NAME

http-analyze – a fast log analyzer for web servers

SYNOPSIS

http-analyze [**–{hdmV}**] [**–3aefnvxy**] [**–c** *cfgfile*] [**–o** *outdir*] [**–p** *privdir*] [**–s** *opt,...*]
[**–t** *num,...*] [**–u** *time*] [**–w** *hits*] [**–F** *format*] [**–G** *suffix,...*] [**–H** *idxfile,...*]
[**–I** *date*] [**–E** *date*] [**–O** *virtname,...*] [**–P** *prolog*] [**–R** *docroot*]
[**–S** *srvname*] [**–T** *tldfile*] [**–U** *srvurl*] [**–W** *3Dwin*] [*logfile* [...]]

DESCRIPTION

http-analyze analyzes the logfile of a web server and creates a detailed statistics of the servers's access load and it's responses in graphical and tabular form. In auto-sense mode (default), **http-analyze** tries to recognize the logfile format automatically. Supportet formats for logfiles are the *common logfile format (CLF)* and two forms of *extended* or *combined logfile formats*, which are basically common format plus user-agent and referrer URL. Those formats are used by most popular web servers such as the Netscape Enterprise server, the NCSA httpd, the Apache server, the Ximati server and many others.

http-analyze has been highly optimized to process large logfiles at maximum speed. There are two modes of operation with different levels of detail in the logfile analysis:

Short statistics ("daily" mode, option **–d**):

> **http-analyze** generates a short summary of server usage per day. In this mode, it uses a history file to skip entries which have been processed already. By avoiding detailed analysis of the logfile entries, **http-analyze** requires only a fraction of the time which would be required to generate a full statistics report.

Full statistics ("monthly" mode, option **–m**):

> In this mode, the analyzer generates a full report for a whole month, which creates much more details than in the short statistics mode. The history file is used to produce a summary for the last 12 months without having to analyze the logfiles for those previous periods again. This is the default if no mode is specified explicitely.

Usually, you run **http-analyze** in full statistics mode only, since this also includes the short statistics. However, if your logfiles are rather large and if the analyzer causes significant load while generating the full statistics, you could run it in short statistics mode using very short update intervals (in the range of 30 minutes to some hours) to create an up-to-date statistics, and then run it in full statistics mode less often (for example once per day, per week or even per month) to generate a detailed report. The operation modes have been named after their maximum useable update intervals, namely "daily" and "monthly" mode for the short and full statistics respectively.

http-analyze maintains a history of the results from previous periods. In short statistics mode, the history contains the daily summary from the first day of the current month until the previous day (yesterday). In full statistics mode, the history contains only summaries for previous months, not for the current one. This means that you should rotate the logfile at the first day of a new month after having generated a full statistics report for the previous month.

If disk space is a concern, you can set up a scheme where the logfiles are rotated once per week or even once per day. In this case, to generate a full statistics report you have to feed **all** logfiles for a month into **http-analyze** in ascending order of the date. If a detailed report for a previous month has been generated, you can save the corresponding logfile(s) somewhere and remove them from your production system.

LOGFILE FORMATS

http-analyze understands the three most important logfile formats:

Common Logfile Format (CLF)

This format is supported by most web servers. The entries contain following information:

```
dns-name - auth-user [date] "clf-request" clf-status ct-length
```

where the fields have following meaning:

dns-name The full qualified domain name (FQDN) of the host accessing the server. If there is no FQDN available for the host, the IP number is logged by the server.

– Unused.

auth-user The user ID provided by the client to access documents which require the user to authenticate itself.

[date] The date of the access as `[DD/MMM/YYYY:HH:MM:SS ±ZZZZ]`.

"clf-request" The request in format `"method URI proto"`, where *method* is one of **GET**, **HEAD**, **POST**, **PUT**, **BROWSE**, **OPTIONS**, **DELETE** or **TRACE**; *URI* is the *Uniform Resource Identifier*, and *proto* is the protocol parameter containing the HTTP version.

clf-status This is the numerical response code from the server.

ct-length This number reflects either the size of the document or the data actually sent over the wire depending on the server.

Combined Logfile Format (DLF)

Some server use the so-called *Combined Logfile Format* to add the referrer URL and user-agent (browser) to the logfile entries. It looks like the CLF format followed by the referrer URL and the user-agent, where the latter two fields are surrounded by double quotes:

```
CLF "referrer_URL" "user_agent"
```

Unfortunately, the double quotes sometimes appear in broken referrer URLs as, for example, in

```
"http://www.some.host/document.html TARGET=newwin""
```

Since sometimes there are even referrer URLs which contain double quotes followed by blanks, those entries are not pareseable in an unambiguous way. Although **http-analyze** recognizes the *Combined Logfile Format* automatically and tries to do it's best to parse the referrer URL correctly, the following format, which avoids this ambiguity should be preferred if possible.

Extended Logfile Format (ELF)

The *Extended Logfile Format* contains also the user-agent and the referrer URL, but in the opposite order and without the surrounding double quotes:

```
CLF user_agent referrer_URL
```

If this *Extended Logfile Format* is used, **http-analyze** searches backwards for the protocol specification of the referrer URL (to be precise, it looks for the colon in **http:**) and then for the preceeding blank. This way, even broken referrer URLs which contain blanks are handled correctly in nearly every case. To select this format, just edit the configuration file of your web server manually and reverse the order of the user-agent and referrer URL fields in the logfile format specification (see the Online Documentation of **http-analyze** for more examples).

SUMMARY REPORTS

Starting with version 2.0, **http-analyze** generates all reports for a year in separate subdirectories to reduce the number of files created for the reports. Those subdirectories are named **www***YYYY*, where *YYYY* is the year of the corresponding period. All filenames listed below are relative to the appropriate subdirectory unless stated otherwise. If you upgrade from the 1.9e version, use the *cvt_files* script included in the distribution to convert old report files into the new directory structure (see the file *INSTALL* for an example how to do this).

There are two user interfaces to the reports, a conventional interface as in previous versions of the analyzer and a frames-based interface. Although the frames-based interface is the preferred method to browse through the statistics report, the navigation in the non-frames version has been improved in **http-analyze 2.0** by optionally using JavaScript for navigation control and different windows for the display of the reports:

Main window
: This window is used by default for most reports such as the yearly, monthly, daily and weekly summaries, the *Top N* lists or the overviews. If there are hotlinks in the *Top N* lists, they are most often links to the corresponding pages or sites, which will show up in the *Viewer window* if those links are followed. The hotlinks in the overviews point to the detailed lists, which are displayed in the *List window*.

