissertation, describing this in excruciating detail, is *A Spreading Activation Model for Information Retrieval*, University of Illinois, 1981. You might be able to get it from University Microfilms if you're really interested. If you're at Thinking Machines, Dave Waltz had a copy once, but may well have shed it in the last decade. The machine readable form, alas, no longer exists (it lived on a long-dead PDP10)]

scott

```
--
scott preece
motorola/mcg urbana design center 1101 e. university, urbana, il   61801
uucp: uunet!uiucuxc!udc!preece,  arpa: preece@urbana.mcd.mot.com
phone: 217-384-8589   fax: 217-384-8550
```

6. Relevant protocols

The WorldWideWeb software can pick up information from many information sources, using existing protocols. Among these are file and news transfer protocols.

6.1 HyperText Transfer Protocol (HTTP)

WWW's own protocol, a faster search and retrieve protocol. It is "HTTP" not because it is only for transfering hypertext, but because it operates in a fast, stateless way as is needed for hypertext jumps. For more details:

- The protocol as implemented initially (1991)
- The full protocol as defined in 1992 and implemented in March 93
- Also see related specs:
- RFC 1341: Mutipurpose Internet Mail Extensions

6.2 File Transfer Protocol (FTP)

The file transfer protocol currently most used for accessing fairly stable public information over a wide area is "Anonymous FTP". This means the use of the internet File Transfer Protocol without authentication. As the WWW project currently operates for the sake of public information, anonymous FTP is quite appropriate, and WWW can pick up any information provided by anonymous FTP. FTP is defined in RFC 959 which includes material from many previous RFCs. (See also: file address syntax). Directories are browsed as hypertext. The browser will notice references to files which are in fact accessible as locally mounted (or on DECnet on VMS systems) and use direct access instead.

See also the prospero project and the shift project, for more powerful file access systems.

6.3 Network News

The "Network News Transfer Protocol" (NNTP) is defined in RFC 977 by Kantor and Lampsley. This allows transient news information in the USENET news format to be exchanged over the internet. The format of news articles is defined in RFC 850, Standard for Interchange of USENET Messages by Mark Horton. This in turn refers to the standard RFC 822 which defines the format of internet mail messages. News articles make good examples of hypertext, as articles contain references to other articles and news groups. News groups appear like directories, but more informative.

6.4 Gopher

The Gopher distributed information system uses a lightweight protocol very similar to HTTP. Therefore, it is now included in every WWW client, so that the Gopher world can be browsed as part of the Web. Gopher menus are easily mapped onto hypertext links. It may be that future versions of the Gopher and HTTP protocols will converge.

6.5 Z39.50

With the use of the freeWAIS software from CNIDR, the W3 software now accesses WAIS servers directly. WAIS is a variant of the z39.50 protocol. This is being developed from earlier versions which did not have the functionality required for NIR. −− see draft standards documents .

6.6 HTTP 0.9

This document defines the Hypertext Transfer protocol (HTTP) as originally implemented by the WorldWideWeb initaitive software in the prototype released. This is a subset of the full HTTP protocol, and is known as HTTP 0.9. No client profile information is transferred with the query. Future HTTP protocols will be back-compatible with this protocol.

This restricted protocol is very simple and may always be used when you do not need the capabilities of the full protocol which is backwards compatible.

The definition of this protocol is in the public domain (see policy).

The protocol uses the normal internet-style telnet protocol style on a TCP-IP link. The following describes how a client acquires a (hypertext) document from an HTTP server, given an HTTP document address .

6.6.1 Connection

The client makes a TCP-IP connection to the host using the domain name or IP number , and the port number given in the address.

If the port number is not specified, 80 is always assumed for HTTP.

The server accepts the connection.

Note: HTTP currently runs over TCP, but could run over any connection-oriented service. The interpretation of the protocol below in the case of a sequenced packet service (such as DECnet(TM) or ISO TP4) is that that the request should be one TPDU, but the response may be many.

6.6.2 Request

The client sends a document request consisting of a line of ASCII characters terminated by a CR LF (carriage return, line feed) pair. A well-behaved server will not require the carriage return character.

This request consists of the word "GET", a space, the document address , omitting the "http:, host and port parts when they are the coordinates just used to make the connection. (If a gateway is being used, then a full document address may be given specifying a different naming scheme).

The document address will consist of a single word (ie no spaces). If any further words are found on the request line, they MUST either be ignored, or else treated according to the full HTTP spec .

The search functionality of the protocol lies in the ability of the addressing syntax to describe a search on a named index .

A search should only be requested by a client when the index document itself has been descibed as an index using the ISINDEX tag .

6.6.3 Response

The response to a simple GET request is a message in hypertext mark-up language (HTML). This is a byte stream of ASCII characters.

Lines shall be delimited by an optional carriage return followed by a mandatory line feed chararcter. The client should not assume that the carriage return will be present. Lines may be of any length. Well-behaved servers should retrict line length to 80 characters excluding the CR LF pair.

The format of the message is HTML - that is, a trimmed SGML document. Note that this format allows for menus and hit lists to be returned as hypertext. It also allows for plain ASCII text to be returned following the PLAINTEXT tag .

The message is terminated by the closing of the connection by the server.

Well-behaved clients will read the entire document as fast as possible. The client shall not wait for user action (output paging for example) before reading the whole of the document. The server may impose a timeout of the order of 15 seconds on inactivity.

Error responses are supplied in human readable text in HTML syntax. There is no way to distinguish an error response from a satisfactory response except for the content of the text.

6.6.4 Disconnection

The TCP-IP connection is broken by the server when the whole document has been transferred.

The client may abort the transfer by breaking the connection before this, in which case the server shall not record any error condition.

Requests are idempotent . The server need not store any information about the request after disconnection. Tim BL

6.7 HyperText Transfer Protocol Requirements

These are dsicussions of requirements for HTTP. See also:

- The HTTP2 specification
- Why a new protocol? ,
- Other protocols used
- the HTTP protocol as currently implemented
- Protocol design issues .

The definition of this protocol is in the public domain (see policy).

6.7.1 Underlying protocol

Current HTTP uses ASCII transmission over a telnet-style internet protocol, to make it simple to program, so that it will catch on: conversion to run over an OSI stack will be simple as the structure of the messages is well defined.

HTTP2 similiarly runs ove an ASCII telnet-style link. the fiedls are represened in RFC-xxxx style mail message format, and as far as possible are taken from the equivalent mail header names.

6.7.2 Idempotent

This protocol is stateless, in that no state is kept by the server on behalf of the client. (This does not rule out caching by the server internally).

6.7.3 Request: Information transferred from client

Parameters below, however represented on the network, are given in upper case, with parameter names in lower case. This set assumes a model of format negociation in which in which the client says what he can take, and the server decides what to give him. One imagines that each function would return a status, as well as information specified below.

