Ari Luotonen, CERN
Tim Berners-Lee, CERN

CERN httpd Reference Manual

A Guide To A World-Wide Web HyperText Daemon

CERN Server User Guide

Generated from the Hypertext

March 4, 1994

Table of Contents

1. CERN Server 2.16beta User Guide

If you are still using 2.14 or older, please upgrade (then see also 2.15beta release notes). If you insist on running the old version, or if you are a VMS user, refer to the old documentation. VMS port is on its way. 2.15beta documentaion is also still available.

For VMS users, see Foteos Macrides' <macrides@sci.wfeb.edu> modifications to VMS version 2.14.

CERN WWW Server (`httpd`, HyperText Transfer Protocol Daemon) is a full featured server for serving files, real or synthesized, running scripts to produce virtual documents, to handle clickable images, fill-out forms, and searches etc. CERN server can also be run as a gateway to allow people behind firewalls to use the Web as if the firewall was not present. **A powerful new feature is caching performed by the gateway.**

`httpd` is a general server program which runs WWW protocol, HTTP. This is a TCP/IP based protocol running by convention on port 80.

1.1 In This Guide...

Installation	The steps necessary to install CERN server.
Administration	How to set up document protection, index search, clickable images, server-side scripts, ...

1.2 About documents generated from hypertext

Paper manuals generated from hypertext are made for convenience, for example for reading when one has no computer to turn to. We have tried to make the hypertext into fairly conventional paper documents, but they may seem a little strange in some ways.

All the links have been removed. Therefore, it is worth looking at the table of contents to see what there is in the manual. Something which is not explained in place may be explained in detail elsewhere.

We have tried to keep related matter together, but sometimes necessarily you might have to check the table of contents to find it.

Please remember that these are for the most part "living documents". That is, they are constantly changing to reflect current knowledge. If you see a statement such as "Product xxx does not support this feature", remember that it was the case when the document was generated, and may not be the same now. So if in doubt, check the online version. Of course, the living document may be out of date too, in which case it is helpful to mail its author.

2. Installing CERN Server

2.1 Getting the Program

CERN server distribution is available from `info.cern.ch` anonymous ftp account. Often you don't need to compile the server yourself, precompiled versions can be found under directory `/pub/www/bin` (in the subdirectory corresponding your machine architecture).

If there is no precompiled version for your platform, of if it doesn't work (e.g. your machine doesn't use NIS, but the binary was linked with a library using NIS), you may need to get the source code. In that case you need both the WWW Common library and the Daemon distribution from the `/pub/www/src` directory. In this case, see compilation instructions.

2.2 Installation

In Unix you can run the server either as stand-alone, or from Internet Daemon `(inetd)`.

A stand-alone server is typically started once at system-boot time. It waits for incoming connections, and forks itself to serve a request. **This is much faster** than letting `inetd` spawn `httpd` every time a request comes. **We therefore recommend that you run CERN httpd in stand-alone mode.**

There are special instructions if you are installing under VMS.

2.2.1 Stand-alone Installation

A stand-alone server is started from the bootstrap command file (for example `/etc/rc.local)` so that it runs continuously like the `sendmail` daemon, for example.

This method has the advantage over using the `inetd` that the response time is reduced.

Add a line starting `httpd` to your system startup file (usually `/etc/rc.local` or `/etc/rc`). Specify the port with `-p` command line option:

```
/usr/etc/httpd -p 80 &
```

(Ampersand is no longer needed in versions 2.15 and newer - `httpd` goes background automatically when `-p` option is specified. Or a little more safely in case httpd is removed:

```
if [ -f /usr/etc/httpd ]; then
    (/usr/etc/httpd -p 80 && (echo -n ' httpd') ) & >/dev/console
fi
```

Naturally you can use any of the other command line options, too.

2.3 Registering Your Server

Once you have your `httpd` up and running, and you have documents to show the word, announce your server, so that others can find it.

2.4 If It Doesn't Work...

...first see the debugging chart and the FAQ. If you can't figure out what's going wrong, feel free to send mail to **httpd@info.cern.ch**

2.5 Compiling the CERN Server

- If you need to compile yourself, uncompress and untar cern_httpd.tar.Z distribution tar file.

```
uncompress cern_httpd.tar.Z
tar xvf cern_httpd.tar
```

- Go to newly-created `WWW` directory, and give command `BUILD`:

```
cd WWW
./BUILD
```

- Executable `httpd` appears in directory `.../WWW/Daemon/sun4` (if you have a Sun4 machine), or in another subdirectory corresponding to your machine architecture.

2.6 Installing httpd Under inetd

This is how to to set up `inetd` to run `httpd` whenever a request comes in. (These steps are the same for any daemon under unix: you will probably find a similar thing has been done for the FTP daemon, `ftpd,` for example.)

2.6.1 Step 1: Install httpd Binary

Copy `httpd` into a suitable directory such as `/usr/etc.` Make it owned by `root`, and make it writable only to `root,` for example by saying:

```
chmod 755 httpd
```

2.6.2 Step 2: Add http Service to /etc/services

Put "http" in the `/etc/services` file, or use the name of a specific service of your own if you want to use a special port number. Standard port number for HTTP is 80.

```
http    80/tcp          # WWW server
```

Exceptions:

- On a NeXT, see using the NetInfomanager
- On any machine running NIS (yellow pages), see specicial instructions.

2.6.3 Step 3: Add a Line to /etc/inetd.conf

Put a line in the internet daemon configuration file, `/etc/inetd.conf.`

```
http  stream  tcp  nowait  root  /usr/etc/httpd  httpd
```

First word is the same as in `/etc/services` file.

If you want to pass command line options or parameters to `httpd,` they would listed be in the end of line, for example to set the rule file to something else than the default `/etc/httpd.conf:`

```
http  stream  tcp  nowait  root  /usr/etc/httpd  httpd -r /my/own/rules
```

Note: For `httpd` version 2.15 and later we recommend that it is run as user `root`.

Running `httpd` as `root` is safe, since it automatically resets its user-id to `nobody`. However, if you decide to use access authorization features, and you need to serve protected files, `httpd` will have to be able to set its user-id to some other uid as well. In any case, `httpd` always sets its user-id to something other than `root` before serving the file to the client.

Note: `/etc/inetd.conf` syntax varies from system to system, for example all systems don't have the field specifying the user name, in which case the default is `root`. If in doubt, sopy the format of other lines in your existing `inetd.conf`.

Note: There seems to be a limit of 4 arguments passed across by `inetd`, at least on the NeXT.

2.6.4 Step 4: Send HUP Signal to inetd

When you have updated `inetd.conf`, find out the process number of `inetd`, and send a "HUP" signal to it.

For example on BSD unix do this:

```
> ps -aux | grep inetd | grep -v grep
root     85   0.0  0.9 1.24M  304K ?  S  0:01 /usr/etc/inetd
> kill -HUP 85
```

For system V, use `ps -el` instead of `ps -aux`.

Be aware that on some systems your local file /etc/services may not be consulted by your system (see notes on debugging).

2.6.5 Test It!

Test the server with the LineMode Browser giving its address explicitly:

```
www http://myhost.dom.ain/welcome.html
```

This assumes that you have a file "welcome.html" in your exported directory. If it doesn't work, you have probably missed something. See notes on debugging.

Naturally you can use whichever browser you prefer, but LineMode Browser gives more verbose explanations with `-v` command line option.

2.6.6 Using NIS (Yellow Pages)

If your machine is running Sun's "Network Information Service", originally know as "yellow pages", read this.

You must:

- First make an addition to the `/etc/services` file just as for a normal unix system.
- Then, change directory to `/var/yp` and run `make`.

This will load the `/etc/services` file info the NIS information system.

Some people have found that they needed to reboot he system afterward for the change to take effect.

2.6.7 Adding a Service on the NeXT

The NeXT uses the the "netinfo" database instead of the `/etc/services` file. This is managed with the `/NextAdmin/NetInforManager` application. Here's how to add the service `http`:

- Start the NetInfomanager by double-clicking on its icon.
- If you are operating in a cluster, open either your local domain `(/hostname)` or if you have authority, the whole cluster domain `(/)`. If you're not in a cluster, just use the domain you are presented with.
- Select `"services"` from the browser tree.
- Select `"ftp"` from the list of services.
- Select `"dupliacte"` from the edit menu.
- Select `"copy of ftp"` and double-click on its icon to get the property editor.
- Click on `"name"` and then on the value `"copy of ftp"`. Change this to `"http"` by typing "http" in the window at the botton, and hitting return.
- Click on `"port"`, and then on the value `21`. Change it to `80`.
- Use `"Directory:Save"` menu `(Command/s)` to save the result. You will have to give a root password or netinfo manager password.

2.7 Priviliged ports

The TCP/IP port numbers below 1024 are special in that normal users are not allowed to run servers on them. This is a security feaure, in that if you connect to a service on one of these ports you can be fairly sure that you have the real thing, and not a fake which some hacker has put up for you.

The normal port number for W3 servers is port 80. This number has been assigned to WWW by the Internet Assigned Numbers Authority, IANA.

When you run a server as a test from a non-priviliged account, you will normally test it on other ports, such as 2784, 5000, 8001 or 8080.

2.7.1 Under Unix

The Internet Daemon `inetd` (running as root) can listen for incomming conections on port 80 and pass them down to a process with a safer uid for the server itself. However, the `httpd` versions 2.14 and later can be safely run as `root` since they automatically change their user-id to `nobody` or some other user-id depending on server setup.

2.7.2 Under VMS

Under UCX, the process running as a server needs BYPASS privilege to listen to ports below 1024. This might mean you have to install the server. With other TCP/IP packages, privilege of some sort is similarly required.