Navigation window
: If JavaScript is enabled in your browser and a summary for a year or a month is loaded in the main window, a small window containing a navigation panel will pop up. If JavaScript is disabled, the navigation panel appears at the bottom of the full summary for the corresponding month in the *Main* window. In this case, use the *Back* button of your browser to go back to the navigation panel when browsing through the reports.

List window
: This window is used for the complete lists of URLs, sites, browser types and referrer URLs. Displaying them in a separate window allows easy navigation and causes this rather large lists to be loaded only once if navigated through by following the links in the overviews.

Viewer window
: This window is used for external pages which are referred to through the hotlinks in the statistics reports. This way, you can visit the pages shown in the reports without having to go forth and back between the summary and the pages listed there.

3D window
: This window is used for the 3D (VRML) model of the statistics. If you have JavaScript enabled, the window's size will be set to the smallest possible size so that the 3D model fits. You can specify the dimensions of this window in the configuration file using the **3DWinSize** directive.

Do not close the windows if you switch reports, they will be re-used for the appropriate lists.

Short statistics mode

In short statistics ("daily") mode, **http-analyze** writes the summary into the output file **stats.html** and updates the daily summaries in the history file **www-stats.hist**. The short statistics report includes the following information (see the section *Interpretation of results* for an explanation of the numbers):

- the number of *hits* per day,
- the number of *files* per day,
- the number of *pageviews* per day,
- the number of *sessions* per day,
- the amount of *data sent* (KBytes) per day.

Full statistics mode

In full statistics ("monthly") mode, **http-analyze** updates the short summary in **stats.html** and additionally creates the following files:

index.html
: in a **www***YYYY* directory is the main page for a given year and contains the total number of *hits*, *files*, *pageviews*, *sessions* and *data sent* per month in tabular and graphical form for the last 12 months. At the end of the year, this file reflects the values for the whole year, while the values for the last 12 months will be written into another index file in a new directory **www***YYYY*. This page is displayed in the *Main window*.

stats*MMYY***.html** and **totals***MMYY***.html**
: contain the total summary for the month *MM* of year *YY* in tabular form. The file **totals***MMYY***.html** is the frames version of the report in **stats***MMYY***.html**. In the conventional interface, this page is shown in the *Main window*.

jsnav.html and **nav***MMYY***.html**
: Navigation panels for JavaScript-capable browsers, shown in the *Navigation window*.

days*MMYY***.html**
: contains the number of hits, files, pageviews, sessions and data sent per day for the month *MM* of year *YY*. This report is shown in the *Main window*.

avload*MMYY***.html**
: shows a graphical representation of the average hits per weekday/hour and the top seconds, minutes, hours, and days of the current period. This report is displayed in the *Main window*.

country*MMYY***.html**
: contains the list of all countries the visitors of your web server came from. This information is determined by analyzing the *top-level domain (TLD)* of the hostname.

  If you have disabled domain name lookups in your web server to decrease response times of your server or if the host isn't configured in the Domain Name System (DNS) for whatever reason, **http-analyze** cannot determine the country a visitor is coming from. All hosts without a name will show up as *Unresolved* in the country list. Note: Sometimes, systems are intentionally not configured in the DNS, so a percentage of up to 30% for unresolved IP numbers is absolutely normal. The country report shows up in the *Main window*.

3Dstats*MMYY***.html**, **3Dstats***MMYY***.wrl.gz**, **3Dstats***YYYY***.html**, **3Dstats***YYYY***.wrl.gz**
: are pre-requisites for the 3D models of the statistics in the *Virtual Reality Modeling Language (VRML)*. Those models are created if the option **−3** was given at the invocation of **http-analyze**.

  The yearly models are suitable only on graphic workstations, therefore they are created only if a prolog file is specified using either the option **−P** or the **VRMLProlog** directive in the configuration file. The file **3Dprolog.wrl** is provided as an example of such a world. You can build your own worlds with embedded monthly models by creating your own prolog file. For the PC platform, the yearly model file is replaced by a stubs file named **3Dlogo.wrl.gz** also included in the distribution.

  To view any of those models, you need a VRML 2.0 compatible plug-in such as the free *CosmoPlayer* from Cosmo Software, which is currently available for Netscape Navigator and MSIE on IRIX and Win95/WinNT platforms. See *http://cosmo.sgi.com/* for more information about Cosmo Software. 3D models show up in the *3D window* so that it can be compared to the graphs in the conventional reports.

topurl*MMYY***.html**, **topdom***MMYY***.html**, **topuag***MMYY***.html**, **topref***MMYY***.html**

Those files contain the *Top Ten* lists (actually it's *Top N*, where *N* is a configurable number) of the files requested, the domains, the browser types and the referrer URLs. The URLs shown in **topurl***MMYY***.html** are either the real URLs requested by the visitor or an *item* (arbitrary text) you choosed to collect certain file names under (see the **HideURL** directive in the configuration file).

The domains shown in **topdom***MMYY***.html** are either the second-level domains of the hosts accessing your server if the DNS name is available or an item you choosed to collect certain hostnames under (see the **HideSys** directive in the configuration file). Unresolved IP numbers show up as *Unresolved* again.

The browser types in **topuag***MMYY***.html** are the different browser types *(user agents)* which have been used by the users to access your web site. If possible, **http-analyze** reduces the name of the browser to the model including the firsdt digit of the version number in this list. Otherwise, the full name as sent by the browser is used.

The referrer URLs are the URLs of those web pages, which have a link to some page on your server, and which have been visited by the user before following them to reach your site. If the user did specify an URL manually in his browser, no referrer URL is logged. Also, the browser can choose to not send a referrer URL anyway. The referrer URL reports are displayed in the *Main window*.

files*MMYY***.html**, **sites***MMYY***.html**, **agents***MMYY***.html**, **refers***MMYY***.html**

Those files contain a complete overview of the files requested, the domains, the browser types and the referrer URLs, similar to the Top N lists.

lfiles*MMYY***.html**, **lsites***MMYY***.html**, **lagents***MMYY***.html**, **lrefers***MMYY***.html**

Those files contain the complete lists of all files requested, all domains, all browser types and all referrer URLs, similar to the previous reports, but sorted by item (if any) and hits. On frequently accessed sites, this lists can become rather large, so they are shown in the *List window*. If you follow the links in the overviews, the appropriate items will be shown in this window so that the lists have to be loaded just once.

rfiles*MMYY***.html**

contains all invalid URLs which caused the server to respond with a *Code 404 (Not found)* status. If there are large number of hits for certain files the server couldn't find, it's probably due to missing inline images or other HTML objects embedded in other pages. This report is displayed in the *Main window*.

rsites*MMYY***.html**

This file contains a list of reverse domains sorted by the top-level domain. This report is shonw in the *Main window*.

frames.html, **header.html**

This two files are required for the frames-based user interface. All other files are shared with the ones for the non-frames UI. In the frames-based UI, the *Main* window is inside the frame, while the *List* window is still an external window. The *3D window* may be inside the frame or an external window (see the **3DWindow** directive).

gr-icon.gif

This is a small icon displayed on the main page under the root directory for the statistics reports (option **–o** or the **OutputDir** directive in the configuration file), which is used as a switchboard to the various statistics (currently WWW stats only; more coming soon.).