When running over a byte stream protocol, SGML would be an encoding possibility (as well as ASN/1 etc).

GET document_name HTRQ Please transfer a named document back. Transfer the results back in a standard format or one which I have said I can accept.

SEARCH keywords Please search the given index document for all items with the given word combination, and transfer the results back as marked up hypertext. This could elaborate to an SQL query. There are many advantages in making the search criterion just a subset of the document name space.

SINCE datetime For a search, refer to documents only dated on or after this date. Used typically for building a journal, or for incremental update of indexes and maps of the web.

BEFORE datetime For a search, refer to documents before this date only.

ACCEPT format penalty I can accept the given formats . The penalty is a set of numbers giving an estimate of the data degradation and elapsed time penalty which would be suffered at the CLIENT end by data being received in this way. Gateways may add or modify these fields.

PORT See the RFC959 PORT command. We could change the default so that if the port command is NOT specified, then data must be sent back down the same link. In an idempotent world, this information would be included in the GET command.

HEAD doc Like GET, but get only header information. One would have to decide whether the header should be in SGML or in protocol format (e.g. RPC parameters or internet mail header format). The function of this would be to allow overviews and simple indexes to be built without having to retrieve the whole document. See the RFC977 HEAD command. The process of generation of the header of a document from the source (if that is how it is derived) is subject to the same possibilties (caching, etc) as a format convertion from the source.

USER id The user name for logging purposes, preferably a mail address. Not for authentication unless no other authentication is given.

AUTHORITY authentication A string to be passed across transparently. The protocol is open to the authentication system used.

HOST The calling host name - useful when the calling host is not properly registered with a name server.

Client Software For interest only, the application name and version number of the client software. These values should be preserved by gateways.

6.7.4 Response

Suppose the response is an SGML document, with the document type a function of the status. (Example)

Status A status is required in machine-readable format. See the 3-figure status codes of FTP for example. Bad status codes should be accompanied by an explantory document, possible conianing links to futher information. A possibility would be to make an error response a special SGML document type. Some special status codes are mentioned below .

Format The format selected by the server

Document The document in that format

6.7.5 Status codes

Success Accompanied by format and document.

Forward Accompanied by new address. The server indicates a new address to be used by the client for finding the document. the document may have moved, or the server may be a name server.

Need Authorisation The authorisation is not sufficient. Accompanied by the address prefix for which authorisation is required. The browser should obtain authoisation, and use it every time a request is made for a document name matching that prefix.

Refused Access has been refused. Sending (more) authorization won't help.

Bad document name The document name did not refer to a valid document.

Server failure Not the client's fault. Accompanied by a natural language explanation.

Not available now Temporary problem - trying at a later time might help. This does not i,ply anything about the document name and authorisation being valid. Accompaned by a natural language explaination.

Search fail Accompanied by a HTML hit-list without any hits, but possibly containing a natural explanation.

Tim BL

6.7.6 Rules for Penalty calculation

There are two sorts of transformation: (a) conversion from one format into another, for reasons of presentation, and whereby information generally is lost, and (b) encoding, for reasons of speed (compaction), security (encryption), or transmission, and whereby the information and format remain untouched.
There are two questions to consider when deciding on different possible transfer formats between servers and clients: Information degradation and elapsed time.

Degradation

When information is converted from one format to another, it may be degraded. For example, when a postscript file is rendered into bitmap, it loses its potentially infinite resolution; when a TeX file is rendered into pure ASCII, it loses its structure and formatting.
This degradation is difficult to guess from simply the file type. and for a given file it is quite subjective. Any attempt to estimate a penalty will therefore be very aproximate, and only useful for distinguishing widely differing cases. A suitable unit would be the proportion, between 0 and 1, of the information which is not lost. Let's call it the degradation coefficient. One would hope that these coefficiemnts are multiplicative, that is that the process of converting a document into one format with degradation coeficient c1 and then further converting the result of that with coeficient c2 would in all be a process with coeffcient c1*c2. This is not, in fact, necessarily the case in practice but is a reasonable guess when we know no better.

Elapsed time

The elapsed time is another penalty of conversion. As an aproximation one might assume this to be linear in the size of the file. It is not easy to say whether the constant part or the size-proportional part is going to be the most important. The server, of course, knows the size of the file. It can in fact as a result of experience make improving guesses as to the conversion time. The conversion time will be a function also of local load. For particular files, it may be affected by the caching of final or intermediate steps in a conversion process. Given a model in which the server makes the decision on the basis of information supplied by the client, this information could include, for each type, both the constant part (seconds) and the size-related part (seconds per byte).
Tim BL, RC

6.8 Why a new protocol?

Existing protocols cover a number of different tasks.

- Mail protocols allow the transfer of transient messages from a single author to a small number of recipients, at the request of the author.
- File transfer protocols allow the transfer of data at the request of either the sender or receiver, but allow little processing of the data at the responding side.
- News protocols allow the broadcast of transient data to a wide audience.
- Search and Retrieve protocols allow index searches to be made, and allow document access. Few exist: Z39.50 is one and could be extended for our needs.

The protocol we need for information access (HTTP) must provide

- A subset of the file transfer functionality
- The ability to request an index search
- Automatic format negotiation.
- The ability to refer the client to another server

Tim BL

7. WWW Names and Addresses: URIs

Uniform Resource Locators (URLs) is the Internet name for a WWW address. The specification of URLs is a draft RFC, and comprises the following sections:

- Summary
- Terms
- Requirements
- Recommendations
- Specific Schemes
- BNF-Style syntax definition
- Security Considerations
- Conclusion
- Acknowledgements
- References

WWW will use any new forms of naming which give features such as persistence and redundancy, when the are available, by extension of the set of schemes in the URI.

This subject is discussed by the URI working group of the IETF (mail uri-request@bunyip.com to join). The following material may also be of interest:

- URI WG discussion list archive
- Discussion materials
- a discussion of design issues involved ,

7.1 UR Terms

URI Uniform Resource Idenifier. (originally, Universal). The generic set of all names/addresses which are short strings which refer to objects. The exact properties of each URI scheme depend on the scheme you are talking about. (Originally UDI in some www documents).

URL Uniform Resource Locators. Term introducted by the IETF in forming the URI working group to point out that currently available URIs are mainly addresses rather than names. Exactly what consitutes a locator as opposed to a name is basically lack of persistence, but this is a much discussed point and impossible to define precisely. In practice, the set of schmes referring to existing protocolls, listed in the URL specification.