2.8 Under VMS

The daemon runs just as under unix, for which the rest of the documentation was written. These instructions are my ideas about how to run it under VMS but it is a long time since I did anything like this, so please tell me what is wrong. We don't have effort available to distribute HTTPD.EXE, sorry.

Compilation of the daemon for VMS requires taking the library and Daemon source files from the unix release, copying them all onto the VMS system, compiling them all. The object files from the Library should go into a libwww.olb file. The object files from the Daemon should be linked together and with the libwww.olb library.

But check out what Mark Donszelmann has done.

When compiling the sources, you must use a compiler flag to specify whether you have Multinet, UCX or Wollongong TCP/IP. (cf rebuilding the line mode browser). The flags should be one of:

```
/DEF=MULTINET
/DEF=WIN_TCP
/DEF=UCX
```

2.8.1 Running

The daemon works with document names which look like unix-style filenames. At the point of reading a file, these are converted into VMS style filenames.

2.8.2 Testing it

Suppose you have compiled and linked httpd successfully. You write a "welcome.html" file as an introduction to your server for those from outside, and you put it in some suitable directory which you wish to export, say sys$disk[my.public]welcome.html

You run it as an ordinary user on a port over 1024 from a terminal window.

```
httpd == $sys$disk:[my.directory]httpd.exe
httpd -p 8000 -v "/sys$disk/my/public"
```

Note that the directory to be exported is given in unix style. Don't panic. Watch the trace (enabled by the -v option) . The server should end up waiting for a message.

From another terminal window, you test the server, giving the internet node name of your machine in place if mynode.dom.ain and the same port number. We assume you have the lin mode browser installed. You could test it with a GUI browser, but the trace might be more difficult to find.

```
www -v "http://mynode.dom.ain:8000/welcome.html"
```

You should now get your welcome page displayed on the terminal. theer will be a lot of trace as well which may make it almost unreadable, but if it works of course you run both server and/or client next time without the -v.

2.8.3 Installing properly.

Check whwther your TCP/IP brand contains an inetd daemon. If it does, that is great, you just run it under the inetd daemon following the manufacturer's instructions. Set the daemon up to run on any TCP connection to port 80. (The service name for port 80 is http).In this case the only command line parameter which you will need to pass to httpd is the directory name. Omit the port number to tell httpd that it is running under the inet daemon. If you find that this daemon is too slow (very possible under VMS), then switch to using the method below.

If you don't have an Inet daemon, then you have to run the daemon as a detached process. To do this you have to add something to one of the many VMS boot startup files like SYSTARTUP.COMor some such. You need to be the system manager to do this, and if you are, you probably know where you personally like to put these things. The command line should be as in the example when you tested it, except the port should be 80 (not 8000), and there should be no trace requested.

In practice it seems that under VMS you always have to start a DCL environment to run a server, because if you just detach HTTPD you can't pass it any parameters. So you use the usual trick of running loginout.exe to create a DCL environment:

```
$RUN/DETACH/IN=SYS$EXE:HTTPD.COM/OUT=SYS$TEMP:HTTPD.LOG -
      SYS$SYSTEM:LOGINOUT.EXE
```

where HTTPD.COM is

```
$ httpd == $sys$disk:[my.directory]httpd.exe
$ httpd -p 80 "/sys$disk/my/public"
```

Check that out and tell us if it doesn't work...

2.9 Running Under Shell

You don't have to run the daemon under the `inetd` if it doesn't work (and we recommend running it standalone anyway). You can run it from a shell session.

Run `httpd` from your terminal turned on, with a different port number like 8080:

```
httpd -p 8080
```

Note: You must be `root` (under VMS, have some privilege) to run with a port number below 1024. If you select a port above 1024, then you can run as a normal user. This way, anyone can publish files on the net. Howeever, it isn't very reliable, as your server will not automatically come back up if the machine is rebooted. In the long term it is best to install it to be started from the system startup file `/etc/rc` or `/etc/rc.local`.

You may not be able to use a port number which has been used by a daemon process recently (port may still be bound), so you may have to switch port number if you ˆC and restart `httpd`. When it is running like this, you can also read the debugging messages (when running with `-v` option), and use a debugger on it if necessary. (See also: telnetting to the server).

2.9.1 Debugging using Trace

If you can't understand why a server refuses to give back a document, then run with the `-v` option to turn on debugging messages. Use `-v` as the very first command line option (this way debugging is turned on right away). You will see the daemon setting up the rules for translating requests into local URLs, and you will see its attept to access the file (assuming you map requests onto files).

```
httpd -v -p 8080
```

Try to access the document from a client using another terminal window. Look at the debugging output. It will probably explain what is happening. If you still can't figure out the problem, mail your local guru help desk or if desperate `httpd@info.cern.ch` **enclosing** a copy of debugging output.

2.9.2 Even simpler

For testing a daemon very simply, without using a client, you can make the terminal be the client. With `httpd` try just running it with the terminal and typing `GET` */document/url* into its input:

```
httpd -v
GET /document/url
```

2.9.3 Telnetting to httpd

Most implementations of telnet allow you to specify a port number. Under unix this is often just a second parameter, under VMS a `/PORT` option.

The HTTP protocol is a telnet protocol, so you can simulate it just by typing things in. This will help you to see exactly what a sending back, and it will check you that it really is the server not the browser which has a problem.

Here is a simple example (keybord input is in **boldface**):

```
> telnet myhost.domain 80
Connected to myhost.domain on port 80
Escape is ^[
GET /document/url
...document or error message...
```

2.9.4 Debugging httpd

Suppose you think you have installed `httpd` but it doesn't work. Here we assume you have used port 80. If you have a situation not handled by this problem-solving guide, please mail `httpd@info.cern.ch`.

Type

```
www http://myhost.domain/
```

What happens?

Connection Refused

The browser tries to connect to the daemon but gets this status in the trace.

This means that nobody was listening on that port number. Check the port numbers match between server and client. Make sure you specify the port number explicitly in the document address for `www`.

If you are running the daemon standalone (as you should be), check that it is actually running by taking a list of processes, and that it is listening to the correct port (specified with `-p` *port* option), or try running it from the terminal with `-v` option as well. The trace for the server should say `"socket, bind and listen all ok".` If it does, and you still get "`connection refused`", then you must be talking to the wrong host (or, conceivably, different ethernet adapters on the same host).

If you are running with the inet daemon, then check both the services file `(/etc/services)` or database (yellow pages, netinfo) if your system uses it, and the `/etc/inetd.conf` file. Check the service name matches between these two (e.g. `http`).

Did you remember to kill -HUP the `inetd` when you changed the `inetd.conf` file?

Be aware that on some systems your local file `/etc/services` will not be consulted E.g. when `ypbind` is running on Suns, then you should type

```
ypwhich -m services
```

and ask the administrator of the machine named to change its own `/etc/services`.

Try running the deamon from a shell window to see better what happens.

Cannot Connect To Information Server

The usual cause of this is that the server is not running, or it's running on a different port.

There is more information you can get. Use the "verbose" option on the LineMode browser to find out what went wrong:

```
www -v http://myhost.domain:80/
```

What do you get? A load of trace messages. There are several cases.

- The browser can't look up the name of the host. If it can, it will display "Parsed address as" message. If not, try fixing your name server or `/etc/hosts` file, or quoting the IP number of the host in decimal notation (like 128.141.77.45) instead.
- The browser can get to the host but gets `"Connection refused"` status back.
- Your browser gets an error number but prints "error message not translated". This is because when it was compiled on your platform it didn't know what form the error message table took. Try the same thing form a unix platform for example.
- You get some network error like "network unreachable". Depending on whether the IP network is your responsibility or not, and your attitude to life, either fix it, try again in an hour's time, or complain to someone.

Unable To Access Document

Typical cause of this is that the configuration file is incorrect, or files are not readable by the user-id under which the server runs. When you are running the server as `root,` it will automatically switch it to `nobody` just before serving the document. This can be changed with the `UserId` configuration directive.

An Empty Document Is Displayed

The document sent back is empty, but there is no error message.

The `inetd` has started a process to run your server but it immediately failed. Possibilities include:

- When running from `inetd`, the daemon may not be in the file specified, or may not be executable by the specified user (or, if a user id is not specified in your variety of `inetd.conf`, `root`).
- For some reason server crashes when it's trying to serve the request. If you can, try to tract down when this happens, and send mail to `httpd@info.cern.ch`. Try running the daemon from a terminal window to see what happens.
- Script fails to produce any result, which may be due to the fact that there is no empty line after the header section output by the script, causing server to read the entire generated document as the header section.

Document Address Invalid Or Access Not Authorized...

...or some similar kind of error message. This means either:

- You have been passed a bad document address. If you are following a link, check with the author of the document which contained the link.
- The document has been moved. Check with the server administrator. You should be able to find out who runs the server by going to the welcome page (type "g /" with the line mode browser) and seeing a link to information about the maintainers.

If you are the server administrator, and you can't understand why the daemon refuses to deliver the file,

- Check the configuration file (rule file, by default `/etc/httpd.conf`) if you have one. Think out way the document name will be mapped successively by each line, and what the result will be.
- Run the daemon in debug mode from a terminal session to get trace information.

Bad Output

A document is displayed, but not the one you wanted.
These are some ideas:

- Try running the server from the terminal.
- Check the HTML source the daemon produces with

  ```
  www -source http://my.host.domain/
  ```

- Try telnetting to httpd and simulating the client:

  ```
  > telnet my.host.domain 80
  Connected to my.host.domain on port 80
  Escape is ^[
  GET /document/name
  ```

3. Command Line of CERN httpd

The command line syntax for `httpd` allows a number of options and an optional directory argument:

```
httpd  [-opt -opt -opt ...] [directory]
```

The directory argument, if present, indicates the directory to be exported. If not present, either a rule file is be used, to export combinations of directories, or else the default is to export the `/Public` directory tree.