If no output directory has been specified when **http-analyze** was run, the reports are created in the current directory. The files containing the detailed lists of files, hosts, browser types, and referrer URLs may optionally placed into a "private" subdirectory to be able to protect them by server authentication (see the option **–p** and the **PrivateDir** directive in the configuration file).

INTERPRETATION OF RESULTS

The statistics report contains among others the following information:

- the number of hits, 304's, files, pageviews, sessions, data sent (in KB)
- the amount of data requested, transferred, and saved by cache (in KB)
- the number of unique URLs, sites, and sessions per month
- the number of all response codes other than 200 (*OK*)
- the average hits per weekday and for last week
- the maximum/average hits per day and per hour
- the number of hits, files, 304's, sites, data sent by day
- the top 5 days, 24 hours, 5 minutes and 5 seconds of the summary period
- the top 30 most commonly accessed URLs (hits, 304's, data sent)
- the 10 least frequently accessed URLs (hits, 304's, data sent)
- the top 30 client domains accessing your server most often
- the top 30 browser types
- the top 30 referrer hosts
- the overview/detailed list of all files requested
- the overview/detailed list of all sites by domain and reverse domain
- the overview/detailed list of all browser types
- the overview/detailed list of all referrer URLs

The following section describes the meaning of all those numbers in the summary report which are not self-explaining:

Hits (color key: green) A hit is any response from the server on behalf of a request sent from a browser. This includes **any** response from the server, not only text files or documents. If, for example, a HTML page has two images embedded, the server generates three hits if this page is requested: one hit for the HTML page itself and two hits for the two inline images.

Files (color key: blue) If the user requests a document and the server successfully sends back a file for this request, this is counted as a Code 200 (*OK*) response. Any such response is counted for as a file. Again, "file" here means any kind of a file.

Code 304 (Not Modified)

(color key: yellow) A Code 304 (*Not Modified*) response is generated by the server if a document hasn't been updated since the last time it was requested by the user and therefore there was no need to actually send the files for this document. This happens if the browser (or a caching proxy server between the browser and your web server) still has an up-to-date copy of the page in it's local storage (cache) and therefore can display the page without requesting the actual content. This technique is used to reduce network traffic, but it also causes an inaccuracy in the statistics reports regarding the number of visitors, because the browser or proxy usually sends only one such a conditional request per user session if it still holds an up-to-date copy of the file. However, the ratio between "files" and "304's" reflects the efficiency of overall caching mechanisms for at least those hits which made it's way to the server.

Pageviews

(color key: magenta) Pageviews are all files which either have a text file suffix (*.html*, *.text*) or which are directory index files. This number allows to estimate the number of "real" documents transmitted by your server. If defined correctly, the analyzer rates text files (documents) as pageviews. Those pageviews do not include images, CGI scripts, Java applets or any other HTML objects except all files ending with one of the predefined pageview suffixes, such as **.html** or **.text**. See also the **Pageview** directive.

Other responses

There are much more responses than only Code 200 *(OK)* and Code 304 *(Not Modified)* responses, especially in the coming standard, the HTTP 1.1 protocol specification. For example, the server could generate a Code 302 *(Redirected)* response if a page has moved, a Code 401 *(Unauthorized Request)* response if access to the document is denied or a Code 404 *(Not Found)* response if the requested page does not exist on this server. See the HTML specification at *http://www.w3.org/* for information about all valid responses from a web server. Note that **http-analyze** does recognize HTTP/1.1 responses according to RFC 2068.

KBytes transferred

(color key: orange) This is the amount of data sent during the whole summary period as reported by the server. Note that some servers do log the size of a document instead of the actual number of bytes transferred. While in most cases this is the same, if a user interrupts the transmission by pressing the browser's stop button before the page has been received completely, some servers (for example all Netscape web servers) do not log the amount of data transferred but the amount of data which *would* have been transferred if the user would have completely loaded the page.

KBytes requested

This is the amount of data requested during the whole summary period. **http-analyze** computes this number by summing up the values of *KBytes transferred* and *KBytes saved by cache* (see below).

KBytes saved by cache

The amount of data saved by various caching mechanisms such as in proxy servers or in browsers. This value is computed by multiplying the number of Code 304 (*Not Modified*) requests per file with the size of the corresponding file. Note: Because **http-analyze** can determine the size of a file only if the file has been requested at least once in the same summary period, the values for *KBytes saved by cache* and *KBytes requested* are just approximations of the real values.

Unique URLs

Unique URLs are the number of all different, valid URLs requested in a given summary period. This shows you the number of all different files requested at least once in the corresponding summary period.

Unique sites

This is the sum of all unique hosts accessing the server during a given *time-window* . The time-window is hardwired to the length of the current month. This means that if a host accesses your server very often, it gets counted only once during the whole month. Only the sum of the unique hosts per month is listed in the statistics report.

Sessions

(color key: red) Similar to unique sites, this is the number of unique hosts accessing the server during a given *time-window*. This time-window is one day by default for backward compatibility, but it can be changed with the option **–u** or the **Session** directive in the configuration file. For example, if the time-window is two hours, all accesses from a certain host in less than 2 hours after the first access from this host are lumped together into one session. All following accesses more than 2 hours apart from the first access will be counted as a new session. This way you may get an estimated number of how many sessions are started on different sites to access your server.

OPTIONS

–h print a short help list explaining the usage of the options. Use **–hh** to print an even more detailed help.

–d *(daily mode)* generate a short statistics report for the current month only. If a history file exists, the values for the previous days will be read from this history file and the corresponding logfile entries are skipped. If the history file does not exist, the whole logfile will be processed and a history file will be created (unless **–n** is also given).

–m *(monthly mode)* generate a full statistics report for a whole month. Although the values from the history file are used usually to create a summary page for the last 12 months, the actual logfile entries always have preceedence over any records in the history file unless the option **–e** is also given. The option **–m** includes **–d**

–V *(version)* print the version number of **http-analyze** and exit immediately.

–3 create a 3D (VRML) model of the statistics in addition to the tabular reports. To view such a model, you need a VRML 2.0-compatible plug-in like *CosmoPlayer* from Cosmo Software, which is currently available for IRIX and Win95/WinNT platforms.