URN Uniform Resource Name. 1. Any URI which is not a URL. 2. A particular scheme which is currently (1991,2,3) under development by the IETF, which should provide for the resolution using internet protocols of names which have a greater persistence than that currently assiated with internet host names or organizations. When defined, a URN(1) will be an example of a URN(1).

```
| |
|  _______________  _______________ |
| |  ftp: | |  urn: | |
| |  gopher: | |  fpi: ? | |
```

```
| |  http: | | | |
| |  etc | | | |
| |________________| |________________| |
| URLs URNs |
||
  URIs
```

URC Uniform Resource Citation. A set of attribute/value pairs describing an object. Some of the values may be URIs of various kinds. Others may include, for example, athorship, publisher, datatype, date, copyright status and shoe size. Not normally discussed as a short string, but a set of fields and values with some defined free formatting.

URM Something introduced by Michael Mealing along the lines of a URC encoded into a string, with a rather peculair syntax. ;-)

7.2 Overview of URLs

This is an old overview which is not definitive.

The format of a w3 address consists of the name of the naming sub-scheme to be used, then a name in a format particular to that subscheme, then an optional anchor identifier within the document. For example, the format is for all internet-based access methods:

scheme : // host.domain:port / path / path # anchor

A suffix # anchor id allows one to refer to a particular anchor within a document.

A suffix ? followed by words separated by + signs allows one to seach an index (see details).

References from one document to another with a similar name may be abbreviated to a relative name . This imposes certain restrictions on the way that the "path" is represented.

A special format is used to represent a search on an index . See also: the full BNF description , about escaping illegal characters .

7.2.1 Examples

```
file://cernvax.cern.ch/usr/lib/WWW/defaut.html#123
```

This is a fully qualified file name, referring to a document in the file name space of the given internet node, and an imaginary anchor 123 within it.

```
#greg
```

This refers to anchor "greg" in the same document as that in which the name appears.

7.2.2 Naming sub-schemes

Different schemes usually use different protocols on the network. The format of the address after the scheme name is a function of the particular scheme. In practice, all internet-based schemes have a common format for the node name and port. Schemes currently defined are as follows, with links to more details.

file Access is provided to files, using whatever means the browser and/or gateways have to reach files on obscure machines.

news Access is provided to news articles, and newsgroups, normally using the NNTP protocol.

http Access is provided to any other information using the HTTP search and retrieve protocol . The internal addressing of the information system is mapped onto a W3 path.

telnet Access is provided by an interactive telnet session. This is provided ONLY as an interface to other existing online systems which cannot or have not been mapped onto the W3 space.

gopher Access is provided using the "gopher" protocol. The gopher protocol is similar to HTTP but uses separate concepts of menus and text files rather than hypertext.

wais Access is provided using the WAIS adaptaion of the Z39.50 protocol.

Systems which are not accessed directly be W3 servers may be accessed though gateways, in which case the document address is encoded within the http address of the document in the gateway. Browsers which do not have the ability to use certain protocols may be configured to automaticaly use certain gateways for certain addressing schemes.

This could allow, for example, simple PC-based clients to follow links through X500 name servers.

7.3 Address for an index Search

If a given hypertext node is an index, or the server has an index associated with it, then a search may be done on that index by suffixing the name of the index with a list of keywords, after a question mark:

```
address_of_index ? keywordlist
```

The address of the index is a normal hypertext address. In the keywordlist, multiple keywords are separated by plus signs (+) . (See BNF syntax description .) The resulting string still does not contain any spaces. It may be considered to be the hypertext address of a document which is the result of making the keyword search on the index. Normally, if the search was successful, the document returned will contain anchors leading to other documents which match the selection criteria.

The search method, and the logical and lexical functions, weights, etc applied to the keywords will depend on the index address. One actual index may have several hypertext addresses, which when searched on will behave in different ways. For example, one may allow a search on author-given keywords only, while another may be a full text search. These things particular to an index should be descibed in the hypertext page for the index node itself (or in linked documents). For example, a server may allow specific boolean search combinations may be represented by the words "and", "or" and "not".

7.3.1 Example:

```
http://cernvm/FIND/?sgml+cms
```

indicates the result of perfoming a search for keywords "sgml" and

7.4 W3 addresses of files

The format of a hypertext reference to a file is an extension of the unix naming system. The full explicit format is:

file : // node / directories / name

The actual protocols used by the client depend on the implementation of the browser and the environment. Typically, the browser will check to see whether the node is the local node, or a node for which files are available mounted in some form of distributed file system. If neither of these are the case, then the browser may try rpc, anonymous FTP or other protocols.

7.4.1 Examples

```
file://cernvax.cern.ch/usr/lib/WWW/defaut.html
```

This is a fully qualified file name.

```
fred.html
```

This relative name , used within a file, will refer to a file of the same node and directory as that file, but the name fred.html.

7.4.2 Improvements : Directory access

The final file name should be optional. If the address ends with a '/', the browser should retrieve the contents of the specified directory and generate a page of virtual hypertext pointing to its contents. In addition, it could display an information file contained in that directory, if any is present. Suggested file names to search for in order : README.html, *README*.html, README, *README*, *readme*.

7.5 Hypertext address for net News

The format of a hypertext reference to information in the internet/usenet news system can take any of the following forms:

news: newsgroup This refers to a list of articles currently available in the given newsgroup. The newsgroup is a series of alphanumeric characters and dots.

news:* This refers to a list of valid newsgroups.

news: message_id This refers to a given article explicitly. The message_id is optionally surrounded by angle brackets, and must contain an @ sign.

Possible extensions to this are more generous wildcarding for the list of newsgroups. It takes too long to load the whole list, and it would be more useful to be able to browse through a set of newsgroups.

There is no way of referring to "unread" articles. Keeping track of this is the job of the browser.

7.5.1 Examples

```
news:<12345678@cernvax.cern.ch>

news:12345678@cernvax.cern.ch
```

These addresses both refer to the same (imaginary!) article by its unique message-id. (Note the hostname in the message-id is just part of the message id generated by the sender of the message - it is not a news server address).

```
news:comp.sys.next.announce
```

This refers to a list of articles in the newsgroup comp.sys.next.announce.

7.6 Relative naming

The address of a hypertext document is normally given within the context of another hypertext document. Where the addresses of the two documents are the similar, this allows only the difference between the two names to be given, saving space. An example is the address of the destination of a hypertext link , which is specified relative to the source document address.

(A futher practical advantage is that a group of documents may be transmitted without internal changes, or accessed using more than one address.)

This implies that certain characters ("/", "..") have a significance reserved for representing a hierarchical space, and must be recognized as such by both clients and servers.

In the WWW address format , the rules for relative naming are:

- If the " scheme " parts are different, the whole absolute address must be given. Other wise, the scheme is omitted, and:
- If the "host" and/or "port" parts are different, the host name and all the rest of the address must be given. The host name may be given using internet hostname conventions, ie domains may be omitted where different. This is not very well defined: one tends to assume that if any dot is present, then the full domain name is being given, up to the root (.) domain, while if there are no dots, the domain is the same as that of the hostname part of the the base address.
- If the access and host parts are the same, then the path may be given with the unix convention, including the use of ".." to mean indicate deletion of a path element. Within the path:
- If a leading slash is present, the path is absolute. Otherwise:
- The last part of the path of the base address (e.g. the filename of the current document) is removed, and the given relative address appended in its place.
- Within the result, all occurences "xxx/.." or "/." are recursively removed, where xxx is one path element (directory).