3.1 Options

-l ***logfile***	Use *logfile* to log the requests.
-newlog ***logfile***	Use *logfile* to log the requests using the new, common logfile format. This will eventually become the default.
-errlog ***errorlogfile***	Use *errorlogfile* to log errors. If this is not specified, but *logfile* is (with `-l` or `-newlog` option), *logfile.error* is used.
-gmt	Use GMT instead of localtime in logfile (localtime is default).
-nolog ***template***	Don't log accesses from hosts matching *template*. Template is either an IP number mask like `128.141.*.*` or a hostname template containing at most one wildcard, for example `*.cern.ch`
-p ***port***	Listen to port *port*. Without this argument `httpd` assumes that it has been run by `inetd`, and uses `stdin` and `stdout` as its communication channel. **Note** that port numbers under 1024 are privileged.
-r ***rulefile***	Use *rulefile* as configuration file.
-disable ***METHOD***	Disable *METHOD* on this server. You can also use the `Disable` directive in configuration file.
-enable ***METHOD***	Enable *METHOD* on this server. You can also use the `Enable` directive in configuration file.
-setuid	When using user authentication, set server user-id to authenticated user id (for people who have login accounts on the same machine as the documents reside, and nobody else needs to access them).
-v	Verbose, turn on debugging messages.
-vv	Very Verbose, turn on even more verbose debugging messages.
-version	Print version number of `httpd` and `libwww` (the WWW Common Library)

3.1.1 Directory Browsing

You can set these also with the `DirAccess` configuration directive.

-dy	Enable direcory browsing. Directories are returned as hypertext documents. See browsing directories. *Default.*
-dn	Disable directory browsing. An attempt to access a directory will generate an error response.
-ds	Selective directory browsing; enabled only for directories containing a file named `.www_browsable`

3.1.2 README Feature

You can set these also with the `DirReadme` configuration directive.

-dt For any browsable directory which contains a `README` file, include the text of the `README` file at the top of the document before the listing. *Default.*

-db As `-dt` but put the `README` at the bottom, after the listing. The `-db` and `-dt` options may be combined with `-dy` as `-dyb`, `-dty` etc.

-dr Disables the `README` inclusion feature.

3.2 Examples

```
httpd -p 80 -dyt /ftp/pub
```

This is a standalone server running on port 80. It exports the entire `/ftp/pub` tree with browsable directories and `README` files included at the top of directory listings.

Requesting a URL `http://your.machine/misc/file.html` will be mapped to physical file `/ftp/pub/misc/file.html` by `httpd`.

```
httpd
```

`httpd` uses its default configuration file `/etc/httpd.conf`. If that file doesn't exist, `httpd` exports the `/Public` directory tree. This tree may contain soft links to other directory trees. This is an `httpd` reading its `stdin` and writing to its `stdout`, so it is run by `inetd`.

```
httpd -r /usr/local/lib/httpd.conf
```

The same as before, but uses `/usr/local/lib/httpd.conf` as a rule file instead of the default `/etc/httpd.conf`.

4. Configuration File of CERN httpd

4.1 In This Section...

- Example configuration files
- `/etc/httpd.conf` - Default configuration file
- `#` - Comment sign
- `Map` - Map URLs to actual files
- `Pass` - Accept a request
- `Fail` - Fail a request
- `Redirect` - Redirect a request
- `Protect` - Set up protection
- `DefProt` - Default protection setup
- `Exec` - Executable server scripts
- `Search` - Index search facility
- `AddType` - Filename suffix mappings to MIME Content-Types
- `AddEncoding` - Filename suffix mappings to MIME Content-Transfer-Encodings
- `AddLanguage` - Filename suffix mappings to different Content-Languages, multilanguage support
- `UserDir` - User-supported directories, URLs starting **/ũsername**
- `MetaDir` - Directory name for meta-information files
- `MetaSuffix` - Suffix for meta-information files
- `NoLog` - No log entries for listed hosts/domains
- `Disable` - Disable methods that you don't need/want
- `Enable` - Enable a desired method

4.1.1 New In 2.16

- `DirAccess` - Enable/Selective/Disable directory listings
- `DirReadme` - Configure/disable README-feature
- `AccessLog` - Set access log file name
- `ErrorLog` - Set error log file name
- `LogFormat` - Set access log file format
- `LogTime` - Set time zone for log files

- `SuffixCaseSense` - Set suffix case sensitivity
- `InputTimeOut` - Timeout for request
- `OutputTimeOut` - Timeout for response
- `UserId` - Default user to run as (instead of `nobody`)
- `GroupId` - Default group to run as (instead of `nogroup`)

4.1.2 Firewall Gateway Caching and Other Directives

- `CacheRoot` - Set cache root directory for a proxy server
- `CacheSize` - Specify cache size (in megabytes)
- `CacheClean` - Remove everything older than this (in days)
- `CacheUnused` - Remove if has been unused this long (in days)
- `CacheDefaultExpiry` - Default expiry time if not given by remote server (in days)
- `GcTimeInterval` - Interval to do cache garbage collection (in hours)
- `GcReqInterval` - Number of requests between garbage collections
- `GcMemUsage` - Garbage collector memory usage directive
- `CacheLimit_1` - First cache file size limit (kilobytes)
- `CacheLimit_2` - Second cache file size limit (kilobytes)
- `CacheLockTimeOut` - Break cache locks after this amount of seconds.
- `http_proxy`
- `ftp_proxy`
- `gopher_proxy`
- `wais_proxy` - Make firewall gateway (proxy) connect to another gateway

The configuration file (often referred to as the rule file) defines how `httpd` will translate a request into a document name. It allows one to provide an extra level of name mapping above that given by symbolic links in the file system. It allows, for example, out of date names to mapped onto their more recent counterparts.

The configuration file also allows access to be restricted. This is essential, to prevent, for example, unauthorized access to your private documents.

Note: The configuration file is not essential if you want to just export one directory tree, but then you must remember to specify the exported directory in command line:

```
httpd -p 80 /your/exported/directory
```

The server guesses the data types of file from the file suffix. A configuration file **is** necessary to specify any data types which are not in the default set of suffixes. However, the default set is quite extensive.

4.2 Default Configuration File

By default, the configuration file `/etc/httpd.conf` is loaded, unless specified otherwise with the `-r` command line option:

```
httpd -p 80 -r /your/own/httpd.conf
```

See also: example rule files.

4.3 Comments in Configuration File

Each line consists of an operation code and one or two parameters, referred to as the template and the result. Lines starting with a hash sign `#` are ignored, as are empty lines.

4.4 Mapping, Passing and Failing

There are three main rules: `Map, Pass` and `Fail.` The server uses the top rule first, then **each successive rule** unless told otherwise by a `Pass` or a `Fail` rule.

`Map` ***template result***	If the address matches the *template*, use the *result* string from now on for future rules.
`Pass` ***template***	If the address maches the *template*, use it as it is, porocessing no further rules.
`Pass` ***template result***	If the string matches the *template*, use the *result* string as it is, processing no futher rules.
`Fail` ***template***	If the address matches the *template*, prohibit access, processing no futher rules.

The *template* string may contain **one** wildcard asterisk `*`. The *result* string may have the wildcard only if the *template* has one.
When matching,

- Rules are scanned from the top of the file to the bottom.
- If a request matches a `Map` template exactly, the result string is used instead of the original string and applied to successive rules.
- If the request maches a `Map` *template* with wildcard, then the text of the request which matches the wildcard is inserted in place of the wildcard in the *result* string to form the translated request. If the result string has no wildcard, it is used as it is.
- When a `Map` substitution takes place, the rule scan continues with the next rule using the new string in place of the request. This is not the case if a `Pass` or `Fail` is matched: they terminate the rule scan.

4.5 Redirecting Requests Elsewhere

When documents, or entire trees of documents, are moved from one server to another, you can use `Redirect` rule to tell `httpd` to redirect the request to another server. If the client program is smart enough user won't even notice that the document is retrieved from a different server.

`Redirect` ***template result*** Document matching *template* is redirected to *result*, which must be a **full URL** (i.e. containing `http:` and the host name).

4.5.1 Example

```
Redirect   /hypertext/WWW/*   http://www.cern.ch/WebDocs/*
```

This redirects everything starting with `/hypertext/WWW` to host `www.cern.ch` into virtual directory `/WebDocs`. For example, `/hypertext/WWW/TheProject.html` would be redirected to `http://www.cern.ch/WebDocs/ThePr`

4.6 Setting Up User Authentication and Document Protection

Documents are protected by `Protect` and `DefProt` rules. Their syntax is the following:

`DefProt` ***template setup-file*** **`[`*uid.gid*`]`** Any document matching the *template* is associated with protection *setup-file*. The documents are not yet taken to be protected, but they may become protected by an existing access control list file in the same directory as the requested file, or by later matching a `Protect` rule. If that `Protect` rule doesn't specify *setup-file*, the one from the latest `DefProt` rule is used.

`Protect` **`[`*template setup-file* `[`*uid.gid*`]]`** Any document matching *template* is protected. The type of protection is defined in finer detail in *setup-file*.
If *setup-file* is not specified the one from previous matched `DefProt` rule will be used. If none have matched access to the file is forbidden.

setupfile is always a full pathname for the protection setup file which specifies the actual protection parameters.

