Inside the Apache directory structure

Presented by developerWorks, your source for great tutorials

`ibm.com/developerWorks`

Table of Contents

If you're viewing this document online, you can click any of the topics below to link directly to that section.

Section 1. Tutorial tips

Should I take this tutorial?

This tutorial introduces the Apache administrator to the directory layouts used for a given installation. With this knowledge, administrators can then easy locate Apache's executable and utility files, and determine what's necessary for custom configuration.

Conventions

A few conventions are used in this tutorial for clarity:

* Text to be typed in is displayed in a **`bold monospace`** font.
* *Italic font* is used to draw attention to window, dialog box, and feature names.
* A `monospace` font is used for file and path names.

Navigation

Navigating through the tutorial is easy:

* Use the Next and Previous buttons to move forward and backward through the tutorial.
* Use the Main menu button to return to the tutorial menu.
* If you'd like to tell us what you think, use the Feedback button.

Getting help

For technical questions about the content of this tutorial, contact the author, Tom Syroid, at *tom@syroidmanor.com* .

Tom Syroid is currently working as a staff writer for a literary agency. In past lives he has worked as a system consultant, system administrator, truck driver, and heavy duty mechanic. Over the years, he has gained extensive experience in deploying UNIX-based operating systems, and specializes in configuring Apache and Samba. In those rare moments when Tom's not writing something or tweaking a server, you'll find him chasing his 20-month old son around the house, doing crafts with his daughter, or enjoying a quiet summer evening on the back porch with his wife. You can contact Tom by e-mail at *tom@syroidmanor.com* or by visiting his Web site at *syroidmanor.com* .

Section 2. Understanding Apache's directory layouts

Why use directories?

Gone are the days when a software program consisted of a single executable and perhaps one or two data files. Today's typical UNIX program is a mix of one or more executables, several "helper" or ancillary utilities, a configuration file, a log file (or files), a PID (Process Identification) file, and depending on the program's purpose, some form of data files to manipulate or act on.

The most common method to organize these various components is grouping elements in directories based on their role and who will be accessing them. For example, executables that only the root user or system administrator should have access to are usually placed in a directory named sbin; program files that the average user can access are placed in a directory named bin. It is also common practice to segregate log files, user data, and files used by the system to determine run-time status (usually called control files) into separate directories.

Apache directory layouts

When Apache is compiled, a directory structure is either specified or implied, and the program's various components are installed based on this structure or layout. Currently, Apache supports sixteen distinct directory layout options. They include:

* A prefix or ServerRoot directory
* An executable prefix directory (which is often the same as the ServerRoot)
* A bin directory
* An sbin directory
* A library directory
* A man page directory
* An icons directory
* A cgi directory
* A system configuration file directory
* A shared data directory
* A document root or htdocs directory
* An include directory
* Several control or system state directories

Note that not all the above location options need be specified or even used. Likewise, groups of common files can share a common directory. For example, many webmasters prefer all Apache's system control files to be written to a single directory under `/var/apache`.

Determining how Apache is installed

So how do you know what directory layout a given distribution uses? There's an easy way to find out, but it introduces a chicken and egg problem -- first you have to know where Apache's server daemon is. The easiest way to determine this is to type locate httpd in an open terminal window or at the command line. There should be one entry on the list displayed that lives in either a bin or sbin directory. Common locations are `/usr/local/apache/bin`, `/usr/local/bin or ../sbin`, `/var/lib/apache/bin or .../sbin`, etc.

Now either change to this directory, or type the full path to the file and append a '-V' after the httpd (that's capital V). The output of this command displays some of the compile-time settings used to build Apache on the machine in question.

```
[tom]/usr/local/bin> httpd -V
Server version: Apache/1.3.14 (Unix)
Server built:   Oct 13 2000 18:35:32
Server's Module Magic Number: 19990320:10
Server compiled with....
-D EAPI
-D HAVE_MMAP
-D USE_MMAP_SCOREBOARD
-D USE_MMAP_FILES
-D USE_PTHREAD_SERIALIZED_ACCEPT
-D HTTPD_ROOT="/usr/local"
-D SUEXEC_BIN="/usr/local/bin/suexec"
-D DEFAULT_PIDLOG="/var/apache/run/httpd.pid"
-D DEFAULT_SCOREBOARD="/var/apache/run/httpd.scoreboard"
-D DEFAULT_LOCKFILE="/var/apache/run/httpd.lock"
-D DEFAULT_XFERLOG="/var/apache/log/access_log"
-D DEFAULT_ERRORLOG="/var/apache/log/error_log"
-D TYPES_CONFIG_FILE="/etc/apache/mime.types"
-D SERVER_CONFIG_FILE="/etc/apache/httpd.conf"
-D ACCESS_CONFIG_FILE="/etc/apache/access.conf"
-D RESOURCE_CONFIG_FILE="/etc/apache/srm.conf"
```