The use of the slash "/" and double dot ".." in this case must be respected by all servers. If necessary, this may mean converting their local representations in order that these characters should not appear within path elements (see "escaping").

7.7 HTTP Addressing

With an access code of http:, a protocol introduced for the WWW initiative is used to acquire data from a server. This is the "Hypertext Transfer protocol", HTTP , a simple search and retrieve (S and R) protocol.

The syntax of an http address is, with []indicating optional parts (see BNF description),

```
http : // hostname [ : port ] / path [ ? searchwords ]
```

for example, the following are valid addresses:

```
  http://info.cern.ch/hypertext/WWW/TheProject.html
http://crnvmc.cern.ch/FIND?sgml+examples
```

HTTP addresses conform to the WWW conventions, including the possibility of using the search format . The significance of the items in the path part of the document name is completely up to the server. Different paths may be used to select different databases, different views of the same database, etc.

hostname This is the name of the server in internet form. A numeric form (e.g. 128.141.201.74) may be used, by the domain name form (e.g. info.cern.ch) is preferred. The hostname is mandatory.

port This is a numeric port number. If a non-numeric string is used, it must be a defined service name. Note that as there is no central repository for service names (they are defined locaaly for each host), a service name is NOT an appropriate way to specify a port number for a hypertext address. If the port number is omitted the preceding colon must also be omitted. In this case, port number 2784 is assumed [This may change!].

See also: WWW addressing in general , HTTP protocol .
Tim BL

7.8 Telnet addressing

A telnet address is a spcecial case of a W3 address.

When a telnet address is used, information can only be rertrieved using an interactive telnet session. This has the disadvantage that information cannot be indexed, searched, etc automatically, nor can it be gatewayed into other systems. The telnet addressing form is used to allow a pointer to information systems such as library information systems which have not been gatewayed into the web properly yet.

The syntax is, with []indicating optional parts (see full BNF)

```
telnet : / /  [ user @ ] host  [ : port ]
```

There should be no spaces. For example, the following are valid telnet addresses:

```
telnet://www@info.cern.ch:23
telnet://www@info.cern.ch
telnet://info.cern.ch
```

user is the optional name of the user to be used for login. If the username is omitted, then so must be the "@" sign. This is equivalent to the argument used with the -l option on the ucb telnet command. When the username is omitted, some access servers will prompt for a username and password.

host This is the name of the server in internet form. A numeric form (e.g. 128.141.201.74) may be used, by the domain name form (e.g. info.cern.ch) is preferred. The host is mandatory.

port This is a numeric port number. If a non-numeric string is used, it must be a defined service name. Note that as there is no central repository for service names (they are defined locaaly for each host), a service name is NOT an appropriate way to specify a port number for a hypertext address. If the port number is omitted the preceding colon must also be omitted. In this case, port number 23 is assumed.

Tim BL

7.9 W3 address syntax: BNF

This is a BNF-like description of the W3 addressing syntax . We use a vertical line "|" to indicate alternatives, and [brackets]to indicate optional parts. Spaces are representational only: no spaces are actually allowed within a W3 address. Single letters stand for single letters. All words of more than one letter below are entites described elsewhere in the syntax description. (Entity names are here linked to their definitions, probably making this difficult to read with the line mode browser.)

An absolute address specified in a link is an anchoraddress . The address which is passed to a server is a docaddress .

anchoraddress docaddress [# anchor]
docaddress httpaddress | fileaddress | newsaddress | telnetaddress | prosperoaddress | gopheraddress | waisaddress
httpaddress h t t p : / / hostport [/ path][? search]
prosperoaddress p r o s p e r o : / / hostport / path
fileaddress f i l e : / / host / path
newsaddress n e w s : groupart
waisaddress waisindex | waisdoc
waisindex w a i s : / / hostport / database [? search]
waisdoc w a i s : / / hostport / database / wtype / digits / path
groupart * | group | article
group ialpha [. group]
article xalphas @ host
database xalphas
wtype xalphas
telnetaddress t e l n e t : / / [user @]hostport
gopheraddress g o p h e r : / / hostport [/ gtype [/ selector]][? search]
hostport host [: port]
host hostname | hostnumber
hostname ialpha [. hostname]
hostnumber digits . digits . digits . digits
port digits
selector path
path void | xalphas [/ path]
search xalphas [+ search]
user xalphas
anchor xalphas
gtype xalpha
xalpha alpha | $ | _ | @ | ! | % | ^ | & | * | (|) | . | digit
xalphas xalpha [xalphas]
ialpha alpha [xalphas]
alpha a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z
digit 0 |1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9
digits digit [digits]
alphanum alpha | digit
alphanums alphanum [alphanums]
void

See also: General description of this syntax, Escaping conventions.
Tim BL

7.10 Escaping illegal characters

The W3 address syntax allows a path to contain most printable ASCII characters, but some are inevitably used for punctuation are excluded. W3 addresses are sometimes used to represent addresses in some other space. This happens when an HTTP server, for example, uses file names as its document names, or when addresses from some other protocol (Gopher, WAIS, etc) are mapped into the W3 web.

In these cases, a convention is normally used to map illegal characters in these "foreign" names onto the allowed set.

In the case of an HTTP server, any mapping may be used.

A suitable convention is that a percent sign (%) followed by two hexadecimal digits (0-9 or a-f) stands for the single character with ASCII hexadecimal code represented by those two digits (Most significant digit first).

A percent sign itself must therefore be represented by %25, as 25 hex is the ASCII code for "%".

Tim BL

7.11 Gopher addressing

Gopher addresses indicate that the gopher protocol should be used to access the information. The Gopher protocol is a simple internet protocol similar to HTTP . It allows the transfer of menus or plain text files. (HTTP expresses both menus and plain text files as special cases of hypertext files). See the gopher protocol notes .

