Introduction to the Apache Web Server

Rich Bowen, Cooper McGregor, Inc

May 2, 2005

Contents

1 History **1**

1.1 Pre-history . . . 1

1.1.1 As We May Think . . . 1

1.1.2 WWW . . . 1

1.1.3 NCSA . . . 2

1.1.4 Netscape . . . 2

1.2 Apache . . . 2

1.3 The Apache Software Foundation . . . 3

2 Installing **5**

2.1 Section objectives . . . 5

2.2 Building from source . . . 5

2.3 Contents of distribution file, 1.3 . . . 7

2.4 Contents of distribution file, 2.0 . . . 8

2.5 Running configure . . . 9

2.5.1 Building with mod_perl . . . 10

2.5.2 Other, more complex installs . . . 10

2.5.3 Apache ToolBox . . . 11

2.5.4 2.0 . . . 11

2.6 apxs . . . 12

3 Starting and Stopping **13**

3.1 Apache process architecture . . . 13
3.1.1 Apache 1.3 . . . 13
3.1.2 Apache 2.0 . . . 14
3.2 apachectl . . . 17
3.3 httpd . . . 18
3.3.1 start . . . 18
3.3.2 stop . . . 18
3.3.3 Other options . . . 19
3.4 Starting at boot . . . 19

4 Configuration files **21**
4.1 Introduction . . . 21
4.2 The files . . . 21
4.3 Config file syntax . . . 22
4.3.1 Directives . . . 22
4.3.2 Anatomy of directive docs . . . 22
4.3.3 Comments . . . 23
4.4 Sections . . . 24
4.5 Options . . . 26
4.6 Different config file . . . 26
4.6.1 Running multiple apache daemons . . . 26
4.7 GUI Configuration Tools . . . 26

5 .htaccess files **27**
5.1 Configuration . . . 27
5.1.1 AccessFileName . . . 27
5.1.2 AllowOverride . . . 27
5.2 Performance . . . 27
5.3 Exercise . . . 29

6 Virtual Hosts **31**

6.1 IP-based virtual hosts . . . 31

6.2 Name-based virtual hosts . . . 32

6.3 General caveats, comments . . . 33

6.4 Exercise . . . 33

6.5 Additional notes, examples, etc . . . 34

6.5.1 /etc/hosts . . . 34

6.5.2 Your example virtual host sections . . . 34

6.5.3 #apache . . . 34

6.5.4 mod_vhost_alias . . . 35

7 MIME **37**

7.1 HTTP headers . . . 37

7.2 MIME configuration . . . 38

7.2.1 AddType . . . 38

7.2.2 RemoveType . . . 39

7.2.3 DefaultType . . . 39

7.2.4 ForceType . . . 39

7.3 mod_mime_magic . . . 40

7.4 Encoding . . . 40

7.4.1 AddEncoding . . . 40

7.4.2 RemoveEncoding . . . 40

7.4.3 mod_gzip . . . 41

7.5 Language . . . 41

7.6 Multiple file extensions . . . 41

7.7 Experiment . . . 41

8 URL Mapping **43**

8.1 URL Mapping procedure . . . 43

8.2 Location . . . 43

8.3 Alias . . . 44
8.4 ScriptAlias . . . 44
8.5 AliasMatch and ScriptAliasMatch . . . 45
8.6 Regular Expressions . . . 45
8.7 Redirect . . . 45
8.8 RedirectMatch . . . 46
8.9 RedirectTemp and RedirectPermanent . . . 46
8.10 DocumentRoot . . . 46
8.11 Error documents . . . 46
8.12 Error documents in Apache 2.0 . . . 47
8.13 Other modules that handler URL mapping . . . 49
8.13.1 mod_speling . . . 49
8.13.2 mod_rewrite . . . 50
8.13.3 mod_userdir and public_html . . . 50

9 Content Negotiation **53**
9.1 Client configuration . . . 53
9.1.1 Accept* headers . . . 53
9.1.2 Quality factors . . . 53
9.2 Negotiation Methods . . . 54
9.2.1 MultiViews . . . 54
9.2.2 Type map files . . . 54
9.3 Caching . . . 55

10 Indexing with mod_autoindex **57**
10.1 Options . . . 57
10.2 DirectoryIndex . . . 57
10.3 IndexOptions . . . 58
10.4 Additional directives . . . 60
10.4.1 HeaderName . . . 60

10.4.2 ReadmeName . . . 60
10.4.3 IndexIgnore . . . 60
10.5 Searching and sorting . . . 60
10.5.1 Apache 1.3 . . . 60
10.5.2 Apache 2.0 . . . 60
10.6 Security Concerns . . . 61

11 Performance Tuning **63**

11.1 Optimization, benchmarking and profiling . . . 63
11.2 ab . . . 63
11.3 Perl . . . 64
11.4 Optimizing hardware . . . 64
11.5 Tuning configuration settings . . . 65
11.5.1 HostnameLookups . . . 65
11.5.2 Symbolic links . . . 65
11.5.3 .htaccess files . . . 65
11.5.4 Negotiation . . . 65
11.5.5 Caching and proxying . . . 66
11.5.6 mod_mmap_static . . . 67
11.6 Process Creation . . . 67
11.6.1 MaxRequestsPerChild . . . 67
11.7 KeepAlive . . . 67
11.8 CGI/Other dynamic content . . . 67

12 CGI programming **69**

12.1 Introduction - The CGI . . . 69
12.2 Apache configuration . . . 70
12.3 How a CGI program works . . . 70
12.4 Common problems . . . 73

13 SSI **75**

13.1 Configuration for SSI . . . 75

13.2 XBitHack . . . 76

13.3 mod_include configuration directives . . . 76

13.4 SSI directives . . . 77

13.4.1 config . . . 77

13.4.2 timefmt . . . 77

13.4.3 echo . . . 78

13.4.4 exec . . . 78

13.4.5 fsize . . . 78

13.4.6 flastmod . . . 78

13.4.7 include . . . 78

13.4.8 printenv . . . 79

13.5 Variables and flow control . . . 79

13.6 Security . . . 79

14 Handlers and Filters **81**

14.1 Handlers . . . 81

14.1.1 Configuration directives . . . 81

14.1.2 Standard handlers . . . 83

14.1.3 Custom handlers . . . 86

14.2 Filters . . . 86

14.2.1 Chaining filters - CGI + SSI . . . 87

14.2.2 mod_deflate . . . 87

15 mod_perl **89**

15.1 Overview - What is mod_perl? . . . 89

15.2 Installation . . . 89

15.3 mod_perl installation caveats . . . 89

15.4 Configuration . . . 90

15.4.1 PerlRequire . . . 90
15.5 Connecting to your database . . . 90
15.6 CGI under mod_perl . . . 90
15.6.1 Apache::PerlRun . . . 90
15.6.2 Apache::Registry . . . 92
15.7 Apache handlers with mod_perl . . . 92
15.7.1 Installing a mod_perl handler from CPAN . . . 92
15.8 Writing a mod_perl handler . . . 93
15.8.1 Example mod_perl handlers . . . 93
15.8.2 Installing the example mod_perl handler . . . 94
15.8.3 Configuring the mod_perl handler . . . 94
15.9 Common problems . . . 94
15.9.1 Don't exit . . . 94
15.9.2 Restart the server . . . 94
15.9.3 Global values . . . 95
15.10 Other phases . . . 95
15.10.1 PerlAccessHandler . . . 95
15.10.2 PerlLogHandler . . . 96
15.10.3 Perl configuration sections . . . 96
15.11 More information . . . 96

16 Logging **97**

16.1 Standard log files . . . 97
16.1.1 access_log . . . 97
16.2 Location and format of the log file . . . 99
16.3 mod_log_io . . . 99
16.4 Exercises . . . 100
16.4.1 Error logs . . . 100
16.4.2 LogLevel . . . 100

16.5 Typical errors . . . 100
16.5.1 Things to remember! . . . 101
16.6 Logfile reporting . . . 101
16.6.1 What your log file tells you . . . 101
16.6.2 What your log file does not tell you . . . 101
16.6.3 Log file parsing . . . 102
16.7 Logging to a process . . . 102
16.8 Logging to syslog . . . 102
16.9 Rotating log files . . . 103
16.9.1 Logfile::Rotate . . . 103
16.9.2 rotatelogs . . . 104
16.9.3 logresolve . . . 104
16.10Logging for multiple virtual hosts . . . 104

17 Authentication, Authorization, Access Control 105

17.1 Definitions . . . 105
17.2 Basic Authentication . . . 105
17.3 Configuration . . . 105
17.4 FAQ . . . 106
17.5 Basic Auth Caveats . . . 107
17.6 Digest Auth . . . 107
17.7 Configuration for Digest auth . . . 107
17.8 Authentication against other things . . . 107
17.8.1 mod_auth_db . . . 108
17.8.2 mod_auth_mysql . . . 108
17.9 Access Control . . . 108
17.9.1 Satisfy . . . 109

18 Spiders 111

18.1 Introduction . . . 111

18.2 Potential problems . . . 111
18.3 Spiders in the logs . . . 111
18.4 Excluding spiders from your site . . . 112
18.4.1 robots.txt . . . 112
18.4.2 ROBOTS metatag . . . 112
18.4.3 Yell at the operator . . . 112
18.4.4 Block by address . . . 113
18.4.5 Blocking with Deny from Env . . . 113
18.5 Writing your own spider . . . 113

19 Security **115**

19.1 Overview . . . 115
19.2 File permissions . . . 116
19.2.1 Content directories . . . 116
19.2.2 Library . . . 116
19.2.3 bin . . . 116
19.2.4 logs . . . 117
19.2.5 proxy . . . 117
19.2.6 public_html . . . 118
19.3 Configuration . . . 118
19.3.1 ServerTokens . . . 118
19.3.2 ServerTokens - hacking the source . . . 119
19.3.3 ServerSignature . . . 120
19.4 SSI . . . 120
19.5 CGI . . . 120
19.5.1 CGI exploit example - trusting form input . . . 121
19.5.2 CGI exploit example - hidden form fields . . . 121
19.6 Default file system settings . . . 122
19.7 UserDir . . . 122

19.8 Modules . . . 123
19.9 suexec . . . 123
19.10mod_security . . . 124
19.11mod_dosevasive . . . 124

20 SSL **127**

20.1 Intro . . . 127
20.2 Installing SSL . . . 127
20.3 Certificates . . . 128
20.4 Configuration . . . 128

21 modules **131**

21.1 Module list . . . 131
21.1.1 Apache 1.3 modules: . . . 131
21.1.2 Apache 2.0 modules: . . . 132
21.1.3 What's new, and what's missing . . . 132
21.2 mod_access . . . 132
21.3 mod_actions . . . 133
21.4 mod_alias . . . 133
21.5 mod_asis . . . 133
21.6 mod_auth . . . 134
21.7 mod_auth_anon . . . 134
21.8 mod_auth_db . . . 134
21.9 mod_auth_dbm . . . 134
21.10mod_auth_digest . . . 135
21.11mod_autoindex . . . 135
21.12mod_cern_meta . . . 135
21.13mod_cgi . . . 136
21.14mod_digest . . . 136
21.15mod_dir . . . 136

21.16mod_env . . . 136
21.17mod_example . . . 137
21.18mod_expires . . . 137
21.19mod_headers . . . 137
21.20mod_imap . . . 137
21.21mod_include . . . 138
21.22mod_info . . . 138
21.23mod_log_agent . . . 138
21.24mod_log_config . . . 138
21.25mod_log_referer . . . 139
21.26mod_mime . . . 139
21.27mod_mime_magic . . . 139
21.28mod_mmap_static . . . 139
21.29mod_negotiation . . . 140
21.30mod_proxy . . . 140
21.31mod_rewrite . . . 140
21.32mod_setenvif . . . 141
21.33mod_so . . . 141
21.34mod_speling . . . 141
21.35mod_status . . . 142
21.36mod_unique_id . . . 142
21.37mod_usertrack . . . 142
21.38mod_vhost_alias . . . 142

22 WebDAV **143**

22.1 mod_dav on 1.3 . . . 143
22.2 mod_dav on 2.0 . . . 143
22.3 DAV clients . . . 144

23 mod_proxy **145**

23.1 Caching 145
23.2 Proxying 146
23.3 Rewrite and proxying 146
23.4 mod_proxy_html 146
23.5 General comments 146

Section 1

History

This section is an attempt to acquaint the student with the history, as well as the historical roots, of the Apache project - how it came to be, why it came to be, and how it has progressed. Additionally, we attempt to give a little bit of context of the surrounding people and projects which shaped the web, and the Apache project.

Finally, we try to give some understanding of the Apache Software Foundation, and what its goals are.

1.1 Pre-history

1.1.1 As We May Think

Apache came into existence 4 years after the creation of the World Wide Web. The Internet had been around for a while by then, and frameworks such as Gopher were already in place and in widespread use. But the ideas that formed the Web had been around for at least 45 years.

As We May Think, in In 1945, **Vannevar Bush** wrote a paper called which he discusses the way that we think, the way that our minds move from one topic to another, and the ways that technology needed to evolve to service the way that we think. Reading his thoughts on this matter, couched in terms of the technology that was then available, is provides interesting insights into our own time. You can obtain this entire document at
`http://www.theatlantic.com/unbound/flashbks/computer/bushf.htm`

1.1.2 WWW

The initial uses of the Internet were primarily communication driven. Email has always been the "killer app" of the Internet.

In 1990, Tim Berners Lee (TBL) was working at the CERN (European Center for Nuclear Research) and needed a way to make large amounts of information available to the researchers in a quick, efficient manner. Building on ideas already in widespread use, like HyperCard and its relatives, he invented the World Wide Web, and coined the terms hyperlink and hypertext.

He wrote the CERN web server, and the `www` client software, terming it a "browser".

1.1.3 NCSA

In 1992 or thereabouts, a project started at NCSA (The National Center for Supercomputing Activities) at UIUC (University of Illinois at Urbana-Champaign to develop a graphical web browser, which was code-named "Mosaic", although the name "Mozilla" was also used.

As part of that project, the NCSA HTTPd was also written.

Mosaic was not the first graphical web browser, as has sometimes been claimed since then, but it quickly became the most popular.

You can read more about this at `http://www.webhistory.org/www.lists/www-talk.1993q1/0262.html`

Rob McCool was the primary author of the NCSA NTTPd code, and his name still appears in the default `httpd.conf` file.

1.1.4 Netscape

In 1995, Mosaic Communications, quickly renamed to Netscape Communications, was founded taking most of the programming talent from the NCSA HTTPd project

This left a void for those web sites which were running the NCSA software, particularly in a growing economy where people had come to rely on their web sites.

1.2 Apache

The Apache project was started simply as a place to collect patches for the NCSA HTTPd. The original "Apache Group" consisted of 8 guys who wanted to add functionality to, and fix problems with, the existing HTTPd code.

- Brian Behlendorf
- Roy T. Fielding
- Rob Hartill
- David Robinson
- Cliff Skolnick
- Randy Terbush
- Robert S. Thau
- Andrew Wilson

The name “Apache” was, apparently, picked by Ben Laurie out of respect for the Apache people.[1]

The other form of this story is that Apache is simply another way of saying that it was “A Patchy” server.

See also `http://httpd.apache.org/ABOUT\_APACHE.html` and
`http://www.geocrawler.com/mail/thread.php3?subject=name&list=417`

Randy Terbush wrote the Apache Software License, based on the BSD Software License, ensuring that the software will be free and open.

- C2.net donates server space for Apache.org
- April 1995 - Apache 0.6.2 released
- December 1995 - Apache 1.0, a complete rewrite, released.
  - Main advance here is modularization of code.
  - Project codenamed ”Shambala”
  - Robert Thau main developer of this code
  - Same basic code base in use today in 1.3.x
- 1997 - Open/Free software goes mainstream
  - 1997 - The Cathedral and The Bazaar
  - Apache deal with IBM - Apache forms codebase for WebSphere
  - Apache Software Foundation formed
  - Revision of License to be more palatable to IBM - Advertising clause removed
  - 1997 - Apache 1.3 released, with Windows support
- May 2000 - Apachecon Orlando - Apache 2.0 alpha released
- October 2000 - ApacheCon Europe - Douglas Adams speaks, one of his final speaking ops before he suddenly died.
- April 2001 - ApacheCon Santa Clara - Apache 2.0 initial beta release
- April 2002 - Apache 2.0.35 releases as GA (General Availability). 2.0.36 follows shortly after with some important fixes.
- May 2002 - Apache 1.3 enters maintenance mode (No new features, just bug fixes and documentation updates).
- `http://uptime.netcraft.com/up/today/top.avg.html`

1.3 The Apache Software Foundation

The ASF was formed for a number of reasons. The catalyst was the IBM deal, and their desire to deal with an actual legal entity. However, the impact of creating the ASF was rather larger than that.

The goals of the ASF are:

[1] I recommend reading Chasing Shadows by Shelley Ann Bowen Hatfieid (no relation) about the Apache people. http://www.amazon.com/exec/obidos/asin/0826318533

- provide a foundation for open, collaborative software development projects by supplying hardware, communication, and business infrastructure;
- create an independent legal entity to which companies and individuals can donate resources and be assured that those resources will be used for the public benefit;
- provide a means for individual volunteers to be sheltered from legal suits directed at the Foundation's projects; and,
- protect the 'Apache' brand, as applied to its software products, from being abused by other organizations.

Section 2

Installing

http://httpd.apache.org/docs/install.html

2.1 Section objectives

In this section, the students will each have to install Apache themselves. They should each have a server system which does not have Apache installed. They should download the Apache source, verify the pgp signature and MD5 sum, unpack and install from source. They should experiment with the arguments to ./configure and should install Apache with and without DSO support. If the machines are reasonably fast, they should try multiple installations, and see what happens.

Students should install Apache 1.3 and Apache 2.0, and should verify that both are functioning correctly, at the same time, running on different ports.

Students should then reinstall Apache 1.3 using Apache ToolBox, enabling any modules which you wish to use in the remainder of the course. In particular, you will probably want to enable mod_ssl, mod_perl, and mod_php.

Make sure that Apache 2 is installed with DAV and dav-fs enabled.

2.2 Building from source

Although many Unixes come with some version of Apache preinstalled, there are many arguments for installing the server yourself.

- Exactly the way you need it
- Optimized for your hardware
- Ensure nothing strange added
- Directory structure that makes sense to you

This is a good point to encourage discussion of package management systems, particularly if your students are already fond of a particular unix distribution that is tied to a package management system. The arguments for installing from source seem rather weak when weighed against the convenience of a package management system. However, there are times when it is necessary to install from source, and so it is useful to know how to do it.

Get source from http://httpd.apache.org

Latest releases are 1.3.33 and 2.0.52. 2.1 is in development, and there is not yet a downloadable packaged version of it.

The site is mirrored around the world, and downloading Apache from a mirror site is of great benefit to the ASF, financially. Our monthly bandwidth bill is astronomical.

Next, you really need to verify the distribution using the MD5 sum, and, if possible, the PGP signature that are available from the site. However, you should get these from the official Apache site, rather than the mirror site. The rationale here is that if the distro was compromised, the signature files probably were also.

Verify the distribution

There are two ways to verify the distribution. The MD5 sum can be used as follows:

```
md5sum httpd_2.0.52.tar.gz
```

Verify that the output of that command matches the contents of the .md5 file. Note that the `md5sum` utility is just called `md5` on some Unixes.

