Compiling and Installing

This document covers compilation and installation of Apache on Unix and Unix-like systems only. For compiling and installation on Windows, see Using Apache with Microsoft Windows[1]. For other platforms, see the platform[2] documentation.

Apache 2.0's configuration and installation environment has changed completely from Apache 1.3. Apache 1.3 used a custom set of scripts to achieve easy installation. Apache 2.0 now uses `libtool` and `autoconf` to create an environment that looks like many other Open Source projects.

Topics

See also

- Starting Apache[3]
- Stopping and Restarting[4]

Overview for the impatient

Download	`$ lynx http://www.apache.org/dist/httpd/httpd-2_1_`*`NN`*`.tar.gz`
Extract	`$ gzip -d httpd-2_1_`*`NN`*`.tar.gz` `$ tar xvf httpd-2_1_`*`NN`*`.tar`
Configure	`$ ./configure --prefix=`*`PREFIX`*
Compile	`$ make`
Install	`$ make install`
Customize	`$ vi `*`PREFIX`*`/conf/httpd.conf`
Test	`$ `*`PREFIX`*`/bin/apachectl start`

NN must be replaced with the current minor version number, and *PREFIX* must be replaced with the filesystem path under which the server should be installed. If *PREFIX* is not specified, it defaults to `/usr/local/apache2`.

Each section of the compilation and installation process is described in more detail below, beginning with the requirements for compiling and installing Apache HTTPD.

Requirements

The following requirements exist for building Apache:

Disk Space
Make sure you have at least 50 MB of temporary free disk space available. After installation Apache occupies approximately 10 MB of disk space. The actual disk space requirements will vary considerably based on your chosen configuration options and any third-party modules.

ANSI-C Compiler and Build System
Make sure you have an ANSI-C compiler installed. The GNU C compiler (GCC)[5] from the Free Software Foundation (FSF)[6] is recommended (version 2.7.2 is fine). If you don't have GCC then at least make sure your vendor's compiler is ANSI compliant. In addition, your `PATH` must contain basic build tools such as `make`.

Accurate time keeping
Elements of the HTTP protocol are expressed as the time of day. So, it's time to investigate setting some time synchronization facility on your system. Usually the `ntpdate` or `xntpd` programs are used for this purpose which are based on the Network Time Protocol (NTP). See the Usenet newsgroup comp.protocols.time.ntp[7] and the NTP homepage[8] for more details about NTP software and public time servers.

Perl 5[9] [OPTIONAL]
For some of the support scripts like apxs[10] or dbmmanage[11] (which are written in Perl) the Perl 5 interpreter is required (versions 5.003 or newer are sufficient). If no such interpreter is found by the `configure' script there is no harm. Of course, you still can build and install Apache 2.0. Only those support scripts cannot be used. If you have multiple Perl interpreters installed (perhaps a Perl 4 from the vendor and a Perl 5 from your own), then it is recommended to use the `--with-perl` option (see below) to make sure the correct one is selected by `./configure`.

Download

Apache can be downloaded from the Apache HTTP Server download site[12] which lists several mirrors. You'll find here the latest stable release.

After downloading, especially if a mirror site is used, it is important to verify that you have a complete and unmodified version of the Apache HTTP Server. This can be accomplished by testing the downloaded tarball against the PGP signature. This, in turn, is a two step procedure. First, you must obtain the `KEYS`[13] file from the Apache distribution site, too. (To assure that the `KEYS` file itself has not been modified, it may be a good idea to use a file from a previous distribution of Apache or import the keys from a public key server.) The keys are imported into your personal key ring using one of the following commands (depending on your pgp version):

```
$ pgp < KEYS
```

or

```
$ gpg --import KEYS
```

The next step is to test the tarball against the PGP signature, which should always be obtained from the main Apache website[12]. A link to the signature file is placed behind the corresponding download link or may be found in the particular directory at the Apache distribution site[14]. Its filename is identical to the source tarball with the addition of `.asc`. Then you can check the distribution with one of the following commands (again, depending on your pgp version):

```
$ pgp httpd-2_1_NN.tar.gz.asc
```

or

```
$ gpg --verify httpd-2_1_NN.tar.gz.asc
```

You should receive a message like

```
Good signature from user "Martin Kraemer <martin@apache.org>".
```

Depending on the trust relationships contained in your key ring, you may also receive a message saying that the relationship between the key and the signer of the key cannot be verified. This is not a problem if you trust the authenticity of the `KEYS` file.

Extract

Extracting the source from the Apache HTTPD tarball is a simple matter of uncompressing, and then untarring:

```
$ gzip -d httpd-2_1_NN.tar.gz
$ tar xvf httpd-2_1_NN.tar
```

This will create a new directory under the current directory containing the source code for the distribution. You should `cd` into that directory before proceeding with compiling the server.

