Contents

Chapter 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 freely available to anyone. 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.

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.

Making a web is as simple as writing a few SGML files which point to your existing data. Making it public involves running the FTP or HTTP daemon , and making at least one link into your web from another. In fact, any file available by anonymous FTP can be immediately linked into a 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 Digital's VMS/Help, Technical Univerity of Graz's ”Hyper-G”, and Thinking Machine's ”W.A.I.S.” systems.

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.

Try it

You can try the simple line mode browser by telnetting to info.cern.ch with user name "www" (no password). You can also find out more about WWW in this way.

It is much more efficient to install the browser on your own machine. The line mode browser is currently available in source form by anonymous FTP from node info.cern.ch [currently 128.141.201.74] as

```
/pub/WWWLineMode_v.vv.tar.Z.
```

(v.vv is the version number - take the latest.) ¡P¿ Also available is a hypertext editor for the NeXT using the NeXTStep graphical user interface in file

```
/pub/WWWNeXTStepEditor_v.vv.tar.Z
```

and a skeleton server daemon, available as

```
/pub/WWWDaemon_v.vv.tar.Z
```

Documentation is readable using www. A plain text version of the installation instructions is included in the tar file.

____ 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. Address: 1211 Geneva 23, Switzerland.

1.2.1 Eelco van Asperen

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

1.2.2 Carl Barker

Carl is at CERN for a six month period during his degree course at Brunell Univerity, UK. Carl will be working on the server sode, possibly on client authentication. Tel: 8265. Email: barker@cernnext.cern.ch

1.2.3 Tim Berners-Lee

Currently in CN division. Before comming to CERN, Tim worked on, among other things, document production and text processing. He developped 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

1.2.4 Robert Cailliau

Currently in ECP division, Programming Techniques group. Robert has been interested in document production since 1975. He ran the 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 contributing browser software for the Macintosh platform, and will be analysing the needs of physics experiments for online data access. Phone: 5005(office), 4646 (Lab). Email: cailliau@cernvm.cern.ch

1.2.5 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.

1.2.6 "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.

1.2.7 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 berowser/editor. Dave also has a strong interest in server technology and intelligent information retrieval algorithms.

1.2.8 Karin Gieselmann

With experience as librarian of the "FIND" database on cernvm, interfacing with authors and readers, Karin has volunteered to be a "hyperlibrarian" and look after the content of hypertext databases. Email: Karin@cernvm.cern.ch

1.2.9 Jean-Francois Groff

Provided some useful input in the "design issues". Currently in ECP/DS as "cooperant", J-F joined the project in September 1991. He wrote the gateway to the VMS Help system , and is looking at new browsers (X-Windows, emacs) and integration of new data sources. Jean-Francois is also working on the underlying browser architecure. Phone: 3755, Email: jfg@cernvax.cern.ch

1.2.10 Willem von Leeuwen

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

1.2.11 Nicola Pellow

Nicola joined the project in November 1990. She is a student at Leicester Polytechnic, UK, and left CERN at the end of August 1991. She wrote the original line mode browser .

1.2.12 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

1.2.13 Pei Wei

Pei is the author of "Viola", a hypertext browser, and the ViolaWWW variant which is a WWW browser. He is at the University of Califorfornia

1.3 Policy

This outlines the policy of the W3 project 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 a very wide range of 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 write, and additional data to be incorporated into the "web". We have already been fortunate enough to have several 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. In practice the interpretation of this in the case of the W3 project is that the programs are freely available to academic bodies. 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.

Where CERN code is based on public domain code, that code is also public domain.

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

____ Tim BL

1.3.4 WorldWideWeb 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.

Archive Directory structure

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

src/WWWLineMode_v.vv.tar.Z	The Line mode browser - all source, and binaries for selected systems.
WWWLineModeDefaults.tar.Z	A subset of WWWLineMode_v.vv.tar.Z. Basic documentation, and our current home page.
src/WWWNextStepEditor.tar.Z	The Hypertext Browser/editor for the NeXT – source and binary.
src/WWWDaemon_v.vv.tar.Z	The HTTP daemon, and WWW-WAIS gateway programs.
doc/WWWBook.tar.Z	A snapshot of our internal documentation - we prefer you to access this on line – see warnings below.
bin/xxx/www	Executable binaries for system xxx

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).

NeXTStep Browser/Editor

The browser for the NeXT is those files contained in the application directory WWW/Next/Implementation/WorldWideWeb.app and is compiled.Whe 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.

