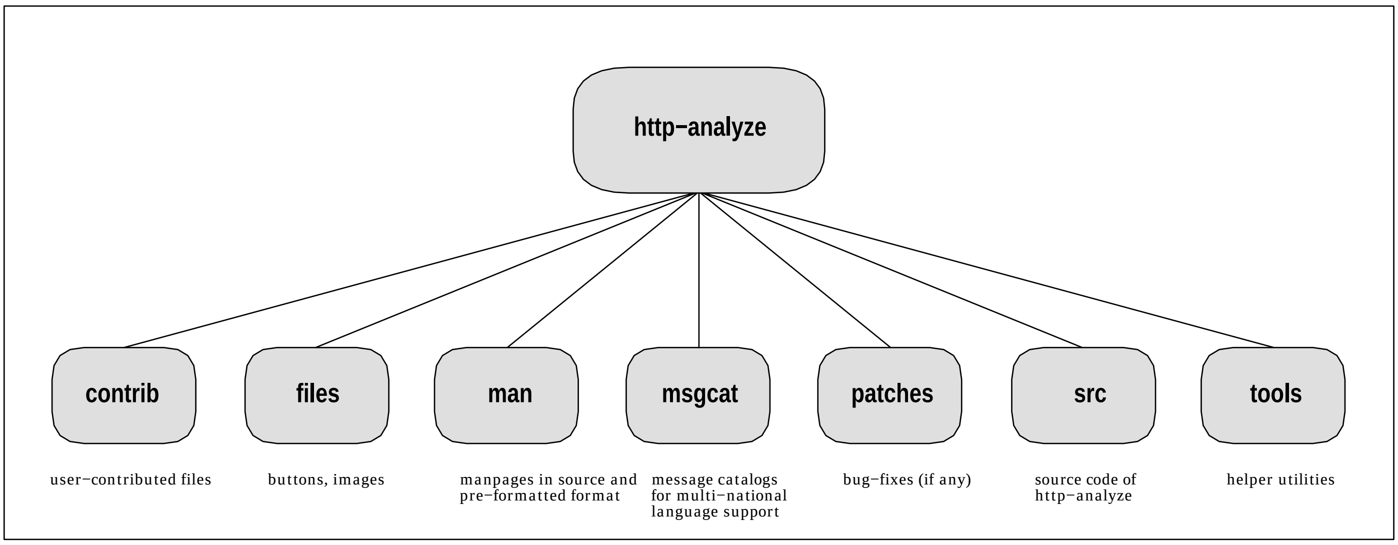
Technical Report 05−2003−09−09

Configuration and compilation of the http−analyze logfile analyzer

Although **http−analyze** comes in pre−compiled, installable format, sometimes people want to compile their own customized version of the executable. This report describes the compilation parameters and settings needed on certain platforms to gain maximum performance from the **http−analyze** software. The second part of the report deals with compilation of **ipresolve** and **ha−sort**, two helper utilities.

After unpacking the **http−analyze** source distribution, the following directories have been created:


http-analyze
contrib
files
man
msgcat
patches
src
tools
user−contributed files
buttons, images
manpages in source and pre−formatted format
message catalogs for multi−national language support
bug−fixes (if any)
source code of http−analyze
helper utilities

The directory **contrib** contains contributed software from users of **http−analyze**. Files required by the analyzer, such as buttons and logos, are in the subdirectory **files**. The subdirectory **man** contains the manpages in source and pre−compiled format. **msgcat** contains message catalogs for X/Open and SVR4 multi−national language support. **patches** contains important bug−fixes if any (there are none since version 2.4 patchlevel 3), **src** contains the source code of **http−analyze** and **tools** contains helper applications such as **rotate−httpd**.