To verify the pgp signature, you will need to first import the keyring from the Apache site:

Download `http://www.apache.org/dist/httpd/KEYS`:

```
wget http://www.apache.org/dist/httpd/KEYS
gpg --import KEYS
```

You can use `curl` or `GET`, if you have one of those installed. The `GET` utility gets installed when you install the `LWP` suite of modules for Perl.

Finally, verify the signature with:

```
gpg --verify httpd_2.0.52.tar.gz.asc
```

Unpack the distribution

```
tar -vzxf httpd-2.0.52.tar.gz
```

Note that the -z flag is a gnu-tar thing, and may not be available in all versions of tar, although it is more common now than it used to be.

Change into the directory

Build it ...

```
./configure --prefix=/usr/local/apache
make
make install
```

Note that if you run `./configure` as a non-root user, Apache will be configured to run on port 8080. This is very annoying, but being aware of it mostly solves the problem. Tell students that they really only need to be root in order to `make install`, and then see what happens. This is mostly for your own information, since at least one student will do this, and then you'll be left wondering why it's not working.

2.3 Contents of distribution file, 1.3

We now take a brief step back to look at the contents of the distribution file, so that the student knows what we're working with. You'll need to familiarize yourself with the contents of each of the directories in the distribution, so that you can explain any files that they may ask about. Note that the layout for 2.0 is different from that for 1.3, having been reorganized for a variety of reasons.

- cgi-bin
  - Sample CGI programs
  - Usually not executable, for security reasons
    The rationale here is that, if every web server on the planet has a particular CGI program installed and enabled, and some day, for some reason, someone finds a security exploit in it, it will be a simple matter to crack any server on the planet. Thus, we ship sample cgi programs, but don't enable them by default, and recommend that they be used for testing only.
  - `--cgidir=DIR`
- conf
  - Starter configuration files
  - Will not overwrite your existing configs
    That is, if you modify your configuration file, and then reinstall Apache, your changes will be preserved. This is particularly important for this exercise, as, after installing Apache the first time, we want to install it again, perhaps several times, with a very different configuration. Thus, each time, we want to remove our configuration file before installing Apache again. Make sure that you explicitly talk about this at this point, as at least one student will forget to do this, and you need to have mentioned it so that you don't look like this problem caught you off-guard.
  - Variables filled in based on your build arguments

    ```
    ServerRoot @@ServerRoot@@
    LockFile @rel_logfiledir@/accept.lock
    ```

Show the students these variables in the pre-install configuration file, and demonstrate to them how these variables are filled in after the fact. Demonstrate this with several different arguments for the various directories, or with a –with-layout= flag.

  - `--sysconfdir=DIR`

- htdocs
  - "It's working" page

    This page is available in 20 or 30 languages. You should get the correct one, based on your browser preferences. You might want to demonstrate this as a preview of the content negotiation chapter.
  - Manual
  - `--manualdir=DIR` Specifying the –manualdir flag will cause the entire user manual to be placed elsewhere, and an Alias to be inserted in the configuration file pointing to that location

- icons
  - Icons used in auto directory listings
  - `--iconsdir=DIR`

  The manual and the icons are the only directories of content that are generally installed by default. icons is always installed outside of the document directory, but the manual is usually installed inside the document directory. There was an attempt to change this, several versions ago, and move the manual outside of the document directory by default. For some reason, this confused people, and it was made an option, with the default being inside the document directory. Thus, the `--manualdir` flag is fairly new.

- logs
  - Initially empty

- src
  - Source code for Apache server
  - Subdirs for a variety of things - main, modules, helper apps, etc

  It would be useful at this point to demonstrate modifying a source code file, and rebuilding Apache with the change. The recommended example follows:

  Apache 1.3

  `src/include/httpd.h`, line 429 (in version 1.3.27, anyways)

  Modify the name of the product (SERVER_BASEPRODUCT) to be something else, like "Harry's Happy HTTPd", then rebuild. Demonstrate what a HEAD request returns now.

  Note that this is just a gimmic, not a security measure. See
  `http://httpd.apache.org/docs/misc/FAQ.html#serverheader` for more discussion on this matter.

2.4 Contents of distribution file, 2.0

Mostly the same, somewhat different layout.

Root-level directories:

- build
- docs
- include
- modules
- os
- server
- srclib
- support
- test

`docs` contains several subdirs that used to be elsewhere:

- cgi-examples
- conf
- docroot
- error (Error documents)
- icons
- man
- manual

These were placed in a docs/ folder primarily so that the documentation team could have one cvs checkout on which to work. In other words, it got moved because I complained. It's nice to have a little influence sometimes! `;-)`

The cgi examples and the default configuration are, technically, considered to be the realm of the documentation team, because they are about best practices.

2.5 Running configure

The `configure` script, located in the root directory of the Apache distribution, configures your Apache compilation.

```
./configure --help

./configure --prefix=/home/httpd --show-layout

./configure --prefix=/usr/local/apache --enable-module=most
  --enable-shared=max

./configure --prefix=/usr/local/apache --enable-module=speling

./configure --prefix=/usr/local/apache --enable-module=speling
--enable-shared=speling
```

You need to run each of the above commands on the big screen, and explain to the students what each of these commands do. Of particular importance are the lines relating to shared objects, as this will come up repeatedly later. When you are done with this section, you really should end up with an Apache installation where everything is a `so`. We will later build a module with `apxs`, and swap out the `so`, to show how easy this is.

You should also demonstrate the `config.layout` file, and building Apache with a different layout. This should be demonstrated with the `--show-layout` flag, rather than doing an actual install. You will find that installing everything in `/usr/local/apache`, as the default setting, will give you less to clean up afterwards, and make it easy for everyone to find things that they need to find during the class. It is important for everyone to have files in the same place, in order for your examples not to confuse the folks who put it somewhere else.

2.5.1 Building with mod_perl

```
gunzip mod_perl-1.xx.tar.gz
tar -vxf mod_perl-1.xx.tar
cd mod_perl-1.xx
perl Makefile.PL APACHE_SRC=../apache_1.3.20/src
  DO_HTTPD=1 USE_APACI=1 EVERYTHING=1
make
make install
```

2.5.2 Other, more complex installs

`README.configure` is a wonderful resource for installing strange and wonderful combinations of modules. And any 3rd party modules should have detailed instructions for building them.

You should read `README.configure` at least once through, to see the sorts of things that are mentioned in there. Going through one of the example installs in there, particularly one for module(s) that you're not familiar with, can be a very educational experience.

2.5.3 Apache ToolBox

http://www.ApacheToolBox.com/

- Automates build process for any combination of modules.
- Downloads libraries, modules, other stuff, that you don't already have installed
- Has to be updated for each new rev of Apache, so can be behind a little

Apache Tool Box is a shell script which automates the process of downloading and building packages. It is useful in that it prevents you from making silly typing errors, and in that it knows how to install modules with strange requirements. Also, it will download any prerequisites that you don't have, and verify the signatures on those files. This makes (usually) for a very easy installation.

Unfortunately, ApacheToolbox is only available for 1.3, Fortunately, installing additional modules on 2.0 tends to be easier and less prone to problems.

Note!

When we reinstall from Apache ToolBox, make sure that you remove/move/backup your configuration file, as `make install` will not overwrite your configuration file.

2.5.4 2.0

```
./configure --help
```

The 2.0 configure processes uses GNU autoconf, rather than the home-rolled thing that 1.3 used. So the `configure` process will look much more familiar to people used to building Unix software.

For the purposes of this class, you should built Apache 2.0 with:

```
./configure --with-mpm=worker --enable-ssl
```

We are going to add other modules as we go along. Hopefully you will become very familiar with `apxs`.

- Change `User` and `Group` directives to be valid
- Start it up

2.6 apxs

`apxs` is a Perl utility for the purpose of installing modules as shared objects. You must have `mod_so` already installed for this to work.

We're going to add `mod_rewrite` to our existing Apache installation using `apxs`

```
/usr/local/apache/bin/apxs -cia mod_rewrite.c
```

Section 3

Starting and Stopping

http://httpd.apache.org/docs/invoking.html

http://httpd.apache.org/docs/stopping.html

In this section we discuss the utilities for starting and stopping Apache, including persuading Apache to start when your system boots up. Along the way, we'll talk about the various Apache process architectures, and how to choose the one that's right for you.

3.1 Apache process architecture

It is useful to understand the Apache architecture before we go much farther.

While this may seem like a bit of a tangent, this is by far the best place to put this in, and helps understand some basic things about how Apache uses your system. It also helps understand why you have to start it as root, but why this is still secure, as well as numerous other things which will be useful in the long run.

This section is also crucial for understanding many future sections, such as performance, the `User` and `Group` directives, and security.

3.1.1 Apache 1.3

With Apache 1.3, Apache runs as multiple processes, each of which is capable of handling incoming HTTP requests. A single parent process, running as `root`, manages the pool of available servers, creating new ones as they are needed, and reaping excess ones when load is reduced.

The size of this pool of servers is controlled by the directives `MinSpareServers` and `MaxSpareServers`, which specify how many idle processes there should be at any given time. When a new client request comes in, one of the idle processes is delegated to handle that request. If this causes the number of idle processes to dip below `MinSpareServers`, then Apache will create a new child process to add to the pool. Likewise, when a client disconnects, Apache will, if necessary, kill off child processes to ensure that there are no more than `MaxSpareServers` idle processes in the pool.

The "correct" value for these directives will vary a great deal from one web site to another. The goal is to make sure that your spare server pool is sufficient to soak up any spikes in traffic.

On Windows, there is no reliable way to fork processes, and so a different model is used by default. There is a parent process, and a single threaded child process. Each thread is able to handle client connections. However, there is a fixed number of threads.

3.1.2 Apache 2.0

Apache 2.0 introduces the MPM model

The MPM - Multi Processing Module - is a way for the particular multi-processing technique of a given platform to be abstracted out. Two examples of multi-processing are threads and forking, and these are two of the available MPMs. With Apache 1.3, this code was contained in if blocks, which were long, icky, and confusing. With 2.0, they are moved out into modules, and you pick the one that is most appropriate for your particular needs and platform. On Unix, you have a number of choices. On non-Unix systems, you are usually limited to a single choice.

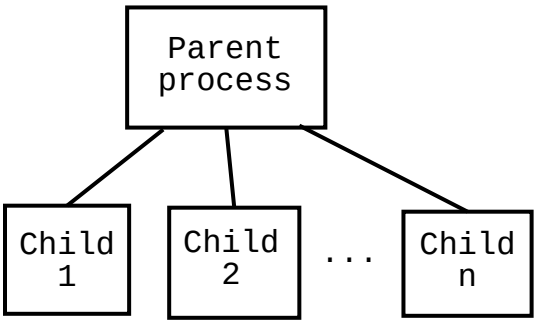
- Prefork
- Worker
- Perchild
- Win32
- OS/2
- Netware
- Various others

At this point, you may wish to rebuild Apache 2.0 using the worker MPM, so that you can see the difference that it makes in your process list, if nothing else. Since it takes a significant time to rebuild, you may want to do this and then send folks on a coffee break while it compiles.

```
./configure --with-mpm=worker

./configure --with-mpm=prefork
```

prefork

`http://httpd.apache.org/docs-2.0/mod/prefork.html`

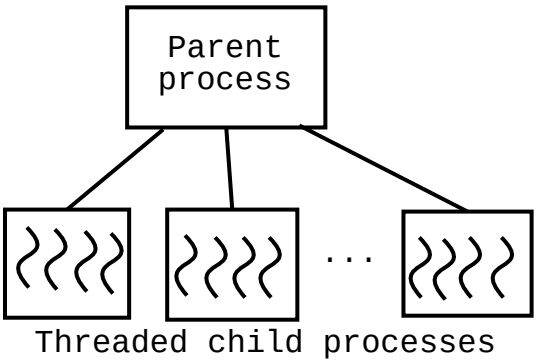


- This is the default
- Looks just like Apache 1.3
- Very robust, but perhaps slower than worker

Robust, because a crash takes out one connection only. However, creating new child processes is slower and more expensive than creating new threads. And, since they take up a bigger memory footprint, you can run fewer of them - hence, less scalable.

worker

`http://httpd.apache.org/docs-2.0/mod/worker.html`



- Multi-process, multi-threaded
- Each child, fixed number of threads
- Launch, reap child processes to deal with changes in load
- ThreadsPerChild

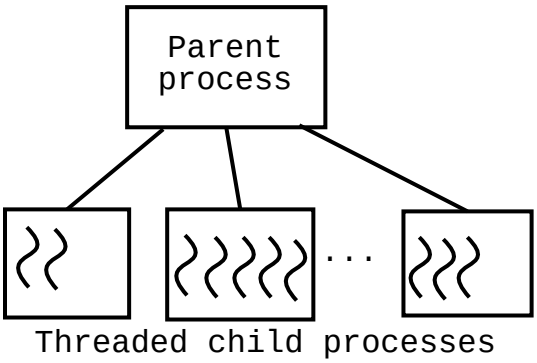
Faster, and less memory use. But less robust because a crash takes out a large number of connections, and so a single problem affects many users. This is the most-recommended MPM, with the following caveats:

- Threading does not work very well on some platforms, like FreeBSD

- Some modules, like mod_php, don't work very well in a threaded environment, and so you need to stick with prefork if you're using php.
- Of course, some modules just don't work at all on 2.0 yet, and in that case you have to stick with 1.3

perchild

`http://httpd.apache.org/docs-2.0/mod/perchild.html`



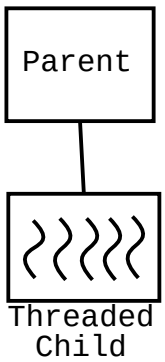
- Does not work yet
- Multi-process, multi-threaded
- Allows configuring things per child process

In a nutshell, here's what perchild does, and why it is cool:

perchild lets you configure Apache per child. Hence the name. In other words, you can actually have a different configuration for each Apache child process.

This allows you to run virtual hosts as a particular user (as opposed to just the cgi programs, like suexec lets you do). It lets you configure one vhost to run 10 threads, and another to run 200 threads. To specifically assign a particular child process to a particular vhost. And so on. It is very cool. But we should probably quit talking about it, since it is largely imaginary, and likely to remain so for some time.

win32



- Windows only
- Way faster than Apache 1.3 on Win32
- Uses completion ports for additional performance

Clearly, this course could use more meat on the Win32 sections. It would be nice to have at least screen shots of a Win32 installation, and the Apache Monitor thingy that they've added. I'm ashamed to admit that I now know almost nothing about Apache in Windows, having given a number of presentations at ApacheCon about it in the past.

3.2 apachectl

- start
- stop
  - Signals parent process
  - Parent kills child processes
  - Kills self
  - If you just kill child processes, parent will respawn them
- restart
  - Unceremoniously kills child processes
  - Re-reads config file
  - Respawns child processes
- graceful

  Like `restart`, but waits for each child to finish what it is doing.

  Note that if you have long-running processes, this will cause a restart to take a very long time. During the restart, no new connections will be accepted.
- startssl
  - Starts SSL
  - Uses -D argument
  - Talk about this when we get to SSL
- configtest
  - Checks conf files
  - Talk about this when we get to configuration
- status
  - Checks status
  - Need to have mod_status installed
  - Talk about this when we get to handlers

- help

apachectl symlink

You may want to:

```
cd /usr/local/bin
ln -sf /usr/local/apache/bin/apachectl ./
```

so that apachectl is in your path. This will make life easier throughout the rest of the course.

3.3 httpd

3.3.1 start

```
/usr/local/apache/bin/httpd

/usr/local/apache/bin/httpd -f /path/to/other/config.file

/usr/local/apache/bin/httpd -DSSL -DOtherVar
```

3.3.2 stop

```
cat /usr/local/apache/logs/httpd.pid | xargs kill
killall -9 httpd
rm -f /usr/local/apache/logs/httpd.pid
```

- httpd.pid contains the PID (process ID) of the parent Apache process
- Each child runs as a separate process
- If you kill children first, parent relaunches them

3.3.3 Other options

```
# /usr/local/apache/bin/httpd -h
Usage: /usr/local/apache/bin/httpd [-D name] [-d directory] [-f file]
                                   [-C "directive"] [-c "directive"]
                                   [-v] [-V] [-h] [-l] [-L] [-S] [-t] [-T]
Options:
 -D name          : define a name for use in <IfDefine name> directives
 -d directory     : specify an alternate initial ServerRoot
 -f file          : specify an alternate ServerConfigFile
 -C "directive"   : process directive before reading config files
 -c "directive"   : process directive after  reading config files
 -v               : show version number
 -V               : show compile settings
 -h               : list available command line options (this page)
 -l               : list compiled-in modules
 -L               : list available configuration directives
 -t -D DUMP_VHOSTS: show parsed settings (currently only vhost settings)
 -t               : run syntax check for config files (with docroot check)
 -T               : run syntax check for config files (without docroot check)
```

Have the students run this command, and experiment with various of the command line options. Encouraging the students to experiment tends to be very hard work, but it is richly rewarding for them later.

3.4 Starting at boot

Various systems will have different ways of doing this. Most Unixes have scripts in `/etc/rc.d` which run on system startup. Somewhere in here, you will want to put ...

```
/usr/local/apache/bin/apachectl start
```

.. in one of those startup scripts, like, perhaps, `rc.local` or `rc.httpd`

Note that some Unixes, like Solaris and RedHat Linux, for example, have a more developed concept of run levels, and you have to work within that framework if you want different things to happen at different run levels.

Also, many Linux'es have rather advanced ideas of what a startup script should or should not do. This can cause confusion in later tech support questions, but knowing how the underlying system works tends to simplify these questions.

Section 4

Configuration files

`http://httpd.apache.org/docs/mod/directives.html` `http://httpd.apache.org/docs/mod/core.html#include`

4.1 Introduction

Apache is configured with plain text files. While there are GUI (Graphical User Interface) tools for configuring Apache, most of this section, as well as the rest of this book, will assume that you will configure Apache using a text editor, and editing these files. (See Section 4.7 for information about GUI configuration tools.)

4.2 The files

Configuration may be in several places:

- `httpd.conf`

1.3	There used to be 3 distinct configuration files, each of which could contain only a particular type of configuration directive. `httpd.conf` was the main server configuration file, and contained directives relating to the operation of the primary functions of the server. `srm.comf` (**S**erver **R**esource **M**anagement) contained directives relating to the resources that the server could use, such as files and directories. `access.conf` contained directives relating to access control and authentication. There was a great deal of confusion as to what directives should go into which files. This confusion was actually increased when the restriction was lifted, and any configuration directive could go anywhere. Seems that people would rather be told where to put what than to be able to choose. Finally, in 1.3.7, this whole state of affairs was put to rest, and the three configuration files were combined into `httpd.conf`. The two other files were retained, but contained nothing more than a comment encouraging you not to use the files for real directives.

`httpd.conf` is the main server configuration file. Some third party distributions of Apache may call this file something different. Popular choices are `apache.conf` and `apache2.conf`.

The location of this file is built into the Apache binary, and can be ascertained by running:

```
% /usr/local/apache2/bin/httpd -V | grep SERVER_CONFIG
```

(Of course, the location and name of the `httpd` binary can vary too, so you may need to find that first.)