–a ignore all URLs which required authentication. If your statistics report is available to the public, you probably do not want to have secret URLs listed in the report.

–e use the history file even if it contains expired data. If this option is present and you analyze the log entries for several months at once (either in different files or in one single logfile), **http-analyze** uses the values recorded in the history file for previous months which are present there and therefore skips all logfile entries up to the first day of a month not recorded in the history (usually the current month). This is useful if you rotate your logfile once per quarter and want to have the analyzer skip all entries for a previous month which has been completely processed already some time before.

–f create a frames-based user interface for the reports in addition to the conventional (non-frames) interface. To view this frames-based version of the report, your browser must support JavaScript.

–n *(no update)* do not update the history file. Useful to generate statistics for previous months (before the last month) without overwriting the current state of the history. Since the history is used to create the report for the last 12 months, this option can be used to not mess up the actual statistics report when analyzing an older period. On the other side, the numbers for this last 12 months on the main page for the current year are not updated, while the numbers on the main page of the corresponding year being analyzed will be updated according to the actual logfile entries read by **http-analyze**.

–v (verbose) comment ongoing processing. Warnings are printed only in verbose mode. If your statistics report contains zero hits, try this option to see whether the logfile were corrupted or whether they are in unrecognized format. If you double **–v**, **http-analyze** will print a dot for each new day discovered in the logfile.

–x don't comprise images under the item "All images", but list any filename literally. Normally, **http-analyze** collects the values of all images *(*.gif, *.jpg, *.ief, *.pcd, *.rgb, *.xbm, *.xpm, *.xwd, *.tif)* under the item "All images" to avoid cluttering up the lists with lots of image URLs. If **–x** is given, each image URL is listed literally unless matched by an explicit **HideURL** directive in the configuration file.

–c *cfgfile*
use *cfgfile* as the configuration file. By using a configuration file, **http-analyze** allows you to pre-define frequently used options and to define the grade of details in the reports. See the section *Configuration File* for a description of the configuration file settings, which are called *directives* in the following text.

–o *outdir*
: use this directory to create the HTML files of the report in. If no directory is specified, the files are created in the current directory. See also the **OutputDir** directive.

–p *privdir*
: place the detailed list of URLs, sites, browsers and referrer URLs into this directory. Useful if you grant public access to your statistics reports and want to restrict access to certain lists to your staff only. You need to define an authentication scheme in your web server for this to work correctly. **http-analyze** just places the output files in this subdirectory. If the name of the directory does not start with a '/', it is considered relative to the output directory specified with **–o**. See also the **PrivateDir** directive.

–F *format*
: use this logfile format. Valid values for *format* are **auto** for auto-sensing the logfile format, **clf** for the *Common Logfile Format*, or **dlf** and **elf** for the two supported forms of the *Combined/Extended Logfile Format*. See the section *Logfile Formats* above for a description of the formats supported by **http-analyze**.

–G *suffix,...*
: rate files ending with *suffix* as pageviews (text files). The suffix **.html** is pre-defined by default. You can add 9 more suffixes here, for example **.shtml**, **.text** and **.htm**. The suffix must start with a '.' (dot). Note that each suffix will require another lookup in this table at an early stage of processing, so many suffixes will increase the processing time of **http-analyze** by a significant amount. To specify more than one suffix with a single **–G** option, use commas to separate them. See also the **PageView** directive.

–H *idxfile,...*
: define an additional directory index filename other than *index.html*. **http-analyze** truncates the URLs containing an index filename so that they merge with '/' or their "base URL", respectively. For example, the "base URL" of */dir/index.html* is */dir/*. The index filename **index.html** is pre-defined already. You can add 9 more names for directory index files here, for example *Welcome.html*, *home.html* or any other filename defined in the web server's configuration file. Note that each name will require another lookup in this table at an early stage of processing, so many names will increase the processing time of **http-analyze**. See also the **IndexFiles** directive.

–I *date*
: skip all logfile entries until this day (exclusive). The date may be specified as *DD/MM/YYYY or MM/YYYY* , where *MM* is the number or the name of a month. Note that in full statistics mode, *DD* defaults to the first day of the month if absent. If you specify any other day in this mode, unpredictable results may occur. For example, `–I Feb` restricts the analysis to the February of the current year.

–E *date*
: skip all logfile entries starting from this day on (inclusive). The date format is the same as in **–I**. To restrict analysis to a certain period, specify the starting date using **–I** and the first date to be ignored using **–E**. For example, `–I Jan/98 –E Feb/98` restricts the analysis to January 1998.

–O *virtname,...*
: define additional virtual names for this server. **http-analyze** uses those names to hide certain referrer URLs *(the self referrer URLs)* in the statistics report. The server's primary name is pre-defined already. See also the **VirtualNames** directive.

–P *prolog*
: use *prolog* as the prolog file for a yearly VRML model (optional). The file **3Dprolog.wrl** is included in the distribution as an example. Note that the resulting VRML model for a whole year is suitable only for viewing on a graphic workstation. The monthly VRML models do not need a prolog file and can be viewed on any platform without problems. See also the **VRMLProlog** directive.

–R *docroot*

restrict the logfile analysis to the given Document Root. Intented for use with (software-) virtual servers which have an own subdirectory as their document root. If *docroot* is prefixed by a '!', the analysis is restricted to all subdirectories of the "real" server except for *docroot*. There may be only one directory name given at any time. See also the **DocRoot** directive in the configuration file.

–S *srvname*

use *srvname* for the server name. Useful if the analyzer runs on another system than the web server is running on. If no server name is defined, **http-analyze** uses either the *uname (2)* or the *gethostname (2)* function to determine the name. On most Unix System V platforms, *uname* returns the nodename only (for example, *host*), while *gethostname* usually returns the full qualified domain name (FQDN, for example, *host.my.domain*) if the DNS is set up properly. See also the **ServerName** directive in the configuration file.

–T *tldfile*

use *tldfile* for the list of valid top-level domains (TLDs). This list currently includes all ISO two-letter country domains, the well-known domains **.net**, **.int**, **.org**, **.com**, **.edu**, **.gov**, **.mil**, **.arpa**, **.nato**, and the upcoming new *CORE* top-level domains **.firm**, **.info**, **.shop**, **.arts**, **.web**, **.rec**, and **.nom**. The length of a top-level domain in the TLD file may not exceed 6 characters (if you need to add longer domain names, use the **AddDomain** directive in the configuration file). **http-analyze** uses it's built-in defaults, if no TLD file is given. See also the **TLDFile** directive and the sample file **TLD** included in the source distribution.

