An Information Provider's Guide to Web Servers

Nathan Torkington

September 16, 1993

Introduction

This document is an introduction to the programs that provide information on the World Wide Web. It is not an introduction to the Web — see the parallel document "World Wide Web Primer" for this (see the section "How to obtain this document" for instructions on obtaining it). It describes the current HTTP servers and their relative features, as well as discussing whether an HTTP is even necessary.

This document is available on the Web, as well as being posted fortnightly to the Usenet newsgroups **comp.infosystems.www**, **alt.hypertext**, **news.answers**, **comp.answers** and **alt.answers**. It is available as LaTeX, plain ASCII, DVI and Postscript files via anonymous ftp. For instructions on retrieving the latest version of this document, consult the last section, called "How to obtain this document".

This document was last revised on *Wed Sep 15 15:26:45 NZT 1993* by *Nathan Torkington*.

Table of Contents

5. CERN server
6. NCSA server
7. Plexus server
8. Obtaining This Document

Hypermedia or Not?

You will probably want to provide information in a hypermedia format (HTML) rather than plain text, because of the power of HTML. Not only can you represent plain text in HTML, but you can also represent gopher-like menus, true hypertext (where certain words in a paragraph can bring up other pages), and hypermedia (where images and audio can be the destination of a hyper-link). If you only want to provide plain text and menus, you might want to try something like gopher (which is accessable by Web browsers).

Running an HTTP isn't the only way to put hypermedia on the Web (browsers can access FTP sites and gopher servers). This is because (on the Web) the protocol and the data format are different — you can provide both hypermedia and plain text via FTP and HTTP. Knowing this, you can decide how you want to provide information.

FTP servers have the benefit that they may already be set up at your site, they have reasonable logging, and automatic filename indexing (via archie). There is a lot of behind-the-scenes overhead for the browser programs in obtaining files via FTP, however (the anonymous user and password need to be sent each time a file is retrieved). Because hypertext often consists of little files, with lots of links, this overhead may prove the deciding factor against using an FTP server.

Gopher also has automatic title indexing (Veronica), and a fairly simple setup struture (see Chapter *n*). Because gopher's HTML type isn't recognised by gopher clients, hypertext cannot be easily served through gopher and for this reason I recommend setting up a gopher server only if you don't need to serve hypertext.

HTTP servers are geared toward hypertext, and because plain text is a degenerate case of hypertext they do equally well at serving plain text. There are three main HTTP servers in use, and all three are briefly described below.

Load generated by Hypermedia Servers

The load generated by the servers varies, depending on the task requested by the browser. Resource intensive tasks such as searching files, translating between data formats, or starting other programs, will cause a larger load than simple document delivery. In general, a well-used server such as that run by NCSA or CERN, should sit on a devoted low-to-mid-range machine, whereas less-used servers can exist quite happily on a multiuser machine.

CERN Server

CERN is a high-energy physics organisation, based in Switzerland. They started the World Wide Web project, and provided much of the initial software that helped it gain acceptance. E-mail regarding the server should be sent to `www-bugs@info.cern.ch`.

Features

Remapping of requests This enables requests for files to remapped onto requests for other files, not necessarily on the same server. For instance, I can specify in my rule file that requests for `/cern/*` can be remapped into requests for `http://info.cern.ch/*` and the server will remap requests anything in the `/cern/` directory to requests for files from a machine in Switzerland.

Mapping from filename suffix to file type You can specify rules to convert file suffixes (`.tex`, for instance) for instance, onto MIME types (`application/tex`, in this case).

HTTP/1.0 ability The initial, simple, implementation of HTTP had no way of specifying which data formats clients could cope with, which version of HTTP was being used, and no MIME typing. HTTP/1.0 is a version of HTTP which does provide all these features (and more).

Ability to act as a gateway to WAIS Using the remapping above, wais: queries can be passed on to other machines.

Ability to act as a gateway through a firewall Also implemented using the remapping feature.

Both standalone and inetd capability Being able to be run "standalone" means that you don't have to be superuser to use it. "inetd" is a Unix system utility that provides a nice interface between TCP ports and programs, but requires superuser access to add programs to.

Automatically presents directory listings nicely When a browser requests a directory, rather than a file, the server will produce a menu of the files in the directory.

Support for README files If a browser requests a directory, and there is a README file present, the server will prepend the README file to the directory listing.

Multiformat documents If you have the same document stored in multiple formats, the server will return a format that the browser can understand (if the browser is using the HTTP/1.0 protocol).

Logging For each request, the server logs the date, time, IP number of the machine originating the request, and the text of the request (without the HTTP/1.0 MIME information).

Access Control Simple user authentication and access control is new in this version.

The CERN server is available as ftp://info.cern.ch/pub/www/src/WWWLineMode_XXX.tar.Z where XXX is the latest version number. It requires the CERN WWW library, available as ftp://info.cern.ch/pub/www/src/WWWLibrary_XXX.tar.Z where XXX is the library version number.