- `.htaccess` files
- Include another file
- Include a directory
  - Path relative to `ServerRoot`
  - All files - watch out for temp files

4.3 Config file syntax

4.3.1 Directives

```
ServerName www.apacheadmin.com
ServerAlias www apache.rcbowen.com apache
```

4.3.2 Anatomy of directive docs

Syntax

A description of the syntax of the directive

Default

The default value, if any. Many directives have a setting in the default configuration file, and that this is not necessarily the same thing as the default value.

Context

Where are you allowed to use this directive? One or more of the following:

server config - Means that it can be used in the main body of the server configuration file, outside of any containers or sections.

virtual host - Can be used within a `VirtualHost` container.

directory - Can be used in a `<Directory>` section, or similar section (ie, `<Location>`, `<Files>`, etc.

.htaccess - May be used within a `.htaccess` file, if `AllowOverride` is set appropriately. (See the value of Override for details.)

Override

What AllowOverride setting is necessary in order to permit the use of this directive in `.htaccess` files?

Status

What is the status of the module which provides this directive? (Core, Base, Extension, Experimental)

Module

What module provides this directive

Compatibility

Are there differences between different Apache versions that you need to know about?

```
AcceptPathInfo Directive

Description:   Resources accept trailing pathname information
Syntax:        AcceptPathInfo On|Off|Default
Default:       AcceptPathInfo Default
Context:       server config, virtual host, directory, .htaccess
Status:        Core
Module:        core
Compatibility: Available in Apache 2.0.30 and later
```

4.3.3 Comments

```
   # This is a comment. I like comments.
```

Comments are ignored during the loading of the configuration file. The default configuration file is roughly 70% comments, which serve to explain the configuration directives. A comment is any line that begins with a # character. The # should be the first non-whitespace character in the line. You cannot add comments onto the end of a directive line. A comment appears by itself on a line. There is no block comment notation - each line must begin with a #.

4.4 Sections

May also see them referred to as containers, or scope.

```
<Directory /usr/local/apache/htdocs>
... directives ...
</Directory>
```

A section defines the scope in which a particular directive, or directives, are effective. In the above example, the directives are effective only when content is being loaded from the directory `/usr/local/apache/htdocs`.

Sections apply to everything beneath them. That is to say, content coming out of `/usr/local/apache/htdocs/rodents` is also subject to these directives.

- `<Directory>`

  ```
  <Directory /usr/local/apache/htdocs/example>
      Options +Indexes
  </Directory>
  ```

  Restricts the contained directives to the specified directory, and subdirectories thereof.

- `<DirectoryMatch>`

  ```
  <DirectoryMatch [Dd]ownload>
      Options +Indexes
  </DirectoryMatch>
  ```

  Restricts the contained directives to directories that match the specified pattern. In the given example, any directory that contains the string `download` or `Download` will have the directive applied to it.

- `<Files>`

- `<FilesMatch>`

  Allowing you to be a little more fine-grained, you can specify a particular file or set of files. The `<Files>` directive takes just one file as an argument, which is perhaps not expected, given the name of the directive. To specify more than one file, you can use FilesMatch:

  ```
  <FilesMatch (one|two|three).html>
      SetType text/plain
  </FilesMatch>
  ```

- `<IfDefine>`

  If a particular -D variable is defined, then use this configuration. For example, you might start the server with:

```
httpd -DUSESSL
```

Then you could add a config like:

```
<IfDefine USESSL>
    SSLEngine On
</IfDefine>
```

- `<IfModule>`

  Checks to see if a particular module is loaded.

```
<IfModule mod_perl.c>
    SetHandler perl-script
</IfModule>
```

  The configuration section is completely ignored if the specified module is not loaded.

- `<Limit>`
- `<LimitExcept>`

  Limits directive scope by method. Very seldom useful.

```
<Limit GET POST>
    order deny,allow
    deny from all
    allow from 192.168
</Limit>
```

  You'll see many Authentication tutorials that tell you to use this syntax when setting up password protection. Ignore them.

- `<Location>`
- `<LocationMatch>`

  Map a URL to a handler. We'll look at Location more when we talk about handlers. See Section 14 for more details.

- `<VirtualHost>`

  Specifies the layout and function of a virtual host.

  Much more about this at a later date. See Section 6 for more details.

- `<Proxy>`

  Allows configuration to be applied to proxied content. See Section 23 for more details.

- Other

  Any third-party module is free to define its own sections. For example, `mod_perl` lets you create `<Perl>` sections containing Perl code to be executed at server startup.

4.5 Options

- `ExecCGI` (See CGI)
- `FollowSymLinks` (See security, performance)
- `SymLinksIfOwnerMatch` (See security, performance)
- `Includes` (See SSI)
- `IncludesNOEXEC` (See SSI)
- `Indexes` (See autoindexing)
- `MultiViews` (See Content Negotiation)
- `All`
- `None`

4.6 Different config file

```
apachectl -f /usr/local/apache/conf/other.conf
```

- Multiple Apache daemons
- Test configurations
- Restarting to a backup config when something new breaks

4.6.1 Running multiple apache daemons

You can run multiple Apache servers off of the same Apache binary, simply by starting it up with different configuration files.

Each Apache server process **must** run on a different port and/or address. Also make very sure that you set the `PidFile` directive differently for each server, so that each process can be managed independently.

You can talk to the correct process by using the `-f` flag on `apachectl`

```
apachectl -f /etc/apache/server1.conf -k restart
apachectl -f /etc/apache/server2.conf -k stop
```

4.7 GUI Configuration Tools

Section 5

.htaccess files

5.1 Configuration

5.1.1 AccessFileName

Configures the name of the file that will be looked for in each directory.

5.1.2 AllowOverride

- AuthConfig
- FileInfo
- Indexes
- Limit
- Options
- All
- None

http://httpd.apache.org/docs-2.0/mod/core.html#allowoverride

5.2 Performance

Checks for .htaccess in EVERY directory up to the location of the file that is being served. Possibly a big performance hit.

For example, if you are serving a file out of the directory `/usr/local/apache/htdocs/services/training/apache/tutorial` then Apache will (possibly) look for the files:

```
/.htaccess
/usr/.htaccess
/usr/local/.htaccess
/usr/local/apache/.htaccess
/usr/local/apache/htdocs/.htaccess
/usr/local/apache/htdocs/services/.htaccess
/usr/local/apache/htdocs/services/training/.htaccess
/usr/local/apache/htdocs/services/training/apache/.htaccess
```

Note that this is the absurd worst case, and never happens in practice. In order to make this happen, you'd have to have `AllowOveride All` set for `/` which nobody in their right mind would ever do. On the other hand, we should not assume anything. People unaware of the impact of this might indeed to that for convenience. But in practice, you end up checking for .htaccess files down to whatever lowest level you have permitted `AllowOverride`.

Thus, if you:

```
<Directory /usr/local/apache/htdocs>
    AllowOverride All
</Directory>
```

Then you'd really only get requests for:

```
/usr/local/apache/htdocs/.htaccess
/usr/local/apache/htdocs/services/.htaccess
/usr/local/apache/htdocs/services/training/.htaccess
/usr/local/apache/htdocs/services/training/apache/.htaccess
```

Which is still a little much, but less serious. Note, however, that dropping the `AllowOverride` directive in this scenario gives 2- or 3-fold performance improvement, even for directories where you're not using `.htaccess` files at all, based on my benchmarks. I'd recommend that you do your own benchmarks to verify these numbers.

- Directives found will be applied in the order that they are found, overriding previous settings.
- This is done every time a file is served
- Can have multiple settings for `AccessFileName`

  ```
      AccessFileName .htaccess .acl directory.conf
  ```

  In which case, it will look for each of these files in each directory

5.3 Exercise

Create `.htaccess` file in your document directory, containing the following, or something like it:

```
DirectoryIndex default.htm
```

Create a `default.htm` file in that directory, and get the `.htaccess` file working.

Students seem to require a lot of hand-holding here. They will need to do a number of things to get this working, and the idea is to get them to do as much of it by themselves as possible. They will need to:

Create `default.htm`

Create `.htaccess`

Enable `AllowOverride` appropriately for the directory in question.

Encourage them to use the correct `AllowOverride` setting, rather than just using `All`. Impress upon them the dangers of using `AllowOverride Options`, which is, of course, included in `All`.

Section 6

Virtual Hosts

http://httpd.apache.org/docs/vhosts/index.html

A virtual host (vhost for short) is a means of running more than one web site on the same Apache server. This can be done one of two ways:

- IP-based virtual hosts
- Name-based virtual hosts

Fortunately, the procedure is almost identical. There's really just one small difference.

What you'll need today

- Ability to edit hosts file
- Basic understanding of DNS and/or name resolution in general
- Some patience

6.1 IP-based virtual hosts

http://httpd.apache.org/docs/vhosts/ip-based.html

- Requires a unique IP address for each host
- The preferred (only, really) way to do SSL hosts
- A little less confusion as to which vhost gets picked

```
<VirtualHost 192.168.1.7>
    ServerName www.foo.com
    DocumentRoot /home/foo/htdocs

    CustomLog /home/foo/logs/access_log common
</VirtualHost>
```

You should use the IP address, rather than the server name, as the argument to the `VirtualHost` directive.

Most directives are valid in a VirtualHost section. See the docs for a particular directive to see what context they are valid in.

If you like, you can have students bind a secondary IP address to their network card, and set up virtual hosting that way. Alternately, and much easier, you can have them set up vhosts on 127.0.0.2, 127.0.0.3, etc. This can be entertaining, and often students learn something they didn't know about networking in the process.

Also, setting up IP-based vhosts first tends to help people understand name-based vhosts a little better.

6.2 Name-based virtual hosts

http://httpd.apache.org/docs/vhosts/name-based.html

- One IP address, multiple hosts
- Can't do SSL this way
- Can end up having overlaps and/or conflicts if you misunderstand the configuration

```
NameVirtualHost 192.168.1.7

<VirtualHost 192.168.1.7>
    ServerName www.foo.com
    ...
</VirtualHost>

<VirtualHost 192.168.1.7>
    ServerName www.boxofclue.com
    ...
</VirtualHost>
```

Notes:

- Argument to NameVirtualHost and to VirtualHost must the exactly the same literal string, or it will not know that you are dealing with the same address Best to use * rather than a particular address

```
NameVirtualHost *

<VirtualHost *>
    ServerName www.coopermcgregor.com
    DocumentRoot /usr/local/apache/vhosts/cmi
</VirtualHost>

<VirtualHost *>
    ServerName www.boxofclue.com
    ServerAlias boxofclue.com clueful.com
    DocumentRoot /usr/local/apache/vhosts/clue
</VirtualHost>
```

- Can specify a port:

```
NameVirtualHost *:443
NameVirtualHost 192.168.1.5:8080
```

- The only real difference between name-based and IP-based (in the configuration, that is) is the NameVirtualHost directive.
- You only need to put directives in one of these sections where they differ from the global setting
- See later section on logging for info about per-vhost logs

Make sure that the class understands that these vhosts are running on the same IP address, and the same port. The only way that the server knows which vhost you want is the `Host:` header that gets sent with the request. This is why old browsers (ie, pre 1996 - nothing to worry about) and telnet requests tend to get the wrong vhost.

6.3 General caveats, comments

- If you set one host up as a vhost, you should probably set all hosts up that way. 50+ % of the vhost problems that we see on #apache are because default configurations are overriding vhost configurations
- Use Listen not Port
- Consider putting each vhost config in its own file. This will save you hassle and confusion later

6.4 Exercise

1. Add 2 or more additional names to your "/etc/hosts" file (virtual1 and virtual2)
2. Set up a virtual host for each one (put vhosts in /usr/local/apache/vhosts/`servername`)

3. Verify that they are serving content out of different directories
4. If you run into any problems, ask the folks on #apache for help. They have promised to be nice.

6.5 Additional notes, examples, etc

6.5.1 /etc/hosts

`/etc/hosts` is a file containing mappings from name to IP address for your machine. Entries in the file look like:

```
192.168.1.104 virtual1
```

Once that entry has been added, the name `virtual1` will immediately start resolving to the IP address 192.168.1.104. This is just for you. other people will not see this mapping. You may need to restart your browser to have this take effect, as most browsers cache name records.

6.5.2 Your example virtual host sections

For the exercise above, you should have ended up with a configuration that looked something like:

```
NameVirtualHost *

<VirtualHost *>
    ServerName virtual1
    DocumentRoot /usr/local/apache2/vhosts/virtual1
</VirtualHost>

<VirtualHost *>
    ServerName virtual2
    DocumentRoot /usr/local/apache2/vhosts/virtual2
</VirtualHost>
```

6.5.3 #apache

#apache, referred to in the above section, is an IRC channel. If you are not familiar with IRC, this may seem a little odd. IRC - Internet Relay Chat - is real-time chat over the Internet. #apache is the name of a "channel" on which people talk about (most of the time) Apache and related topics. #apache is on the `irc.openprojects.net` network.

If you have a IRC client installed (XChat is nice) you can connect to this server, and join this channel. There is usually at least one person there who knows what they are talking about.

6.5.4 mod_vhost_alias

At this point in the course, if we have time, we'll experiment with mod_vhost_alias, which is a module allowing bulk virtual hosting. You'll find a sample configuration in the `examples` directory on your CD, which should look something like this:

```
VirtualDocumentRoot /usr/local/apache/vhosts/%1.1/%1.2/%1.3+/htdocs
VirtualScriptAlias /usr/local/apache/vhosts/%1.1/%1.2/%1.3+/cgi-bin
```

As this is primarily a hands-on experiment, little space is given to this in the notes. Please see http://httpd.apache.org/docs-2.0/mod/mod_vhost_alias.html for more information.

Students tend to want more experimentation here, so I've attempted to add some better examples that they can play with. Have them try one or more of the following:

```
VirtualDocumentRoot /usr/local/apache/vhosts/%2/%1/htdocs
```

Which should map `www.foo.com` to `/usr/local/apache/vhosts/foo/www/htdocs` and `bob.foo.com` to `/usr/local/apache/vhosts/foo/bob/htdocs`

or ...

```
VirtualDocumentRoot /usr/local/apache/vhosts/%1.1/%1.2+/htdocs
```

Which should map `www.foo.com` to `/usr/local/apache/vhosts/w/ww/htdocs` and `bob.foo.com` to `/usr/local/apache/vho`

In order to actually try these things, they will need to create a lot of directories and add a lot of hostfile entries. This is only useful in practice for wildcard DNS entries.

Section 7

MIME

http://httpd.apache.org/docs/mod/mod_mime.html

MIME - Multipart Internet Mail Extensions - was created in order that attachments could be sent via email, rather than having email restricted to plain text only. HTTP is built around MIME, and headers in general. Perhaps this section would better be labeled **HTTP Headers**.

- Multipart Internet Mail Extensions
- RFC 2045 - 2049
- HTTP based entirely on MIME standards
- MIME header tells the browser what type of document it is getting
- Content-Type: major/minor
- Content-Type: text/html
- Content-Type: image/gif
- Content-Type: Application/Unknown
- The browser has a list of mappings to applications, so that it knows how to display the content.

7.1 HTTP headers

Slight rewind here - HTTP is all about headers. Most of the information about a transaction is contained in the headers. The body is actually quite uninteresting (at least from a protocol perspective).

http://webtools.mozilla.org/web-sniffer - Tool for viewing the complete HTTP transaction, including all headers.

Or, try `http://www.web-caching.org/showheaders.html`

or, `http://www.mdb.ku.dk/tarvin/http_tool`

- Content-Length
- Content-Encoding
- Location
- Format is Header: value
- Headers are not case-sensitive
- Headers are terminated by a blank line

```
Header: value
Header: value
Header: value

Body here
```

- The message is over when `Content-Length` bytes have been served.
- On dynamic documents, either the size is calculated before the document is delivered, or it is delivered in chunks, with a `Content-Length` header on each chunk. (This is called "chunked encoding".)

7.2 MIME configuration

TypesConfig

- conf/mime.types
- Maps types to file extensions
- audio/x-realaudio ra
- video/mpeg mpeg mpg mpe
- Don't add mime types here
- Not case sensitive, dot not required

The reason that you don't edit the TypesConfig file is that on "make install" the TypesConfig file is overwritten, always, no matter what changes you have made to it. Changes may have been made to this file at the standards-board level, and you need to get those new file types, or whatever. So you should make your changes using the directives below to alter the MIME types mappings.

7.2.1 AddType

- AddType image/png .png
- `AllowOverride FileInfo` lets you put these in .htaccess files

```
AddType application/x-tar tgz
```

Note that a surprising number of students will want to get pedantic about the "x-" in the mime type above. It's not clear why it is there, but browsers recognize it, and expect it. Although Internet Explorer will also work without it it, some older versions of Netscape will not.

7.2.2 RemoveType

- Removes a mapping that was previously in place

  The scenario here is a site that has a cgi directory with examples of the software, and then a download subdirectory containing the source code, so that people can download and examine the code. While better directory organization might be more in order, this at least illustrates the concept. See also RemoveEncoding.

```
<Directory /www/docs/products>
Options +Includes +ExecCGI
AddType application/x-httpd-cgi cgi
</Directory>

<Directory /www/docs/products/conference>
Options -ExecCGI
RemoveType cgi
</Directory>
```

- Removes the mapping
- Remember that directives trickle down through the directory tree unless explicitly overridden like this

7.2.3 DefaultType

- This type is used unless another is explicitly set
- Core directive, not a mod_mime directive.
- text/html by default, if not set

7.2.4 ForceType

- Sets the MIME type for all files in the scope, regardless of filename
- ForceType image/gif

Example: Images uploaded from digital camera called `dc00034`, `dc00035`, `dc00036`, etc, without a file extension. Rather than having to rename all the files to `something.jpg`, I can just:

```
<Directory /usr/local/apache/htdocs/photos>
    ForceType image/jpg
</Directory>
```

7.3 mod_mime_magic

- Determines file type based on content of file.
- Runs the `file` program to determine
- magic.conf contains mappings to mime types

7.4 Encoding

- Usually compression
- Can be other encoding, such as uuencode
- Is additional to content type
- Can have multiple file extensions to convey this information
- resume.doc - Microsoft Word document
- resume.doc.zip - PKZipped Microsoft Word document
- The default is that a file is sent as is, with no encoding

7.4.1 AddEncoding

- Adds an encoding mapping to a particular file extension
- AddEncoding pkzip .zip
- AddEncoding gzip .gz

7.4.2 RemoveEncoding

- RemoveEncoding gz
- Removes any encoding that has been associated with the specified file extension

This directive has been useful on the Apache download site itself. Files with a .tar.gz file extension should probably not be sent with a gzip encoding, as this will cause them to be uncompressed upon arrival, which is typically not the desired behavior. Likewise, on the Apache site, we want .tar.gz.asc files that contain the gpg signature for the corresponding .tar.gz file, but are not themselves either a .tar or a .gz file. Thus, when sent with a gzip content encoding, they arrive as a zero-byte file, since there is no valid gzip content in the file.

7.4.3 mod_gzip

More will be said about mod_gzip later. Files are compressed as they are sent out to the client, and an additional `Content-Encoding` header is attached to the file to let the browser know that the content needs to be decoded (uncompressed) before it can be displayed.

7.5 Language

- Content-Language header specifies the language that is being sent
- Browser configuration can determine the preferred language (See Content Negotiation)
- Can be set in addition to other attributes

7.6 Multiple file extensions

Files can have more than one file extension in order to convey more than one of the above pieces of information.

```
file.tar.gz.en
file.tar.Z.fr
file.html.gz.de
```

There's no great value of going into much detail here with the language stuff, as we will get into much greater depth in the Content Negotiation section, which is just a few down the line.