–U *srvurl*

define *srvurl* as the server URL which should be used as a prefix for the hotlinks in the URL list. Useful if the statistics report is created on a different system than the server is running on and for virtual hosts. See also the **ServerURL** directive.

–W *3Dwin*

define the window for the VRML model. The keyword *3Dwin* may be either **extern** or **intern** for display of the VRML model in a new, external window or in the lower half of the main frame respectively (meaningful only in the frames-based interface).

–s *opt,...*

suppress certain lists in the report. *opt* may be one of:

AVLoad	to suppress the average load report (top seconds/minutes/hours),
URLs	to suppress the overview of URLs/items and the detailed lists,
URLList	to suppress the detailed list of URLs,
Code404	to suppress the detailed list of Code 404 *Not Found* responses,
Sites	to suppress the overview of domains and the hostname lists,
RSites	to suppress the overview of reverse domains,
SiteList	to suppress the detailed list of hostnames,
Agents	to suppress the overview of browser types and the detailed lists,
Referrer	to suppress the overview of referrers and the detailed lists,
Country	to suppress the country list,
Pageviews	to suppress pageview rating (304's are shown instead),
Graphics	to suppress all graphs and pie charts,
Hotlinks	to suppress the hypertext links to the pages in the URL lists,
Interpol	to suppress interpolation of the graphs.

You can specify more than one *opt* with a single **–s** option by separating them with a ',' as in: `–s Agents,Referrer,Hotlinks`. See also the **Suppress** directive.

–t *num* define the size of certain lists. *num* is either a positive number or the value 0 to suppress the corresponding list. You specify the list by appending one of the following characters to the number shown here as '*#*' (note that the characters are case-sensitive):

*#***U**	set the number of entries in the Top N URL list (default: 30),
*#***L**	set the number of entries in the least N URL list (default: 10).
*#***S**	set the number of entries in the Top N domain list (default: 30),
*#***A**	set the number of entries in the Top N agent/browser list (default: 30),
*#***R**	set the number of entries in the Top N referrer URL list (default: 30),
*#***d**	set the number of entries in the Top N days table (default: 5),
*#***h**	set the number of entries in the Top N hours table (default: 24),
*#***m**	set the number of entries in the Top N minutes table (default: 5),
*#***s**	set the number of entries in the Top N seconds table (default: 5),
*#***F**	set the font size for text in detailed lists (default: 2),
*#***H**	set the font size for headers in detailed lists (default: 3).
*#***N**	set the size of the navigation frame (default: 120)

The list of least frequently accessed URLs is generated only if the number of all unique URLs is greater than the sum of the entries in the top URL lists regardless of the list's size actually defined. Use commas to sepcify more than only one *num* per **–t** option. See also the directives beginning with **Top*** in the configuration file.

–u *time* define the time-window for counting *sessions*. See *Sessions* in the section *Interpretation of results* for an explanation of this term.

–w *hits* set the noise-level to *hits*. If a noise-level is defined, all URLs, sites, agents and referrer URLs with hits below this level are collected under the item *Noise* in the Top N lists and overviews to avoid cluttering up those reports. See also the **NoiseLevel** directive.

logfile(s)
This are the name(s) of the logfile(s) to process. If more than one file is given, they are processed in the order in which their names appear on the command line. **http-analyze** checks for the existance of all files before processing them. If a '-' is specified as the filename, standard input is read. If no file is given, the analyzer either processes the default logfile specified in the configuration file or the standard input.

CONFIGURATION FILE

The option **–c** allows to define a configuration file which contains pre-defined, server-specific default settings for **http-analyze**. Command line options always take preceedence over the definitions in this configuration file. The configuration file contains a single directive per line. Except for **IndexFiles**, **PageView**, **AddDomain**, **Ign***, **VirtualNames**, and **Hide***, each directive may appear only once in the configuration file.

Following a directive field there are one or two value fields, which must be separated from the directive and each other by one or more tabulators. Blanks are considered a part of the string for the third field only if there is such a field. All directive names are case-insensitive.

3DWinSize *width*×*height*
Defines the size of the 3D window. Useful for Netscape Navigator 3.X, which displays scrollbars in the 3D window with standard size (520×420 pixels). Example:
```
3DWinSize        540x450
```

3DWindow *keyword*
Defines the 3D window the VRML model is displayed in (same as option **–W**). The *keyword* may be **extern** (default) or **intern** for display of the VRML model in a new, external window or in the lower half of the main frame respectively. Example:
```
3DWindow         intern
```

AddDomain *domain string*

Add names to the domain table causing certain *domains* to be allocated to some "artificial" top-level-domain *string*. Do not use wildcards here, they are ignored anyway. This directive is useful to collect certain hostnames, for example the local network or the hosts of world-wide operating online services, under some arbitrary *string* (item) instead of under the country they seem to originate from. Example:

```
AddDomain        .compuserve.com CompuServe
```

CustLogoW *image srvurl* and **CustLogoB** *image srvurl*

Define images for use as customer logos in the statistics report. This feature is available only in the commercial version of the analyzer. You will have to create two logos, approx. 72×72 pixels in size, one for use on pages with white background (**CustLogoW**) and another one for use on pages with black background (**CustLogoB**). Example:

```
CustLogoW  btn/mycompany_sw.gif   http://www.mycompany.com/
CustLogoB  btn/mycompany_sb.gif   http://www.mycompany.com/
```

DefaultMode *mode*

The default operation mode of **http-analyze**. The value field contains either the keyword **daily** or **monthly** for short or full statistics summaries (see also options **−d** and **−m**). If left undefined, the default is now **monthly**. Example:

```
DefaultMode      daily
```

DocRoot *docroot*

Restricts logfile analysis to the given Document Root (same as option **−R**). Intented for use with (software-) virtual servers which have their documents under a certain subdirectory. If *docroot* is prefixed by a '!', analysis is restricted to all directories except for this subdirectory. There may be only one directory name given at any time. Example:

```
DocRoot          /customer/
```

FontSize *size* and **HeadSize** *size*

The font size for text (default: 2) and headers (default: 3) in detailed lists. Example:

```
FontSize         3
HeadSize         4
```

HTMLPrefix *prefix* and **HTMLTrailer** *trailer*

The HTML *prefix* and *trailer* to be printed after the header section and at the end of the page. If defined, the **HTMLPrefix** string must include the <BODY> tag. If a *filename* is given instead of the *prefix* or *trailer*, the HTML code is taken from this file. Example:

```
HTMLPrefix       <BODY BGCOLOR="#FF0000">
HTMLTrailer      <A HREF="/intern/">Back</A> to the internal page.
```