Line Mode browser

Binaries of this for some systems are in subdirectories of /pub/www/bin. If the binary exists for your system, take that and also the /pub/www/WWWLineModeDefaults.tar.Z. Unwrap the documentation, and put (link) its directory into /usr/local/lib/WWW on your machine. Put the www executable into your path somewhere, and away you go.

If no binary exists, procede as follows. Take the source tar file WWWLineMode_v.vv.tar.Z , uncompress and untar it. You will then find the line Mode browser in WWW/LineMode/Implementation/... (See Installation notes)

Subdirectories to that directory contain Makefiles for systems to which we have already ported. If your system is not among them, make a new subdirectory with the system name, and copy the Makefile from an existing one. Change the directory names as needed. PLEASE INFORM US OF THE CHANGES WHEN YOU HAVE DONE THE PORT. This is a condition of your use of this code, and will save others repeating your work, and save you repeating it in future releases.

Whe you install the browsers, you may want to configure the 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.

Documentation

The archive /pub/www/doc/WWWBook.tar.Z is an extract of the text from the WorldWideWeb (WWW) project documentation.

This is a snapshot of a changing hypertext system. The text is provided as example hypertext only, not for general distribution. The accuracy of any information is not guaranteed, and no responsibility will be accepted by the authors for any loss or damage due to inaccuracy or omission. A copy of the documentation is inevitably out of date, and may be inconsistent. There are links to information which is not provided in that tar file. If any of these facts cause a problem, you should access the original master data over the network using www, or mail us.

Servers

The Daemon tar file contains (in this release) the code for the basic HTTP daemon for serving files, and also for the WWW-WAIS gateway. To compile the WAIS gateway, you will need [a link to] a WAIS distribution at the same level as the WWW directory.

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

1.3.5 Copyright CERN 1990-1992

The information (of all forms) in these directories is the intellectual property of the European Particle Physics Laboratory (known as CERN). It is freely availble for non-commercial use in collaborating non-military academic institutes. Commercial organisations wishing to use this code should apply to CERN for conditions. Any modifications, improvements or extensions made to this code, or ports to other systems, must be made available under the same terms.

No guarantee whatsoever is provided by CERN. No liability whatsoever is accepted for any loss or damage of any kind resulting from any defect or inaccuracy in this information or code.

Tim Berners-Lee

CERN

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

1.4 History to date

A few steps to date in the WWW project history are as follows:

March 1989	First project proposal written and circulated for comment (TBL) . Paper "HyperText and CERN" (in ASCII or WriteNow format) produced as background.
October 1990	Project proposal reformulated with encouragement from CN and ECP divisional management. RC is co-author.
November 1990	Initial WorldWideWeb prototype developped on the NeXT (TBL) .
November 1990	Nicola Pellow joins and starts work on the line-mode browser . Bernd Pollermann helps get interface to CERNVM "FIND" index running. TBL gives a colloquium on hypertext in general.
Christmas 1990	Line mode and NeXTStep browsers demonstrable. Acces is possible to hypertext files, CERNVM "FIND", and internet news articles.
Febraury 1991	workplan for the purposes of ECP division.
26 February 1991	Presentation of the project to the ECP group.
March 1991	Line mode browser (www) released to limited audience on priam vax, rs6000, sun4.
May 1991	Workplan produced for CN/AS group

17 May 1991	Presentation to C5 committee. General release of www on central CERN machines.
12 June 1991	CERN Computer Seminar on WWW
August 1991	Files available on the net, posted on alt.hypertext (6, 16, 19th Aug), comp.sys.next (20th), comp.text.sgml and comp.mail.multi-media (22nd).
October 1991	VMS/HELP and WAIS gateways installed. Mailing lists www-interest and www-talk@info.cern.ch mailing lists started. One year status report. Anonymous telnet service started.
December 1991	Presented poster and demonstration at HT91 . W3 browser installed on VM/CMS. CERN computer newsletter announces W3 to the HEP world.
15 January 1992	Line mode browser release 1.1 available by anonymous FTP. See news . Presentation to AI-HEP'92 at La Londe.
12 February 1992	Line mode v 1.2 annouced on alt.hypertext, comp.infosystems, comp.mail.multi-media, cern.sting, comp.archives.admin, and mailing lists.

Chapter 2

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 .
Suggest someone else does	Maybe you know a system which it would be neat to have on the web. How about suggesting 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 UDI to allow w3 users to pick them straight up.

____ Tim BL

2.1 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?