You may also wish to make a note that multiple file extensions conveying the same piece of information will cause all but the last one to be ignored. Thus file.doc.tar.txt is a text file, not a doc or tar file.

7.7 Experiment

- In your vhost directory, create a file called `something.abc`
- Add a MIME type to this file extension
- Verify that your browser loads it with this MIME type, and asks you what it is supposed to do with it.

The purpose of this exercise is to illustrate that you can make up your own file types, and cause the browser to behave a certain way upon receiving that file type. This is how people come up with custom plug ins, file handlers, or whatever. You have some client-side application that gets mapped to a particular MIME type header.

Note also that Internet Explorer occasionally thinks that it is smarter than you. That is, it will, in some conditions, follow the file extension rather than the MIME type. Thus, a txt file with an Application/Unknown

MIME type (in order that the user will be forced to choose a file location to download and save the file, for example) may in fact be displayed in the browser as plain text by IE, which follows the file extension rather than the MIME type.

While this is clearly undesirable behavior, you should note that somewhere between 70 and 98% of your audience will be running IE, and you must plan accordingly.

Fortunately, this is not a web design course. However, this is a reminder to test with several browsers in order to assure correct behavior everywhere.

Section 8

URL Mapping

http://httpd.apache.org/docs/sections.html

8.1 URL Mapping procedure

The process of translating a URL into an actual something that is sent out to the user.

While most folks seem to subconsciously assume that URLs map to some file on the server, this is not always the case. In fact, as the web becomes progressively more dynamic, this is less and less the case. URLs map to resources, where the definition of "resource" varies greatly.

The URL mapping phase is when the server tries to figure out what a URL means - ie, what "resource" the URL refers to, whether it is actually a file, or something else entirely.

This section covers the various ways that the server administrator can force a particular URL to map to a particular resource, with the default behavior being to try to look for a file of the specified name.

8.2 Location

- Not tied to file space
- Usually maps to a handler or script

The `Location` container is used to limit the scope of directives to a range of URLs - a subset of "url-space", if you will. It is often used for the purpose of mapping URLs to a particular handler, but this is by no means the only way that it can be used. It can occasionally be used as a substitute for the Directory directive, but the meaning is somewhat different, in that it matches a URL rather than a directory, and so applies to URLs that don't actually map to a directory.

8.3 Alias

- Maps a URL to a directory, often outside of the DocumentRoot directory

Used to map a URL to a directory - usually a directory outside of the main document directory, although this is not necessarily the case. The default distro, for example, comes with an Alias for the documentation (/manual) which points to a directory within the document root. This is so that you can move it if you want to, but is really rather redundant. For one release (1.3.25 perhaps?) /manual actually did move outside of the document root, but it moved back in the next release because it irritated people who don't like change.

The Alias for /icons/, on the other hand, points to a directory that has always been outside of the document root.

Important note about Alias. The slashes must match. That is, if the first argument contains a slash, the second one should also. Thus:

```
Alias /foo /var/www/foo
```

Note that the first argument (`/foo`) has no trailing slash, and, thus, the second argument (`/var/www/foo`) also should not. When the slashes don't match, bad things happen. Alias is taken very literally. The string in the URL is replaced verbatim with the argument provided. This can result in file paths with too many slashes, or two few, depending on which side you erred.

For example, if you have:

```
Alias /icons/ /usr/local/apache/icons
```

You will end up with errors in your logs which say something like:

```
File /usr/local/apache/iconssomething.gif not found
```

Note the missing / between `icons` and `something.gif`. That's your clue that this is what is happening.

8.4 ScriptAlias

- Maps a URL to a directory, and indicates that the directory contains CGI programs

```
ScriptAlias /cgi-bin/ /usr/local/apache/cgi-bin/
```

Equivalent to ...

```
Alias /cgi-bin/ /usr/local/apache/cgi-bin/
<Directory /usr/local/apache/cgi-bin/>
Options +ExecCGI
SetHandler cgi-script
</Directory>
```

8.5 AliasMatch and ScriptAliasMatch

- Just like Alias and ScriptAlias, but with regular expressions

8.6 Regular Expressions

This is a bit of an aside, but is useful for the rest of this stuff to make sense.

Regular expressions are a means of matching arbitrary patterns in text. It can be a very full-featured library of pattern matches. Here's the smaller list of them:

You can actually spend a pretty substantial amount of time on Regular Expressions if you really want to. Here we try to keep it down to the basics, with a little more detail in the next few pages. Apache 1.3 uses the regex engine from egrep (or at least that same one) and Apache 2.0 uses PCRE, which is more full-featured.

.	Matches anything
+	One or more of the previous character
	Zero or more of the previous character
[ ]	Character class - match one thing in here
?	Optional
^	Start of string
$	End of string
^	(Inside a character class) Not

8.7 Redirect

- Maps a URL to an external URL. (Alias is always to an internal document)

```
Redirect /HyperCal.html http://www.coopermcgregor.com/products/hypercal/
```

8.8 RedirectMatch

- With regexes

```
RedirectMatch [sS]upport(.*) http://www.coopermcgregor.com/support/
Redirectmatch [dD]r[Bb]acc?h?us.* http://www.drbacchus.com/
RedirectMatch (.*) https://otherserver.com$1
```

8.9 RedirectTemp and RedirectPermanent

- Generate different redirect codes

8.10 DocumentRoot

- If all else fails, it must be a request for an actual document, so we look in the DocumentRoot for the path requested.

8.11 Error documents

- Maps an error condition to a more useful error message
- ErrorDocument 404 /cgi-bin/404.cgi
- ErrorDocument 404 http://www.errors.com/
- ErrorDocument 500 /errors/500.html
- ErrorDocument 403 "You need to log in first"

Quotes on ErrorDocument

In Apache 1.3, when you specify a string argument to `ErrorDocument`, you start with quotes, but do not close the quotes on the end of the string. In Apache 2.0, you need to close the quotes.

Example of a 404 CGI handler.

In your configuration file, put:

```
ErrorDocument 404 /cgi-bin/404.cgi
```

Then `/cgi-bin/404.cgi` will look like:

```
#!/usr/bin/perl
use Mail::Sendmail;
use strict;

my $message = qq~
Document not found: $ENV{REQUEST_URI}
Link was from: $ENV{HTTP_REFERER}
~;

my %mail = (
To => 'admin@server.com',
From => 'website@server.com',
Subject => 'Broken link',
Message => $message,
);
sendmail(%mail);

print "Content-type: text/html\n\n";
print "Document not found. Admin has been notified";
```

This is a good hands-on exercise for the students, in that it will suggest to them things that they can do in their own environment that will be more useful. The tests can be made conditional, or set up to send batch email rather than one per error, or other things. You may wish to implement a number of these on your own, so that you can display a few example alternatives.

8.12 Error documents in Apache 2.0

Apache 2.0 has a new way of handling ErrorDocument that will mean much more customizable error messages, rather than the same old boring "Document Not Found" errors.

In you Apache 2.0 default configuration file, you will see the following:

```
<IfModule mod_negotiation.c>
<IfModule mod_include.c>
   Alias /error/ "@@ServerRoot@@/error/"

   <Directory "@@ServerRoot@@/error">
       AllowOverride None
       Options IncludesNoExec
       AddOutputFilter Includes html
       AddHandler type-map var
       Order allow,deny
       Allow from all
       LanguagePriority en es de fr
       ForceLanguagePriority Prefer Fallback
   </Directory>

   ErrorDocument 400 /error/HTTP_BAD_REQUEST.html.var
   ErrorDocument 401 /error/HTTP_UNAUTHORIZED.html.var
   ErrorDocument 403 /error/HTTP_FORBIDDEN.html.var
   ErrorDocument 404 /error/HTTP_NOT_FOUND.html.var
   ErrorDocument 405 /error/HTTP_METHOD_NOT_ALLOWED.html.var
   ErrorDocument 408 /error/HTTP_REQUEST_TIME_OUT.html.var
   ErrorDocument 410 /error/HTTP_GONE.html.var
   ErrorDocument 411 /error/HTTP_LENGTH_REQUIRED.html.var
   ErrorDocument 412 /error/HTTP_PRECONDITION_FAILED.html.var
   ErrorDocument 413 /error/HTTP_REQUEST_ENTITY_TOO_LARGE.html.var
   ErrorDocument 414 /error/HTTP_REQUEST_URI_TOO_LARGE.html.var
   ErrorDocument 415 /error/HTTP_SERVICE_UNAVAILABLE.html.var
   ErrorDocument 500 /error/HTTP_INTERNAL_SERVER_ERROR.html.var
   ErrorDocument 501 /error/HTTP_NOT_IMPLEMENTED.html.var
   ErrorDocument 502 /error/HTTP_BAD_GATEWAY.html.var
   ErrorDocument 503 /error/HTTP_SERVICE_UNAVAILABLE.html.var
   ErrorDocument 506 /error/HTTP_VARIANT_ALSO_VARIES.html.var

</IfModule>
</IfModule>
```

In the directory `@@ServerRoot@@/error/` you will find all of those .html.var files, which contain SSI directives for building custom ErrorDocument pages. And, thanks to the efforts of several people, they are avaialable in several languages. These error documents can be customized to your heart's content.

The 404 page, for example, looks like the following, in English:

```
------------------------------

 <!--#set var="TITLE" value="Object not found!" -->
 <!--#include virtual="include/top.html" -->

    The requested URL was not found on this server.

  <!--#if expr="$HTTP_REFERER" -->

    The link on the
    <a href="<!--#echo encoding="url" var="HTTP_REFERER"-->">referring
    page</a> seems to be wrong or outdated. Please inform the author of
    <a href="<!--#echo encoding="url" var="HTTP_REFERER"-->">that page</a>
    about the error.

  <!--#else -->

    If you entered the URL manually please check your
    spelling and try again.

  <!--#endif -->

 <!--#include virtual="include/bottom.html" -->

-----------------------------------
```

As you have the full array of SSI variables at your disposal, this lets you customize this page as much as you like.

8.13 Other modules that handler URL mapping

8.13.1 mod_speling

- Corrects common typos
- CheckSpeling On

Handles transposition of characters, common mistypes (l instead of 1, o instead of 0, etc) and mis-capitalizations. Makes things run slower, but is very useful in migrating from Windows to Apache, for example.

Also, if there are several possible matches, you will get a listing of choices.

Exercise: Turn on `mod_spelling` checking for your server. Experiment with URL correction

8.13.2 mod_rewrite

- Alter URLs on the fly as they come in
- Will be covered in more detail on day 5 Examples in the slides, but we'll not dwell on them, as they will be discussed in detail later.

8.13.3 mod_userdir and public_html

- URLs that begin with ~ (tilde) map to that user's directory

```
http://www.uky.edu/~rbowen/
```

- UserDir specifies where that home directory is supposed to be

```
# Serve files out of /home/username/public_html
UserDir public_html

# Serve files out of somewhere else
UserDir /www/users/*/htdocs
```

- Be careful with permissions.

  There should be a discussion here about file permissions, home directory security, and why things are the way that they are. Possibly talk about perchild here, although that may be getting a little silly, now that perchild has been untouched for more than 6 months.

  If UserDir is set to public_html, then you must assure that /home and /home/username and /home/username/public_htm are all readable and executable by the Apache user (defined in the User directive). The x is necessary in order for Apache to get directory listings.

  If the users are concerned that this creates an unacceptable security situation, then they are paying attention. Good job.

  Users should not put anything directly in /home/userdir, but should put it in subdirectories thereof, and this really should take care of any security concerns that they may have. I've never quite understood why people got so uptight about this.

  On the other hand, the contents of the public_html directory is world readable, which means that even content that is password protected on the web will be fully available to anyone that has an account on the server itself. This can be a concern on systems where multiple users are rending web space. There's not really any way around this, unfortunately.

- Disable for certain users

```
UserDir enabled
UserDir disabled root hackerdude rbowen
```

  Or, better still, only enable for a few trusted users:

```
UserDir disabled
UserDir enabled rbowen sungo krietz
```

Section 9

Content Negotiation

http://httpd.apache.org/docs/mod/mod_negotiation.html

http://httpd.apache.org/docs/content-negotiation.html

Content negotiation is a server-side means of choosing for the user the document that best suits their preferences, as configured in their browser settings

This is accomplished by a combination of client-side settings and server-side configuration.

Content Negotiation does not translate documents. Try not to laugh when a student asks you this. It may seem absurd, but at least one student will ask how it manages to translate the documents into different languages, and how many languages it knows, or variations on that theme.

9.1 Client configuration

9.1.1 Accept* headers

Sent by the client to say what content types, and languages, they wish to receive. See content-negotiation.html (full URL above) for more complete discussion.

`Edit -> Preferences -> Navigator -> Languages` for language configuration options.

At this point, you should show the students the output of printenv, or another of the standard CGI programs that displays environment variables, and discuss the Accept* headers at length, making sure that they understand what each means. You should configure your browser to request several languages in order to make these headers more interesting.

9.1.2 Quality factors

Associated with each content type, these further specify what document types (language, content type, character set) are preferred over others.

9.2 Negotiation Methods

9.2.1 MultiViews

```
Options +MultiViews
```

Multiple variants of a particular document are placed in a directory, and `MultiViews` turned on for that directory. Following the algorithm described in the documentation, the file that most closely matches the preferences specified by the user is chosen and returned.

Example:

Client requests the resource `index`

In the directory, we have the following files:

```
index.html.en
index.html.fr
index.html.de
index.txt
index.pdf
```

If the client browser specifies that it prefers to receive documents in German, then this client will receive the document `index.html.de`, because it most closely matches the client's requirements.

Content negotiation via `MultiViews` is very slow, as it must get a directory listing in order to consider the files that match the name of the resource requested.

Important to note that the client can request the resource as `index` rather than `index.html`, in order to consider a wider range of possible variants of the file. If the resource `index.html` is requested instead, the files `index.txt` and `index.pdf` would not even be considered.

9.2.2 Type map files

Rather than leaving Apache to fend for itself, you can do some of the work for it by creating a type-map file, listing all variants of a particular document, and the information about these variants.

This file would be called either `example.var` or `example.html.var`, depending on whether you wanted to negotiate for the URL "example" or "example.html".

```
URI: example

URI: example.html.en
Content-type: text/html
Content-language: en

URI: example.html.fr
Content-type: text/html;charset=iso-8859-2
Content-language: fr
```

You can also specify content quality:

This file would be called `picture.var`

```
URI: picture

URI: picture.jpg
Content-type: image/jpeg; qs=0.8

URI: picture.gif
Content-type: image/gif; qs=0.5

URI: picture.txt
Content-type: text/plain; qs=0.01
```

The scenario here is that you have a particular picture available in a high-quality jpeg image, a low-quality gif image, and as ASCII art. Depending on the browser's preferences, they may get one version or another. Presumably, a text-only browser which rejects images entirely could still get the ASCII art version of the image, and still (sort of) appreciate the experience.

You'll need the following to enable this:

```
AddHandler type-map .var
```

9.3 Caching

Caching can be problematic because it may mean that a client might get a document that was right for someone else, but not for themselves.

For example, Pierre down the hall goes to CNN.com first thing in the morning, and gets the site in French, which is then cached. The rest of the day, everyone else keeps getting it in French, because they are getting it from the caching proxy server.

Note, however, that CacheNegotiatedDocs is off by default, and it's unlikely that anyone will ever turn it on.

```
CacheNegotiatedDocs Off
```

Section 10

Indexing with mod_autoindex

http://httpd.apache.org/docs/mod/mod_autoindex.html

If a directory is requested, such as `http://www.rcbowen.com/imho/`, then Apache can ...

1. Serve an index file
2. Display a file listing
3. Product an error message

10.1 Options

The `Indexes` option turns on the ability to display a directory listing in the event that there is no index file in the directory.

```
<Directory /usr/local/apache/icons>
Options +Indexes
</Directory>
```

10.2 DirectoryIndex

This directive specifies the file that is to e served when a directory is requested. Multiple files can be listed in the priority order in which they are to be considered.

```
DirectoryIndex index.html index.php index.cgi
```

10.3 IndexOptions

The arguments to `IndexOptions` are as follows

You should demonstrate each of these options so that the students can see what happens Note that for the ones that are Apache 2.0 specific, you will need to switch over to that configuration file, which will confuse the students that are following along. Encourage them to try each of these options as well.

- None

  IndexOptions None causes a directory listing to be generated as a simple bullet-list of items with links to the file itself.

- FancyIndexing

  This option is necessary for all following ones. That is, turning on FancyIndexing enables the use of all other index options.

- DescriptionWidth

- NameWidth

  Allows you to set the number of characters available for the description of the file or directory. Or the name of the file or directory. Respectively.

- FoldersFirst

  Display the folders first in the listing, as people are used to seeing in various file managers.

- HTMLTable (Apache 2.0)

  The default directory index listing is not HTML-complient, because it contains formatting and image tags inside pre tags. Some people get uptight about stuff like this. Displaying the listing as an HTML table gets around this, and produces HTML-complient listings. The purpose of the pre tags is to get the columns to line up, and pre-dates the availablilty of HTML tables. This feature is only available in Apache 2.0, because nobody has cared to back-port it to 1.3.

- IconHeight, IconWidth

  Adds width= and height= tags to the HTML img tag to facilitate rapid page rendering.

- IconsAreLinks

  The icons are not usually links to the file. This option makes them such. However, it also put the large blue border around each icon, which is irritating.

- IgnoreClient

- SuppressColumnSorting

  These two options should be considered together, as it is easy to get confused which is which, and why you need both of them.

  IgnoreClient ignores arguments passed in the URL for reordering the entries by column. However, the links are still at the top of each column to re-order the entries.

  SuppressColumnSorting removes the links at the top of each column, but still will honor the arguments in a URL to reorder, if the user types it in, or if it is linked to explicitly.

  So, it really only makes sense to use both of these options, or neither. I'm not sure why anyone would feel the need to do this, but perhaps a particular ordering is required for some pages.

- ScanHTMLTitles

  For HTML files, the file is opened and the value of the title tag is placed into the description field for the directory listing. Note that this is a HUGE performance hit.

- SuppressHTMLPreamble

  Useful when using HeaderName. The HTML preamble (the head and body tags, as well as the "Index of /foo" text) are ommitted, and you can replace them with the contents of the HeaderName file instead. Otherwise, any head or body tags that you put in the HeaderName file will probably be ignored by the client.

- SuppressDescription
- SuppressIcon
- SuppressLastModified
- SuppressSize

  Don't display the column in question

- SuppressRules
- TrackModified

  Send a `Last-Modified` header which reflects the last time a file in this directory was modified. Otherwise, the `Last-Modified` header will always be the time of the request, because the resulting document is being generated fresh each time.

- VersionSort

  1.1, 1.2, 1.10 rather than 1.1, 1.10, 1.2

Other directives are:

- AddIcon
- AddIconByType
- AddIconByEncoding
- DefaultIcon
- AddDescription

  Note that this directive is a substring match, not a literal file name, or even a file extension. This can cause confusion if you happen to have multiple files with similar names, and wish to give them different descriptions. Thus:

  ```
  AddDescription "The Foo page" foo.html
  AddDescription "Other foo stuff" foo
  ```

  Will not work as desired, whereas:

  ```
  AddDescription "Other foo stuff" foo
  AddDescription "The Foo page" foo.html
  ```

  will. Get it?