Setup file can be omitted from `Protect` rule, but it is obligatory in `DefProt` rule. If setup file is omitted it is not possible to give the *uid.gid* part, either.

uid.gid are the Unix user id and group id (either by name or by number, separated by a dot) to which the server should change when serving the request. These are only meaningful when the server is running as `root.` If they are missing they default to *nobody.nogroup*.

Note: Uid and gid are inherited from `DefProt` rule to `Protect` rule **only** when the *setup-file* is also inherited. If *setup-file* is specified for `Protect` rule but *uid.gid* is not, they default to *nobody.nogroup* regardless of the previous `DefProt` rule.

This is to avoid accidentally running the server under wrong user id with wrong setup file. This information should logically go into the protection setup file, but for safety reasons it cannot be done, because a non-trustworthy collaboration could specify it to be `root`. This way only the main `webmaster` can control user and group ids.

4.7 Executable Server Scripts

Document address is mapped into a script call by `Exec` rule:

```
Exec template script
```

In both *template* and *script* there **must be a `*` wildcard, that matches everything starting from the script filename.** This is to enable `httpd` to know what is the script name and what is the extra path information to be passed to the script.

4.7.1 Example

You want to map everything starting with `/your/url/doit` to execute the script `/usr/etc/www/htbin/doit.` You do this by saying:

```
Exec  /your/url/*  /usr/etc/www/htbin/*
```

Here asterisk mathes the script name `doit` (and everything else that follows it). Usually people use some fixed keyword in front of the pathname in URL to point out that the document is actually a script call. Often this keyword is `/htbin`. That is, usually your `Exec` rule looks like this:

```
Exec  /htbin/*  /usr/etc/www/htbin/*
```

and all the URLs pointing to the scripts start with `/htbin`, for example `/htbin/doit` in the previous example.

4.7.2 Historical Note (HTBin Rule)

CERN `httpd` versions 2.13 and 2.14 had a hard-coded handling of URL pathnames starting `/htbin` that mapped them to scripts in a directory specified via `HTBin` rule:

```
HTBin /your/htbin/directory
```

This is still handled automatically by `httpd`, by translating it to its equivalent `Exec` form:

```
Exec /htbin/*  /your/htbin/directory/*
```

Always use `Exec` instead –– it is more general.

4.8 Index Search Facility

Server automatically calls a script to perform search, if the **absolute pathname** of search script is supplied by a `Search` field in rule file:

```
search /search/script/pathname
```

This script is called with URL pathname of the document from which the query was issued from, in `PATH_INFO` environment variable, and absolute (translated) document pathname in `PATH_TRANSLATED` environment variable. Keyword part of the URL is (undecoded) in `QUERY_STRING` environment variable, and also decoded as command line parameters, one in each of `argv[1]`, `argv[2]`, ...
Search script must conform to CGI/1.0 rules, that is, it has to output either a `Location:` field, or start its output with:

```
Content-Type: text/html
```

followed by a blank line. (The Content-Type can, of course, be also other than `text/html` –– this was just an example.

4.9 Binding Suffixes to MIME Content-Types

As well as any mapping lines in the rule file, the rule file may be used to define the data types of files with particular suffixes. CERN `httpd` has an extensive set of predefined suffixes, so usually you don't need to specify any.
The syntax is:

```
AddType .suffix representation encoding [quality]
```

The parameters are as follows:

suffix The last part of the filename. There are two special cases. `*.*` matches to all files which have not been matched by any explicit suffixes but do contain a dot. `*` by itself matches to any file which does not match any other suffix.

representation A MIME Content-Type style description of the repreentation in fact in use in the file. See the HTTP spec. This need not be a real MIME type - it will only be used if it matches a type given by a client.

encoding A MIME content transfer encoding type. Much more limited in variety than representations, basically whether the file is ASCII (7bit or 8bit) or binary. A few other encodings are allowed, and maybe extension to compression.

quality Optional. A floating point number between 0.0 and 1.0 which determines the relative merits of files `xxx.*` which differ in their suffix only, when a link to `xxx.multi` is being resolved. Defaults to 1.0.

4.9.1 Examples

```
AddType .html text/html              8bit     1.0
AddType .text text/plain             7bit     0.9
AddType .ps   application/postscript 8bit     1.0
AddType *.*   application/binary     binary   0.1
AddType *     text/plain             7bit
```

4.9.2 Historical Note (Suffix Directive)

`AddType` was previously called `Suffix.` The old name is still understood, but may be misleading since suffixes are also used to determine Content-Transfer-Encoding and language. Always use `AddType` instead.

4.10 Binding Suffixes to MIME Content-Transfer-Endocings

Suffixes are also used to determine the Content-Transfer-Encoding of a file (`.Z` suffix for `x-compressed`, for example). Syntax is:

```
AddEncoding .suffix  encoding
```

4.10.1 Example

```
AddEncoding .Z  x-compress
```

4.11 Multilanguage Support

Multilanguage support is also built on using suffixes to determine the language of a document. Suffix is bound to a language by `AddLanguage` rule (`.en` suffix for english, for example). Syntax is:

```
AddLanguage .suffix  encoding
```

4.11.1 Examples

```
AddLanguage .en  en
AddLanguage .uk  en_UK
```

4.12 User-Supported Directories

User-supported directories, URLs of form **/ũsername**, are enabled by `UserDir` directive:

```
UserDir dir-name
```

The *dir-name* argument is the directory in each user's home directory to be exported, for example `WWW` or `Web`.

4.13 Meta-Information

It is possible to tell `httpd` to add meta-information to response. Meta-information is stored in a directory specified by `MetaDir` directive, under the same directory as the file being retrieved:

```
MetaDir  dir-name
```

Meta-information is stored in a file with the same name as the actual document, but appended with a suffix specified via `MetaSuffix` directive:

```
MetaSuffix  .suffix
```

Meta-information files contain RFC822-style headers.

4.14 Suppressing Log Entries For Certain Hosts/Domains

It's not always necessary to collect log information of accesses made by local hosts. The `NoLog` directive can be used to prevent log entry being made for hosts matching a given IP number or host name template:

```
NoLog  template
```

4.14.1 Examples

```
NoLog 128.141.*.*
NoLog *.cern.ch
NoLog *.ch  *.fr  *.it
```

4.15 Enabling and Disabling HTTP Methods

You can enable/disable methods that you do/don't want your server to accept:

```
Enable  method
Disable method
```

By default `GET`, `HEAD` and `POST` are enabled, and the rest are disabled.

4.15.1 Examples

```
Enable POST
Disable DELETE
```

5. New In 2.16

5.1 Controlling Directory Browsing

`DirAccess on` Enable directory browsing in all directories (which are not forbidden by rules). Synonym with `-dy` command line option. *Default.*

`DirAccess off` Disable directory browsing. Synonym with `-dn` command line option.

`DirAccess selective` Enable selective directory browsing - only directories containing the file `.www_browsable` are allowed. Synonym with `-ds` command line option.

5.2 README Feature

`DirReadme top` For any browsable directeory containing a `README` file, include the text at the top of the directory listing. Synonym with `-dt` command line option. *Default.*

`DirReadme bottom` Same as previous, but contents of `README` appear on the bottom. Synonym with `-db` command line option.

`DirReadme off` Disables the `README` inclusion feature. Synonym with `-dr` command line option.

5.3 Access Log File

Access log file contains a log of all the requests. The name of the log file is spesified either by `-l` *logfile* command line option, or with `AccessLog` directive:

```
AccessLog /absolute/path/logfile
```

5.4 Error Log File

Error log contains a log of errors that might prove useful when figuring out if something doesn't work. Error log file name is set by `ErrorLog` directive:

```
ErrorLog /absolute/path/errorlog
```

If error log file is not specified, it defaults to access log file name with `.error` extension. If the filename extension already exists, `.error` will replace it.

5.5 Log File Format

Previously every server used to have its own logfile format which made it difficult to write general statistics collectors. Therefore there is now a *common logfile format* (which will eventually become the default). Currently it is enabled by

```
LogFormat  common
```

The old CERN `httpd` format can be used by

```
LogFormat  old
```

5.6 Log Time Format

Times in the log file are by default local time. That can be changed to be GMT time by `LogTime` directive:

```
LogTime  gmt
```

Default is:

```
LogTime  localtime
```

5.7 Suffix Case Sensitivity

Suffix case sensitivity is by default *off.* You can make suffixes case sensitive with `SuffixCaseSense` directive:

```
SuffixCaseSense On
```

5.8 Timeouts

You can specify timeouts for how long the server will wait for request from the client, and to send response. Timeouts are all specified in seconds. The defaults are:

```
InputTimeOut 120
OutputTimeOut 1200
```

That is, 2 minutes and 20 minutess, respectively.

5.9 Default User Id

`UserId` directive sets the default user to run as instead of `nobody`. This directive is only meaningful when running server as `root.`

```
UserId whoever
```

5.10 Default Group Id

`GroupId` directive sets the default group to run under instead of `nogroup`. This directive is only meaningful when running server as `root.`

```
GroupId whichever
```

6. Caching

6.1 Setting Cache Directory

Caching is enabled on a server running as a gateway (proxy) by `CacheRoot` directive, which is used to set the absolute path of the cache directory:

```
CacheRoot /absolute/cache/directory
```

6.2 Cache Size

`CacheSize` directive sets the maximum cache size in megabytes. Default value is 5MB, but its preferable to have several megabytes of cache, like 50-100MB, to get best results. Cache may, however, temporarily grow a few megabytes bigger than specified.

6.2.1 Example

```
CacheSize 20
```

sets cache size to 20 megabytes.

6.3 Maximum Time to Keep Cache Files

All cache files older than specified by `CacheClean` directive will be removed. This value overrides expiry date in that no file can be stored longer than this value specifies, regardless of expiry date. Default value is 21 days.