To start with configuration, see the **Makefile** in the subdirectory **src**. There are also some pre−defined macro settings for several platforms:

macro	*default values*	*meaning*
HA_BINDIR	**/usr/local/bin**	is the directory where the executable is installed
HA_LIBDIR	**/usr/local/lib/http-analyze**	is the directory where the required files are installed
REQINCS	**-I/usr/local/include**	location of the include files for library functions
REQLIBS	**-L/usr/local/lib -lgd -lpng -lz**	ld options for required libraries
OPTIM	**-O3**	optimization level
DEFINES	**-DIRIX**	platform−specific options also passed to *lint(1)*
PLATFORM	**-n32 -mips3 -xansi**	platform−specific options also passed to *ld(1)*
COMDEFS	**-DTIME_STATS -DNDEBUG -DHA_LIBDIR=\"$(HA_LIBDIR)\"**	are definitions common to all platforms such as **HA_LIBDIR**
EXTRA_LIBS	*(undefined)*	extra libraries (beside *gd*, *png* and *libz*)

Further customization takes place in file **config.h**, which selects appropriate macro definitions depending on the platform selected via the **DEFINES** macro in the **Makefile**. Currently the following platforms are supported:

IRIX	SGI IRIX 6.x	**BSD**	BSD/OS, FreeBSD
SUN	SUN SunOS 4.x/Solaris 2.x	**SVR4**	AT&T (vanilla) Unix System V.4
DEC	DIGITAL UNIX V4.0	**SVR3**	AT&T (vanilla) Unix System V.3
AIX	IBM AIX 4.2	**BEOS**	BeOS
HPUX	HP/UX 10.20	**WIN32**	Windows NT/Netware
linux	GNU/Linux systems	**OS2**	OS/2

Continued on page 2

The header file **config.h** uses C preprocessor definitions to enable or disable certain functionality or to include missing functions. All directives beginning with »**NO_**« mark a missing function such as *gethostname(2)* or *uname(2)*. Directives beginning with »**USE_**« are most often optional functions, which could improve overall performance of the analyzer. Directives beginning with »**NEED_**« are most often macros to include missing functions, which are de−facto standard elsewhere, but have not yet made it into the C standard library, such as *snprintf(3)* on some systems. A short overview follows. More information is available in the file **config.h**.

NO_GETHOSTNAME	Do not use *gethostname(2)* to obtain the system name.
NO_UNAME	Do not use *uname(2)* to obtain system name and platform/OS.
NO_REGEX	Has no POSIX 1003.2 compliant regular expressions.
NO_NO_SIGSET	Has no SVR4 reliable signals.
NO_FLOCK	Has no *flock(2)* function to lock the database during updates (**ipresolve** only).
NO_SYMLINK	Has no symlink capabilities.
NEED_GETOPT	include own *getopt(3)*
NEED_STRCASECMP	include own *strcasecmp(3)*
NEED_STRERROR	include own *strerror(3)*
NEED_SNPRINTF	include own *snprintf(3)*
NEED_SNPRINTF	include own *snprintf(3)*
NEED_VSNPRINTF	include own *vsnprintf(3)*
FAST_CTYPE	If defined, use own character classification functions.
FAST_MALLOC	If defined, use fast memory allocator available on SGI IRIX.
VRML	If defined, include code for generation of VRML model.
LOGF_DEFAULT	Defines the default format of the logfile (let undefined for auto−detection).
TIME_STATS	If defined, includes code for performance analysis.
USE_STDIO	If defined, use *fgets(3)* (or *gzgets(3L)* for *gzip*'ed files) to read input data.
USE_STBLKSIZE	If defined, use *stat(2)* to determine the best size of an I/O buffer.
USE_MMAP	If defined, use *mmap(2)* to map certain files into the address space.
USE_XPGCAT	If defined, use X/Open multi−national language support.
USE_SVR4CAT	If defined, use SVR4 native multi−national language support.
USE_MYCAT	If defined, useown (built−in) multi−national language support.
USE_NDBM	If defined, use the Berkeley DB or NDBM functions (**ipresolve** only).

The meaning of the first twelve directives should be clear.

FAST−CTYPE includes faster character classification functions, which depend on use of the ASCII character set in URLs, hostnames and other logfile entries.

FAST_MALLOC uses fast memory allocation functions from *libmalloc.so* on SGI IRIX systems. It requires inclusion of this library in the **EXTRA_LIBS** macro in the **Makefile**.

The macro **VRML** defines whether to include code for the − still optional − generation of a VRML (3D) model.. For historical reasons, you can switch between the old and the new (default) layout of the VRML world. The analyzer creates this model if command line option **−3** is given.