2.1.1 You have a few files

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

2.1.2 You have a NeXT

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. 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.

2.1.3 Using a shell script

An HTTP daemon 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. More details ...

2.1.4 You have many files

In this case, for speed of access, the HTTP daemon will probably be best. You can write a tree of hypertext in HTML linking the text files, or you can even generate the tree automatically from your directory tree. 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 .

2.1.5 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.

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 . You can also write one from scratch, for example in perl. An advantage of a gateway is that you can maintain your existing procedures for creating text and managing the database. See: Tips for server writers.

-- Tim BL

2.2 Etiquette

There are a few conventions which will make for a more useable, less confusing, web.

2.2.1 Sign it!

An important aspect of information which helps keep it up to date is that one can trace its author. Doing this with hypertext is easy – all you have to do is put a link to a page about the author (or simply to the author's phone book entry).

Make a page for yourself with your mail address and phone number. At the bottom of files for which you are responsible, put a small note – say just your initials – and link it to that page. The address style (right justified) is useful for this.

Your author page is also a convenient place to put and disclaimers, copyright noitices, etc which law or convention require. It saves cluttering up the mesages themselves with a long signature.

If you are using the NeXT hypertext editor, then you can put this link from your default blank page so that it turns up on the bottom of each new document.

2.2.2 Give the status of the information

Some information is definitive, some is hastily put together and incomplete. Both are useful to readers, so do not be shy to put information up which is incomplete or out of date – it may be the best there is. However, do remember to state what the status is. When was it last updated? Is it complete? What is its scope? For a phone book for example, what set of people are in it?

2.2.3 Refer back

You may create some data as part of an information tree, but others may may make links to it from other places. Don't make assumptions about what people have just read. Make links from your data back to more general data, so that if people have jumped in there, and at first they don't undertand what it's all about, they can pick up some background information to get the context.

2.2.4 A root 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. I suggest you put a "map" 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. ____ Tim BL

2.3 Things to be done

There are many of these ...if you have amoment, take your pick! There are also special lists of things to do for the line mode browser , the NeXT browser , and the daemon .

2.3.1 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 ("knowbots?") 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. (html to tex?)

2.3.2 Server side

Server upgrade	Easier to install, porrt. Export directories as hypertext. Run shell scripts embedded in the directory for virtual documents and searches.
More Servers	See the list of things we have thought of or been pointed to.
WAIS integration	WAIS protocol extensions tro allow hypertext; HTML data type, docids to be conforming UDIs. Integrate WAIS in client too.
Integrate client and server	A client which generates HTML becomes a general purpose gateway. Especially useful for sites where general access to news, external internet, etc, is limited.

2.3.3 Other

Mail server	Update listserv to supply www documents from UDI, including at the bottom a list of references with their UDIs.
Gateways	JANET and DECnet for example. Real need.
HTTP enhancements	Format conversion, authorization, better logging information for statistics.

———————————————————— Tim BL

Chapter 3

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 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, know-bots etc., format conversion , gateways.

3.1 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.

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

3.2 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 PC with graphics

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

3.3 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.

3.3.1 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 .

3.3.2 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.

3.3.3 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.

3.3.4 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 ").

See also: HyperText and Information Retrieval

3.3.5 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.

3.3.6 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

3.3.7 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

3.4 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

3.5 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

3.6 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.

3.6.1 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.

3.6.2 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.

3.6.3 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

3.6.4 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.

3.6.5 Annotation

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

3.7 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.

3.8 Topology

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

3.8.1 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.

3.8.2 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.

3.8.3 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.

3.8.4 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 .

3.8.5 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.

3.8.6 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

3.9 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

3.9.1 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) They might include:

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.

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

3.10 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.

3.10.1 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 "123715924234624423423234234242346876234236 8" 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.

3.10.2 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.

3.10.3 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

3.11 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.

3.11.1 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:

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).

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.

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.