6.3.1 Example

```
CacheClean 14
```

would cause everything older than two weeks to be removed.

6.4 Maximum Time to Keep Unused Files

All cache files that have been unused longer than specified by `CacheUnused` directive will be removed. Default value is 14 days.

6.4.1 Example

```
CacheUnused 7
```

would set this to one week.

6.5 Default Expiry Time

Files for which the server gave neither `Expires:` nor `Last-Modified:` header will be kept at most the number of days specified by `CacheDefaultExpiry` directive. Default value is 7 days.

6.5.1 Example

```
CacheDefaultExpiry 1
```

would set this to one day.

6.6 How Often to Do Garbage Collection

Garbage collection is launched right away when cache size limit is reached. However, expired files should to be removed even if there is still cache space remaining. There are two directives controlling garbage collection scheduling:

```
GcTimeInterval hours
GcReqInterval requests
```

`GcTimeInterval` specifies the number of hours after which time to do garbage collection. Default value is 24 hours.

`GcReqInterval` specifies the maximum number of requests between successive garbage collections. Default value is 10000 requests.

6.7 Memory Usage of Garbage Collector

Garbage collector performs its job best if if can read information about the whole cache into memory at once. This is not possible if the machine doesn't have enough main memory.

`GcMemUsage` directive advices garbage collector about how much memory to use. You may imagine this is the number of kilobytes to use for gc data, but it may vary greatly according to dynamic things, like the directory structure of cached files.

Default is 500; if gc fails because memory runs out make this smaller. If your machine has so much memory that it just can't run out, make this very big.

6.7.1 Example

```
GcMemUsage 100
```

if you have very little memory.

6.8 Cache File Sizes

There are two limits controlling the size factor of a file when its value is being calculated. `CacheLimit_1` sets the lower limit; under this all the files have equal size factor. `CacheLimit_2` sets up higher limit; files bigger than this get extremely bad size factor (meaning they get removed right away because they are too big).

Sizes are specified in kilobytes, and defaults values are 200K and 4MB, respectively.

6.8.1 Examples

```
CacheLimit_1 200
CacheLimit_2 4000
```

would set the same values as the defaults, 200K and 4MB.

6.9 Cache Lock Timeout

During retrieval cache files are locked. If something goes wrong a lock file may be left hanging. `CacheLock-TimeOut` directive sets the amount of time after which lock can be broken. Time is specified in seconds, default value is 1200 seconds (20 minutes), the same as default `OutputTimeOut`. **CacheLockTimeOut should never be less than OutputTimeOut!**

6.9.1 Example

```
CacheLockTimeOut 1800
```

would set lock timeout to half an hour.

6.10 Going Through Many Gateways

If there is a need to make an (inner) proxy server connect to the outside world via another (outer) proxy server, you can use the same environment variables as are used to redirect clients to the proxy to make inner proxy use the outer one:

- `http_proxy`
- `ftp_proxy`
- `gopher_proxy`
- `wais_proxy`

E.g. your (inner) proxy server's startup script could look like this:

```
#!/bin/sh
http_proxy=http://outer.proxy.server:8082/
export http_proxy
/usr/etc/httpd -r /etc/inner-proxy.conf -p 8081
```

This is a little ugly, so there are also the following directives for the configuration file:

- `http_proxy` *http://outer.proxy.server/*
- `ftp_proxy` *http://outer.proxy.server/*
- `gopher_proxy` *http://outer.proxy.server/*
- `wais_proxy` *http://outer.proxy.server/*

7. Configuration File Examples

A basic configuration file for httpd might look like this:

```
Pass    /           file:/u/john/welcome.html
Pass    /*          file:/u/john/public/*
Fail    *
```

The first line maps the root document onto a specific document about the server, and accepts it. (See etiquette about the welcome page.)

The second line maps all document names onto filenames in a particular directory and accepts them.

The third line disallows access to all other documents. (This is actually no longer necessary, because `httpd` by default fails everything that isn't explicitely `Pass`'ed.

7.1 Second Example

```
UserDir  public_html
Exec     /htbin/*     /u/john/scripts/*
Map      /            /tnotes/welcome.html
Map      /tnotes/*    file:/u/john/public/*
Map      /seminars/*  file:/u/jane/seminars/*
Pass     file:/u/john/public/*
Pass     file:/u/jane/seminars/*.html
```

The first line specifies that user-supported directories are called `public_html` under each user's home directory. This is where URLs of form **/ũsername/...** get mapped.

The second line specifies that all the starting with `/htbin/` will be handled as requests to run a server script in directory `/u/john/scripts`. It's important to have the `Exec` rule before `Maps` and `Passes`, to avoid script requests from being served as regular file requests.

The first `Map` rule maps the root document onto a specific document about the server. Because it is `Map` and not `Pass`, it **doesn't** accept it but passes it on for futher mapping by lines futher down.

The second `Map` rule maps all document names starting with `/tnote/` onto filenames in a particular directory where john maintains the technical notes. If someone else takes over the technical notes, we can change this. Here we are starting to distinguish between document names and file names. This can be carried much further if necessary, but one level of mapping is enough to allow for changes of administration of different areas.

The third `Map` separately maps the seminar information into Jane's `seminars` directory.

The `Pass` rules enable access to anything in John's `public` directory, and any `.html` file in Jane's `seminar` directory tree. Note here that the `*` maps to any sequence **including slashes** so all files in any subdirectory of /u/jane/seminars will be enabled so long as they end in `.html`.

Remember that currently the wildcard character can only appear once.

It is important to understand that rules are translated from top to bottom, and the first `Pass`, `Exec` or `Fail` rule that matches the current template will terminate rule translation.

7.2 Configuration file for a WAIS gateway

`httpd` can be used as a WAIS gateay if it has been compiled with the necessary options and linked with the freeWAIS software. A suitable configuration file is

```
Pass    /*   wais://*
```

8. CERN Server CGI/1.0 Script Support

Server scripts are used to handle searches, clickable images and forms, and to produce synthesized documents on the fly. See calendar and finger gateway for examples.

8.1 In This Section...

- Using `Exec` rule to allow scripts
- CGI/1.0 Interface −− Script Input
- CGI/1.0 Interface −− Script Output
- NPH-Scripts −− No Parsing of Headers
- Setting up a search script

8.2 Important Note!

CERN `httpd` versions 2.15 and newer have **two** script interfaces. The other one is the official CGI/1.0, Common Gateway Interface, which enables scripts to be shared between different server implementations (NCSA server, Plexus, etc). The other one is the original, very easy-to-use, interface, that was introduced in version 2.13.

Use of CGI/1.0 instead of the old interface is strongly encouraged.

IMPORTANT: If you have, or wish to write, scripts that use the old interface, your script name has to end in `.pp` suffix (comes from "Pre-Parsed"). URLs referring to these scripts should not contain this suffix. This is to make it easier to later upgrade to CGI/1.0 scripts, so you only need to change the script name in the file system, and not the documents pointing to it. If you absolutely want to use the old interface (which is nice for quick hacks that don't need to be portable), see the doc.

8.3 Setting Up httpd To Call Scripts

The server knows that a request is actually a script request by looking at the beginning of the URL pathname. You can specify these special strings in the configuration file `(/etc/httpd.conf)` by `Exec` rules:

```
Exec /url-prefix/*  /physical-path/*
```

Where */url-prefix/* is the special string that signifies a script request, and */physical-path/* is the absolute filesystem pathname of the **directory** that contains your scripts.

8.3.1 Example

```
Exec  /htbin/*  /usr/etc/cgi-bin/*
```

makes URL paths starting with `/htbin` to be mapped to scripts in directory `/usr/etc/cgi-bin.` I.e. requesting

```
/htbin/myscript
```

causes a call to script

```
/usr/etc/cgi-bin
```

8.3.2 Historical Note

In `httpd` versions before 2.15 there was an `HTBin` directive:

```
HTBin  /physical-path
```

which is now obsolite, but understood by the server to mean

```
Exec  /htbin/*  /physical-path/*
```

Use of `Exec` rule instead is recommended for its generality.

8.4 Information Passed to CGI/1.0 Scripts

CGI/1.0 scripts get their input mainly from environment variables and standard input (when using `POST` method). Search scripts get keywords also as command line arguments.

Most important environment variables are:

QUERY_STRING The query part of URL, that is, everything that follows the question mark. This string is URL-encoded, meaning that special characters like spaces and newlines are encoded into their hex notation (%xx), and characters like `+ = &` have a special meaning. The contents of this variable can be easily parsed using the `cgiparse` program.

PATH_INFO Extra path information given after the script name, for example with `Exec` rule:

```
Exec  /htbin/*  /usr/etc/cgi-bin/*
```

a URL with path

```
/htbin/myscript/extra/pathinfo
```

will execute the script `/usr/etc/cgibin/myscript` with `PATH_INFO` environment variable set to `/extra/pathinfo`.

PATH_TRANSLATED Extra pathinfo translated through the rule system. (This doesn't always make sense.)

See also NCSA's primer to writing CGI/1.0 scripts.

8.5 Results From Scripts

Scripts return their results either outputting a document to their standard output, or by outputting the location of the result document (either a full URL or a local virtual path).