The macro **LOG_DEFAULT** may be used to define the default logfile format. If left undefined, **http−analyze** tries to automatically detect the logfile format. Note that only the three most common logfile formats are supported for the sake of a high−speed parser, which isolates fields in logfile entries at maximum speed. The default setting for this macro is undefined.

The macro **TIME_STATS** is defined by default to include code for performance measurement of **http−analyze**.

The macro **USE_STDIO** can be defined to use *fgets(3)* (or *gzgets(3)* for *gzip*'ed logfiles) rather then thebuilt−in, high−speed readline function of **http−analyze**. While good implementations of *fgets(3)* may yield similar results as the built−in function, *gzgets(3)* has shown a dramatical increase in processing time required for I/O compared to *gzread(3)* used in the built−in function of **http−analyze**. Therefore, you probably should left this macro undefined and use the built−in function.

The macro **USE_STBLKSIZE** should be defined. It enables **http−analyze** to use a dynamically allocated I/O buffer depending on the best I/O size for an input file as reported by the *stat(2)* system−call. For example, on SGI IRIX systems, the best I/O size for pipes is 16 KB, while for hard−disk files the best size ranges from 64 KB to 256 KB depending on the underlying disk hardware.

The macro **USE_MMAP** scan be defined to use memory−mapped files when compressing the VRML model. It is currently not used otherwise.

The macros **USE_XPGCAT, USE_SVR4CAT** and **USE_MYCAT** define the type of multi−national language support − if any at all. The default setting is X/Open MNLS (**USE_XPGCAT**). If you encounter any problems, use the native (built−in) support of **http−analyze**.

The macro **USE_NDBM** includes database support in **ipresolve**. IP number/hostname pairs can be saved in a database over the invocations of **ipresolve**.

Continued on page 3

To customize the compilation environment, define the appropriate macros in **config.h** and in the **Makefile**. The following are definitions for a system running SGI IRIX 6.5:.

Makefile:

```
REQINCS  = −I/usr/local/include
REQLIBD  = −L/usr/local/lib32
COMDEFS  = −DTIME_STATS −DVRML −DNDEBUG −DHA_LIBDIR=\"$(HA_LIBDIR)\"
DEFINES  = −DIRIX
OPTIM    = −O2 −OPT:Olimit_opt=on
PLATFORM = −n32 −mips3 −xansi
EXTRA_LIBS = −lmalloc
```

config.h:

```
#if defined(IRIX)
# define USE_STBLKSIZE  /* Best block size for I/O depends on file type */
# define USE_XPGCAT     /* Use X/Open message catalogs */
# define USE_MMAP       /* Use mmap(2) for file I/O if possible */
# define USE_NDBM       /* Use NDBM database function (ipresolve only) */
# define FAST_CTYPE     /* Use fast character classification: for ASCII only! */
# define FAST_MALLOC    /* Use fast memory allocation functions */
```

The following definitions are for FreeBSD systems:

Makefile:

```
REQINCS  = −I/usr/local/include
REQLIBD  = −L/usr/local/lib
COMDEFS  = −DTIME_STATS −DVRML −DNDEBUG −DHA_LIBDIR=\"$(HA_LIBDIR)\"
DEFINES  = −DBSD −Dunix
OPTIM    = −O2
CC       = gcc
```

config.h:

```
#elif defined(BSD)
# undef  USE_STBLKSIZE  /* Sun reports 8KB for block size, this sucks */
# define USE_MYCAT      /* Use built−in message catalogs */
# define USE_MMAP       /* Use mmap(2) for file I/O if possible */
# define USE_NDBM       /* Use NDBM database function (ipresolve only) */
# define FAST_CTYPE     /* Use fast character classification: for ASCII only! */
```

The following definitions are for Sun Solaris systems:

Makefile:

```
REQINCS  = −I/usr/local/include
REQLIBD  = −L/usr/local/lib
COMDEFS  = −DTIME_STATS −DVRML −DNDEBUG −DHA_LIBDIR=\"$(HA_LIBDIR)\"
DEFINES  = −DSUN −Dunix
OPTIM    = −O2
PLATFORM = −D_LARGEFILE_SOURCE −D_FILE_OFFSET_BITS=64
```

config.h:

```
#elif defined(SUN)
# undef  USE_STBLKSIZE  /* Sun reports 8KB for block size, this sucks */
# define USE_XPGCAT     /* Use X/Open message catalogs */
# define USE_MMAP       /* Use mmap(2) for file I/O if possible */
# define USE_NDBM       /* Use NDBM database function (ipresolve only) */
# define FAST_CTYPE     /* Use fast character classification: for ASCII only! */
# define NEED_GETOPT    /* use own getopt since stdlib don’t know ‘optopt’ */
```