3.11.2 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,
- 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

3.12 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

3.12.1 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)________________________________

3.13 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
```

Chapter 4

Relevant protocols

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

4.1 File Transfer

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

4.2 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. (See news address syntax)

4.3 Search and Retrieve

The WWW project defines its own protocol for information transfer, which allows for negotiation on representation. This we call HTTP, for HyperText Transfer Protocol .See also HyperText Transfer Format , and the HTTP address syntax)

4.4

Whilst the HTTP protocol provides an index search function, another common protocol for index search is Z39.50, and the version of it

4.5 HTTP as implemented in WWW

This document defines the Hypertext Transfer protocol (HTTP) as currently implemented by the WorldWideWeb initaitive software. This is a subset of the proposed full HTTP protocol. No client profile information is transferred with the query. Future HTTP protocols will be back-compatible with this protocol.

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 .

4.5.1 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.

During development, the default HTTP TCP port number is 2784 – this will change when an official port number is allocated.

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 repose may be many.

4.5.2 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 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 .

4.5.3 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.

4.5.4 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 will not record any error condidtion.

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

____ Tim BL

4.6 HyperText Transfer Protocol

See also: Why a new protocol? , Other protocols used

This is a list of the choices made and features needed in a hypertext transfer protocol. See also the HTTP protocol as currently implemented .

4.6.1 Underlying protocol

There are various distinct possible bases for the protocol - we can choose

- Something based on, and looking like, an Internet protocol. This has the advantage of being well understood, of existing implementations being all over the place. It also leaves open the possibility of a universal FTP/HTTP or NNTP/HTTP server. This is the case for the current HTTP.
- Something based on an RPC standard. This has the advantage of making it easy to generate the code, that the parsing of the messages is done automatically, and that the transfer of binary data is efficient. It has the disadvantage that one needs the RPC code to be available on all platforms. One would have to chose one (or more) styles of RPC. Another disadvantage may be that existing RPC systems are not efficient at transferring large quantities of text over a stream protocol unless (like DD-OC-RPC) one has a let-out and can access the socket directly.
- Something based on the OSI stack, as is Z39.50. This would have to be run over TCP in the internet world.

Current HTTP uses the first alternative, 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.

4.6.2 Idempotent ?

Another choice is whether to make the protocol idempotent or not. That is, does the server need to keep any state informat about the client? (For example, the NFS protocol is idempotent, but the FTP and NNTP protocols are not.) In the case of FTP the state information consists of authorisation, which is not trvial to establish every time but could be, and current directory and transfer mode which are basically trivial. The propsed protocol IS idempotent.

This causes, in principle, a problem when trying to map a non-dempotent system (such as library search systems which stored "result sets" on behalf of the client) into the web. The problem is that to use them in an idempotent way requires the re-evaluation of the intermediate result sets at each query. This can be solved by the gateway intelligently caching result sets for a reasonable time.

4.6.3 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	Please transfer a named document back. Transfer the results back in a standard format or one which I have said I can accept. The reply includes the format. In practice, one may want to transfer the document over the same link (a la NNTP) or a different one (a la FTP). There are advantages in each technique. The use of the same link is standard, with moving to a different link by negociation (see PORT).
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 dat 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 foir 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.

4.6.4 Response

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

4.6.5 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.
Not Authorized	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.
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

4.6.6 Penalties

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 attept to estimate a penalty will therfore be very aproximate, and only useful for distinguishing widely differing cases. A suitable unit would be the proportion, betwen 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, necessaily 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 particlular 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 consant part (seconds) and the size-related part (seconds per byte).

4.7 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

Chapter 5

W3 Naming Schemes

(See also: a discussion of design issues involved , BNF syntax , W3 background)

The format of a hypertext name consists of the name of the naming subscheme 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 .

5.1 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.

5.2 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.
x500	Format to be defined.

Systems (such as WAIS) which are not currently 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 (in principle) be configured to automaticaly use certain gateways for certain addressing schemes.

This will allow, for example, simple PC-based clients to follow links

5.3 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 keuwordlist, 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".

5.3.1 Example:

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

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

5.4 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.

5.4.1 Examples

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

This is a fully qualified file name.