Apache's layout.config file

While it's certainly possible to individually configure all sixteen directory options manually, there's an easier approach. Apache provides a very slick mechanism that preconfigures directory layouts based on popular Linux distributions. The file is called `layout.config` and can be found in the root directory where Apache's source code was unpacked. If you did not install Apache's source code with your distribution, it's a good idea to do so -- having access to this one file is worth the effort. As an alternative, go to *www.apache.org* and download the latest source code.

`layout.config` contains thirteen pre-defined installation path layouts including Classical Apache, GNU FSF, RedHat 5.x, SuSE 6.x, Slackware 7.x, several Mac layouts, two BSD layouts, and a Solaris layout. Print this file out and keep it handy for reference to Apache's various installation layouts.

If you ever need to compile Apache under one of the listed layouts, simply change to the root directory containing the source code and `layout.config` file, and type **`./configure --with-layout=`*`layoutname`*** . Each installation layout listing in `layout.config` is named uniquely.

Using show-layout to view an installation

There is another way to display an installation layout under Apache. This approach uses the `--show-layout` option in conjunction with the named section of `layout.config`. While using this method presupposes knowledge of the available layouts, it has an extremely useful role in ensuring all the right rocks are going in the right jars before you start the compile process. The listing below shows the RedHat layout option.

```
[tom@velocity apache_1.3.14]$ ./configure --with-layout=RedHat --show-layout
Configuring for Apache, Version 1.3.14
 + using installation path layout: RedHat (config.layout)

Installation paths:
prefix: /usr
exec_prefix: /usr
bindir: /usr/bin
sbindir: /usr/sbin
libexecdir: /usr/lib/apache
mandir: /usr/man
sysconfdir: /etc/httpd/conf
datadir: /home/httpd
iconsdir: /home/httpd/icons
htdocsdir: /home/httpd/html
cgidir: /home/httpd/cgi-bin
includedir: /usr/include/apache
localstatedir: /var
runtimedir: /var/run
logfiledir: /var/log/httpd
proxycachedir: /var/cache/httpd

Compilation paths:
HTTPD_ROOT: /usr
SHARED_CORE_DIR: /usr/lib/apache
DEFAULT_PIDLOG: /var/run/httpd.pid
DEFAULT_SCOREBOARD: /var/run/httpd.scoreboard
DEFAULT_LOCKFILE: /var/run/httpd.lock
DEFAULT_XFERLOG: /var/log/httpd/access_log
DEFAULT_ERRORLOG: /var/log/httpd/error_log
TYPES_CONFIG_FILE: /etc/httpd/conf/mime.types
```

Secret: Creating a custom layout.config file

Can't find a directory layout in layout.config that matches your needs or current installation? Create your own. Copy the section closest to your installation, and paste this new section anywhere in the layout file. Edit the title (the text between the angle brackets), and adjust the paths as desired. Now when you compile Apache using the `--with-layout` *`option`* , simply specify the name you used for the new layout title. For example, to use a newly added section titled "Layout Unique", the compile command would be `./configure --with-layout=Unique`. Note that section titles are case sensitive.

Section 3. Modifying Apache's default directory locations

The one constant is change...

OK, your Apache installation is configured and serving Web pages flawlessly. But over time you find yourself running out of space on the filesystem where your pages or logs are stored, or you want to add a development branch to your document tree and test out a new Web design. Can you tell Apache to look somewhere other than the compiled default location for a particular file or files?

Generally speaking, the answer is yes. It doesn't make sense to change the location of some files (for example, Apache's executables), but the file locations that do commonly change in an installation can be modified without recompiling Apache.