HideAgent *agent string*

Hide certain browsers under an arbitrary *string* (item). Useful to map a browser's name to an arbitrary *string* (item). Since only the leading part of the browser type is compared against *agent*, there is no need to specify wildcards. In fact, a wildcard suffix is removed from the string, while a wildcard prefix is taken literal. Example:

```
HideAgent  Mozilla/4.0 (compatible; MSIE 4.       MSIE 4.*
```

HideRefer *referrer string*

Hide certain referrer URLs under an arbitrary *string* (item). Useful to map different referrer URLs for a given host to a common name. Since only the leading string of the referrer URL is compared against *referrer*, there is no need to specify wildcards. As in **HideAgent**, a wildcard suffix is removed from the string, while a wildcard prefix is taken literal. If the second argument contains a string in square brackets, this defines the CGI parameter which specifies the search key for search engines. In this case, the search key will be extracted from the argument list and prominently displayed after the name of the search engine/web server. See the file **sample.conf** for more examples on how to use **HideRefer**. Example:

```
HideRefer  http://search.yahoo.com        Yahoo [p=]
HideRefer  http://altavista.digital.com/  AltaVista [q=]
```

HideSys *hostname string*

Hide a *hostname* under an arbitrary *string* (item). The string may contain blanks. If the first character of *string* is a '[', this item is suppressed in the *Top N* lists. Hidden items are accounted for separately, but in the summary they are collected under the description defined with this directive. You may use the wildcard character '*' as either a prefix or as a suffix of the *hostname* (as in ***.host.com** and **192.168.12.***). Hostnames are case-insensitive. When building the list of countries, **http-analyze** determines the country from the top-level domain given in *hostname*. If *hostname* is an IP number, you can add the country's top-level domain in square brackets to the *string*. Example:

```
HideSys          *.mycompany.com MY COMPANY
HideSys          192.168.12.*    MY COMPANY [COM]
```

HideURL*url string*

Hide an *URL* under an arbitrary *string* (item). The string may contain blanks. If the first character of *string* is a '[', this item is suppressed in the *Top N* lists. Hidden items are accounted for separately, but in the summary they are collected under the description defined with this directive. You may use the wildcard character '*' as either a prefix or as a suffix of the *URL* (as in ***.map** and **/subdir/***). URLs are case-sensitive. Note, that images are hidden automatically under the term *All images* unless **−x** was specified. See the **sample.conf** file included in the distribution for more examples. Example:

```
HideURL          /newsletter/*   MyCompany's Monthly Newsletter
HideURL          /robots.txt     [Robot control file]
```

IgnURL *url* and **IgnSys** *hostname*

Ignore entries with a specific URL or accesses from a certain system. You may use the wildcard character '*' as either a prefix or as a suffix of the URL or the hostname (as in ***.gif** ,**/subdir/file*** and ***.host.com**). Note that all URLs/hostnames are compared against any entry in this list while **http-analyze** reads the logfile as opposed to **HideURL**/**HideSys**, which are later looked up for when all URLs/hostnames have been reduced to the set of unique URLs/hostnames. Therefore, using **IgnURL**/**IgnSys** will increase processing time of **http-analyze** by a significant amount. Example:

```
IgnURL           *.gif,*.jpg,*.jpeg
```

IndexFiles *idxfile [,idxfile ...]*

Defines additional directory index files (same as option **−H**). **http-analyze** truncates the URLs containing an index filename so that they merge with '/' or their "base URL", respectively. For example, the "base URL" of */dir/index.html* is */dir/*. You can define up to 9 more names in addition to the pre-defined name *index.html*; common names are *Welcome.html* and *home.html*, but any other name can be defined in the web server's configuration file. Note that each name requires a linear lookup in this table at a very early stage of processing, so many entries will increase the processing time. Example:

```
IndexFiles       Welcome.html,home.html,index.htm
```

LogFile *filename*

The name of the server's logfile. If you define a default name for the logfile, this file is processed if no other filenames are explicitely specified on the command line. Without such a definition, **http-analyze** always reads *stdin* if no other filename is given. Example:

```
LogFile          /usr/ns-home/www/logs/access
```

LogFormat *format*

use this logfile format. Valid values for *format* are **auto** for auto-sensing the logfile format, **clf** for the *Common Logfile Format*, or **dlf** and **elf** for the two supported forms of the *Combined/Extended Logfile Format*. See the section *Logfile Formats* above for a description of the formats supported by **http-analyze**. Example:

```
LogFormat        clf
```

NavWinSize *width*×*height*

Defines the size of the navigation window which pops up in the conventional interface if JavaScript is enabled. Useful if the browser displays scrollbars when the default size of 420×190 is used. Example:

```
NavWinSize       440x200
```

NavigFrame *size*

Defines the size of the navigation frame in pixels. Useful if the browser displays scrollbars when the default size of 120 pixels is used. Example:

```
NavigFrame       140
```

NoiseLevel *hits*

set the noise-level to *hits*. If a noise-level is defined, all URLs, sites, agents and referrer URLs with hits below this level are collected under the item *Noise* in the Top N lists and overviews to avoid cluttering up those reports. Example:

```
NoiseLevel       7
```

OutputDir *directory*

The name of the directory where the output files should be created (same as option **–o**). If left undefined, output files are created in the current directory. Example:

```
OutputDir        /usr/www/htdocs/stats
```

PageView *suffix [,suffix ...]*

define additional pageview suffixes (same as option **–G**). All files with a certain *suffix* are rated as pageviews (text files, documents). You can define up to 9 more pageview suffixes in addition to the pre-defined suffix *.html*. Common text file suffixes are **.shtml**, **.text** and **.htm**, but any other suffix can be defined in the web server's configuration file. Note that each suffix requires a linear lookup in this table at a very early stage of processing, so many entries will increase the processing time by a significant amount. Example:

```
PageView         .shtml,.text,.htm
```

PrivateDir *directory*

The name of a private directory where the overviews and detailed list of URLs, sites, browsers, and referrer URLs should be created (same as option **–p**). Access to this private directory may be granted to staff only by using server authentication. Pathnames not beginning with a '/' are relative to **OutputDir**. Note that you have to turn on authentication in the server's configuration file also in order to secure this subdirectory. Example:

```
PrivateDir       lists
```

RegInfo *customer_name registration_ID*

Defines the customer's name and the registration ID, which are both shown on the main page in the summary report. Example:

```
RegInfo          MyCompany      3745JMJZ00000311300000682344
```

ReportTitle *title*

The document title and header to use in the statistics report. **http-analyze** appends the server's name to this string. Example:

```
ReportTitle      WWW Access usage for
```

ServerName *srvname*

The official name of the server (same as option **–S**). Defaults to the current system name. Useful if the analyzer runs on another system the the web server is running on. Example:

```
ServerName       www.mycompany.com
```