Then compile the software and test it. You can first let **http−analyze** display the version info and the help list:

```
$ http−analyze −V
Copyright 2003 by RENT−A−GURU(TM)
$Id: http−analyze.c,v 2.4.1.3 1999/11/04 12:15:19 stefan Stab $
Personal edition − see http://www.netstore.de/Supply/http−analyze/register.html

$ http−analyze −hh
Usage:
    http−analyze [−{hdmBVX}] [−3aefgnqvxyM] [−b bufsize] [−c cfgfile]
        [−i newcfg] [−l libdir] [−o outdir] [−p prvdir] [−s subopt,...]
        [−t num,...]  [−u time] [−w hits] [−F logfmt] [−L lang] [−C chrset]
        [−I date] [−E date] [−G suffix,...]  [−H idxfile,...]  [−O vname,...]
        [−P prolog] [−R docroot] [−S srvname] [−T TLDfile] [−U srvurl]
        [−W 3Dwin] [−Z showdom] [logfile ...]
...
```

Continued on page 4

Next, test the statistics generation with the included logfile(s). Change into subdirectory **files** to avoid **http−analyze** from complaining about missing buttons:

```
$ cd ../files
$ mkdir testd
$ ../src/http−analyze −3fmv −o testd logfmt.elf
http−analyze 2.4pl3 (IP32; IRIX 6.5; XPG4 MNLS; PNG)
Copyright 2003 by RENT−A−GURU(TM)
Generating full statistics in output directory ‘testd’
Reading data from ‘logfmt.elf’
Best blocksize for I/O is set to 64 KB
Hmm, looks like Extended Logfile Format (ELF)
Start new period at 01/Jan/1999
NOTE: output files will be created in subdirectory ‘www1999’
Creating VRML model for January 1999
Creating full statistics for January 1999
... processing URLs
... processing hostnames
... processing user agents
... processing referrer URLs
Total entries read: 8, processed: 8
Clear almost all counters at 03/Jan/1999
Start new period at 01/Feb/1999
No more hits since 02/Feb/1999
Creating VRML model for February 1999
Creating full statistics for February 1999
... processing URLs
... processing hostnames
... processing user agents
... processing referrer URLs
... updating ‘www1999/index.html’: last report is for February 1999
Total entries read: 3, processed: 3
Statistics complete until 28/Feb/1999
$
```

If your distribution contains the helper utilities **ipresolve** and **ha−sort**, compile them as follows:

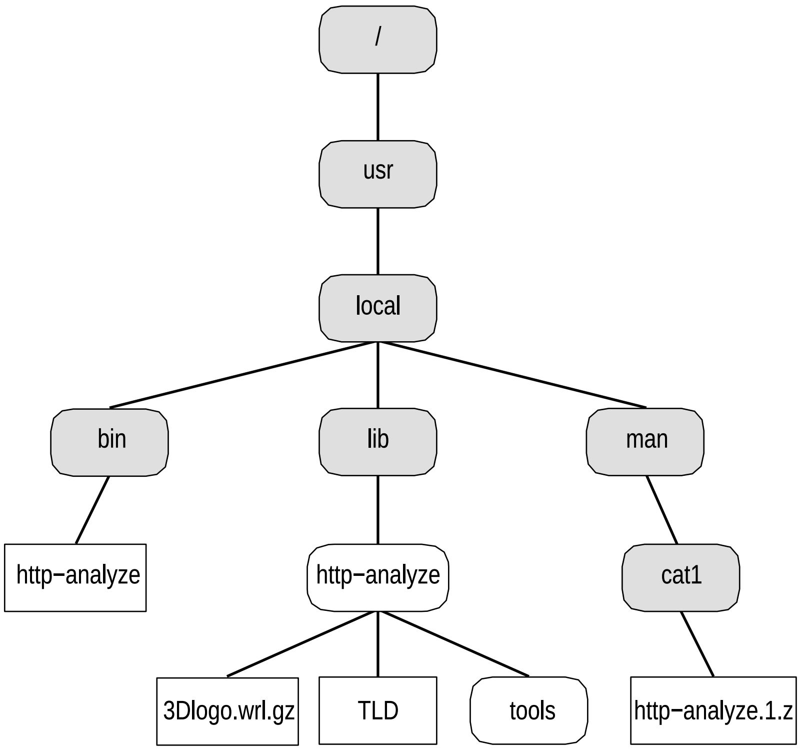
```
cc −n32 −mips3 −O3 −fullwarn −I/usr/local/include −DIRIX −c ipresolve.c
cc −n32 −mips3 −O3 −fullwarn −I/usr/local/include −DIRIX −c readline.c
cc −n32 −mips3 −O3 −fullwarn −o ipresolve ipresolve.o readline.o −L/usr/local/lib32 −lz
cc −n32 −mips3 −O3 −fullwarn −o ha−sort ha−sort.c
```