There are two approaches depending on what option you want to change:

* Invoke Apache's httpd daemon with a command-line option
* Modify the appropriate directive in Apache's configuration file (`httpd.conf`)

Command line directory options

Locate the Apache daemon for your installation and run it with the help option. For example, if httpd is located in the directory /usr/local/apache/sbin, type:
`/usr/local/apache/sbin/httpd -h`

In the output generated, there are two options specific to directory locations:

* -d directory - specifies an alternate ServerRoot
* -f file - specifies an alternate ServerConfigFile

The -d directory option

The -d option allows you to specify an alternate ServerRoot directory, which points Apache to the top of the directory tree under which the server's conf and log subdirectories are located. This option is typically used for testing and/or debugging an alternate Apache configuration.

For example, based on the following hypothetical directory structure, issue the command **`/usr/local/apache/sbin/httpd -d /home/website1`** to instruct Apache to start using the configuration file found in the directory `/home/website1/conf.`

```
/public/website1
/public/website1/html
/public/website1/logs
/public/website1/conf
```

Tip: Understanding the ServerRoot value

As noted in the previous panel, the ServerRoot tells Apache where the "root" directory for its configuration, error, and log files are located. The key to understanding how Apache utilizes the ServerRoot value lies in understanding the difference between explicit and relative paths.

If a control or log file is referenced from within httpd.conf that begins with "/" then Apache will use that explicit path. If, however, a control or log file does not include an explicit or fully qualified path, then Apache prepends the value of ServerRoot to the filename. So if the ServerRoot for an installation was set to `/usr/local/apache`, the directive `logs/test.log` would be interpreted as `/usr/local/apache/logs/test.log`.

Apache's -f file option

The second httpd command line option available is to use is the `-f` *`file`* approach. This is different from the previously discussed -d option in that with the -f option you specify a filename, not a directory structure, and you must also include the -d option as well. For example:

```
/usr/local/apache/sbin/httpd -d /home/website1 -f
httpd-test.conf
```

The advantage of the -f option is that you could have more than one configuration file in the *conf* subdirectory and chose an alternate directly from the command line. Using just the -d option necessitates a unique ServerRoot directory structure for each configuration used.

Modifying directory defaults from httpd.conf

The second way to modify Apache's default directory locations is from the configuration file, *httpd.conf*. The advantage of modifying a directory location from within httpd.conf is that there are far more options available than the two command line options previously discussed. The disadvantage is the inconvenience of physically editing the configuration file, and the potential for introducing typographical errors.

Some of the more common directory locations that can be modified from httpd.conf are:

* ServerRoot
* DocumentRoot
* The location of the cgi-bin directory
* The proxy cache directory

The number and range of available directory locations will depend on the needs of your particular installation, and the Apache layout currently in use. See the httpd.conf file itself for more details; all available options are extremely well documented.

Global and virtual host options

The configuration directives in httpd.conf are grouped into three broad sections: the "global" environment, directives that apply to the "main" server, and directives that apply to any configured virtual hosts. In Apache parlance, virtual hosting is a way to configure the server to respond to page requests for more than one IP or domain.

Note that any directives modified in the "main" configuration section of httpd.conf become the default values used by the virtual hosts section. In the context of directory structures, it's very unlikely that you'll want to use the main server's DocumentRoot or log directories for a virtual host entry, so for each virtual host a unique DocumentRoot and log location is usually specified.

Again, refer to the httpd.conf file and the examples it contains regarding configuring virtual hosts.

Section 4. Wrapup

Resources

As you can see, the topic of Apache directory structures is not complex, but it's often not well understood. You should now know the essentials of the Apache directory structure and where to find key executables.

A number of exellent resources include lots of additional information about Apache. A few of these are:

Your feedback

Please let us know whether this tutorial was helpful to you and how we could make it better. We'd also like to hear about other tutorial topics you'd like to see covered. Thanks!

For technical questions about the content of this tutorial, contact the author, Tom Syroid, at *tom@syroidmanor.com* .

Colophon

This tutorial was written entirely in XML, using the developerWorks Toot-O-Matic tutorial generator. The Toot-O-Matic tool is a short Java program that uses XSLT stylesheets to convert the XML source into a number of HTML pages, a zip file, JPEG heading graphics, and two PDF files. Our ability to generate multiple text and binary formats from a single source file illustrates the power and flexibility of XML.