10.4 Additional directives

In addition to the `IndexOptions`, there are a few other directives that allow you to adjust how directory listings are displayed.

10.4.1 HeaderName

Display a file as a header in the directory listing. If this file is HTML, you can make the listing appear in the same look as the rest of your site. See `SuppressHTMLPreamble`.

```
HeaderName header.html
```

10.4.2 ReadmeName

Might perhaps be better named `FooterName`. This displays the contents of a file at the bottom of the directory listing.

```
ReadmeName footer.html
```

10.4.3 IndexIgnore

List files that you don't want to show up in the directory listing.

```
IndexIgnore .htaccess *.swp *.tmp
```

10.5 Searching and sorting

10.5.1 Apache 1.3

- Can sort by Name, Modified, Size, and Description
- `http://server/directory?S=D` - Sort descending by size
- `http://server/directory?M=A` - Sort ascending by modified date

10.5.2 Apache 2.0

- Much more full-featured sorting and searching

- `http://server/directory?P=*.jpg` - 'glob' style patterns
- IndexOrderDefault

```
IndexOrderDefault Ascending Size
```

10.6 Security Concerns

- Get to documents that are not linked
- Security by obscurity is not really security at all

Section 11

Performance Tuning

http://httpd.apache.org/docs/misc/perf-tuning.html

http://httpd.apache.org/docs/programs/logresolve.html

http://httpd.apache.org/docs/programs/ab.html

11.1 Optimization, benchmarking and profiling

Optimize the right thing

People have a tendency to spend an inordinate amount of time optimizing the wrong thing. Like, for example, optimizing something that takes 5% of the time, ignoring the thing that takes 80% of the time. Or whatever. Benchmarking something gives you at least a clue as to what is taking all the time. We're talking here about client benchmarking, and not really about profiling your actual CGI code, which is more about programming than about managing Apache, and so is, thankfully, out of scope for this course.

Note that no matter what you do, the network will always be the bottleneck

Of course, that's not really true, but it's a good myth to keep in mind. If most of your users are dialup users, or home users in general, this may be the case. If most of your users are business users, this may not be the case, but the network will still be a very big portion of any user's performance experience.

11.2 ab

ApacheBench. Benchmarking for web content.

```
/usr/local/apache/bin/ab -n 1000 http://localhost/index.html
```

Make sure that everyone runs this, ponders the results, compares it to other students' output, and so on. Make sure that they look at all the fields, and know what they mean. This may seem like a silly exercise at the time, but most of this stuff will come in useful throughout the rest of this section, and the rest of the course.

`-k` flag for KeepAlive.

Can run this against other servers as well as your own. Try not to do this without the permission of the server administrator.

Some sites will detect this as an attack, and block your address. That would be annoying.

11.3 Perl

```
use Benchmark;
use LWP::Simple;

timethese($count, {
    'Slow' => '$content = get("http://server/slow.cgi");',
    'Fast' => '$content = get("http://server/fast.cgi")',
    'EvenFaster' => '$content = get("http://server/mod_perl_handler")',
});
```

- Gives comparative times for the two documents
- Note that this includes network time
- And, of course, this can be used to time any pieces of code that you are interested in comparing
- It will be interesting to come back to this code after we have covered mod_perl

11.4 Optimizing hardware

- More RAM
- If you have to swap, all bets are off. When in doubt, buy more RAM.
- Fast disk access. RAID is good.
- Faster CPU
- Pretty much the obvious stuff. Apache does not require any custom hardware.

11.5 Tuning configuration settings

11.5.1 HostnameLookups

```
HostNameLookups Off
```

- DNS lookups take a long time
- Used to be on by default
- Now is off by default. Leave it that way
- /usr/local/apache/bin/logresolve < /usr/local/apache/logs/access_log > report
- Do log resolution and reporting somewhere other than on your production web server

11.5.2 Symbolic links

- Allow symlinks to improve performance
- FollowSymlinks requires that every file path get checked for symlinks. Not just on the file itself but on every directory leading up to it.
- SymLinksIfOwnerMatch requires not only this, but that we check ownership of every file along the way.
- For best performance, always use FollowSymLinks, and never use SymlinksIfOwnerMatch

11.5.3 .htaccess files

- Very very bad
- Must check for the existence and contents of a particular file for every directory in the path to the target.
- AllowOverride None
- Put directives in the main server configuration file

11.5.4 Negotiation

- It is slow. Turn it off if you don't need it
- On an increasingly global web, more and more sites will need this.
- Try running `ab` against `index.html` vs `index.html.en`

11.5.5 Caching and proxying

Apache ships with a caching proxy server which you can configure to cache incoming or outgoing requests.

mod_proxy

```
ProxyRequests On
CacheRoot /var/httpd/cache
```

Squid

```
http://www.squid-cache.org
```

Benefits of a caching proxy

- Faster retrieval of remote content

  If you put your organization behind a caching proxy server, and have your users proxy all of their content through it, then commonly-fetched content will be cached, so that they will access it across the LAN, rather than across the Internet.

- Faster serving of content, sometimes

  Using mod_proxy and mod_rewrite, you can have incoming requests farmed out to a list of servers, and have this content served through the proxy server.

  ```
  http://apache13/mod/mod_rewrite.html#RewriteMap
  ```

Disadvantages

- Don't always have fresh content
- Don't always get the negotiated document that you really wanted
- May get incorrect data from cached CGI program output.

Note: These last two cases should actually never happen. With `CacheNegotiatedDocs Off` and properly configured CGI programs, these resources should always specify that they don't want to be cached.

11.5.6 mod_mmap_static

Map static files into memory

Replaced in 2.0 by mod_file_cache

```
MMapFile /usr/local/apache/htdocs/index.html
```

11.6 Process Creation

- MinSpareServers
- MaxSpareServers
- StartServers

When additional servers are needed, they are started on the following schedule. One is launched in the first second, 2 in the second, 4 in the third, and so on exponentially until 32 are being launched each second. This rate is maintained until the MinSpareServers requirement is again satisfied.

11.6.1 MaxRequestsPerChild

- 0 means never kill the child
- On Solaris, there are Apache leaks. Set this to something non-zero
- On Windows, never set this to anything other than 0 * KeepAlive * KeepAliveTimeout
- Setting this too high, (or too low) causes a loss of the performance benefit of KeepAlive

11.7 KeepAlive

- KeepAlive On
- KeepAliveTimeout

11.8 CGI/Other dynamic content

- The greatest bottleneck I have had has been bad code.
- Poor algorithms can cause code to work well in low-stress testing environments, but perform very poorly when faced with large amounts of real data.
- Test with realistic data.
- If the user hits "reload" before a document loads the first time, you have just doubled the load.
- Split dynamic content over several steps if this improves performance

Section 12

CGI programming

http://httpd.apache.org/docs/howto/cgi.html

http://httpd.apache.org/docs/mod/mod_cgi.html

Today you are going to write a CGI program and get it working. This CGI program will parse form contents and put a record in a database. (Note, if DBI is not available, we'll put this in a text file or similar.)

12.1 Introduction - The CGI

- Provides an interface for arbitrary programs to provide content for web pages
- CGI spec: `http://hoohoo.ncsa.uiuc.edu/cgi/interface.html`
- CGI 2: `http://cgi-spec.golux.com/`
- Advantages of CGI
  - Easy to write
  - Easy to maintain
  - Readily available examples for download
- Disadvantages
  - Slow startup → slow runtime
  - No maintenance of state
  - Most of the 'readily available examples' are very badly written
- Alternatives to CGI
  - mod_perl
  - PHP
  - FastCGI
  - ASP

  - JSP
  - etc, etc, etc
- Each addresses the above problems in different ways, but with many of the same general concepts:
  - Persistent database connections
  - In-memory interpreter
  - Direct interface to the web server API

12.2 Apache configuration

- `Options +ExecCGI`
- `AddHandler cgi-script`
  - Adds cgi execution to particular files
- `SetHandler cgi-script`
  - Adds cgi execution to all files in the range
- `ScriptAlias`
  - This is the preferred method
  - Keep track of your CGI programs
  - Don't have to expose the mechanisms (.cgi) – See Jakob Nielsen

12.3 How a CGI program works

- Input

  Input comes in from the browser in several different formats
  - Environment variables:

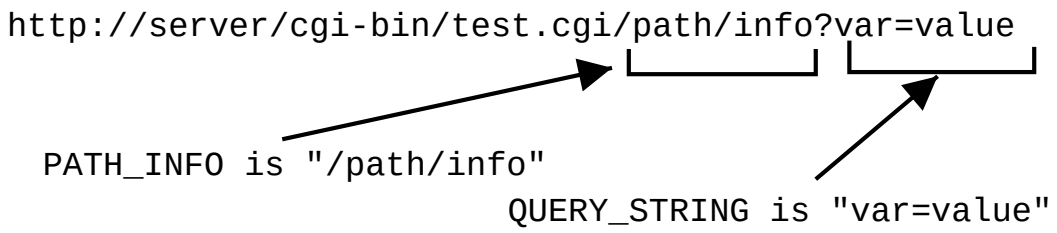
```
SERVER_SOFTWARE
SERVER_NAME
GATEWAY_INTERFACE

SERVER_PROTOCOL
SERVER_PORT
REQUEST_METHOD
PATH_INFO
PATH_TRANSLATED
SCRIPT_NAME
QUERY_STRING
REMOTE_ADDR
REMOTE_HOST
AUTH_TYPE
REMOTE_USER
REMOTE_IDENT
```

```
CONTENT_TYPE
CONTENT_LENGTH

HTTP_USER_AGENT
HTTP_ACCEPT
```

- `GET` requests

  Arguments following the end of the URL are available in the variables `QUERY_STRING` and `PATH_INFO`

```
http://server/cgi-bin/script.cgi/path/info?foo=bar&one=two
```



  `PATH_INFO` is `/path/info`

  `QUERY_STRING` is `foo=bar&one=two`

- Form input

- `POST` requests

  Form content has a similar format to `GET` information. It comes in over `STDIN`, and is formatted as `variable=value&variable=value`

- Decoding form data

  Most programming languages have some library available to decode form contents. We'll be using primarily Perl for the purpose of this tutorial, but other languages can be used.

- Output

- Content-type header

- Content

- Example programs

  Simple CGI programs in Perl

  Example 1

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "Hello world";
```

  Example 2

```
#!/usr/bin/perl
use CGI;
my $cgi = new CGI;
$form = $cgi->Vars;

print "Content-type: text/html\n\n";
print "<h2>Form values ...</h2>";
foreach my $key (keys %$form) {
    print "$key => $form->{$key}<br>";
}
```

The above example needs an HTML form, something like:

```
<form action="/cgi-bin/example3.cgi" method="POST">
   One <input name="variableone" value="default"><br>
   Password <input type="password" name="pass"><br>
   <input type="hidden" name="x" value="y"><br>
   Checkbox <input type="checkbox" name="yesno"><br>
   Input area <textarea name="textarea" rows="5" cols="40"></textarea><br>
   <input type="submit" value="Submit Form">
</form>
```

Example in sh

```
#!/bin/sh
echo Content-type: text/html
echo
echo Hello, World
```

Example in C

```
#include <stdio.h>

int main()
    printf("Content-type: text/html\n\n");
    printf("Hello, world!\n");
    return 0;
```

12.4 Common problems

- Permissions
- Syntax errors
- Invalid headers
- Asking for help in a newsgroup

Section 13

SSI

http://httpd.apache.org/docs/howto/ssi.html

http://httpd.apache.org/docs/mod/mod_include.html

Directives written into the HTML pages, which are processed by the server as the page is being served.

- Add dynamic content to an existing HTML page
- Include external text files (headers, footers)

13.1 Configuration for SSI

- `Options +Includes`
- `Options +IncludesNOEXEC`

I

f you have `AllowOverride Options` enabled (or `AllowOverride All`, which includes `Options`) then people can put these directives in their .htaccess files, overriding your security precautions.

- Enabling by file extension

```
AddType text/html .shtml
AddHandler server-parsed .shtml
```

This causes all files with a `.shtml` extension to be parsed for SSI directives.

The disadvantage of this approach is that you have to

  - Expose the mechanism (ie, everyone knows how you are generating the effect)

– You have the change the name of all the files, and break all the links to those files.

You could also set all `.html` files to be parsed for SSI directives:

```
AddHandler server-parsed .html
```

- Additional overhead on EVERY file
- Parsing files that have no directives in them
  Fortunately, there is an alternative.

13.2 XBitHack

The XBitHack directive tells Apache to parse files for SSI directives if they have the execute bit set.

```
XBitHack On
```

- on - parse files with u+x
- off - don't parse files with u+x
- full - g+x means send the last-modified date on the file itself, rather than the current time

Several advantages

- Don't have to change file names
- Don't expose the mechanism
- Guessable URLs
- Not supported on Windows - there's no x-bit

13.3 mod_include configuration directives

- SSIStartTag
- SSIEndTag
- SSIErrorMsg
- SSITimeFormat
- SSIUndefinedEcho

```
SSIStartTag "<%"
SSIEndTag "%>"
```

13.4 SSI directives

`Syntax: <!--#element attribute=value attribute=value ... -->`

Elements are ...

- `config`
- `echo`
- `exec`
- `fsize`
- `flastmod`
- `include`
- `printenv`

13.4.1 config

```
<!--#config errmsg="[It's broken]" -->
```

```
<!--#config sizefmt="bytes" --> (or abbrev)
```

13.4.2 timefmt

This supports the same time formats as strftime(). (man strftime for a complete listing.)

```
<!--#config timefmt="%B %e, %Y" -->
```

13.4.3 echo

13.4.4 exec

- `cgi`
- `cmd`

13.4.5 fsize

- `file` (full path)
- `virtual` (URL path)

13.4.6 flastmod

Displays the last modified time of a file. This is different from the `LAST_MODIFIED` variable in that you can refer to a different file than the one currently being viewed.

```
<!--#flastmod virtual="/index.html" -->
```

```
<!--#flastmod file="otherdir/index.html" -->
```

You can only display this information file files that are accessible via the web site in some way. The `file` argument is relative to the current directory, and cannot start with a leading slash, or contain `..` in the path.

The output of this directive is subject to the `#config timefmt` directive.

13.4.7 include

Include a file, or the output of a CGI program.

```
<!--#include virtual="/cgi-bin/counter.pl" -->
```

With `include` you can provide additional arguments to the CGI program, which you cannot do with `exec`:

```
<!--#include virtual="/cgi-bin/counter.pl?page=foo.html" -->
```

Most commonly, `include` is used to include a file, such as a header or footer. By using this technique, a footer can be maintained at one location, and included into any number of pages. Then, when the content of the footer is updated, it is immediately changed across the entire web site.

```
<!--#include virtual="/includes/footer.html" -->
```

13.4.8 printenv

```
<!--#printenv -->
```

13.5 Variables and flow control

```
<!--#if expr="$Mac && $InternetExplorer" -->
  Apologetic text goes here
<!--#else -->
  Cool JavaScript code goes here
<!--#endif -->
```

Also available is the `elif` keyword.

13.6 Security

- exec considered harmful
- Use 'include' rather than 'exec' for CGI programs
  - Removes security concerns with exec
  - Able to pass `QUERY_STRING` arguments

Section 14

Handlers and Filters

http://httpd.apache.org/docs/handler.html

http://httpd.apache.org/docs-2.0/filter.html

http://httpd.apache.org/docs-2.0/mod/mod_actions.html

14.1 Handlers

Handlers are functions in Apache modules which produce dynamic content in response to a URL request. They are usually configured via a `<Location>` directive, or can be mapped to a particular file type.

14.1.1 Configuration directives

AddHandler

Maps a handler to a particular file extension. That is, associates a particular action or process to a given file type.

```
AddHandler cgi-script cgi pl py
```

The above directive tells Apache to consider any program with a `cgi`, `pl`, or `py` extension to be a CGI program, execute it, and pass the resulting output back to the client.

This handler is provided by the `mod_cgi` module.

SetHandler

Much like the `AddHandler` directive, but specifies that all files (or URLs) in the given scope (usually a `Location`, `Files`, or `Directory` section is to be handled by the specified handler.

```
<Location /server-status>
    SetHandler server-status
</Location>
```

The `server-status` handler is provided by the module `mod_status`.

RemoveHandler

Removes the action of a handler from a specified file extension.

```
RemoveHandler .html
```

Action and Script

Provides for the creation of custom handlers, by mapping a file type to a CGI URL.

There are two ways that this can be handled.

You can map an action to a particular type (MIME type) of file using just the `Action` directive:

```
Action image/gif /cgi-bin/watermark.cgi
```

Or, you can map a file extension to an `Action` in two steps:

```
AddHandler my-handler .gif
Action my-handler /cgi-bin/watermark.cgi
```

These two techniques are essentially equivalent. The latter creates a named handler, and then defines that handler to be the specified CGI program.

The `Script` directive is a little bit different, and seldom used. `Script` specifies that a particular script is to be used every time a particular HTTP method is used in a request.

```
Script POST /cgi-bin/post.cgi
```

14.1.2 Standard handlers

The following are the standard handlers - that is, those handlers that are defined by modules that come standard with Apache.

default-handler

Defined by the Apache core (rather than by an extension module) this is the handler that deals with requests for files. This is the default manner of dealing with a URL request if there is nothing else special about it.

This is the handler that was in use in the URL mapping section yesterday.

send-as-is

Defined by mod_asis, the `send-as-is` handler sends a file without prefacing it with any headers. It is assumed that the file itself will contain the headers as part of the content of the file.

```
Status: 301 Now where did I leave that URL
Location: http://xyz.abc.com/foo/bar.html
Content-type: text/html

<HTML>
<HEAD>
<TITLE>Lame excuses'R'us</TITLE>
</HEAD>
<BODY>
<H1>Fred's exceptionally wonderful page has moved to
<A HREF="http://xyz.abc.com/foo/bar.html">Joe's</A> site.
</H1>
</BODY>
</HTML>
```

The `send-as-is` handler is enabled using the `AddHandler` directive:

```
AddHandler send-as-is .asis
```

It is customary to name these pages `something.asis`, however, in the case of the above example, when

advertising a page that has moved, you may want to use a `<Files>` section to map the `send-as-is` handler to just a particular file.

cgi-script

Provided by `mod_cgi`, the `cgi-script` handler executes a program and returns the output of it to the client.