The syntax is, with []indicating optional parts (see BNF)

```
gopher:// hostname [: port ] [ / [gtype [selector] ] ] [ ? search ]
```

There should be no spaces. For example, the following are valid addresses:

```
gopher://gopher.micro.umn.edu:70
gopher://gopher.micro.umn.edu:70/1/
gopher://gopher.micro.umn.edu:70
```

The W3 address for a gopher item may be derived from the fields of a gopher menu line which has the format

host This is the name of the server in internet form. A numeric form (e.g. 128.141.201.74) may be used, by the domain name form (e.g. info.cern.ch) is preferred. The hostname is mandatory.

port This is a numeric port number. If a non-numeric string is used, it must be a defined service name. Note that as there is no central repository for service names (they are defined locaaly for each host), a service name is NOT an appropriate way to specify a port number for a hypertext address. If the port number is omitted the preceding colon must also be omitted. In this case, port number 70 is assumed.

gtype This is a gopher item type number, a (hopefully printable!) ASCII character. Currently these types are all ASCII decimal digit characters. Character "0" (hex 30) signifies a plain text file. Character "1" signifies a Menu. Character "7" signifies a searchable index. Character "8" should not be used in a W3 address: use telnet addressing instead. In general W3 terms, the type is the first part of the path. The rest of the path is the gopher selector string. The type field is a hint to the client as to how to represent the anchor, and how to follow it.

selector This is the string to be sent to the gopher server to identify the information required. NOTE that many but NOT all selector strings start with the gtype field, so often one sees the gtype field character repeated in a gopher URL..

7.11.1 Finger URLs

It is possible to use gopher URLs to point at "finger" resources, by specifying port 79 and setting the gtype field to "0". Note that this is not designed in, but happens to work due to the simplicity and similarity between the two protocols. The selector string contains the username to be "fingered".

Example

```
gopher://wsinis04.info.win.tue.nl:79/0reinpost
```

This refers to the result of the "finger reinpost@wsinis04.info.win.tue.nl" command. Tim BL

7.12 W3 addresses for WAIS servers

Servers using the WAIS ("Wide Area Information Systems") protocols from Thinking Machines may be accessed as part of the web using addresses of the form (see BNF description)

w a i s : / / hostport / database ...

Access (currently) goes through a gateway which stores the "source" files which contain the descriptions of WAIS servers. This address corresponds to the address of an index. To this may optionally be appended either a search string or a document identifier.

Note that changes have been proposed to WAIS document id format, so this representation of them may have to change with that format. Currently the WAIS document address necessary for retrieval by a client requires the following information, which is orginally provided by the server in the hit list.

Document format This is normally "TEXT" but other formats such as PS, GIF, exist.

Document length This is needed by the client who must loop to retrie the whole document in slices.

Document identifier This is an entity consisting of numerically tagged fields. the binary representation used by WAIS is transformed for readability into a sequence of fields each consisting of a decimal tag, an equals sign (=) , the field value, and a semicolon. Within the field value, hex escaping is used for otherwise illegal characters.

See also: Other W3 address formats, BNF definition.

8. HTML Overview

The WWW system uses marked up text to represent a hypertext document for transmision over the network. The hypertext markup language is an SGML format.

To find out how to write HTML, or to write a program to generate it, read the following sections:-

Text and Markup	An introduction to SGML
The elements	A list of the tags used in HTML with their significance.
Entities	Special characters are represented by SGML entities
HTML Specification	Guidance for parser implementors
DTD	The SGML document type definition for HTML. (Hypertext version)

The following do not form part of the specification

Style Guide	A guide to how to organize and write online hypertext
Beginner's guide	Marc Andressen's quick start to writing HTML
Example	A file containing a variety of tags used for test purposes, and its source text . See also finding examples on the web .
Future directions	Changes suggested for HTML improvements
HTMLplus	A more sophisticated document format for more demanding applictions than HTML. DTD and report .
Constraints	Design constraints for HTML which might explain some of its properties.
LibHTML	A conformant parsing code library

Other specifications which might be of tangential interest are

text/enriched	An internet draft on a simpler non-SGML MIME data type.
HTML+	A more sophisticated document type under development. Will not supercede HTML, but may be a superset.

8.1 HTML Elements

This is a list of elements used in the HTML language. Documents should (but need not absolutely) contain an initial HEAD element followed by a BODY element.

Old style documents may contain a just the contents of the normal HEAD and BODY elements, in any order. This is deprecated but must be supported by parsers.

See also: Status of elements

8.1.1 Properties of the whole document

Properties of the whole document are defined by the following elements. They should appear within the HEAD element. Their order is not significant.

TITLE	The title of the document
ISINDEX	Sent by a server in a searchable document
NEXTID	A parameter used by editors to generate unique identifiers
LINK	Relationship between this document and another. See also the Anchor element , Relationships . A document may have many LINK elements.
BASE	A record of the URL of the document when saved

8.1.2 Text formatting

These are elements which occur within the BODY element of a document. Their order is the logical order in which the elements should be rendered on the output device.

Headings	Several levels of heading are supported.
Anchors	Sections of text which form the beginning and/or end of hypertext links are called "anchors" and defined by the A tag.
Paragraph marks	The P element marks the break between two paragraphs.
Line Breaks	Like paragraph marks, but just forces a new line.
Horizontal Rule	A horizontal dividing line etc
Address style	An ADDRESS element is displayed in a particular style.
Blockquote style	A block of text quoted from another source.
Lists	Bulleted lists, glossaries, etc.
Preformatted text	Sections in fixed-width font for preformatted text.
Character highlighting	Formatting elements which do not cause paragraph breaks.

8.1.3 Graphics

IMG The IMG tag allows inline graphics.

8.1.4 Obsolete elements

The other elements are obsolete but should be recognised by parsers for back-compatibility.

8.2 SGML

ISO 8879:1986, Information Processing –– Text and Office Systems –– Standard Generalized Markup Language (SGML)

This is an ISO standardised derivative of an earlier IBM "GML". It allows the structure of a document to be defined, and the logical relationship of its parts. This structure can be checked for validity against a " Document Type Definition ", or DTD. The SGML standard defines the syntax for the document, and the syntax and semantics of the DTD. See books –– Eric van Herwijnen's "Practical SGML" and Charles Goldfarb's "SGML Handbook". Some of the points generally broght up in (frequent) discussions of SGML follow.

See also: Klensin on SGML

8.2.1 High level markup

An SGML document is marked up in a way which says nothing about the representation of the document on paper or a screen. A presentation program must marge the document with style information in order to produce a printed copy. This is invaluable when it comes to interchange of documents between different systems, providing different views of a document, extracting information about it, and for machine processing in general. However, some authors feel that the act of communication includes the entire design of the document, and if this is done correctly the formatting is an essential part of authoring. They resist any attempts to change the representation used for their documents.

8.2.2 Syntax

The SGML syntax is sufficient for its needs, but few would say that it is particularly beautiful. The language shows its origins in systems where text was the principle content and markup was the exception, so a document which contains a lot of SGML is clumsy. There is always, of course, an element of personal taste to syntax.

There are few obvious comments one could make.

8.2.3 Tools

For many years, SGML was generated by hand, by people editing the source. This has lead to a hatred of SGML among those who prefer their own mark-up language which may have been quicker, more powerful, or more familiar. The advent of WYSIWYG editors and solid SGML applications should improve that facet of SGML.

8.2.4 Archive

There are a number of SGML archive sites. In Germany, there the Darmstadt archive .