8.5.1 Outputting a Document

Script result must begin with a `Content-Type:` line giving the document content type, followed by **an empty line**. The actual document follows the empty line. Example:

```
Content-Type: text/html

<HEAD>
<TITLE>Script test>
</HEAD>
<BODY>
<H1>My First Virtual Document</H1>
....
</BODY>
```

8.5.2 Giving Document Location

If the script wants to return an existing document (local or remote), it can give a `Location:` header followed by an empty line: Example:

```
Location: http://info.cern.ch/hypertext/WWW/TheProject.html
```

This causes the server to send a redirection to client, which then retrieves that document. If `Location` starts with a slash (is not a full URL), it is taken to be a virtual path for a document on the same machine, and server passes this string right away through the rule system and serves that document as if it had been requested in the first place. In this case clients don't do the redirection, but the server does it "on the fly".
Example:

```
Location: /hypertext/WWW/TheProject.html
```

Understand, that this is a **virtual path**, so after translations it might be, for example, `/Public/Web/TheProject.html`.
Important: Only **full** URLs in `Location` field can contain the *#label* part of URL, because that is meant only for the client-side, and the server cannot possibly handle it in any way.

8.5.3 NPH-Scripts (No-Parse-Headers)

Script wishing to output the entire HTTP reply (including status line and all response headers) should be named to begin with `nph-` prefix. This makes `httpd` connect script's output stream directly to requesting client reducing the overhead of server needlessly parsing the response headers.

Example Of NPH-Script Output

```
HTTP/1.0 200 Script results follow
Server: MyScript/1.0 via CERN/2.15
Content-Type: text/html

<HEAD>
<TITLE>Just testing...</TITLE>
</HEAD>
<BODY>
<H1>Output From NPH-Script</H1>
Yep, seems to work.
</BODY>
```

8.6 Setting Up A Search Script

There is a special `Search` directive in the configuration file givin the **absolute** pathname of the script performing the search:

```
Search /absolute/path/search
```

Every time a document is searched, this script is called with

Command line	containing the search keywords decoded, one in each of `argv[1]`, `argv[2]`, ...
`QUERY_STRING`	containing the query string encoded, as it came in the URL after the question mark.
`PATH_INFO`	Virtual path of the document that the search was issued from.

PATH_TRANSLATED Absolute filesystem path of the document.

Search results are output in the usual way:

```
Content-Type: text/html

...generated document...
```

8.7 cgiparse Manual

`cgiparse` handles `QUERY_STRING` environment variable parsing for CGI scripts. It comes with CERN server distributions **2.15** and newer.

If the `QUERY_STRING` environment variable is not set, it reads `CONTENT_LENGTH` characters from its standard input.

8.7.1 Command Line Options

Main Options

cgiparse -keywords Parse `QUERY_STRING` as search keywords. Keywords are decoded and written to standard output, one per line.

cgiparse -form Parse `QUERY_STRING` as form request. Outputs a string which, when `eval`'ed by Bourne shell, will set shell variables beginning with `FORM_` appended with field name. Field values are the contents of the variables.

cgiparse -value ***fieldname*** Parse `QUERY_STRING` as form request. Prints only the value of field *fieldname*.

cgiparse -read Just read `CONTENT_LENGTH` characters from `stdin` and write them to `stdout`.

cgiparse -init If `QUERY_STRING` is not defined, read `stdin` and output a string that when `eval`'d by Bourne shell it will set `QUERY_STRING` to its correct value. This can be used when the same script is used with both `GET` and `POST` method. Typical use in the beginning of Bourne shell script:

```
eval `cgiparse -init`
```

After this command the `QUERY_STRING` environment variable will be set regardless of whether `GET` or `POST` method was used. Therefore `cgiparse` may be called multiple times in the same script (otherwise with `POST` it could only be called once because after that the `stdin` would be already read, and the next `cgiparse` would hang).

Modifier Options

-sep ***separator*** Specify the string used to separate multiple values. With

- `-value` default is newline
- `-form` default is ", "

-prefix ***prefix***

- Only with `-form`. Specify the prefix to use when making up environment variable names. Default is "*FORM_*".

-count With

- `-keywords` outputs the number of keywords
- `-form` outputs the number of unique fields (multiple values are counted as one)
- `-value` *fieldname* gives the number of values of field *fieldname* (no such field is zero, one field gives 1, one multiple 2, etc).

−*number*, e.g. −2 With

- `-keywords` gives *n*'th keyword
- `-form` gives all the values of *n*'th field
- `-value` *fieldname* gives *n*'th of the multiple values of field *fieldname* (first value is number 1).

-quiet Suppress all error messages. (Non-zero exit status still indicates error.)

All options have one-character equivalents: `-k -f -v -r -i -s -p -c -q`

8.7.2 Exit Statuses

- `0` Success
- `1` Illegal command line
- `2` Environment variables not set correctly
- `3` Failed to get requested information (no such field, `QUERY_STRING` contains keywords when form field values requested, etc).

8.7.3 Examples

Note: In real life, of course, `QUERY_STRING` is already set by the server.
Here `$` is the Bourne shell prompt.

Keyword Search

```
$ QUERY_STRING="is+2%2B2+really+four%3F"
$ export QUERY_STRING
$ cgiparse -keywords
is
2+2
really
four?
$
```

Parsing All Form Fields

```
$ QUERY_STRING="name1=value1&name2=Second+value%3F+That%27s right%21"
$ export QUERY_STRING
$ cgiparse -form

FORM_name1='value1'; FORM_name2='Second value? That'\''s right!'

$ eval `cgiparse -form`
$ set
...
FORM_name1=value1
FORM_name2=Second value? That's right!
...
$
```

Extracting Only One Field Value

```
QUERY_STRING as in previous example.
$ cgiparse -value name1
value1
$ cgiparse -value name2
Second value? That's right!
$
```

9. CERN Server Clickable Image Support

CERN Server versions 2.14 and newer have a `htimage` program in the distribution, which is an `/htbin` program handling clicks on sensitive images. For versions 2.15 and newer it is a CGI/1.0 program (uses the Common Gateway Interface to communicate with `httpd`). See demo.

9.1 In This Section...

- `htimage` installation
- Writing documents that contain clickable images
- Image configuration file
- Output of `htimage`

9.2 Installing htimage Binary

After compiling `htimage` you should move the executable binary to the same directory as your other server scripts are, and remember to set up an exec rule. For example if your scripts are in `/usr/etc/cgi-bin`, you could have an `Exec` rule like this:

```
Exec   /htbin/*   /usr/etc/cgi-bin/*
```

Often `htimage` is one of the most often used scripts, and it would therefore be nice to refer to it with as short a name as possible, like `/img`, so you could have a `Map` rule just before the `Exec`:

```
Map    /img/*     /htbin/htimage/*
Exec   /htbin/*   /usr/etc/cgi-bin/*
```

9.3 Writing a Document With Clickable Images

To create a clickable image in your HTML document, you'll need to:

- specify `ISMAP` in your inlined image call, and
- make that image an anchor, with an `HREF` to the script handling the request `(htimage)` with image configuration file name appended to it.

Each clickable image has to be described to `htimage` via an image configuration file. These files are referred to by the extra path information in the URL causing the call to `htimage`:

```
<A HREF="/htbin/htimage/image/config/file">
<IMG SRC="Image.gif" ISMAP></A>
```

Image configuration file can be:

- either a virtual path, that is translated through rule system,
- or an absolute path in your filesystem.

`htimage` will look for both of these (afterall, it gets both `PATH_INFO` and `PATH_TRANSLATED` environment variables from `httpd` anyway).

You can even do some very smart mappings in the rule file to allow very short references to `htimage` and picture configuration files. Let's suppose all your image configuration files are in directory `/usr/etc/images`. Then you can use the following two rules in your server's configuration file (by default `/etc/httpd.conf`):

```
Map   /img/*    /htbin/htimage/usr/etc/images/*
Exec  /htbin/*  /usr/etc/cgi-bin/*
```

In this case you can refer to your image mapper very easily; if you have an image configuration file `Dragons.conf` in `/usr/etc/images` directory, all you need to say in the anchor is this:

```
<A HREF="/img/Dragons.conf">
<IMG SRC="Image.gif" ISMAP></A>
```

9.4 Image Configuration File

There are four keywords:

`default` ***URL*** — *URL* which is used if click is in none of the given shapes. This should always be set!

`circle` ***(x,y) r URL*** — Circle with center point *(x,y)* and radius *r*.

`rectangle` ***(x1,y1) (x2,y2) URL*** — Rectangle with (any) two opposite corners having coordinates *(x1,y1)* and *(x2,y2)*.

`polygon` ***(x1,y1) (x2,y2) ... (xn,yn) URL*** — Polygon having adjacent vertices *(xi,yi)*. If the path given is not closed (first and last coordinate pairs aren't the same) the first and last coordinate pairs will be connected by `htimage.` So first point is added also as the last one if necessary.

These can be abbreviated as `def, circ, rect, poly.`

Shapes are checked in the order they appear in config file, and the URL corresponding to the first match is returned. If none match, the `default` URL is returned.

*URL*s are

- either full URLs (with access method, machine name and path), in which case server sends a redirection to client,
- or a partial URL containing only pathname part of it (always starting with a slash), in which case server considers that as the original request, translates it through the rule system, access authorization and serves it normally (faster than sending redirection).

9.5 Output Produced by htimage

`htimage` prints a single `Location:` field to its `stdout`, or an error message with preceding `Content-Type: text/html` so in fact `htimage` behaves exactly as any other CGI/1.0 program (script), and is not in any way handled specially by the server. Therefore, you can rename `htimage` to whatever you prefer, like we called it `/img` in the above example.

Server understands this `Location:` field, and either directly sends that file to the client (non-full URL), or sends a redirection to client causing it to fetch the document, maybe even from another machine.

Note that URLs returned by `htimage` may well be other script requests - there is no reason for being limited to just regular documents.

10. Protected CERN Server Setup

Access can be restricted according to user name, internet address, or both. Access control can be tree-level, file level, or both.