```
fred.html
```

This ¡A NAME=0 HREF=Relative.html¿relative name¡/A¿ , used within a file, will refer to a file of the same node and directory as that file, but the name fred.html.

5.4.2 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*.

5.5 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.

5.5.1 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.

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

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

5.6 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 the 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

5.7 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

5.8 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

5.9 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 — gopheraddress — waisaddress
httpaddress	h t t p : / / hostport [/ path] [? search]
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 — $ — _ — @ — ! — % — ˆ & — * — (—) — . — 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

5.10 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

5.11 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.

—————————————————————————— Tim BL

5.12 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 .

Chapter 6

HTML

The WWW system uses marked-up text to represent a hypertext document for transmision over the network. The hypertext mark-up language is an SGML format. HTML parsers should ignore tags which they do not understand, and ignore attributes which they do not understand of tags which they do understand.

To find out how to write HTML, or to write a program to generate it, read:

The tags	A list of the tags used in HTML with their significance.
Example	A file containing a variety of tags used for test purposes, and its source text .

You can use th line-mode browser to get the source text (-source option) of an html document you find on the network, so you can use any existing documents as examples.

6.1 Default text

Unless otherwise defined by tags, text is transmitted as a stream of lines. The division of the stream of characters into lines is arbitrary, and only made in order to allow the text to be passed through systems which can only handle text with a limited line length. The recommended line length for transmission is 80 characters. The division into lines has no significance (except in the case of example sections and PLAINTEXT) apart from indicating a word end. Line breaks between tags have

6.2 HTML Tags

This is a list of tags used in the HTML language. Each tag starts with a tag opener (a less than sign) and ends with a tag closer (a greater than sign). Many tags have corresponding closing tags which identical except for a slash after the tag opener. (For example, the TITLE tag).

Some tags take parameters, called attributes. The attributes are given after the tag, separated by spaces. Certain attributes have an effect simply by their presence, others are followed by an equals sign and a value. (See the Anchor tag, for example). The names of tags and attributes are not case sensitive: they may be in lower, upper, or mixed case with exactly the same meaning. (In this document they are generally represented in upper case.)

Currently HTML documents are transmitted without the normal SGML framing tags, but if these are included parsers will ignore them.

6.2.1 Title

The title of a document is given between title tags:

```
<TITLE> ... </TITLE>
```

The text between the opening and the closing tags is a title for the hypertext node. There should only be one title in any node. It should identify the content of the node in a fairly wide context, and should ideally fit on one line.

The title is not strictly part of the text of the document, but is an attribute of the node. It may not contain anchors, paragraph marks, or highlighting. the title may be used to identify the node in a history list, to label the window displaying the node, etc. It is not normally displayed in the text of a document itself. Contrast titles with headings .

6.2.2 Next ID

This tag takes a single attribute which is the number of the next document-wide numeric identifier to be allocated (not good SGML). Note that when modifying a document, old anchor ids should not be reused, as there may be references stored elsewhere which point to them. This is read and generated by hypertext editors. Human writers of HTML usually use mnemonic alpha identifiers. Browser software may ignore this tag. Example of use:

```
<NEXTID 27>
```

6.2.3 Base Address

Anchors specify addresses of other documents, in a from relative to the address of the current document. Normally, the address of a document is known to the browser because it was used to access the document. However, is a document is mailed, or is somehow visible with more than one address (for example, via its filename and also via its library name server catalogue number), then the browser needs to know the base address in order to correctly deduce external document addresses.

The format of this tag is not yet specified. NOT CURRENTLY USED

6.2.4 Anchors

The format of an anchor is as follows:

```
<A NAME=xxx HREF=XXX> ... </A>
```

The text between the opening tag and the closing tag is either the start or destination (or both) of a link. Attributes of the anchor tag are as follows.

HREF	If the HREF attribute is present, the anchor is senstive text: the start of a link. If the reader selects this text, he should be presented with another document whose network address is defined by the value of the HREF attribute . The format of the network address is specified elsewhere . This allows for the form HREF=#identifier to refer to another anchor in the same document. If the anchor is in another document, the atribute is a relative name , relative to the documents address (or specified base address if any).
NAME	The attribute NAME allows the anchor to be the destination of a link. The value of the parameter is that part of a hypertext address which follows the hash sign .
TYPE	An attribute TYPE may give the relationship described by the hyertext link. The type is expressed by a string for extensibility. Strings for types with particular semantics will be registered by the W3 team. The default relationship if none other is given is void.

All attributes are optional, although one of NAME and HREF is necessary for the anchor to be useful.

6.2.5 IsIndex

This tag informs the reader that the document is an index document. As well as reading it, the reader may use a keyword search.

Format:

```
<ISINDEX>
```

The node may be queried with a keyword search by suffixing the node address with a question mark, followed by a list of keywords separated by plus signs. See the network address format .

6.2.6 Plaintext

This tag indicates that all following text is to be taken litterally, up to the end of the file. Plain text is designed to be represented in the same way as example XMP text, with fixed width character and significant line breaks. Format:

```
<PLAINTEXT>
```

This tag allows the rest of a file to be read efficiently without parsing. Its presence is an optimisation. There is no closing tag.

6.2.7 Example sections

These styles allow text of fixed-width characters to be embedded absolutely as is into the document. The format is:

```
<LISTING>
...
</LISTING>
```

The text between these tags is to be portrayed in a fixed width font, so that any formatting done by character spacing on successive lines will be maintained. Between the opening and closing tags:

- The text may contain any ISO Latin printable characters, including the tag opener, so long as it does not contain the closing tag in full.

- Line boundaries are significant, and are to be interpreted as a move to the start of a new line.
- The ASCII Horizontal Tab (HT) character should be interpreted as the smallest positive nonzero number of spaces which will leave the number of characters so far on the line as a multiple of 8. Its use is not recommended however.

The LISTING tag is portrayed so that at least 132 characters will fit on a line. The XMP tag is portrayed in a font so that at least 80 characters will fit on a line but is otherwise identical to LISTING. The examples of markup are here given using the XMP tag.

6.2.8 Paragraph

This tag indicates a new paragraph. The exact representation of this (indentation, leading, etc) is not defined here, and may be a function of other tags, style sheets etc. The format is simply

```
<P>
```

(In SGML terms, paragraph elements are transmitted in minimised form).

6.2.9 Headings

Several levels (at least six) of heading are supported. Note that a hypertext document tends to need less levels of heading than a normal document whose only structure is given by the nesting of headings. H1 is the highest level of heading, and is recommened for the start of a hypertext node. It is suggested that the first heading be one suitable for a reader who is already browsing in related information, in contrast to the title tag which should identify the node in a wider context.

```
<H2>, <H3>, <H4>, <H5>, <H5>, <H6>
```

These tags are kept as defined in the CERN SGML guide. Their definition is completely historical, deriving from the AAP tag set. A difference is that HTML documents allow headings to be terminated by closing tags:

```
<H3>Second level heading</h2>
```

6.2.10 Address

This tag is for address information, signatures, etc, normally at the top or bottom of a document. typically, it is italic and/or right justified or indented. The format is:

```
<ADDRESS> text ... </ADDRESS>
```

6.2.11 Highlighting

The highlighted phrase tags may occur in normal text, and may be nested. For each opening tag there must follow a corresponding closing tag. NOT CURRENTLY USED.

```
<HP1>...</HP1>   <HP2>... </HP2> etc.
```

6.2.12 Glossaries

A glosary (or definition list) is a list of paragraphs each of which has a short title alongside it. Apart from glossaries, this format is useful for presenting a set of named elements to the reader. The format is as follows:

```
<DL>
<DT>Term<DD>definition pagagraph
<DT>Term2<DD>Definition of term2
</DL>
```

6.2.13 Lists

A list is a sequence of paragraphs, each of which is preceded by a special mark or sequence number. The format is:

```
<UL>
<LI> list element
<LI> another list element ...
</UL>
```

The opening list tag must be immediately followed by the first list element. The representation of the list is not defined here, but a bulleted list for unordered lists, and a sequence of numbered paragraphs for an ordered list would be quite appropriate. Other possibilities for interactive display include embedded scrollable browse panels.

Opening list tags are:

UL	A list multi-line paragraphs, typically separated by some white space.
MENU	A list of smaller paragraphs. Typically one line per item, with a style more compact than UL.
DIR	A list of short elements, less than one line. Typical style is to arrange in four columns or provide a browser, etc.

6.3 SGML

The "Standard Generalised Mark-up Language" 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.

6.3.1 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.

6.3.2 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.

6.3.3 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.

See also: HyTime , HTML , Hypertext Document formats . ______________________________ Tim BL

6.3.4 AAP

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

Chapter 7

Coding Style Guide

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.)

The style guide is divided into sections on Language features , Macros , Module header , Function header , Code style , Identifiers , Include files , Directory structure .

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

7.1 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 compiled 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 condidtion in a comment. Don't put it as code - it won't pass all compilers.
const	This keyword does not exist in K&R C, s use the macro CONST which expands to "const" under standard C and nothing otherwise. – See HTUtils.h

(part of: style guide) ____

7.2 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

7.3 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.

7.3.1 Format

See a sample procedure heading . Note:-

- White space of two lines separating functions.
- The name 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).

7.3.2 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-condidtions.

The pre condidtions 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.

7.3.3 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

7.4 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.

7.4.1 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 spaces.
- 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() */
```

Tim BL

7.5 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 ¡A NAME=1 HREF=Coding.html¿Overview¡/A¿)¡P¿

7.6 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 (..).

7.7 Include Files

7.7.1 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 itse own .h file.

A .h file should be protected so that errors do not result if it is #included twice.

An interface which relies on other intrefaces 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.

7.7.2 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) Tim BL