See also: HyTime , HTML , Hypertext Document formats . Davenport. group .

A public domain parser, SGMLS, exists. There is an archive of SGML related information at ifi.uio.no Tim BL

8.2.5 AAP

AAP stands for the American Asociation of Publishers, one of the first groups to fix on a common SGML DTD.

8.3 Design Constraints

When designing the HTML document type, consideration was given to a certain simplicity in order to allow many browsers and hopefully editors to be developed on many platforms.

8.3.1 Lack of nesting

Many text editing systems (Microsoft Word, The NeXT text object, the Mac text object, etc) handle text in a variety of styles but do not have any concept of nestable structure in the SGML sense.

The constraint here is therefore that HTML be able to be mapped into a sequence of paragraphs of styled text, and that if that text is edited that the editor should be able to map the sequence of styles back onto a sequence of elements in a well-defined way. This allows some limited trivial nesting (eg LI within UL) but no general nesting, as a finite and small set of styles is used. In particular, the styles are not parameterized by the nesting level.

8.3.2 Logical Markup

It is required that HTML be a common language between all platforms. This implies no device-specific markup, or anything which requires control over fonts or colors, for example. This is in keeping with the SGML ideal.

Lack of specific semantics

Just as the markup must not be device-specific

9. Library Internals –– OBSOLETE –– see new document

The WWW browsers and servers share a common architrecure, and a libarry of common code. (See also: Browser operation , and utility modules which are used throughout, and using the common library .).

In the contol flow diagram , common code is to the right of the grey line.

Application This module is the main program, and is window-system-dependent. In the line mode browser , it is HTBrowse . The application is called by the operating system, and manages the overall running of the program. It asks the navigation module to load the default page.

Navigation The module which acually loads documents is based in HTAccess.c. This uses all the protocol modules . Given an anchor ID to jump to, it asks the anchor object for the address in order to load it.

History This module records and replays on request the documents which the user vists.

Format manager The format manager uses the parser modules to load the document as appropriate. It can also decide on the format of a file from its name.

Anchor object The HTAnchor module takes care of creating anchors, managing the links between them and their attributes. This module is independent of the type of graphics object (text, line drawing etc). It stores hypertext addresses of anchors, and ensures that anchors with the same address are the same anchor. (More)

9.1 Protocol modules

A protocol module is invoked by the navigation module in order to access a document. Each protocol module is responible for extracting information from a local file or remote server using a particular protocol. Depending on the protocol, the protocol module either builds a graphic object (e.g. hypertext) itself, or it passes a socket descriptor to the format manager for parsing by one of the parser modules.

File access HTFile.c provides access to files, using HTFTP.c for remote access. The latter uses HTTCP for common TCP routines.

HTTP access The HTTP module handles document search and retrieve using the HTTP protocol.

News access The NNTP internet news protocol is handled by HTNews which builds a hypertext.

Gopher access The internet gopher access to menus and flat files (and links to telnet nodes etc) is handled by HTGopher .

WAIS access is implemented in a separate gateway program .

9.2 Format conversion modules

These modules allow different formats to be used to generate graphic objects. They invoked by the format manager. Currently we only parse HTML and plain text, but obviously other formats can be added.

HTML Basic hypertext parsing is done by HTML.c which uses the simple SGML engine SGML.c as a basic tokeniser and element stack manager.

Plain text This is built directly by the format manager as it is so simple.

9.3 Graphic objects

A graphic object is a (complex) displayable entity. It is built by a protocol module directly or using a parser. Graphic objects are in general necessarily coded differently on diferent window systems. The graphic object is resposible for displaying istelf, catching mouse clicks, and calling the navigation object in order to follow links. We use the more common term "document" to describe the logical entity which a graphics object represents and displays.

Hypertext	This object is window-system dependent. In the line mode browser, the GridText module is the hypertext object, providing the generic functionality of HText.h

Tim BL

9.4 Anchors

Anchors represent parts of graphic objects which may be the sources or destinations of links. Here follows a general description of thir implementation in the WWW architecure . (See definition)

An anchor be the source of no, one, or many links . It has one "main" link for the (common) case in which it is the source for one link. (In the w3 software, only the main link is normally used - jan 92)

An anchor may be the destination of no, one, or many links. The anchor module stores all links known by the program, and so in fact manages a copy of a small part of the web.

There are two types of anchors: Parenet anchors and child anchors.

9.4.1 Parent anchors

These represent whole documents. Every graphic object has an associated parent anchor. Associated with a parent anchor is data including:

- The title of the associated document, if known. This allows the document's title to be displayed in lists of previous nodes visited, etc., even when the document itself has been freed.
- A flag as to whether the document is an index.
- The address of the document. When a new anchor is created, the code ensures that if an anchor with that address already exists, that that anchor is returned instead, so no duplicates can exist.
- A list of children

9.4.2 Child anchors

These represent parts of documents. The graphic object stores the correlation between the id of the anchor and the actual space (time) shape which is referred to. Child anchors contain

- A pointer to the parent.

Tim BL

9.5 Utility modules

HTChunk	A module which manipulates flexibly arrays in memory. See header file .
HTAtom	A module which generates unique 32-bit pointers representeing strings, for rapid comparison. See header file .
HTList	A container object for other objects. See header file .
HTFont	Definition of a font object. See header file . No code. Needs to be changed for different font systems.
HTTCP	General TCP-IP routines to be used in combination of the socket library. See header file .

CBT

9.6 Browser Operation

The WWW browsers operate, in general, in the following manner.

9.6.1 Data flow

The data flow is demonstrated in a separate diagram . The application gives the navigation module the home page address. The anchor is generated (using the Anchor object) and passed via HTAccess to the relevant protocol module for loading. In the case of HTTP or a file, a character stream of data is passed to the parser , which build a hypertext object. In the case of news, the protocol module builds the hypertext object itself.

The hypertext object is built as a stream of text interspersed with style changes and anchor start/end points. The parsers create the anchors, giving their addresses, and just pass the id of the object to the graphic object.

Events such as mouse clicks are picked up by the application, and passed to the hypertext object. This may determine that a link should be followed, in which case it invokes the navigation module again, passing the anchor object's id. The naviagation module asks the anchor object for its address, and so loads the next document. Tim BL

9.7 Authorisaztion in the server

The authorization for a HTTP file server running on a unix system uses the underlying file protection scheme.

The user and password registration tools are those of the system, so two parallel systems do not have to learned by administrators.

The daemon may be run under a non-root uid so that it is less prone to pose a security problem due to an obscure programming bug.

The daemon will therefore only have access to certain files by virtue of the uid under which it runs. This will provide a certain security.

The daemon will itself voluntarily refuse access if the rule file denies access.

The daemon is running in secure mode it will require a user/password pair for any file which does not have public read access. It will check that the password is valid, then that the relevant user would have access to the file. Therefore the file must be accessible by BOTH the uid under which the daemon runs, and the authorized user.