10.1 In This Section...

- Password File
- Group File
- Protect Directive in Configuration File
- Protection Setup File
- Protecting a Tree of Documents
- Protecting Individual Files
- Using Two-Level Protection
- Access Control List File

10.2 Password File

If user-wise access control is used there has to be a password file listing all the users and their encrypted passwords. Password file can be maintained by `htadm` program which is a part ot CERN `httpd` distribution.

Unix password files are understood by CERN daemon (but not vice versa). However, **Unix users are in no way connected to the WWW access authorization.**

10.3 Group File

Group file contains declarations of groups containing users and other groups, with possibly an IP address template. Group declarations as viewed from top-level look like this:

```
groupname: item, item, item
```

The list of items is called a group definition. Each `item` can be a username, an already-defined groupname, or a comma-separated list of user and group names in parentheses. Any of these can be followed by an at sign `@` followed by either a single IP address template, or a comma-separated list of IP address templates in parentheses. The following are valid group declarations:

```
authors: john, james
trusted: authors, jim
cern_people: @128.141.*.*
hackers: marca@141.142.*.*, sanders@153.39.*.*,
         (luotonen, timbl, hallam)@128.141.*.*,
         cailliau@(128.141.201.162, 128.141.248.119)
cern_hackers: hackers@128.141.*.*
```

If an item contains only IP address template part all users from those addresses are accepted (e.g. `cern_people` above). Note the last two declarations: `cern_hackers` group is made up of the `hackers` group by restricting it further according to IP address.

Group definition can be continued to next line after any comma in the definition. Forward references in group file are illegal (i.e. to use group name before it is defined).

Group definition syntax is valid not only in group file, but also in

- `GetMask` in protection setup file, and
- in last field in ACL entries.

10.4 Server Configuration File

Typically you protect a tree of documents by `protect` rule in rule file, and specify authorized persons and IP addresses in the protection setup file or access control list file:

```
Protect /very/secret/*  /WWW/httpd.setup
```

If there are Unix file system protections set up so that there is no world read-permission the daemon naturally has to run as the owner or the group member of those files.

However, if there are protected trees owned by different people this doesn't work. In that case *the daemon has to run as* `root`*, and the user and group ids have to be specified in the* `protect` *rule,* e.g.:

```
Protect /kevin/secret/*   /WWW/httpd.setup1  kevin.www
Protect /marcus/secret/*  /WWW/httpd.setup2  marcus.nogroup
```

10.5 Protection Setup File

Each `protect` rule has an associated protection setup file. It specifies valid authentication schemes, password and group files, and password server-id:

```
AuthType      Basic, KerberosV4
ServerId      OurCollaboration
PasswordFile  /WWW/Admin/passwd
GroupFile     /WWW/Admin/group
```

Password server id needs not be a real machine name. It's only purpose is to inform the browser about which password file it is using (different protection setups on the same machine can use different password file and that would otherwise confuse pseudo-intelligent clients trying to figure out which password to send).

Same server-ids on different machines are considered different by clients (otherwise this would be a security hole).

KerberosV4 authentication is here only as an example; it has not been implemented as part of the library.

10.5.1 Protecting Entire Tree As One Entity

If you want to control access only to entire trees of documents and don't care to restrict access differently to individual files, it suffices to give a `GetMask` in setup file (and you don't need any ACL files):

```
GetMask    group, user, group@address, ...
```

Group definition has the same syntax as in group file.

10.5.2 Protecting Individual Files Differently

When each individual file needs to be protected separately you should use an ACL (access control list) file in the same directory as the protected files. After that no file in that directory can be accessed unless there is a specific entry in ACL allowing it.

In this case you don't need the `GetMask` in setup file.

10.5.3 Restricting Access Even Further

There may be both `GetMask` *and* an ACL, in which case both conditions must be met. This is typically used so that `GetMask` defines a general group of people allowed to access the tree, and ACLs restrict access even further.

10.6 Access Control List File

ACL file is a file named `.www_acl` in the same directory as the files the access of which it is controlling. It looks typically something like this:

```
secret*.html : GET,POST : trusted_people
minutes*.html: GET,POST : secretaries
*.html : GET : willy,kenny
```

It is worth noticing that all the templates are matched agaist (unlike in rule file where translation of rules stops in `pass` and `fail`.. So in the previous example all the HTML files are accessible to `willy` and `kenny`, even those matching the two previous templates.

The last field is just a list of users and group (possibly at required IP addresses), and in fact this field is in same syntax as group file.

When `PUT` method will be implemented it can appear in the middle field separated by a comma from `get`:

```
*.html : GET,PUT : authors
```

10.7 Manual Page For htadm

CERN `httpd` password file can be maintained with `htadm` program which is a part ot CERN `httpd` distribution.

10.7.1 Command Line Options and Parameters

`htadm -adduser` ***passwordfile*** **`[`*username*` [`*password*` [`*realname*`]]]`** adds a user into the password file (fails if there is already a user by that name).

`htadm -deluser` ***passwordfile*** **`[`*username*`]`** deletes a user from the password file (fails if there is no user by that name).

`htadm -passwd` ***passwordfile*** **`[`*username*` [`*password*`]]`** changes user's password (fails if there is no such user).

`htadm -check` ***passwordfile*** **`[`*username*` [`*password*`]]`** checks user's password (fails if there is no such user). Writes either `Correct` or `Incorrect` to standard output. Also indicates password correctness by a zero return value.

`htadm -create` ***passwordfile*** creates an empty password file.

If *`password`* or even *`username`* is missing in either of the previous cases they are prompted interactively. *`passwordfile`* must be always specified. Missing real name is also prompted when adding a new user.

Do NOT use `htadm` to add new users to the actual Unix password file `/etc/passwd`, entries written by `htadm` are missing some necessary fields to Unix.

Passwords should not be longer than 8 characters (this is a restriction from linemode clients using C library function `getpass()` to read the password −− there is no other cause for this restriction; the maximum hardcoded password size is actually much larger, and if you only use GUI or other clients that are able to read this long passwords, feel free to use them).

`htadm` destroys the password from command line as soon as possible so that it is very unlikely to see somebody's password by looking at the process listing on the machine (with `ps`, for example).

11. Firewall Gateways (Proxies)

11.1 In This Section...

- Client Setup
- Server Setup
- Caching

11.2 Setting Up Clients To Use Proxy

Clients can be set up to connect to the outside world through firewall gateways by setting the following environment variables:

- `http_proxy`
- `ftp_proxy`
- `gopher_proxy`
- `wais_proxy`

to URLs of a `cern_httpd` running as a gateway.

11.2.1 Example of a Script Starting a Client

```
#!/bin/sh
http_proxy="http://www.cern.ch:911/"
ftp_proxy="http://www.cern.ch:911/"
gopher_proxy="http://www.cern.ch:911/"
wais_proxy="http://www.cern.ch:911/"
export http_proxy
export ftp_proxy
export gopher_proxy
export wais_proxy
exec Mosaic
```

Synonyms for these variables are:

- `WWW_http_GATEWAY`
- `WWW_ftp_GATEWAY`
- `WWW_gopher_GATEWAY`
- `WWW_wais_GATEWAY`

11.3 Setting Up cern_httpd To Run as Proxy

Server runs as a proxy if its configuration file allows URLs starting with corresponding access method to be passed. Typical proxy configuration file reads:

```
pass http:*
pass ftp:*
pass gopher:*
pass wais:*
```

Note, that `cern_httpd` is capable of running as a regular HTTP server at the same time; just add your normal rules after those ones.

11.3.1 Caching

`cern_httpd` running as a gateway can also perform caching of files retrieved from remote hosts. See configuration file documentation for further details.

12. CERN Server FAQ

If you have problems, first make sure you're using the newest version. You'll find that out by peeking into ftp://info.cern.ch/pub/www/src.

When something goes wrong you should run server in verbose mode (the `-v` flag) to see exactly what is the problem. If you usually run it from inet daemon start it now standalone to some other port (with `-p` *port* flag) with otherwise the same parameters as in `/etc/inetd.conf.`

12.1 My Scripts Get Served As Text Files...

...or are completely unaccessible.

It's important to understand that rules in the configuration file (`Map`, `Pass`, `Exec`, `Fail`, `Protect`, `DefProt` and `Redirect`) are translated from top to bottom, and the first matching `Pass`, `Exec` or `Fail` will **terminate** rule translation.

So, make sure that your `Exec` rule is before any general `Map`pings.

12.2 How do I...

- Set up access authorization?
- Write server-side scripts?
- Get the server to perform searches?
- Make clickable images?
- Handle forms?
- Set up a firewall gateway (proxy)
- Set up firewall gateway caching

12.3 Zombies

If you are running `cern_httpd` standalone an see one zombie process, don't worry, it's normal. If there are lots of them it is a bug.

12.4 Inet daemon complains about looping...

...and terminates WWW service. `:-(`

This is a hard-coded `inetd` limitation on at least SunOS-4.1.* and NeXT, which limits maximum allowed connections from a given host to 40 per minute. This can be exceeded by scripts doing Web-roaming, or documents having masses of small inlined images.

There is a fix for at least SunOS `inetd` (100178-08), and in Solaris this is fixed. You can also run `httpd` standalone (preferably with the `-fork` command line option).

Most importantly, you should stop running `httpd` from `inetd` and rather run it standalone. This is because running from `inetd` is inefficient.

12.5 Server looks at funny directories and finds nothing

From version 2.0 until 2.15, you need to have an explicit map to file system in your rule file, e.g.:

```
Map    /*    file:/*
```

but 2.15 doesn't have this limitation anymore.