ServerURL *srvurl*

The URL of the server to be used for hot links in URL lists (same as option **–U**). Useful if the reports for your web server are published on another server, for example on an internal developement machine. Also necessary for (software-) virtual servers to have **http-analyze** generate correct hypertext links in the reports. Example:

```
ServerURL        http://www.mycompany.com
```

Session *time*

The time-window for counting *sessions*. See *Sessions* in the section *Interpretation of results* for an explanation of this term. Example:

```
Session          4 hours
```

Suppress *option(s)*

Suppress certain lists in the reports (same as **–s**). *option* may be one of:

AVLoad	to suppress the average load report (top seconds/minutes/hours),
URLs	to suppress the overview of URLs/items and the detailed lists,
URLList	to suppress the detailed list of URLs,
Code404	to suppress the detailed list of Code 404 *Not Found* responses,
Sites	to suppress the overview of domains and the hostname lists,
RSites	to suppress the overview of reverse domains,
SiteList	to suppress the detailed list of hostnames,
Agents	to suppress the overview of browser types and the detailed lists,
Referrer	to suppress the overview of referrers and the detailed lists,
Country	to suppress the country list,
Pageviews	to suppress pageview rating (304's are shown instead),
Graphics	to suppress all graphs and pie charts,
Hotlinks	to suppress the hypertext links to the pages in the URL lists,
Interpol	to suppress interpolation of the graphs.

You may specify more than one argument to **Suppress** by separating them with a ','. Example:

```
Suppress         Country,Interpol
```

TLDFile *filename*

use *filename* for the list of top-level domains (same as option **–T**). This list includes all ISO two-letter country domains, the well-known domains **.net**, **.int**, **.org**, **.com**, **.edu**, **.gov**, **.mil**, **.arpa**, **.nato**, and the upcoming new *CORE* top-level domains **.firm**, **.info**, **.shop**, **.arts**, **.web**, **.rec**, and **.nom**. The length of a domain in the TLD file may not exceed 6 characters. **http-analyze** uses it's built-in defaults, if no TLD file is given. Example:

```
TLDFile          /usr/local/lib/http-analyze/TLD
```

Top{**Days,Hours,Minutes,Seconds,URLs,Sites,Agents,Refers**}, **LeastURLs**

Defines the size of certain Top N tables and lists. If set to zero, the corresponding list will be suppressed. Example:

```
TopURLs          20
LeastURLs        0
TopDays          7
TopHours         12
```

VirtualNames *name,...*

The list of additional virtual names of this server. **http-analyze** uses those names to construct a list of *self referrer URLs*, which will appear when a HTML page causes the browser to request inline images. Those self referrers are suppressed from the list of referrer URLs. The entries in the list of referrer URLs therefore gives a good impression about external pages referencing your site. The server's name is set as a self referrer by default (using the **ServerURL** or **ServerName** directive, whatever is specified. If you run virtual web servers, you would typically specify each virtual hostname here. Example:

```
VirtualNames     www.customer.com,customer.com
VirtualNames     www.othername.com,othername.com
```

VRMLProlog *file*

The name of a prolog file for a yearly VRML model (same as option **–P**). Pathnames not beginning with a '/' are relative to **OutputDir**. Example:

```
VRMLProlog       3Dprolog.wrl
```

EXAMPLES

After successful compilation of **http-analyze** you can create a statistics report before you choose to install the program permanently. To do so, create a subdirectory for the output files to avoid cluttering up the directory and install the required files using the **ha-setup** utility:

```
http-analyze setup
------------------
1) Set up an analyzer configuration for a virtual web server
2) Install the required files in a statistics output directory
3) Brand your copy of http-analyze with the registration ID
4) Exit
Please select a function (1-4) [1]: 2
Install required files for http-analyze
---------------------------------------
This script copies the required files (3D*, btn/*) into the statistics
...
Name of the HTML output directory: testd
Directory testd doesn't exist, create it (y/n) [y]: <RETURN>
Now enter the name of the directory containing the required files.
...
Directory containing required files (3D*, btn/*) [files]: <RETURN>
Required files have been copied into testd
```

Then, run the analyzer on your web server's logfile. For example, if the name of the logfile is */usr/ns-home/www/logs/access*, use the following command to create a full statistics including a frames-based interface and a 3D (VRML) model in the newly created directory **testd**:

```
$ http-analyze -vm3f -o testd /usr/ns-home/www/logs/access
http-analyze 2.01pl8 (IP22; IRIX 6.2), Copyright 1998 RENT-A-GURU(TM)
Generating statistics in directory 'testd'
Reading data from '/usr/ns-home/www/logs/access'
Hmm, looks like Extended Logfile Format (ELF)
Start new period at 01/Dec/1997
Creating VRML model for December 1997
Creating full statistics for December 1997
... processing URLs
... processing hostnames
... processing user agents
... processing referrer URLs
... updating 'www1997/index.html': last report is for December 1997
Statistics complete until 31/Dec/1997
$
```

After the analyzer terminates, start your browser and open the file **testd/index.html**.

To permanently install the program, issue a `make install` which copies the required files in the appropriate places. To set up an analyzer configuration for a web server, choose an output directory for the statistics report and use the **ha-setup** utility to install the required files there.

Following are some more examples, which assume that the analyzer has been installed permanently. The first command processes an archived logfile *logYYYY/access.MM* from the server's log directory to create a report for January 1998 in the directory **/usr/htdocs/stats**:

```
$ cd /usr/ns-home/www/logs
$ http-analyze -vm3f -o /usr/htdocs/stats log1998/access.01
```

The next command reads the logfile entries from a pipeline and creates the statistics report for a whole year using a customized configuration file:

```
$ gzcat log1997/access.[01]?.gz |
> http-analyze -c /usr/local/bin/sample.conf -
```

REGULAR INVOCATION VIA CRON

To have statistics generated on a regular base, use the following scheme:

1) Optionally install a cron job which calls **http-analyze –d** frequently to create a short statistics report. The execution interval may range from once per day up to twice per hour depending on the size of your logfile and the time needed to analyze it. On our server, we run the daily statistics once per hour.

2) Install a cron job which calls **http-analyze –m** to create a full statistics report once per week or once per day (again depending on the size of your logfile). Note that the full statistics reports are created for the first time at the second day of a new month. On our server, we create a monthly summary two times per day.

3) Create a script which rotates the server's logfile, restarts the http server, and then creates the final summary for this period. Have *cron* execute this script at 00:00 on the **first day** of a new month. See the script **rotate-httpd** for an example on how to do this for several virtual web servers running on the same machine.