Authorized useds may be dummy (non login) users representing groups of people.

Administration

Every class of user having privileged access will have to be given a (dummy) user id on the system using the normal tools.

Every set of documents requiring a different pattern of access rights should be given a corresponding groups id using the normal tools.

The many-many mapping of groups to users (traditionally, in the "/etc/groups" file) may be used to describe the mapping of the document sets onto the dummy users.

Browser

The browser maintains a list of server/user/password for each protected server that the user has accessed during the session. (Note that the first access of a protected document must necessarily fail, and lead the browser to putting up a panel requesting the user/password pair. As one wants to send the user/password pair only to the correct server, the browser should not be so constructed as to contain or ask for the triples before actual access).tbl ctb rc

10. Coding Standards

This document describes a coding style for C code (and therefore largely for C++ and Objective-C code). The style is used by the W3 project used so that:-

- Code is portable and maintainable.
- Code is easily readable by other project members.

If you have suggestions, do send them. (We do not include points designed to allow automatic processing of code by parsers with an incomplete awareness of C syntax.). IF YOU DO NOT ABIDE BY THIS GUIDE YOUR CODE WILL ANNOY OTHER TEAM MEMBERS AND MAY NOT PORT.

The style guide is divided into sections on Language features , Macros , Module header , Modules in straight C , Function header , Code style , Identifiers , Code management: the use of CVS , Include files , Directory structure , special marks .

(See also pointers to some public domain styles). Tim BL

10.1 Language features

Code to be common shared code must (unfortunately!) be written in C, rather than any objective C or C++, to ensure maximum portability. This section does not apply to code written for specific platforms.

C code must compile under either a conforming ANSI C compiler OR an original Kernighan Ritchie C compiler. Therefore, the __STDC__ macro must be used to select alternative code where necessary.. (example)

Code should compile without warnings under an ANSI C compiler such as gcc with all warnings enabled.

Parameters and Arguments The PARAMS(()) macro is used to give a format parameter list in a declataion so that it will be suppressed if the compiler is not standard C - see example . The ARGS1 macro is for the declaration of the implementation, taking first the type then the argument name. For n arguments, macros ARGn exists taking 2n arguments each.

#endif Do put the ending condition in a comment. Don't put it as code - it won't pass all compilers.

#elif Don't use it. Basic cpp doesn't know it.

#preprocessor statements DON'T indent any preprocessor statements. They must (for some compilers) begin in column 1.

const This keyword does not exist in K C, so use the macro CONST which expands to "const" under standard C and nothing otherwise. −− See HTUtils.h

(part of: style guide) Tim BL

10.2 Module Header

The module header is the comment at the top of a .h or .c file. Information need not (except for the title) be repeated in both the .c and .h files. Of course History sections are separate. See a dummy example . Note:-

Heading To make it easy to spot the file in a long listing, put a header and te file name in the top right-hand corner.

Authors Just a list to make the initials intelligible. Use initials in the history or in comments in the file.

History A list of major changes of the file. You do not need to repeat information carried by a code management system or in an accompanying hypertext file.

Section headings Sections in the file such as public data, private module-wide data, etc should be made visible. Two blank lines and a heading are useful for this.

Tim BL

```
/* Foo Bar Module foobar.c
** ==============
**
** Authors:
** JB J. Bloggs, ACME Widget Company, TX, USA
** JD J. Doe, University of Tamahalahula, Samabaria
**
** History:
**    Jan 1983 First wriiten as a widget sorter (JB)
** 23 Nov 1986 Converted into mangle worzle sorter (JD)
** 38 Dec 1992 Bug fix: Used to carsh on null worzles. (JD, JB)
**
** Copyright:
** CERN copyright -- See Copyright.html
*/


/* Global Data
** -----------
*/
```

Tim BL

10.3 Function Headings

This style concerns the comments, and so is not essential to compilation. However, it helps readability of code written by a number of people. Some of these conventions may be arbitrary, but are none the less useful for that.

10.3.1 Format

See a sample procedure heading . Note:-

- White space of two lines separating functions.
- The identifier of the function right-justified to make it easy to find when flicking through a listing
- The separate definitions for standard and old C.
- The macros PUBLIC and PRIVATE (in HTUtils.h) expand to null and to "static" respectively. They show that one has thought about whether visibility is required outside the module, and they get over the overloading of the keyword "static" in C. Use one or the other. (Use for top level variables too).

10.3.2 Entry and exit condidtions

It is most important to document the function as seen by the rest of the world (especially the caller). The most important aspects of the appearance of the function to the caller are the pre- and post-conditions.

The pre conditions include the value of the parameters and structures they point to. Both include any requirements on or changes to global data, the screen, disk files, etc.

Tim BL

10.3.3 Function Heading: dummy example

```
} /* previous_function() */


/* Scan a line scan_line()
** -----------
** On entry,
** l points to the zero-terminated line to be scanned
** On exit,
** *l The line has null termintors inserted after each
**  word found.
** return value is the number of words found, or -1 if error.
** lines This global value is incremented.
*/
PRIVATE int scan_line ARGS1(const char *, l);
{
/* Code here */


} /* scan_line() */
```

Tim BL

10.4 Function body layout

With the body of functions, this is the way we aim to do it...we're not religious about it, but consistency helps. If you think your way is smarter, you may be right but this is the way we do it.

Whatever you do, NEVER make global changes to the indentation etc in a file without agreement from me first. (It screws up diffs and cde management as well as people).

10.4.1 Indentation

- Put opening { at the end of the same line as the if, while, etc which affects the block;
- Align the closing brace with the START of that opening line;
- Indent everything between { and } by an extra 4 (FOUR) spaces.
- Never indent preprocessor instructions (#ifdef, etc), nor increase the indentation level on account of preprocessor instructions.
- Block comments should start with /* in column 1, end end with */ in column 1. In between, having ** in column 1 is conventional but not necessary for very large comments.
- Comment the closing braces of conditionals and other blocks with the type of block, including the correct sense of the condition of the block being closed if there was an "else", of the function name. For example,

```
    if (cb[k]==0) { /* if black */
foo = bar;
    } else { /* if white */
foo = foobar;
    } /* if white */
}  /* switch on character */
```

```
    }  /* loop on lines */
} /* scan_lines() */
```

10.5 Identifiers

When chosing identifier names,

- Macros should be un upper case entirely unless they mimic and replace a genuine function.
- External names should be prefixed with HT to avoid confusion with other projects' code. Wthin the rest of the identifier, we use initial capitals a la Objective-C (e.g. HTSendBuffer).
- The macro SHORT_NAMES is defined on systems in which external names must be unique to within 8 characters (case insesitive). If your names would clash, at the top of the .h file for a module you should include macros defining distinct short names:

```
#ifdef SHORT_NAMES
#define HTSendBufferHeader HTSeBuHe
#define HTSendBuffer HTSeBuff
#endif
```

(back to Overview)

10.6 Directory structure

This is an outline of the directory structure used to support multiple platforms.