Configuring the source tree

The next step is to configure the Apache source tree for your particular platform and personal requirements. This is done using the script `configure` included in the root directory of the distribution. (Developers downloading the CVS version of the Apache source tree will need to have `autoconf` and `libtool` installed and will need to run `buildconf` before proceeding with the next steps. This is not necessary for official releases.)

To configure the source tree using all the default options, simply type `./configure`. To change the default options, `configure` accepts a variety of variables and command line options. Environment variables are generally placed before the `./configure` command, while other options are placed after. The most important option here is the location prefix where Apache is to be installed later, because Apache has to be configured for this location to work correctly. But there are a lot of other options available for your pleasure.

For a short impression of what possibilities you have, here is a typical example which compiles Apache for the installation tree `/sw/pkg/apache` with a particular compiler and flags plus the two additional modules `mod_rewrite` and `mod_speling` for later loading through the DSO mechanism:

```
$ CC="pgcc" CFLAGS="-O2" \
./configure --prefix=/sw/pkg/apache \
--enable-rewrite=shared \
--enable-speling=shared
```

When `configure` is run it will take several minutes to test for the availability of features on your system and build Makefiles which will later be used to compile the server.

The easiest way to find all of the configuration flags for Apache is to run `./configure --help`. What follows is a brief description of most of the arguments and environment variables.

Environment Variables

The `autoconf` build process uses several environment variables to configure the build environment.

In general, these variables change the method used to build Apache, but not the eventual features of the server. These variables can be placed in the environment before invoking `configure`, but it is usually easier to specify them on the `configure` command line as demonstrated in the example above.

`CC=...`
The name of the C compiler command.

`CPPFLAGS=...`
Miscellaneous C preprocessor and compiler options.

`CFLAGS=...`
Debugging and optimization options for the C compiler.

`LDFLAGS=...`
Miscellaneous options to be passed to the linker.

`LIBS=...`
Library location information ("`-L`" and "`-l`" options) to pass to the linker.

`INCLUDES=...`
Header file search directories ("`-I`*`dir`*").

`TARGET=...` [Default: apache]
Name of the executable which will be built.

`NOTEST_CPPFLAGS=...`

`NOTEST_CFLAGS=...`

`NOTEST_LDFLAGS=...`

`NOTEST_LIBS=...`
These variables share the same function as their non-`NOTEST` namesakes. However, the variables are applied to the build process only after autoconf has performed its feature testing. This allows the inclusion of flags which will cause problems during feature testing, but must be used for the final compilation.

`SHLIB_PATH=...`
Options which specify shared library paths for the compiler and linker.

autoconf Output Options

`--help`
Prints the usage message including all available options, but does not actually configure anything.

`--quiet`
Prevents the printing of the usual "`checking...`" messages.

`--verbose`
Prints much more information during the configuration process, including the names of all the files examined.

Pathnames

There are currently two ways to configure the pathnames under which Apache will install its files. First, you can specify a directory and have Apache install itself under that directory in its default locations.

`--prefix=`*`PREFIX`* [Default: `/usr/local/apache2`]
Specifies the directory under which the Apache files will be installed.

It is possible to specify that architecture-dependent files should be placed under a different directory.

`--exec-prefix=`*`EPREFIX`* [Default: *`PREFIX`*]
Specifies the directory under which architecture-dependent files will be placed.

The second, and more flexible way to configure the install path locations for Apache is using the `config.layout` file. Using this method, it is possible to separately specify the location for each type of file within the Apache installation. The `config.layout` file contains several example configurations, and you can also create your own custom configuration following the examples. The different layouts in this file are grouped into `<Layout FOO>...</Layout>` sections and referred to by name as in `FOO`.

`--enable-layout=`*`LAYOUT`*
Use the named layout in the `config.layout` file to specify the installation paths.

Modules

Apache is a modular server. Only the most basic functionality is included in the core server. Extended features are available in various modules. During the configuration process, you must select which modules to compile for use with your server. You can view a list of modules [15] included in the documentation. Those modules with a status [16] of "Base" are included by default and must be specifically disabled if you do not want them (e.g. `mod_userdir`). Modules with any other status must be specifically enabled if you wish to use them (e.g. `mod_expires`).

There are two ways for a module to be compiled and used with Apache. Modules may be *statically compiled*, which means that they are permanently included in the Apache binary. Alternatively, if your operating system supports Dynamic Shared Objects (DSOs) and `autoconf` can detect that support, then modules may be *dynamically compiled*. DSO modules are stored separately from the Apache binary, and may be included or excluded from the server using the run-time configuration directives provided by `mod_so`. The mod_so is automatically included in the server if any dynamic modules are included in the compilation. If you would like to make your server capable of loading DSOs without actually compiling any dynamic modules, you can explicitly `--enable-so`.

`--enable-`*`MODULE`*`[=shared]`
Compile and include the module *MODULE*. The identifier *MODULE* is the Module Identifier [17] from the module documentation without the "_module" string. To compile the module as a DSO, add the option `=shared`.