12.6 But the document says rule file is no longer needed

True, but it also says you must remember to give your Web directory as a parameter to `httpd,` e.g.

```
httpd  /home/me/MyGloriousWeb
```

13. CERN httpd 2.15 Release Notes

There is one single thing that needs to be done when changing over from `httpd` 2.14 to 2.15:

```
Rename your old /htbin scripts to end in .pp suffix!
```

13.1 General Notes

- Code tested under Purify –– all detected memory leaks and bugs fixed.
- Forking code enhanced –– no longer crashes when running standalone. Everybody should start running CERN httpd standalone instead of from inetd
- Documentation redesigned, but still under construction
- Contains Solaris port, but not VMS

13.2 CGI/1.0, Common Gateway Interface

- CGI/1.0 interface fully implemented
- **Old CERN httpd scripts will continue working if you rename them to end with .pp suffix.** Links referencing these scrips do NOT need to be changed. (This feature does not add any overhead to CGI/1.0 script calls.)
- New product cgiparse for CGI/1.0 scripts to parse QUERY_STRING env.var and to read CONTENT_LENGTH characters from stdin
- `htimage` upgraded to CGI/1.0
- The whole server-environment is propagated to CGI script, except for variables that are reserved for CGI/1.0.
- Scripts are spawned by doing a fork() and exec() instead of system() –– more efficient and secure

13.3 Firewall Gateway Modifications

- Access authorization works thru firewalls
- So does POST, therefore forms also
- -disable/-enable command line options and Disable/Enable configuration directives for dis/enabling HTTP methods. GET, HEAD and POST are enabled by default.
- Fix: text/html and text/plain not passed multiply to servers when running as gateway
- Fix: */*, image/* etc not expanded by the gateway
- Fix: try local search ONLY when accessing local files

13.4 Other New Features

- When started standalone in non-verbose mode automatically disconnects from terminal session and goes background
- User-supported directories enabling URLs starting with **/ũsername**
- Redirection
- Meta-information files to allow RFC-822-style headers to be appended to server response header section
- New, common logfile format, localtime default, `GMT` as an option
- Ability to suppress logging for certain hosts/domains according to given hostname template or IP number mask, like `*.cern.ch` or `128.141.*.*`
- -setuid option to set server uid to authenticated uid (local)
- Multilanguage support: same URL can be used to retrieve a document in different languages
- AddLanguage, AddEncoding and AddType directives to configuration file (AddType replaces Suffix)
- Better multiformat algorithm
- HostName directive to configuration file for servers that want to give CGI/1.0 scripts a different hostname than the actual. Useful if machine has many aliases, or if httpd fails to get the full domainname.
- Exec rule obsoliting HTBin directive −− now multiple script directories possible, with arbitrary mappings
- Get-Mask, Post-Mask and Put-Mask for protection setup files. Get-Mask obsolites Mask-Group
- Groups All/Users and Anybody/Anyone/Anonymous automatically defined. All means anybody that has been authenticated, and Anybody is just anybody
- Server:
- Last-Modified:
- Content-Length:
- Content-Language:
- Content-Encoding:
- Scripts can output also Uri: and Expires: headers (this will eventually be made more general)
- HEAD works, also with stupid scripts that also output the body

13.5 Enhancements, Fixes

- The final explicit Map to filesystem in configuration file no longer required, because it was causing confusion
- Assume Basic authentication scheme even if not explicitly mentioned in setup file
- Get client DNS hostname, for the logfile among other things
- Fail made the default when rules are translated to the end without coming accross with a Pass, Exec or Fail rule (this is to enhance security, it was too easy to forget the Fail * from the end of config file)
- Made config (rule) file understand different ways of writing keywords, e.g.: UserDir, userdir, User-Dir, user_dir, UserDirectory and so on
- The eight misplaced server-side access authorization files moved away from libwww
- Fix: directory indexing works with a trailing slash
- Fix: HTSimplify() might have behaved unexpectably on some systems (called strcpy() with overlapping args)

14. CERN httpd 2.16beta Release Notes

- If you are upgrading from 2.15beta, you need to make **no changes**.
- If you are upgrading from 2.14, there is one single thing that needs to be done:

```
Rename your old /htbin scripts to end in .pp suffix!
```

14.1 Firewall Gateway (Proxy) Additions, Fixes

- `ftp` with binary files work
- `x-compress` and `x-gzip` work correctly over proxy
- Firewalling now works through arbitrary number of proxies; `http_proxy, ftp_proxy, gopher_proxy` and `wais_proxy` configuration directives cause proxy to connect to the outside world through another proxy. Environment variables with the same names have same effects, but config file is user-friendlier for this.
- Now sends all the headers sent by client.
- Proxy log file now gives byte count.
- Proxy log file now gives correct status code also on error.

14.2 Firewall Gateway (Proxy) Caching

- `CacheRoot` directive specifies cache root directory, and turns on proxy caching. Cache root directory must be dedicated to `httpd` - all files in there are subject to garbage collection.
- Cache size (in megabytes) is specified by `CacheSize` directive; cache size should be several megabytes, 50-100MB should give good results. Cache may, however, temporarily grow a few megabytes bigger than specified. Also, space taken up by directories is not calculated in the current version.
- `http, ftp, gopher` with `GET` method get cached.
- However, not caching:
  - HTTP0 responses (you never know if it failed; also confused HTTP1 servers sometimes output garbage in front of HTTP1 headers).
  - Protected documents (request had `Authorization:` field).
  - Queries - they have too often side-effects. (POST should be **always** used with forms, and all script responses should have `Expires:` header when necessary. Until then, we don't cache them.)
- Expiry date is extracted:
  - From `Expires:` header.
  - If not present `Last-Modified:` is used to approximate expires. If a file hasn't changed in five months the chances are it won't change during the next week. On the other hand, if a file has changed yesterday, it will probably change again pretty soon. I know this is heuristic but until all the servers give `Expires:` this works much better than not using it, so no flames about it.
  - If `Last-Modified:` not given use the time given by `CacheDefaultExpiry` directive, default 7 days.

- Format of cache files and directory structure under cache root is subject to change if necessary. No application should yet rely on any certain cache format. Eventually I can see clients accessing cache files directly, bypassing proxy server.
- Caching system understands both time formats, also the one output by old NCSA httpds.
- Cache files get locked during transfer. Lock files time out if something goes wrong. Timeout can be set by `CacheLockTimeOut` directive (default 20 minutes). During the lock is in effect, further requests to the same file get retrieved from the remote host.
- Garbage collection directives:
  -
  - `GcMemoryUsage` to advice gc about how radical to be in memory use (more memory => smarter gc).
  - `GcTimeInterval`, how often to do gc.
  - `GcReqInterval`, after how many requests to do gc.
  - (gc is also automatically started if cache size limit is reached.)
  - `CacheLimit_1`, size in KB until which files are equally valuable despite their size (200K).
  - `CacheLimit_2`, size in KB after which files get discarded because they are too big (4MB).
  - `CacheClean`, remove all files older than this (default 21 days).
  - `CacheUnused`, remove all files that have not been used in this long time (default 14 days).
- Garbage collector always removes all expired, too long unused, and too old files.
- If cache size limit is reached some files need to be sacrified; the current algorithm takes into account:
  - Time remaining to unconditional removal; if it expires tomorrow it might as well be removed today.
  - Time last accessed; if it hasn't been accessed in 5 days, it probably won't be accessed anymore before it expires.
  - Size; huge files get removed move easily.
  - Time it took to load it from the remote host; files that were time-consuming to transfer have much higher value. This compensates the size factor. Load delay is the single most significant value.
  - Time it has already been in cache; ancient files get removed more easily than fresh ones.

14.3 Other New Features

- Error log file.
- `Referer:` field ends up in error log when a request fails.
- `UserId` and `GroupId` to set default uid and gid (used instead of nobody and nogroup).
- Timeout for input and output; default time to wait for a request is 2 minutes, and to send response 20 minutes. Timeout causes a note to error log, and terminates child (no more hanging httpds). **Note:** the one zombie is normal; don't report to me about it, I may do something about it some day, or maybe I won't. Zombie doesn't take up any other system resources except the one process table entry.
- Suffixes are no longer case-sensitive by default; this may be changed via the `SuffixCaseSense` configuration directive.

- Lou Montulli's news and proxy diffs added to the library.
- Most command line options now also available as configuration directives:
  - `DirAccess`
  - `DirReadme`
  - `AccessLog`
  - `ErrorLog`
  - `LogFormat`
  - `LogTime`
- `-vv` command line option for Very Verbose trace output. Outputs also request headers as they came in. Otherwise like `-v` flag.

14.4 Enhancements, Fixes

- NPH-scripts now work from automatically backgrounded standalone server.
- Fixed the many problems with `Content-Transfer-Encoding`:
  - Mosaic uses `Content-Encoding`, although spec says `Content-Transfer-Encoding`; I now output both
  - `Content-Transfer-Encoding` sometimes didn't show up although it should have, fixed.
  - `Content-Transfer-Encoding` didn't come up correctly with ftp, fixed.
- Strange escaping fixed with directory indexing (legal characters got escaped randomly by a gcc-compiled version).
- Timezone bug around midnight with the new logfile format fixed. (New logfile format is not yet default, use `-newlog` command line option, or `LogFormat` directive in configuration file.)
- Dashes for non-existent status codes and byte counts now show up correctly in the log.
- Forking code once again enhanced - fixed a possible hanging situation.
- Log time fixed to be the time of incoming request, not the time of request served.
- Zombies now correctly waited away on HP (this was in fact fixed already in 2.15beta binaries distributed after February 17th - **note,** that this bug had no effect on any other platforms).
- Directory listings no longer have `Content-Length:` (because it was wrong).
- Now understands also the old Accept: syntax, with spaces as separators between actual content-type and its parameters. This will eventually be taken out.
- `htadm` now uses the same file creation mask as in the original password file.