Then test the programs:

```
$ ipresolve −h
Usage:
  ipresolve {−hcV} [−izv] [−b blksize] [−d database] [−m maxage] [−o outfile] [logfile]

Options:
    −h          print this help list and exit
    −c          clean the database (see −d and −m)
    −V          print version information and exit
    −i          suppress unresolveable IP numbers
    −z          force compression of output file (not applicable for stdout)
    −v          increase verbosity level (may be specified more than once)
    −b blksize  set best block size for I/O (determined automatically)
    −d database name of database to save IP/hostname pairs
    −m maxage   set maximum age for DB entries to ‘maxage’ days (default: 14 days)
    −o outfile  name of the output file (use suffix ‘.gz’ for a compressed file)
    logfile     name(s) of the logfile(s) to process (default: stdin)

$ ipresolve
212.86.202.89 − test
www.netstore.de − test

$ ha−sort −h
Usage:
  ha−sort [−hv] [−I date] −E [date] logfile [logfile ...]

    −h          print this help list
    −E date     skip all logfile entries from this date on
    −I date     ignore all logfile entries up to this date
    logfile     name of the logfile(s)

$ ha−sort −v logfmt.clf logfmt.dlf logfmt.elf | wc −l
Phase 1: Gathering logfile data
Phase 2: Sorting logfile data by timestamp
Phase 3: Creating sorted data stream
            33
$
```

Continued on page 5

If everything is working, install the executables and support files with the command »**`make install`**« in the top−level directory. To use multi−national language support, select a default target in **`msgcat/Makefile`**:. By default, installation procedures use pre−formatted manpages from the **`man`** subdirectory. After the executables and required support files have been installed, you can use http−analyze to create and update statistics reports. The figure below shows the files and directories installed by default (white background).



To have **http−analyze** executed automatically, add an entry like the following in the crontab of the server user (do **NOT** use **`root`**'s crontab!):

```
# crontab file for http−analyze statistics
#
# Format of lines:
# min hour daymo month daywk cmd
#
17 1,13 * * * /usr/local/bin/http−analyze −3fm −o /www/stats −c /www/analyze.conf
```

This causes **http−analyze** to be called at 01:17 and 13:17 each day. The name of the logfile to process is passed through the configuration file **analyze.conf**. On the 1st of a month the **rotate−httpd** script, which rotates (saves) the logfile, can also call **http−analyze** explicitely, in this case you should avoid running the analyzer at the same time. See the *crontab(1)* documentation of your system for more information about *cron(8)* jobs.

Some things to keep in mind are:

❏ Always have **http−analyze** process all logfiles for the current month in full statistics mode, otherwise the statistics can get zeroed.

❏ If you analyze older periods than the current month, anything between this period and the current month will be lost except when you specify the **−n** option to not update the history!

❏ If you use **http−analyze** 2.5 or **ipresolve** 2.0, they can both read *gzip*'ed logfiles directly.

☞ See *TR−01−2003−09−08* for information how to run a shell script to update statistics reports for many servers and *TR−02−2003−09−08* for instructions how to rotate (save) the web server's logfile once per month using shell−scripts.

Please send comments, enhancements, tips and tricks to: **`office@rent−a−guru.de`**.