```
AddHandler cgi-script .cgi .pl
```

Use of this handler is permitted by use of the `Options +ExecCGI` directive.

imap-file

Sometimes a little hard to explain because it is completely archaic. There used to be server-side image maps (circa 1996) before client-side image maps came into wide-spread use.

This handler is provided by `mod_imap`, and you should consult the documentation for this module if you are interested in more detail. You are unlikely to ever use this handler.

server-info

Provided by `mod_info`, this handler provides detailed information about the server configuration and what modules are loaded.

Configured with a `<Location>` section:

```
<Location /server-info>
SetHandler server-info
</Location>
```

server-status

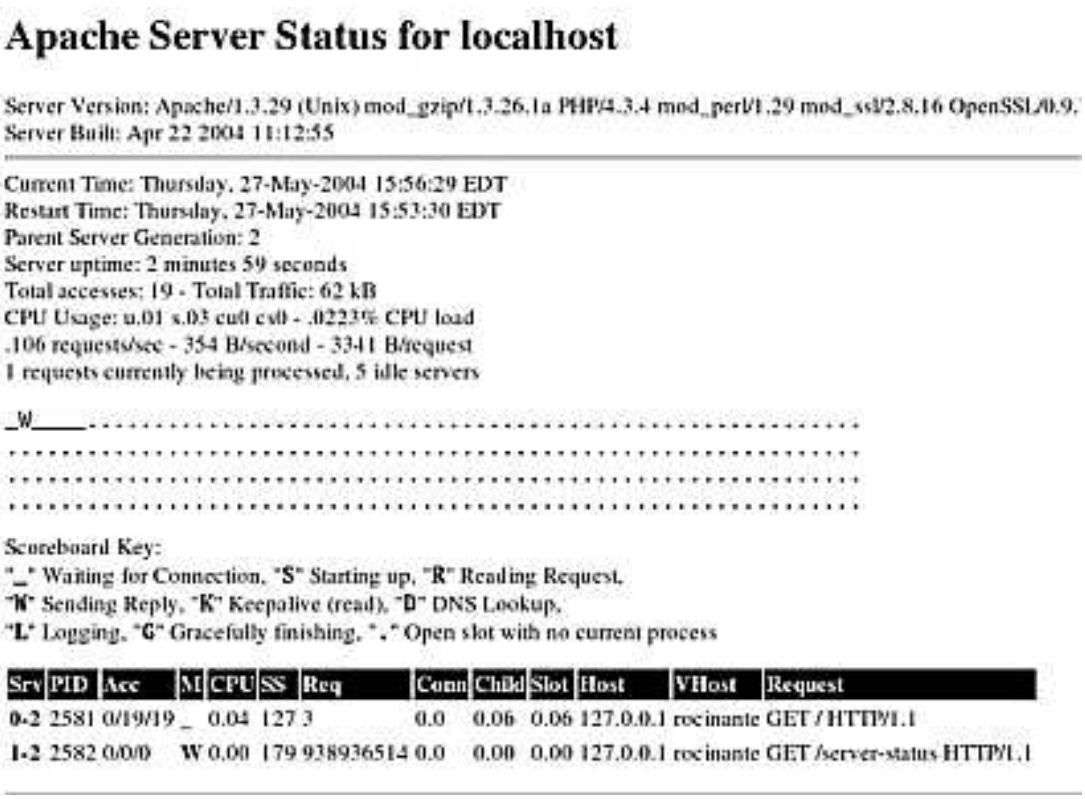
Provided by `mod_status`, this handler provides a snapshot of server activity, displaying what each child process is doing.

See also the `ExtendedStatus` directive.

You can also access this information with `apachectl status`

```
<Location /server-status>
SetHandler server-status
</Location>
```

See http://httpd.apache.org/server-status/ for a good example of this in action.



Apache Server Status for localhost

Server Version: Apache/1.3.29 (Unix) mod_gzip/1.3.26.1a PHP/4.3.4 mod_perl/1.29 mod_ssl/2.8.16 OpenSSL/0.9.
Server Built: Apr 22 2004 11:12:55

Current Time: Thursday, 27-May-2004 15:56:29 EDT
Restart Time: Thursday, 27-May-2004 15:53:30 EDT
Parent Server Generation: 2
Server uptime: 2 minutes 59 seconds
Total accesses: 19 - Total Traffic: 62 kB
CPU Usage: u.01 s.03 cu0 cs0 - .0223% CPU load
.106 requests/sec - 354 B/second - 3341 B/request
1 requests currently being processed, 5 idle servers

_W_____.....
.....
.....
.....

Scoreboard Key:
"_" Waiting for Connection, "S" Starting up, "R" Reading Request,
"W" Sending Reply, "K" Keepalive (read), "D" DNS Lookup,
"L" Logging, "G" Gracefully finishing, "." Open slot with no current process

Srv	PID	Acc	M	CPU	SS	Req	Conn	Child	Slot	Host	VHost	Request
0-2	2581	0/19/19	_	0.04	127	3	0.0	0.06	0.06	127.0.0.1	rocinante	GET / HTTP/1.1
1-2	2582	0/0/0	W	0.00	179	938936514	0.0	0.00	0.00	127.0.0.1	rocinante	GET /server-status HTTP/1.1

server-parsed

Provided by `mod_includes`, this handler parses HTML pages looking for SSI (Server-Side Includes) directives. If found, it processes these directives and replaces them with the result of the action described in the directive.

See section on SSI for more information.

Use of this handler is permitted by using the `Includes` argument to `Options`.

type-map

Provided by `mod_negotiation`, this handler indicates that a particular file is a type map file, describing the variants of a particular document. See Section 9 for more details on content negotiation.

```
AddHandler type-map .var
```

14.1.3 Custom handlers

Custom handlers can be created using the `Action` and `Script` directives described above.

For example, using the following configuration:

```
Action text/html /cgi-bin/footer.pl
```

you can add a footer to the bottom of every HTML page, using a CGI program that looks like:

```
#!/usr/bin/perl

print "Content-type: text/html\n\n";

my $file = $ENV{PATH_TRANSLATED};

open FILE, "<$file";
print while <FILE>;
close FILE;
print qq~

FOOTER GOES HERE
~;
```

14.2 Filters

Filters are a major enhancement that comes with Apache 2.0, and one of the things that has been discussed since the very early days of talking about what would be in 2.0.

Filters give you the ability to chain actions on input or output. The classic example of this is SSI and CGI. For years, a frequently asked question on the mailing list and news groups has been whether you could put SSI directives in the output of a CGI program, and then have them processed correctly before the content was served to the client. The answer to this question is no, because you have only one shot at producing dynamic content. You decide which module will handle a particular request, and that's it. You can't have it both ways.

Filter chains change the answer to no. You can specify that particular content is passed through one or more additional filters on its way out to the client. The `Includes` filter happens to be an available filter. You can

specify that your CGI-generated content will pass through the `Includes` filter on the way out the wire to the client, and those directives will be correctly processed and the desired content filled in.

```
Options +ExecCGI
AddHandler cgi-script cgi
AddOutPutFilter INCLUDES cgi
```

Alternately, you can specify that content be processed by more than one filter. In the example below, files with a `.shtml` extension are processed by the `INCLUDES` filter, and then compressed with the `DEFLATE` filter:

```
AddOutputFilter INCLUDES;DEFLATE shtml
```

14.2.1 Chaining filters - CGI + SSI

With Apache 1.3, people frequently ask how they can have their CGI programs output SSI directives, and then have those SSI directives parsed by `mod_includes`. In 1.3, the answer is that you can't do that. In 2.0, this is now possble.

In your `cgi-bin` directory section the Apache 2.0 `httpd.conf` file, add the following:

```
 Options ExecCGI Includes
 AddOutputFilter INCLUDES .cgi
```

Then, in the `cgi-bin` directory, place the following script, calling it `ssi.cgi`

```
#!/usr/bin/perl

print "Content-type: text/plain\n\n";
print "SSI: <pre><!--#printenv --></pre>";
```

Now, upon accessing `http://localhost:90/cgi-bin/ssi.cgi` you should see the complete output of an SSI `printenv` directive.

14.2.2 mod_deflate

`mod_deflate` compresses content as it is sent out to the client, in much the same way that `mod_gzip` does for Apache 1.3.

```
<Directory "/your-server-root/manual">
    AddOutputFilterByType DEFLATE text/html
</Directory>
```

Since some browsers don't correctly handle gzipped non-html content, it is recommended that you only compress html content. The directive above does that.

If you wish to create a log file a la `mod_gzip`, you'll also need something like the following:

```
DeflateFilterNote ratio

LogFormat '"%r" %b (%{ratio}n) "%{User-agent}i"' deflate
CustomLog logs/deflate_log deflate
```

Section 15

mod_perl

15.1 Overview - What is mod_perl?

mod_perl embeds a Perl interpter into the Apache process, for two purposes:

- Make calls directly to the Apache API to write Apache modules in Perl
- Improve performance of Perl CGI programs upwards of 300%

15.2 Installation

Unpack mod_perl source, change into the mod_perl directory, and then ...

```
perl Makefile.PL APACHE_PREFIX=/usr/local/apache
APACHE_SRC=../apache-1.3.20/src DO_HTTPD=1 USE_APACI=1
EVERYTHING=1
APACI_ARGS='--enable-module=rewrite,--enable-module=speling'

make && make install
```

15.3 mod_perl installation caveats

- Don't install as dso
- php

You can install it in conjunction with PHP if you are really careful, but there are frequent problems with this interaction. In particular, they seem to use conflicting versions of the mysql libraries, and this can cause conflicts if/when they both attempt to connect to a mysql database.

15.4 Configuration

15.4.1 PerlRequire

Perl commands that you want to run at startup. Preload modules into shared memory. Set global variables.

```
PerlRequire /usr/local/apache/conf/preload.pl
```

And `/usr/local/apache/conf/preload.pl` would then contain:

```
use Apache::DBI;
use DBI;
use CGI qw(:Standard);
use MyCompany::Utils;
use lib '/path/to/my/modules';
1;
```

15.5 Connecting to your database

```
Apache::DBI->connect_on_init( $database, $username, $password );
```

15.6 CGI under mod_perl

A major use of mod_perl is as a CGI speed enhancer. This can provide 10 to 20 times speed improvement, in my experience.

This is done one of two ways.

15.6.1 Apache::PerlRun

If you have cgi code that works and you don't want to spend much time ensuring that it is safe to run under mod_perl, you can just use Apache::PerlRun to run these programs and gain some of the benefits of mod_perl.

```
Alias /cgi-perl/ /usr/local/apache/cgi-bin/
<Location /cgi-perl>
    SetHandler perl-script
    PerlHandler Apache::PerlRun
    Options ExecCGI
    PerlSendHeader on
</Location>
```

Maps the URL `/cgi-perl` to your cgi-bin directory, and arranges for mod_perl to execute these cgi programs for you, rather than `mod_cgi`. You can now access your cgi programs with a new URL, and get an immediate speed improvement.

Rather than using the URL

```
http://servername/cgi-bin/test.cgi
```

you can now use the URL

```
http://servername/cgi-perl/test.cgi
```

For the purpose of this exercise, we will use the following CGI program, and call it `test.cgi`. Please type in this program and place it in `/usr/local/apache/cgi-bin`:

```
#!/usr/bin/perl
print "Content-type: text/html\n\n";
print "Hello";
```

You now should run the following command:

```
/usr/local/apache/bin/ab -n 1000 -c 5 http://localhost/cgi-bin/test.cgi
```

After noting the numbers that you get, then run the following:

```
/usr/local/apache/bin/ab -n 1000 -c 5 http://localhost/cgi-perl/test.cgi
```

Wow.

15.6.2 Apache::Registry

If you are certain that your program is well-written, uses `strict` and `warnings`, and does not abuse global varibles, try it under `Apache::Registry`

Add the following configuration:

```
Alias /perl/ /usr/local/apache/cgi-bin/
<Location /perl>
    SetHandler perl-script
    PerlHandler Apache::Registry
    Options +ExecCGI
    PerlSendHeader on
</Location>
```

Now, run:

```
/usr/local/apache/bin/ab -n 1000 -c 5 http://localhost/perl/test.cgi
```

Be impressed.

15.7 Apache handlers with mod_perl

If you really want to take advantage of the power of `mod_perl`, you should write Apache handlers using mod_perl.

15.7.1 Installing a mod_perl handler from CPAN

There are a plethora of existing mod_perl handlers available for download from CPAN (the Comprehensive Perl Archive Network - http://www.cpan.org/). We'll install one, and talk about another one.

Install `Apache::Perldoc` using:

```
# perl -MCPAN -e shell

cpan> install Apache::Perldoc
```

Configure the module using:

```
<Location /perldoc>
    SetHandler perl-script
    PerlHandler Apache::Perldoc
</Location>
```

And then use the module by going to the URL

```
http://localhost/perldoc/Apache::Perldoc
```

Generates full documentation for any Perl module that you have installed.

`Apache::Album` allows you to put image files on your server, and automatically generate image galleries with thumbnail images:

```
http://buglet.rcbowen.com/photos/
```

15.8 Writing a mod_perl handler

A mod_perl handler is a Perl module with a single method called `handler`. This method should take a single argument - the `Apache::Request` object, traditionally called `$r`, and should emit content to get displayed in the browser, using the appropriate methods from the Apache API

15.8.1 Example mod_perl handlers

```
package Apache::HandlerTest;

sub handler {
    my $r = shift; # Apache session object
    $r->content_type('text/html');
    $r->send_http_header;
    $r->print( "Hello, world." );
}
```

15.8.2 Installing the example mod_perl handler

Because Perl looks certain places for Perl modules, this module needs to be placed in the Perl library directory. There are a variety of ways to do this, and for the purpose of this course, we will just copy the file into the Perl library directory manually.

The above file is to be called `HandlerTest.pm`, and is to be placed in an `Apache` subdirectory of the Perl lib directory.

For example, if the Perl lib directory is
`/usr/lib/perl5/site_perl/5.6.0`
then the file should be placed at
`/usr/lib/perl5/site_perl/5.6.0/Apache/HandlerTest.pm`

To find a listing of what the Perl lib directories are on your particular machine, type:

```
perl -le 'print join "\n",@INC;'
```

If you don't know much about Perl, ask the instructor for assitance at this point.

15.8.3 Configuring the mod_perl handler

The handler is configured by adding a `<Location>` section in your configuration:

```
<Location /handlertest>
    SetHandler perl-script
    PerlHandler Apache::HandlerTest
</Location>
```

15.9 Common problems

15.9.1 Don't exit

Calling the Perl command `exit()` causes the Perl interpreter to exit, rendering that Apache child useless. Don't do that.

15.9.2 Restart the server

When you restart, you actually have to stop and start the server, as your code is often cached in the parent process, and so restarting the child processes does not cut it.

See also:

- PerlFreshRestart
- PerlInitHandler Apache::StatINC

15.9.3 Global values

Global values in `mod_perl` are really global. Meaning that not only can variables be seen out of scope, but they can also be seen in other child processes (maybe) and in other client accesses. This can really ruin your whole day.

15.10 Other phases

All of the `mod_perl` handlers that we have looked at so far are content handlers. That is, they return content, much the same way a CGI program would.

`mod_perl` can handle any phase of the Apache lifecycle, including access control, authentication, logging, or configuration. In this section we'll look at a few examples of this.

15.10.1 PerlAccessHandler

A `PerlAccessHandler` will allow you to do access control based on arbitrary criteria, like the phase of the moon. `Acme::Apache::Werewolf`[1] is a module that does just that.

```
<Directory /fullmoon>
    PerlAccessHandler Acme::Apache::Werewolf
    PerlSetVar MoonLength 4
</Directory>
```

The lunar cycle is 28 days, with the full moon falling right in the middle. `MoonLength` allows you to specify how many days you want to consider to be the full moon. This gives you an safety margin around the full moon.

The code itself is extremely simple:

[1]The `Acme` namespace of Perl modules indicates a module that is functional, but was written as a joke.

```
package Acme::Apache::Werewolf;
use strict;
use Astro::MoonPhase;
use Apache::Constants qw(:common);

use vars qw($VERSION);
$VERSION = '1.00';

sub handler {
    my $r = shift;
    my $moonlength = $r->dir_config('MoonLength');
    warn "Moon length is $moonlength";

    my ( $MoonPhase,
         $MoonIllum,
         $MoonAge,
         $MoonDist,
         $MoonAng,
         $SunDist,
         $SunAng ) = phase(time);

    return FORBIDDEN unless abs(14 - $MoonAge) > ($moonlength/2);
    return OK;
}
```

All the hard work in this module is done by `Astro::MoonPhase`, which calculates the phase that the moon is in currently. Given the `MoonAge` value, the module returns `FORBIDDEN` if this value is inside the window defined by the value that you provided for `MoonLength`.

15.10.2 PerlLogHandler

A `PerlLogHandler` handler will let you handle the logging phase from within Perl.

15.10.3 Perl configuration sections

15.11 More information

http://perl.apache.org/

`mod_perl` Developer's Cookbook (Geoff Young)

Practical `mod_perl` (Stas Bekman and Eric Cholet)

Section 16

Logging

Every request to your server results in an entry in a log file. If something goes wrong, it will also result in an entry in the error log file. This means that you always have a "paper trail" for everything that goes on on your server, so that you can look back and find out what happened. This is primarily useful for two purposes - troubleshooting, and statistics gathering.

In this chapter, we look at the standard log files, as well as at the custom log files which you can create to fit non-standard needs.

16.1 Standard log files

`/usr/local/apache/logs/access_log`

First we look at a standard default access log, in Common log format (CLF). While many third-party bundlings of Apache ship with the Combined log format instead, this is still the most common log format, and the Combined is just an extension of it.

16.1.1 access_log

Format ...