It will compile automatically for most systems.

NCSA Server

NCSA, the National Centre for Supercomputing Applications, is based out of the University of Illinois at Urbana-Champaign, in the USA. They are responsible for providing the Mosaic series of browsers, and accelerating the acceptance of Web browsers. E-mail regarding the server should be sent to `httpd@ncsa.uiuc.edu`.

Features

Simple It consists of less than ten source-code files, and is easy to install because of it.

Can operate from a gopher setup The server will map the gopher `.cap` and `.links` files into menus, when directories are requested.

inetd and standalone support See the same section in the description of the CERN server.

Logging For each request, the server logs the hostname of the machine originating the connection, the date, time, and the request (without the HTTP/1.0 MIME information).

HTTP/1.0 ability See the same section in the description of the CERN server.

Because of its extreme simplicity, the NCSA server will compile readily on most systems. It is available in `ftp://ftp.ncsa.uiuc.edu/Web/ncsa_httpd/`. It is a good place to start if you are already running a gopher server.

Plexus

Plexus is written by Tony Sanders (`sanders@bsdi.com`) and is written in perl (an interpreted language, suitable for most text-processing and system management tasks).

Features

Written in perl Because perl is an interpreted language, there is no compilation step between changing the code and running it. Because of this, making changes to the code is quicker than changing (for instance) the CERN code.

Built-in setext, archie, calendar, manual page and finger gateways These provide excellent base services for a local server, as well as giving good indication on how to implement new gateways.

Easily extendable The code is exceptionally easy to understand and add to, and perl is not difficult to learn.

Access control on a per-directory basis You can deny or permit access to files in directories based on the IP address/hostname of the machine the browser is running from.

Recommended only as stand-alone Because the perl interpreter is rather large, it is not recommended that Plexus be run from inetd (which would run perl for each connection), although it does have inetd support if you really want to do this.

Logging For each request, Plexus logs the hostname of the machine originating the connection, the date and time, and the text of the request.

HTTP/1.0 ability See the same section in the description of the CERN server.

Perl is available in `ftp://ftp.uu.net/pub/languages/perl/`.

Configuring Plexus

(this section is for release 2 of Plexus. Release 3 will probably have a different system).

The configuration for Plexus is done in the file `plexus.conf`, and via environment variables. The environment variables are:

$HTTPD The directory base from which the server can serve files.

$HTTPD_CONF The configuration file (relative to the directory base).

$HTTPD_DEBUG Whether debugging should be turned on.

The variables to set in `plexus.conf` are:

$http_support This should be HTML that describes how to report errors. For instance, '<ADDRESS>www-admin@vuw.ac.nz</ADDRESS>.

$http_homepage The file (relative to the directory base above) that should be returned if the user requests `http://host/`

$http_index The filename in a directory that should be returned if the user requests `http://host/path/`

$http_log The filename to place log messages in. This does not need to be relative to the directory base.

$http_indexdirs Set to 1 if directories should be indexed, 0 otherwise.

$http_chroot If non-zero, the server should use the chroot() call.

Also inside `plexus.conf` are the mappings which decide which HTTP commands are understood. These look like: $method{'get'} = "do_get";

Any mapping commented out with a # at the start of the line, will not have the corresponding command recognised by the server. The mappings commented out in the distribution are the mappings which the server doesn't have code for (they are only included for completeness).

After the mappings in `plexus.conf` are the configuration options for the methods, the configuration options for the gateways, and the list of scripts to load. Gateways are implemented by mapping URLs like `http://host/specialstring/blah` into a request for `blah` from the gateway. These mappings are called translations, and are defined after the list of scripts to load.

The mapping from filename extensions to MIME content types is done through the list of assignments after the translations. These all look like $ext{'dvi'} = $ext{'DVI'} = 'application/dvi';

Similarly the MIME encoding definitions follow those for content types.

The remainder of `plexus.conf` is all internal to plexus and should not be changed.

How to obtain this document

The latest version of this document is always available on the Web as **http://www.vuw.ac.nz/non-local/gnat/www-servers.html**, and the most recently posted ASCII version will be available via anonymous FTP from rtfm.mit.edu in the directory /pub/usenet/news.answers/www as **servers**. The ASCII, LaTeX, DVI, and PostScript versions will be available via anonymous FTP from wuarchive.wustl.edu in the directory /doc/misc/www/.

This document is part of a series: "World Wide Web Primer", "An Information Provider's Guide to HTML", and "An Information Provider's Guide to Web Servers". The other documents in the series are available from the archives above.

Please send feedback to the author, Nathan Torkington, at the e-mail address Nathan.Torkington@vuw.ac.nz — all discussion will be treated as public domain and may be used in future versions of this document.