4) Because of *cron*'s scheduling overhead and delays in execution of the script which rotates the logfile, heavy used servers sometimes writes a few entries for the new month in the old logfile. **http-analyze** usually detects and ignores such "white noise" at the end of a month. However, to get correct figures, in this last step you should run **http-analyze –m** on the logfile for the current month immediately after generating the statistics for the previous month.

Note that the cron jobs must run with the user ID of the owner of the directory where the HTML output files are to be created, except for **rotate-httpd**, which must run with the user ID of the server user. You should also take care to avoid running more than one **http-analyze** processes at the same time.

Here are some sample *crontab*(1) entries for the scheme described above:

```
# Generate a full report twice per day at 01:17 and 13:17
17  1,13 * * *  /usr/local/bin/http-analyze -m -c /usr/httpd/analyze.conf

# Generate a short summary each hour except at 01:17 or 13:17
17  2-12 * * *  /usr/local/bin/http-analyze -d -c /usr/httpd/analyze.conf
17 14-23 * * *  /usr/local/bin/http-analyze -d -c /usr/httpd/analyze.conf

# Rotate the HTTPD logfiles at the first day of a new month at 00:00
0 0 1 * *       /usr/local/bin/rotate-httpd
```

TROUBLESHOOTING

If you discover any problems using the analyzer you may find the verbose mode helpful. If the option **–v** is given at invocation, **http-analyze** comments ongoing processing. When the macro *NDEBUG* is undefined while **http-analyze** is being compiled, various assertion checks and debugging messages are included in the executable. If you then increase the verbosity level by specifying more **–v** options, all logfile entries being processed are printed:

```
$ http-analyze -vvvm3f -o testd log1998/access.01
http-analyze 2.01pl8 (IP22; IRIX 6.2), Copyright 1998 RENT-A-GURU(TM)
Generating statistics in directory 'testd'
Reading data from 'log1998/access.01'
Best blocksize for I/O is 64 KB
Hmm, looks like Extended Logfile Format (ELF)
  1 01/Dec/1997:00:02:14 [262929738], req=/stats/, sz=2656 <- Code 200 OK
Start new period at 01/Dec/1997
  2 01/Dec/1997:00:02:17 [262929741], req=/logo.gif, sz=5880 <- Code 200 OK
  3 01/Dec/1997:00:02:17 [262929741], req=/btns.gif, sz=4713 <- Code 200 OK
  ...
```

REGISTRATION

The distribution of **http-analyze** on our web site is made available to you for evaluation purposes only. In this version an "unregistered" button will show up in the statistics reports. To replace this button with the Netstore logo of the free version (for personal and educational use), just click on this "unregistered" button to follow the link to our registration form on our site and register for a free, non-commercial version.

NON-COMMERCIAL VERSION

After registration you will receive a registration ID to be put into the registration or the configuration file and two Netstore logos as replacements for the "unregistered" buttons. In the private version, the Netstore logo and a Copyright note will appear in the statistics reports after registration according to the *NON-COMMERCIAL LICENSE*, under which this personal copy is made available to you.

COMMERCIAL VERSION

If you choose to use **http-analyze** for commercial purposes such as statistics services, you must buy a *Commercial Service License* available from RENT-A-GURU® and their authorized resellers. You will receive a registration ID to be put into the registration or the configuration file and two Netstore logos as replacements for the "unregistered" buttons.

In the commercial version, the Netstore logo and the Copyright note will be supressed from the statistics report except for the main page and the navigation frame in the frames-based user interface. The customer also can define his own logos which will show up in the reports by using the **CustLogoW** and **CustLogoB** directives in the configuration file. Except for this feature, the non-commercial version and the comemrcial version are identical.

BRANDING THE SOFTWARE

For all license types, you can brand your copy of **http-analyze** with the registration ID, either in a system-wide file (usually */usr/local/lib/http-analyze/REGID*) or via the **RegInfo** directives in the configuration file. The former method requires root permissions, while the latter one requires specification of the configuration file each time **http-analyze** is invoked.

If you choose to create a system-wide registration file, you need to brand the software only once. On Unix systems, issue the following commands as root:

```
# mkdir /usr/local/lib/http-analyze
# http-analyze -r "Customer Name" 3745JMJZ00000311300000682344
Registration information saved in file ‘/usr/local/lib/http-analyze/REGID’
#
```

where *Customer Name* is the name of the customer which registered for this license and the following string is the registration ID. Next, install the two registration images we sent you by email into the appropriate buttons subdirectory:

- If you use **http-analyze** for one web server only, copy the registration images into the buttons subdirectory of the corresponding statistics directory defined by **OutputDir/btn**.
- If you analyze more than one server, install the registration images in the central buttons directory */usr/local/lib/http-analyze/btn*, which should have been created during the installation process. Then, you can easily install the required files and buttons using the **ha-setup** utility by copying or linking them into the several **OutputDir**-subdirectories.

After installing the buttons you have completed the registration. Now run the analyzer to create the statistics with the registered version.

COPYRIGHT

CREDITS

Thanks to the numeruous users of **http-analyze** for their valuable feedback. Special thanks to Lars-Owe Ivarsson forr his suggestions to optimize the parser algorithm and the code he provided as an example. Special thanks also to Thomas Boutell *(http://www.boutell.com/)* for his great GD library for fast GIF creation, without **http-analyze** couldn't produce such fancy graphics in the summary reports. *gd 1.2 is copyright 1994, 1995, Quest Protein Database Center, Cold Spring Harbor Labs.*

FILES

This section lists the pre-requisite files, which are needed by **http-analyze** in order to create the statistics reports. Those files are usually installed in the directory */usr/local/lib/http-analyze*, unless specified otherwise in the Makefile. See also the section *Summary Reports* above for the name of the HTML output files making up a statistics report.

btn/.gif*	buttons and icons used in HTML output files
TLD	list of all top-level-domains
ha2.0_*.gif	**http-analyze** logo for your web site (black/white bg)
logfmt.[cde]lf	sample logfiles in CLF, DLF and ELF format
3Dprolog.wrl	prolog file for the yearly VRML model on SGI workstations
3DshelfMotion.wav	sound file for the yearly model on SGI workstations
3Dlogo.wrl.gz	a stubs file for the yearly VRML model on PCs

SEE ALSO

rotate-httpd	script to rotate the web server's logfiles
ha-setup	script to set up the analyzer configuration for a web server
cvt_files	script to convert older files into new 2.0 directory structure
http://www.netstore.de/Supply/http-analyze/	The official homepage of **http-analyze**

NOTES

If you are going to analyze different logfiles in one invocation of **http-analyze**, you must sort them in ascending order of their date, otherwise the logfiles being processed after the first logfile will be silently ignored.

BUGS

You tell me.