- All code is under a subdirectory "Implementation" at the appropriate point in the tree.
- All object files are in a subdirectory Implementation/xxx where xxx is the machine name. See for example WWW/LineMode/Implementation/*.
- Makefiles in the system-specific directories incldue a CommonMakefile which is in the parent Implementation directory (..).

10.7 Include Files

10.7.1 Module include files

Every module in the project should have a C #incldue file defining its interface, and a .c source file (of the same name apart from the suffix) containing the implementation.

The .c file should #include its own .h file.

A .h file should be protected so that no errors occur if it is #included twice.

An interface which relies on other interfaces should #include those interface files. An implemention file which uses other modules should #include the .h file if it not already #included by its own .h file.

10.7.2 Common include files

These are all in the WWW/Implementation directory.

HTUtils.h	Definitions of macros like PUBLIC and PRIVATE and YES and NO. For use in all .c files.
tcp.h	All machine-dependent code for accesing TCP/IP channels and files. Also defines some machine-dependent bits like SHORT_NAMES.
WWW.h	Project-wide definition of constants, etc.

(See also: Style in general , directory structure)

10.8 Macros used in W3 code

These are macros which make it less tedious to program in portable C. They are defined in HTUtils.h . See Language features .

10.8.1 ARGS1

Generates the format parameter list for a function heading. Expands to either an ANSI or non-ANSI argument declaration list. Note in order to do this, it has to take one argument for the type and one for the argument name. This is messy but necessary. See example, also NOARGS, ARGSn, PARAMS .

10.8.2 ARGSn

As ARGS1 , macros ARGS2, ARGS3 etc up to ARGS9 generate arguments lists for function headings with different numbers of arguments. See also NOARGS, PARAMS .

These should be used for private function definitions. If the function is PUBLIC (not static) then there will already be a function prototype in the header file, so a K&R style function definition may be used instead of ARG

10.8.3 CONST

This is used instead of the keyword "const" which does not exist in K&R C. It expands to const when using ANSI C, otherwise nothing.

10.8.4 NOARGS

Generates the argument list for a function header with no arguments.. See also ARGSn, NOPARAMS.

10.8.5 NOPARAMS

This is used to generate the formal parameter list in a function declaration having no parameters. It expands to (void) for ANSI C and () for non-ANSI. See also: PARAMS

10.8.6 PARAMS

Define formal parameter list in function declaration. The normal ANSI function header is given enclosed in DOUBLE parentheses. This causes the whole list to be replaced with just () when a non-ANSI compiler is used. See also: NOPARAMS

10.8.7 PRIVATE

This expands to "static". It is used in a function or variable declaration to signify that the function is private to the module, rather than public (the default). Avoids the confusion in C between "static" used in this sense, and used to declare storage variables within blocks as static as opposed to automatic.

10.8.8 PUBLIC

The alternative to PRIVATE. expands to nothing. One or other of PRIVATE or PUBLIC should be used to ensure that the programmer has thought about it. This prevents unnecessry accidental public definitions which may later clash with other people's routines.

10.8.9 StrAllocCopy

This is a safe string assignment function. It takes as arguments two pointers. If the first is non-zero, it frees the block it points to. It then allocates a block of size sufficient for the zero-terminated string pointed to by the second, and assigns the first pointer to point to that block. Implemented nowadays in terms of a function. Note that unlike the function, StrAllocCopy takes its first pointer by name not address (p, not).

(Part of style guide) Tim BL

10.9 Use of CVS

See also:

- History of introducing CVS in WWW code management
- Short Description of CVS
- CVS Manual , commands

10.9.1 WWW installation

The directory CVSROOT is where CVS stores its bookkeeping files.

The directory CVSRepository is where CVS stores the sources in the RCS format, i.e. one file for all versions and branches of each source file.

Please do not touch CVSROOT nor CVSRepository, they are to be maintained by use of CVS commands only.

10.9.2 Your .cshrc file

Code management in WWW is done with tools to be found in the directory hypertext/WWW/CVS-RCS.

The CVS repository for product xxx (xxx=Daemon, Library, LineMode, etc) is in WWW/xxx/Repository. Set envirnment variables to point to the repository and binary directories, for example:

```
setenv RCSBIN  $(HOME)/hypertext/WWW/CVS-RCS/next
setenv PATH ( $PATH $RCSBIN )
setenv CVSROOT $(HOME)/hypertext/WWW/CVS
setenv CVSREAD YES
```

You must also be a member of group www (gid=69) and have your umask set so that other group members will be able to write to files you create:

```
umask 2
```

10.9.3 Changing a module

This assumes you have NFS access to the WWW source tree.

Get the source

First of all, make yourself a working directory to be equivalent to "WWW" in the main tree. Call it WWW somewhere else, or WWW-joe for example. In that directory, if (say) you want to develop the Library code, do

```
cvs get Library
```

This will build you the sources. Alternatives are Line Mode and Daemon .

- Work on the sources.
- Test them.

Synchronise with other mods

Pick up any changes others have made using, in the Library/Implementation directory, the command

```
cvs update
```

Fix clashes

It is possible but surprisingly unlikely that someone else has been changing the same part of the same file and RCS can't figure out what the result should be. In this case, the file is flagged with big marks in it and a copy of both orginal files are left in your directory. Figure out what to do about it to combine your mods with the other guy's before proceeding.

Commit your work

```
cvs commit -m "Message describing changes and tested status"
```

while in the same (Implementation) directory. Your work will now be picked up by others working on the project.

10.9.4 Setting up a new module

If I remember rightly the sequence was to make a new subproduct xxx as follows:-

```
cd $CVSROOT/CVSROOT
```

Modify the "modules/modules" file under that directory to add your module just like the others

```
cvs commit modules
```

Under WWW, make a directories WWW/xxx and WWW/Implementation.

```
mkdir WWW/xxx
mkdir WWW/xxx/implementation
cd WWW/xxx/Implementation
```

Put the files in here -- actually they could be anywhere. Make sure the current directory has exactly and only the files you want to import. CVS will run through subdirectories recusively ok.

```
cvs import  xxx/Repository vendortag start
```

Note that the repository is relative to CVSROOT, not the pwd!!. The vendortag is for teh whole branch of the product, the release tag (start) anything but having no dots etc.

```
cd ../..        (ie back up to WWW)
cvs get xxx
cd xxx/Implementation
```

10.9.5 CVS manual

The CVS and RCS manuals are in the web.

(Note: Terry Hung introduced it at SLAC, but I believe we got it from the "standard" sources: prep.ai.mit.edu.)

RC