`--disable-`*`MODULE`*
Remove the module *MODULE* which would otherwise be compiled and included.

`--enable-modules=`*`MODULE-LIST`*
Compile and include the modules listed in the space-separated *MODULE-LIST*.

`--enable-mods-shared=`*`MODULE-LIST`*
Compile and include the modules in the space-separated *MODULE-LIST* as dynamically loadable (DSO) modules.

The *MODULE-LIST* in the `--enable-modules` and `--enable-mods-shared` options is usually a space-separated list of module identifiers. For example, to enable `mod_dav` and `mod_info`, you can either use

```
./configure --enable-dav --enable-info
```

or, equivalently,

```
```

```
./configure --enable-modules="dav info"
```

In addition, the special keywords `all` or `most` can be used to add all or most of the modules in one step. You can then remove any modules that you do not want with the `--disable-`*`MODULE`* option. For example, to include all modules as DSOs with the exception of `mod_info`, you can use

```
./configure --enable-mods-shared=all --disable-info
```

In addition to the standard set of modules, Apache 2.0 also includes a choice of Multi-Processing Modules[18] (MPMs). One, and only one MPM must be included in the compilation process. The default MPMs for each platform are listed on the MPM documentation page[18], but can be overridden on the `configure` command line.

`--with-mpm=`*`NAME`*
: Choose the mpm *NAME*.

DBM

Several Apache features, including `mod_authn_dbm` and `mod_rewrite`'s DBM `RewriteMap` use simple key/value databases for quick lookups of information. Apache includes SDBM with its source-code, so this database is always available. If you would like to use other database types, the following `configure` options are available:

`--with-gdbm[=`*`path`*`]`

`--with-ndbm[=`*`path`*`]`

`--with-berkeley-db[=`*`path`*`]`
: If no *path* is specified, Apache will search for the include files and libraries in the usual search paths. An explict *path* will cause Apache to look in *path*`/lib` and *path*`/include` for the relevant files. Finally, the *path* may specify specific include and library paths seperated by a colon.

Suexec

Apache includes a support program called suexec[19] which can be used to isolate user CGI programs. However, if suexec is improperly configured, it can cause serious security problems. Therefore, you should carefully read and consider the suexec documentation[19] before implementing this feature.

Build

Now you can build the various parts which form the Apache package by simply running the command:

```
$ make
```

Please be patient here, since a base configuration takes approximately 3 minutes to compile under a Pentium III/Linux 2.2 system, but this will vary widely depending on your hardware and the number of modules which you have enabled.

Install

Now its time to install the package under the configured installation *PREFIX* (see `--prefix` option above) by running:

```
$ make install
```

If you are upgrading, the installation will not overwrite your configuration files or documents.

Customize

Next, you can customize your Apache HTTP server by editing the configuration files [20] under *PREFIX*/conf/.

```
$ vi PREFIX/conf/httpd.conf
```

Have a look at the Apache manual under docs/manual/ [21] or consult http://httpd.apache.org/docs-2.1/ for the most recent version of this manual and a complete reference of available configuration directives[22].

Test

Now you can start[3] your Apache HTTP server by immediately running:

```
$ PREFIX/bin/apachectl start
```

and then you should be able to request your first document via URL `http://localhost/`. The web page you see is located under the `DocumentRoot` which will usually be *PREFIX*/htdocs/. Then stop [4] the server again by running:

```
$ PREFIX/bin/apachectl stop
```

URI References

[1] http://httpd.apache.org/docs-2.1/platform/windows.html
[2] http://httpd.apache.org/docs-2.1/platform/
[3] http://httpd.apache.org/docs-2.1/invoking.html
[4] http://httpd.apache.org/docs-2.1/stopping.html
[5] http://www.gnu.org/software/gcc/gcc.html
[6] http://www.gnu.org/
[7] news:comp.protocols.time.ntp
[8] http://www.eecis.udel.edu/~ntp/
[9] http://www.perl.org/
[10] http://httpd.apache.org/docs-2.1/programs/apxs.html
[11] http://httpd.apache.org/docs-2.1/programs/dbmmanage.html
[12] http://httpd.apache.org/download.cgi
[13] http://www.apache.org/dist/httpd/KEYS
[14] http://www.apache.org/dist/httpd/
[15] http://httpd.apache.org/docs-2.1/mod/
[16] http://httpd.apache.org/docs-2.1/mod/module-dict.html#Status
[17] http://httpd.apache.org/docs-2.1/mod/module-dict.html#ModuleIdentifier

[18] http://httpd.apache.org/docs-2.1/mpm.html
[19] http://httpd.apache.org/docs-2.1/suexec.html
[20] http://httpd.apache.org/docs-2.1/configuring.html
[21] http://httpd.apache.org/docs-2.1/
[22] http://httpd.apache.org/docs-2.1/mod/directives.html