```
216.35.116.91 - - [19/Aug/2000:14:47:37 -0400] "GET / HTTP/1.0" 200 654
```

- `216.35.116.91` Client address (See HostNameLookups)
- `-` (placeholder) Ident (always blank)
- `-` (placeholder) Username
- `[19/Aug/2000:14:47:37 -0400]` - Date/time

- `"GET / HTTP/1.0"` Request
  - Method type (GET, POST, HEAD)
  - URL requested
  - PROTOCOL (HTTP + version number)
- `200` Status code
- `654` Bytes transferred

- Client address - This is the IP address of the client connecting to your server. There are a number of comments that should be made about this.
  - It is the IP address, not the host name. If you turn on HostNameLookups, you'll get the host name, rather than the IP address. Don't do this, for reasons that will be discussed in the performance section.
  - Some folks will get rather irate that you have this information. Of course, this is silly, as the information is probably worthless. However, there are a variety of good reasons to keep your log files highly confidential. Releasing this information (this IP address visited this page at this time) can cause embarassment, or worse. The log files are useful as statistical information, but individual entries should not be public information.
  - Note that many hosts may be behind one address, and, conversely, one host can show up, at different times, behind many IP addresses.
- Ident - This field is here for historical reasons. Long long ago (Netscape 0.9) this field would contain the email address of the person visiting your web site. The browser happily provided this information along with every request. As you can imagine, the marketing people got hold of this information, and started sending UCE to those addresses. Browsers quickly stopped providing this information, and this field has been blank ever since.

  There is a patch available at http://mm.apache.or.jp/pipermail/apache00-01/2000-July/001181.html which adds a directive (Anonymous_Email_As_Ident) which causes the "password" field supplied to be logged in the Ident field. This makes a lot of sense, actually, and we'll come back to that when we get to the Auth chapter.
- Username - Set only if the resource in question required authentication, and, in that case, will contain the username of the authenticated user.
- Date/Time - The date and time that the request was made. Granularity is 1 second, and, no, there's no way to make that finer.
- Request - Contains the full request as received from the client. This should contain three fields, such as: "GET / HTTP/1.0". The first field - the method - can be one of a dozen or so methods such as GET, POST, HEAD, CONNECT, PROPFIND, and so on. The next field is the requested URL. For a local URL, this will be the path - that is, no hostname. For proxied URLs, it will be the full URL of the remote resource. Finally we have the protocol and version, such as HTTP/1.0 or HTTP/1.1.
- Status Codes:

  100 Informational
  200 OK
  300 Redirect
  400 User error
  500 Server error
- Bytes transferred

16.2 Location and format of the log file

```
CustomLog /usr/local/apache/logs/access_log common
```

Means that the log file is to be located at /usr/local/apache/logs/access_log and is to be in the format 'common'. This format is defined by the LogFormat directive:

```
LogFormat "%h %l %u %t \"%r\" %>s %b" common
```

See `http://httpd.apache.org/docs-2.0/mod/mod_log_config.html#formats` for the full listing of options.

Other log formats:

- Common Log Format (CLF) `"%h %l %u %t \"%r\" %>s %b"`
- Common Log Format with Virtual Host

  ```
  "%v %h %l %u %t \"%r\" %>s %b"
  ```

- NCSA extended/combined log format

  ```
  "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-agent}i\""
  ```

- Referer log format `"%{Referer}i -> %U"`
- Agent (Browser) log format `"%{User-agent}i"`

The CustomLog directive can then be used to create a new log file using the format that you have created.

You can have as many log files as you like.

16.3 mod_log_io

Logs the total bytes transferred, including headers. Adding `mod_log_io` gives you two new variables to use in your log formats: %I and %O

16.4 Exercises

1. Construct a log file containing the exact time taken to serve the request, in microseconds.
2. Construct a log file that shows which virtual host the request was made to.
3. Add `mod_log_io` to your server, using `apxs`, and construct a log file containing the exact number of bytes transferred.

16.4.1 Error logs

Location of the error log

```
ErrorLog logs/error_log
```

Note that path is Relative to ServerRoot

16.4.2 LogLevel

- emerg
- alert
- crit
- error
- warn
- notice
- info
- debug

16.5 Typical errors

- Document error

  ```
  [Fri Aug 18 22:36:26 2000] [error] [client 192.168.1.6] File does not
  exist: /usr/local/apache/bugletdocs/Img/south-korea.gif
  ```

  As with access_log, error message is in several distinct parts

- Authentication error

```
[Tue Apr 11 22:13:21 2000] [error] [client 192.168.1.3] user rbowen
authentication failure for "/cgi-bin/hirecareers/company.cgi":
password mismatch
```

- CGI errors

```
Wed Jun 14 16:16:37 2000] [error] [client 192.168.1.3] Premature
end of script headers:
/usr/local/apache/cgi-bin/TestProg/announcement.cgi
Global symbol "$rv" requires explicit package name at
/usr/local/apache/cgi-bin/TestProg/announcement.cgi line 81.
Global symbol "%details" requires explicit package name at
/usr/local/apache/cgi-bin/TestProg/announcement.cgi line 84.
Global symbol "$Config" requires explicit package name at
/usr/local/apache/cgi-bin/TestProg/announcement.cgi line 133.
Execution of /usr/local/apache/cgi-bin/TestProg/announcement.cgi
aborted due to compilation errors.
```

16.5.1 Things to remember!

- The error log is your friend
- `tail -f /usr/local/apache/logs/error_log`
- Watch the log while you are working on stuff.

16.6 Logfile reporting

16.6.1 What your log file tells you

- Address of remote machine
- Time of visit
- Resource requested
- What's broken

16.6.2 What your log file does not tell you

- Who is visiting
  - Proxies
  - Caches

- Precisely how many visitors
- Name, address, credit card number
- If you want to know something, you have to ask

16.6.3 Log file parsing

- Webalizer (www.mrunix.net/webalizer)
- Analog
- WebTrends
- WWWStat
- Wusage
- Apache::ParseLog Perl module

16.7 Logging to a process

```
CustomLog |/usr/bin/apachelog common
```

Logs to the program /usr/bin/apachelog, rather than to a file, where apachelog looks like ...

```
#!/usr/bin/perl

while (my $log = <STDIN>) {
    DoSomethingUseful($log);
}
```

- Buffering
- Performance
- `http://modules.apache.org/` for modules that already do this

16.8 Logging to syslog

```
CustomLog syslog combined
CustomLog syslog:local1 common
```

16.9 Rotating log files

16.9.1 Logfile::Rotate

Logfile::Rotate is a Perl module. It is on the CD. Or you can get it from
`http://www.cpan.org/modules/by-module/Logfile/`

Unpack it somewhere, run the following:

```
perl Makefile.PL
make
make test
make install
```

Or, to install the module from the CPAN shell:

```
perl -MCPAN -e shell
cpan> install Logfile::Rotate
cpan> quit
```

Then put the following code into a file called `rotatelogs.pl`

```
#!/usr/bin/perl

use Logfile::Rotate;

$logfile = new Logfile::Rotate(
    File => '/usr/local/apache/logs/access_log',
    Count => 5,
    Gzip => '/bin/gzip',
    Post => sub {
      `/usr/local/apache/bin/apachectl restart`;
    }
);

$logfile->rotate();
```

Run the file a few times and see what happens to your log files. You'll want to run this program every month (or week, or whatever) via cron or other scheduler.

16.9.2 rotatelogs

```
CustomLog "|/usr/local/apache/bin/rotatelogs /some/where 86400" common
```

Implemented as a piped log file, which rotates your logs automatically every day.

The `/some/where` is where you want it to put the old log files each day. Specifically, it will create files called `/some/shere.XXXXXXX`, where `XXXXXXX` is the time in Unix time - that is, the number of seconds since Jan 1, 1970. This number can be converted back into human-readable time by the command:

```
perl -le 'print scalar localtime($time);'
```

where $time is the number appended to the end of the file name.

The 86400 is the number of seconds in a day, and means that the log rotation will happen once a day. You may want to set this at 300 seconds (ever 5 minutes) to see what this does.

NEED TO ADD 2.0 STUFF HERE

16.9.3 logresolve

```
cp ../logs/access_log ./
./logresolve -s stats < access_log > resolved.log
```

Resolves IP addresses, generates simple statistics.

16.10 Logging for multiple virtual hosts

- Each vhost should have its own log file
- Alternately, if you have only one log file, make sure you use the extended (combined) log file format which contains the vhost name:

  ```
  "%v %h %l %u %t \"%r\" %>s %b"
  ```

  Note that this is the canonical name for the vhost, not necessarily the name with which the host was accessed. ie, could be www.apacheadmin.com rather than apacheadmin.com even if that's what was actually used.

Section 17

Authentication, Authorization, Access Control

17.1 Definitions

- Authentication
- Authorization
- Access Control

17.2 Basic Authentication

- Provided by mod_auth
- 401 Authentication Required
- Browser supplies credentials if it has them
- Otherwise provides username/password dialog for user
- Credentials passed with every request
- Auth name/realm - used by the browser to cache login
- Password passed plaintext, with every request

17.3 Configuration

- Create a password file
  - htpasswd -c filename username
  - htpasswd filename username

- Set configuration to use this file
  - AuthType Basic
  - AuthName
  - AuthUserFile
  - Require user username
  - Require valid user
  - Use a different password than for your network login
- Optionally, create a group
  - group: user1 user2 user3
  - 8K limit
  - AuthGroupFile
  - Require group groupname

```
<Directory /usr/local/apache/htdocs/private>
  AuthType Basic
  AuthName "Top Sekrit"
  AuthUserFile /usr/local/apache/passwd/passwords
  Require user rbowen sungo
</Directory>
```

Or

```
<Directory /usr/local/apache/htdocs/private>
  AuthType Basic
  AuthName "Top Sekrit"
  AuthUserFile /usr/local/apache/passwd/passwords
  AuthGroupFile /usr/local/apache/passwords/groups
  Require group sekrit
</Directory>
```

`/usr/local/apache/passwords/groups` looks like:

```
    sekrit: rbowen sungo dpitts
```

17.4 FAQ

- How do I log out?
- How do I change what the password box looks like?
- How do I make my login persist across browser sessions?
- Why does it sometimes ask for my password twice?

17.5 Basic Auth Caveats

- Basic auth is not secure
- Username/password passed in the clear
- Content passed in the clear
- Cosmetic security only

17.6 Digest Auth

- Same as basic, except ...
- Username, password, MD5 hashed, and passed.
- Password not stored anywhere in the clear
- Content still passed in the clear
- Not supported by all browsers

Instead of htpasswd ...

```
htdigest -c /usr/local/apache/password/digest realm username
```

17.7 Configuration for Digest auth

```
AuthType Digest
AuthName "Private Area"
AuthDigestFile /usr/local/apache/passwords/digest
Require user drbacchus dorfl
```

Group file is identical to that used with Basic, if you want one. Use `AuthDigestGroupFile` with the same format.

17.8 Authentication against other things

- mod_auth_db
- mod_auth_mysql
- mod_auth_ldap

- mod_auth_nds (Netware Directory Services)
- mod_auth_smb (SMB - NT domain authentication)

17.8.1 mod_auth_db

- Creating a password file

```
dbmmanage passwords.dat adduser montressor
dbmmanage groups.dat add rbowen one,two,three
```

`dbmmanage --help` for full details, or `man dbmmanage`

This is still Basic authentication, with all the concerns pertaining thereto. It's just using a different file for its information.

```
AuthName "Members Only"
AuthType Basic
AuthDBUserFile /usr/local/apache/passwd/passwords.dat
AuthDBGroupFile /usr/local/apache/passwd/groups.dat
require group three
```

17.8.2 mod_auth_mysql

- User and password information in mysql
- Manage this information with whatever tools you're already using for database management.

17.9 Access Control

```
allow from address
deny from address
allow from 192.168
deny from dev.apacheadmin.com
deny from wanadoo.fr
```

The addresses specified can be a host name (partial, or complete) or an IP address (partial or complete).

To be even more specific, you need to use the Order directive:

```
Order deny,allow
Deny from all
Allow from apacheadmin.com
```

This can appear in a `<Directory` section, or in a `.htaccess` file.

These are applied as a series of filters. Everyone is excluded, then `apacheadmin.com` is let in. The other way around, it would be ineffectual.

Alternately,

```
Order allow,deny
Allow from all
Deny from wanadoo.fr
```

17.9.1 Satisfy

Use the satisfy directive when any one of a set of restrictions may be met. For example, if you want people inside your company to get into an area without being asked for a password, but people outside the company to be asked for a password, you could do the following:

```
Require group customers
Allow from internal.subnet.com
Satisfy any
```

See also `satisfy all` for another spin on this.

See also `mod_perl` access control handlers.

Section 18

Spiders

18.1 Introduction

Spiders, also known as robots, or automated user agents, or a variety of other things, are any software which automatically fetches content from the web. This may be done for a variety of different purposes.

- Indexing
- Searching
- Offline browsing
- Testing
- Link checking
- Performance testing (like ab)

18.2 Potential problems

- High server load
- Black holes
- DOS

18.3 Spiders in the logs

- altavista.com
- yahoo.com
- google.com

- etc
- Also, names like 'emailsiphon'

18.4 Excluding spiders from your site

There are a number of ways to exclude robots from your site.

18.4.1 robots.txt

Place a file called `robots.txt` in your `DocumentRoot` directory.

```
User-agent: *
Disallow: /cgi-bin/
Disallow: /datafiles/
```

or

```
User-agent: Scooter
Disallow: /dont-index/
```

18.4.2 ROBOTS metatag

```
<META NAME="ROBOTS" CONTENT="INDEX,NOFOLLOW">
```

- INDEX
- NOINDEX
- FOLLOW
- NOFOLLOW

18.4.3 Yell at the operator

Look up the IP address that it is coming from, and email the admin at that location.

18.4.4 Block by address

```
Order allow,deny
Allow from all
Deny from unfriendly.spider.com
```

18.4.5 Blocking with Deny from Env

```
SetEnvIf User-Agent EmailSiphon Spammers
Order Allow,Deny
Allow from all
Deny from env=Spammers
```

18.5 Writing your own spider

- Don't
- Get one from somewhere else
- Writing a spider is easy
- Writing a *good* spider is hard
- See sourcecode in the book

Section 19

Security

http://httpd.apache.org/docs/misc/security-tips.html

19.1 Overview

Apache has a significantly lower incidence of security problems, and significantly higher speed of resolving those problems, than that other web server. However, the last several releases have been security bug fix releases. And, although we try to ship apache "secure by default", there are a number of things that you can do to improve the situation, as well as a lot of things that you can do to make it worse.

Here's the simplistic list of what you should do.

- Keep file permissions as restrictive as possible
- Disable unused ports
- Remove unnecessary user accounts
- Don't use telnet
- Limit modules (Don't have modules installed that you are not using)
- Avoid FrontPage like the plague
- Avoid SSI where not necessary
- Don't use the system password file for authentication
- Don't put your password file in a document directory
- Develop on a staging server
- Keep up with OS and Apache security patches.
- Restrict CGI
- Use suexec for CGI

Ok, now, in more detail ...

19.2 File permissions

The overriding goal in your file permissions, as with any security precaution, is to be as paranoid as possible. Set file permissions at the most restrictive possible level which still allows people to get their work done. Security is inconvenient. Get over it. Getting hacked is more inconvenient. Train yourself to think in terms of what damage a malicious person could cause if they had the urge to do so.

Remember that the server runs as the `nobody` user (or whatever you have `User` set to) in the `nobody` group (or whatever you have `Group` set to) and that any web-based attacks will be run as that userID. So no files should be owned by, or writeable by, that user and/or group. (Note that I'll contradict this in the DAV chapter (22), but will provide a way around that.) There are a few small exeptions to this, and even there, you should be careful.

19.2.1 Content directories

The content directories (`cgi-bin`, `htdocs`, and `icons`) have the unfortunate requirement that the content in them will need to be modified from time to time. Thus, you'll have to give someone write permission to those directories. And, at the same time, you'll need to give the Apache user read permission to those directories.

Clearly identify the group of people that should have write permission to this content. Put those users in a user group, and give that group write access to that directory.

The `icons` directory is a little different, since the content in there never actually changes (on most servers) and so can safely remain unwriteable. The directory can be 755, and the content in it be 644, all owned by `root`.

19.2.2 Library

Library directories (`libexec` and `include`) contain files that are needed by Apache at startup, and are not required by anyone else at any other time. In particular, `libexec` contains the modules which you have built as shared objects, and `include` contains files that may be needed when building modules using `apxs`.

Solely for the purposes of Apache, these directories may be made mode 700, since the only time they are needed is during startup, when Apache is root, and has not yet relinquished its priveleges. And you don't want anyone to have an opportunity to meddle with these files. Some folks say that this is being overly paranoid.

19.2.3 bin

The `bin` directory is a little bit of a mixed bag, in that it contains files that you want everyone to have access to (or you might) and several that you don't. This somewhat depends on how paranoid you are.

The directory itself must *NEVER* be writeable.

```
drwxr-xr-x    2 root     root         4096 Mar 30 20:27 .
drwxr-xr-x   12 root     root         4096 Mar 30 20:27 ..
-rwxr-xr-x    1 root     root        43452 Mar 30 20:27 ab
-rwxr-xr-x    1 root     root         7107 Mar 30 20:27 apachectl
-rwxr-xr-x    1 root     root        26591 Mar 30 20:27 apxs
-rwxr-xr-x    1 root     root         3636 Mar 30 20:27 checkgid
-rwxr-xr-x    1 root     root        10937 Mar 30 20:27 dbmmanage
-rwxr-xr-x    1 root     root        12788 Mar 30 20:27 htdigest
-rwxr-xr-x    1 root     root        35464 Mar 30 20:27 htpasswd
-rwxr-xr-x    1 root     root       489980 Mar 30 20:27 httpd
-rwxr-xr-x    1 root     root         7288 Mar 30 20:27 logresolve
-rwxr-xr-x    1 root     root         5944 Mar 30 20:27 rotatelogs
-rws--x--x    1 root     root        10476 Mar 30 20:27 suexec
```

Users (ie, not root) will routinely have need of the `htpasswd` and `htdigest` utilities, and possibly also the `dbmmanage` utility, to create password files for use with authentication.

Users may wish to run `httpd -v` or `httpd -l`, for example, to get additional information about the apache server. They should probably be permitted to do this, so that they don't have to contact you every time they have a question.

Users probably do not need to have access to `apachectl`, `ab`, and `logresolve`. These files can safely be made mode 700.

You should not alter the permissions on `suexec` or `rotatelogs`, as they will be run by the unprivileged Apache processes.

`checkgid` is only used, if ever, at install.

19.2.4 logs

The `logs` directory must *never* be writeable by any user other than `root`. If another user can write to the `logs` directory, they can compromise Apache and gain `root`-level access to your server.

The log files themselves are created by Apache, and you should not modify their permissions.

19.2.5 proxy

If you are running `mod_proxy`, then you may have a `proxy` directory created on server install. This directory will be owned by the `nobody` user, or whatever user Apache is configured to run as, because Apache will need to write content to the proxy cache while it is running. This is one of the very few exceptions to the rule that nothing should be owned by the Apache user. And some people will say that even this poses a security risk, as a clever CGI program, for example, could modify content in the cache, and cause a user to send sensitive data to one place, while believing it was going somewhere else.

19.2.6 public_html

If users have web content in their home directory, this poses a special set of problems. In addition to having to set the permissions correctly, you also have to persuade the users, and the system administrators, that it is safe to do it that way.

There are two requirements that have to be satisfied. First, Apache needs to be able to read the files in the directory. Second, Apache needs to be able to traverse the directory tree to get to the files.

The first requirement means that files will typically need to be mode 644 or greater. The second means that the directories themselves need to be mode 755 or greater. Also, parent directories will need to be 711 or greater. For example, if user content is in `/home/eddie/public_html`, the the directory `/home/eddie` will need to be 711.

Usually home directories are created with a mode of 700, and users will tend to gripe at having to loosen these. Although there are some strange ways around this, the very best thing to tell them is:

- Making the directory +x does not actually allow anyone to read files in there, just to cd through it. Since the directory is not +r, it doesn't really matter.
- If they are still concerned, remind them they they should not put confidential files in the root of their home directory anyway, but should instead put it in subdirectories which are themselves 700

Some folks like to make the `public_html` directory owned by the apache `Group`, then make the directory mode 750. This has the distinct disadvantage that new files, created by the user, will not be owned by that group, and neither will subdirectories. While you presumably could make the apache `Group` the users' primary group, this is probably not recommended.

19.3 Configuration

While we try to ship Apache "secure by default", there are always things that you might want to do to further tighten it down. And there are without question a number of things that you can do to make it worse, so need to know about so that you can avoid them.

19.3.1 ServerTokens

Web servers return a `Server:` header as part of the response to the client. This header usually tells the client something about the server, like what server software it is running.

By default, it returns fairly detailed information:

```
Server:  Apache/1.3.29 (Unix) PHP/4.3.4 mod_perl/1.29
```

You may not want to give out this much information, as it gives crackers that tiny extra edge, and saves them the work involved in figuring out what server you're running. This lets them launch slightly more targeted attacks.

Understand that this is cosmetic security. It does not fix your vulnerabilities, it just hides them a little bit.

Setting `ServerTokens` allows you to give out less information in the `Server:` headers.

Example:

`ServerTokens Min`

Full	Apache/1.3.29 (Unix) PHP/4.3.4 mod_perl/1.29
OS	Apache/2.0.41 (Unix)
Min[imal]	Apache/1.3.27
Minor (2.x only)	Apache/2.0
Major (2.x only)	Apache/2
Prod[uctOnly]	Apache

19.3.2 ServerTokens - hacking the source

If you get particularly paranoid, and think that by lying about what the server is you'll get some additional benefit, well, there's no direct way to do this with configuration directives. In fact, there's a whole FAQ about how this isn't a particularly good idea.
`http://httpd.apache.org/docs/misc/FAQ.html#serverheader`

But, I'll tell you how to do it anyway. Because that's they kind of guy I am.

In Apache 1.3, you need to modify `include/httpd.h` and recompile:

```
#define SERVER_BASEVENDOR   "Apache Group"
#define SERVER_BASEPRODUCT  "Apache"
#define SERVER_BASEREVISION "1.3.29"
```

In Apache 2.0, it's in `include/ap_release.h`

```
#define AP_SERVER_BASEVENDOR "Apache Software Foundation"
#define AP_SERVER_BASEPRODUCT "Apache"
#define AP_SERVER_MAJORVERSION "2"
#define AP_SERVER_MINORVERSION "0"
#define AP_SERVER_PATCHLEVEL "48"
```

Please understand that *most* of the attacks against web servers are conducted by automated scripts, which will run attacks regardless of what server you claim to be running, so this has very little actual benefit.

In fact, I tend to think that it has two very significant detrimental effects. First, it makes you feel more secure when you are not, and a false sense of security leads you to do stupid things. Second, it makes you actually forget what version you're running. `httpd -v` returns incorrect information, and you can't determine the version number. So, when a new version comes out, you forget to upgrade, and end up trading cosmetic security for the real security of a patched new version.

19.3.3 ServerSignature

Automatically-generated error documents - such as when you go to a bad URL, or get a server error - have server information in them which looks a lot like the `Server:` headers. However, the information here is controled by a different directive. `ServerSignature` can be set to `On`, which returns the server information, `Off`, which does not, or `Email` which additionally adds the email address supplied in the `ServerAdmin` directive, with a `mailto:` link.

On all versions prior to 2.0.44, the server information returned is the server version number and the `ServerName`. With version 2.0.44 and later, `ServerSignature` will return the details specified by the `ServerTokens` directive.

ServerTokens default settings

Although the default value of this directive is `Off`, the default configuration file sets it to `On`.

19.4 SSI

Server Side Includes, which will be discussed in an upcoming chapter, can be a significant security risk. When enabling SSI, remember that you are allowing someone to execute arbitrary commands on your server. If that doesn't give you pause, then you're entirely too trusting.

Don't enable SSI unless you need to. If you need to, then enable it for the smallest possible scope of directories. And, if possible, use `IncludesNoExec` rather than `Includes` as your `Options` setting.

The only saving grace here is that the arbitrary commands are executed by the Apache user, who, hopefully, doesn't own any content on your server, as specified in the section above on file permissions.

19.5 CGI

As with SSI, CGI allows people to execute arbitrary code on your server. You should, therefore, be very frightened. This is much more flexible than SSI, though, because people can write arbitrarily complex programs in any language, and execute them on your server.

You should see the `suexec` section later for details about running CGI programs as other users, but by default, CGI programs are run as the user and group specificed in the `User` and `Group` directives.

As with SSI, you should limit as much as possible the directories in which CGI execution is permitted. If possible, allow CGI execution in `ScriptAlias` directories only, as this will ease the burden of auditing code, since you won't have to go hunting for it.

Poorly written CGI programs are the easiest, fastest, and most common way to break into an Apache server, or to exploit the resources of an Apache server to do various things like send spam by proxy.

19.5.1 CGI exploit example - trusting form input

One of the most common CGI exploit categories involved accepting data from HTML forms and trusting it. Remember that the user cannot be trusted. Like your mother always told you, don't put that in your command, you don't know where it's been.

Consider the following scenario. You have a form on your web site, which allows users to send you feedback email. Within your CGI code is something like the following:

```
open MAIL, "|/usr/bin/sendmail -s $FORM'subject' $FORM'to'";
print MAIL $FORM'body';
print MAIL "\n\n.\n\n";
close MAIL;
```

Looks pretty straightforward. Data comes in from the form fields `subject`, `to`, and `body`, and this gets passed off to `sendmail`, which delivers the email. Email is composed to the address in `to` with a subject line of `subject` and a body of `body`, all specified in the form.

There are two problems here.

The first is that a clever (or not-so-clever, really) spammer can use this form to send as many email messages as they want, to whomever they want, by posting data to your CGI program. Since that email will come from your server, you will appear to be responsible for that email.

The second problem is a little more subtle, and much more dangerous. It has to do with the awful way that the code here is calling `sendmail`. Rather than using a module/library to send the email, it is calling the system executable, and passing arguments to it. This is universally a bad idea, because it allows a clever person to circumvent your command line and insert their own.

Consider, for example, if I enter into the form field `to` the value:

```
bob@foo.com ; rm -rf /
```

The first part of this is an email address. Great. But then I have the character `;`, which terminates the command and starts a new one. At a regular command line, the ; character allows you to type multiple commands in the same line. Since your code is calling the command line, the same rules apply. So putting this in the form field will cause the `sendmail` command to be abandoned, and my other command to be executed, recursing through the entire file system deleting any file that I happen to have access to.

Fortunately for the hackers, there are several rather popular (read: widely installed) CGI programs that do exactly that, and so all that they need to do is write scripts that attempt to put these sorts of arguments into web forms and see if they can do anything.

19.5.2 CGI exploit example - hidden form fields

Hidden form fields are not hidden. This is another example of cosmetic security.

* Hidden form fields are not hidden.

* Don't put important things in there, like passwords, SQL statements, usernames, etc.

* Don't use "hidden" form fields without aggressively validating that they contain good stuff.

=CGI exploit examples - Doing it yourself

* There are CGI libraries in your language of choice. Use them.

* You haven't thought of all the possible exploits. Neither have the library authors, but they've thought about it more than you have.

19.6 Default file system settings

```
<Directory />
   AllowOverride None
   Options FollowSymlinks

   Order deny,allow
   Deny from all
</Directory>
```

* Don't permit C¡.htaccess¿ files anywhere, unless explicity stated. This also provides performance benefits.

* Don't permit access, unless explicitly stated.

* Note that for any document directory you'll need to add:

```
 <Directory /path/to/directory>
   Order allow,deny
   Allow from all
 </Directory>
```

19.7 UserDir

```
UserDir disabled root
```

* Or, better:

```
UserDir disabled
UserDir enabled bob waldo sally
```

19.8 Modules

The overarching rule on modules is simply: Don't run modules you're not using. Modules that you're not actively using will not get maintained, and will not get patched when there are security problems

If `mod_asis` has a bug, and you don't know what `mod_asis` is, you're not likely to upgrade it.

On the last day, we'll go through the complete list of standard modules and what they do. Unfortunately, if you're running a third-party packaged distribution, it likely contains modules that we've not discussed.

19.9 suexec

- Execute CGI programs as a different user.
- Use `User` and `Group` directives in virtual hosts
- `username` urls automatically execute CGI as that user
- Tries to prevent you from executing unsafe things, or from executing stuff in an unsafe manner.
- Checks file path, ownership, and permissions, before permitting it to be executed.
- Refuses to run with uid and gid that are outside of a particular approved range

`http://apache-server.com/tutorials/LPsuexec.html`

```
suEXEC options:
 --enable-suexec        enable the suEXEC feature
 --suexec-caller=NAME   set the suEXEC username of the allowed caller [www]
 --suexec-docroot=DIR   set the suEXEC root directory [PREFIX/share/htdocs]
 --suexec-logfile=FILE  set the suEXEC logfile [PREFIX/var/log/suexec_log]
 --suexec-userdir=DIR   set the suEXEC user subdirectory [public_html]
 --suexec-uidmin=UID    set the suEXEC minimal allowed UID [100]
 --suexec-gidmin=GID    set the suEXEC minimal allowed GID [100]
 --suexec-safepath=PATH set the suEXEC safe PATH [/usr/local/bin:/usr/bin:/bin]
 --suexec-umask=UMASK   set the umask for the suEXEC'd script [server's umask]
```

(These are `./configure` command line options.)

19.10 mod_security

http://www.modsecurity.org/

mod_security is a third-party module which is intended to give a firewall-like front end to Apache, in order to stave off attacks before they make it to Apache. It lets you write pattern-based rules to intercept requests and block them.

To install mod_security, download and unpack the source, change into the directory for the particular version of Apache that you're running (apache1 or apache2), then type:

```
apxs -cia mod_security.c
```

You will need to provide the full path to apxs as it will probably not be in your path,

The result of running this command will be, hopefully, that you end up with the file /usr/local/apache/libexec/mod_security.so being created, and the following line added to your httpd.conf :

```
LoadModule security_module    libexec/mod_security.so
```

You'll then want to add one or more rule to your configuration file:

```
SecFilterEngine On
SecFilterScanPOST On
SecAuditLog /dev/null
SecFilterDefaultAction "deny,log,status:402"
SecFilter "delete[[:space:]]+from"
SecFilter "insert[[:space:]]+into"
SecFilter "select.+from"
SecFilter "\.\./"
SecFilterSelective "REQUEST_METHOD" "SEARCH"
```

19.11 mod_dosevasive

LINK http://www.nuclearelephant.com/projects/dosevasive/

Prevents a denial of service by one site requesting documents many times within a too-short period of time.

```
apxs -cia mod_dosevasive.c
```

Then ...

```
<IfModule mod_dosevasive.c>
    DOSHashTableSize    3097
    DOSPageCount        2
    DOSSiteCount        50
    DOSPageInterval     1
    DOSSiteInterval     1
    DOSBlockingPeriod   10
</IfModule>
```

Note that the blocking period is only 10 seconds. However, if the client persists in requesting pages, this period is extended with each request, causing this block to last for as long as the persist in the behavior.

Section 20

SSL

http://openssl.org/

http://modssl.org/

- public/private key cryptography
- SSL Certificates
- Signed certificates
  - Thawte
  - Verisign
  - Sign it yourself

20.1 Intro

- Runs on port 443
- Accessed via https://servername/

20.2 Installing SSL

- OpenSSL
- mod_ssl
- Apache 2.0 - ./configure –with-ssl

20.3 Certificates

- Generate a key pair

```
openssl genrsa -rand file1:file2:file3
-out www.domain.com.key 1024
```

Don't enter a pass phrase, or you will have to enter that passphrase every time you restart Apache

- Generate certificate signing request

```
openssl req -new -key www.domain.com.key -out www.domain.com.csr
```

You will be asked for a series of information, to which you need to provide answers. These answers will appear on the certificate itself, and verify its authenticity.

When it asks for your 'Common Name', this is actually the name of the web site - the fully qualified domain name which will be used to serve the site. So this should be, for example, 'www.rcbowen.com' or similar. Not your name. Not your company name.

At this point, you can either sign this yourself, or you can submit the certificate request to one of the CA (Certificate Authorities) for signing.

`http://digitalid.verisign.com/server/apacheNotice.htm`

`http://www.thawte.com/certs/server/request.html`

- Sign it yourself

```
openssl x509 -req -days 365 -in www.domain.com.csr -signkey
www.domain.com.key -out www.domain.com.cert
```

- Install the key - place it in the certs directory (should be either under the openssl directory, or somewhere under the Apache directory.

20.4 Configuration

```
<VirtualHost _default_:443>
ServerName www.domain.com
SSLEngine on
SSLCertificateFile /path/to/www.comain.com.cert
SSLCertificateKeyFile /path/to/www.domain.com.key
</VirtualHost>
```

See also the default SSL configuration that comes with openssl when you install it. It has a lot more stuff than this in it.

You can also use *:443 instead, but, if you do, you need to make sure that all the other vhosts that you are running use the same syntax.

Section 21

modules

21.1 Module list

These are the modules that come with Apache

21.1.1 Apache 1.3 modules:

(To get this moudle list, just type `./configure --help`

```
      [access=yes      actions=yes     alias=yes      ]
      [asis=yes        auth=yes        auth_anon=no   ]
      [auth_db=no      auth_dbm=no     auth_digest=no ]
      [autoindex=yes   cern_meta=no    cgi=yes        ]
      [digest=no       dir=yes         env=yes        ]
      [example=no      expires=no      headers=no     ]
      [imap=yes        include=yes     info=no        ]
      [log_agent=no    log_config=yes  log_referer=no ]
      [mime=yes        mime_magic=no   mmap_static=no ]
      [negotiation=yes proxy=no        rewrite=no     ]
      [setenvif=yes    so=no           speling=no     ]
      [status=yes      unique_id=no    userdir=yes    ]
      [usertrack=no    vhost_alias=no                 ]
```

21.1.2 Apache 2.0 modules:

```
[access         actions         alias      ]
[asis           auth            auth_anon  ]
[auth_dbm       auth_digest     autoindex  ]
[cache          cern_meta       cgi        ]
[cgid           charset_lite    dav        ]
[deflate        dir             env        ]
[example        expires         ext_filter ]
[file_cache     headers         imap       ]
[include        info            isapi      ]
[log_config     mime            mime_magic ]
[negotiation    proxy           rewrite    ]
[setenvif       so              speling    ]
[ssl            status          suexec     ]
[unique_id      userdir         usertrack  ]
[vhost_alias                               ]
```

21.1.3 What's new, and what's missing

The modules `mod_auth_db`, `mod_digest`, `mod_log_agent`, `mod_log_referer`, and `mod_mmap_static` go away in version 2.0.

New in version 2.0 are `mod_cache`, `mod_cgid`, `mod_charset_lite`, `mod_dav`, `mod_deflate`, `mod_ext_filter`, `mod_file_cache`, `mod_isapi`, and `mod_ssl`.

21.2 mod_access

```
Name            mod_access
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_access.html
```

mod_access provides access control based on client hostname, IP address, or other characteristics of the client request.

21.3 mod_actions

```
Name            mod_actions
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_actions.html
```

Provides for executing CGI scripts based on media type or request method.

21.4 mod_alias

```
Name            mod_alias
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_alias.html
```

Mapping different parts of the file system into the document tree, and URL redirection.

21.5 mod_asis

```
Name            mod_asis
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_asis.html
```

Sending files which contain their own HTTP headers.

```
Content-type: 32
Content-type: text/html
Content-language: en
Set-Cookie: name=value

And then the content goes here.
```

21.6 mod_auth

```
Name            mod_auth
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_auth.html
```

Basic HTTP authentication, using text files to contain user and group information.

21.7 mod_auth_anon

```
Name            mod_auth_anon
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_auth_anon.html
```

Anonymous user access to authenticated areas.

21.8 mod_auth_db

```
Name            mod_auth_db
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_auth_db.html
```

Basic HTTP authentication, using Berkeley DB files to contain user and group information.

This module is removed in Apache 2.0, and only `mod_auth_dbm` remains. There's very little chance that you'll ever have a need for both at the same time, or even have more than one dbm implementation install on the same machine.

21.9 mod_auth_dbm

```
Name            mod_auth_dbm
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_auth_dbm.html
```

Basic HTTP authentication, using DBM files to contain user and group information.

21.10 mod_auth_digest

```
Name            mod_auth_digest
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_auth_digest.html
```

MD5 digest authentication.

In Apache 2.0, no longer marked as experimental.

21.11 mod_autoindex

```
Name            mod_autoindex
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_autoindex.html
```

Automatic directory listings

21.12 mod_cern_meta

```
Name            mod_cern_meta
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_cern_meta.html
```

Support for HTTP header metafiles

21.13 mod_cgi

```
Name            mod_cgi
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_cgi.html
```

Support for execution of CGI programs

21.14 mod_digest

```
Name            mod_digest
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_digest.html
```

Provides MD5 authentication, but has been replaced by mod_auth_digest

21.15 mod_dir

```
Name            mod_dir
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_dir.html
```

Provides for mapping URLs with a trailing slash to an index file, typically called index.html

21.16 mod_env

```
Name            mod_env
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_env.html
```

Handles the passing of environment variables to CGI programs

21.17 mod_example

```
Name            mod_example
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_example.html
```

An example module, demonstrating the Apache API, and the technique of writing Apache modules

21.18 mod_expires

```
Name            mod_expires
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_expires.html
```

Gives the ability to apply Expires: headers to resources.

21.19 mod_headers

```
Name            mod_headers
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_headers.html
```

Add arbitrary HTTP headers to resources

21.20 mod_imap

```
Name            mod_imap
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_imap.html
```

Handles server-side image map files

21.21 mod_include

```
Name            mod_include
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_include.html
```

Server-parsed documents (Server-side includes)

21.22 mod_info

```
Name            mod_info
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_info.html
```

Provides the server-info handler, for providing information about server configuration.

21.23 mod_log_agent

```
Name            mod_log_agent
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_log_agent.html
```

Logging of user agent (browser). This module is superseded by the LogFormat directive in mod_log_config

21.24 mod_log_config

```
Name            mod_log_config
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_log_config.html
```

Allows you to build custom log files. See Chapter 24 for detailed treatment of this module and the functionality it provides.

21.25 mod_log_referer

```
Name            mod_log_referer
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_log_referer.html
```

Provides logging of document references. That is, logs the places that have links to your content. This module is superseded by the LogFornat directive in mod_log_config

21.26 mod_mime

```
Name            mod_mime
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_mime.html
```

Determining document types by file extensions. See Chapter 8.

21.27 mod_mime_magic

```
Name            mod_mime_magic
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_mime_magic.html
```

Determining document types using "magic numbers" - that is, by looking at the contents of the file, and, based on the frequency of occurance of certain patterns or characters, determining what the file type probably is.

21.28 mod_mmap_static

```
Name            mod_mmap_static
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_mmap_static.html
```

Mapping files into memory to improve performance of serving static document. This module is marked as experimental.

21.29 mod_negotiation

```
Name            mod_negotiation
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_negotiation.html
```

Content negotiation. See Chapter 10

21.30 mod_proxy

```
Name            mod_proxy
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_proxy.html
```

A caching proxy server.

21.31 mod_rewrite

```
Name            mod_rewrite
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_rewrite.html and
                http://httpd.apache.org/docs/misc/rewriteguide.html
```

Provides the ability to rewrite incoming URL requests in order to do all of the things that you wish mod_alias did.

21.32 mod_setenvif

```
Name            mod_setenvif
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_setenvif.html
```

Set environment variables based on client information. Can be used for things such as access control:

```
SetEnvIf User-Agent ^KnockKnock/2.0 let_me_in
<Directory /docroot>
Order Deny,Allow
    Deny from all
    Allow from env=let_me_in
</Directory>
```

21.33 mod_so

```
Name            mod_so
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_so.html
```

Dynamically load modules as shared objects at runtime.

21.34 mod_speling

```
Name            mod_speling
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_speling.html
```

Automatically correct minor typos in URLs, such as character transposing, wrong capitalization, or other small errors.

21.35 mod_status

```
Name            mod_status
On by default?  Yes
Docs            http://httpd.apache.org/docs/mod/mod_status.html
```

Display server status in a convenient HTML report.

21.36 mod_unique_id

```
Name            mod_unique_id
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_unique_id.html
```

Generate unique identifiers for each incoming request for tracking purposes.

21.37 mod_usertrack

```
Name            mod_usertrack
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_usertrack.html
```

User tracking using cookies. Make sure you know what you are doing before enabling this. It sends a lot of cookies.

21.38 mod_vhost_alias

```
Name            mod_vhost_alias
On by default?  No
Docs            http://httpd.apache.org/docs/mod/mod_vhost_alias.html
```

Dynamically configure a large number of virtual hosts without changing your server configuration file.

Section 22

WebDAV

WebDAV, Distributed Authoring and Versioning, is, to grossly oversimplify, file transfer over HTTP. The advantages over FTP are numerous, but can be simplified to this:

- Configuration via your HTTP server
- Uses whatever HTTP authentication you already have in place
- Security without creating user accounts

There are other advantages, but those are the ones that I primarily care about.

22.1 mod_dav on 1.3

The `mod_dav` module is available for Apache 1.3, and can be obtained from `http://webdav.org/mod_dav/`

If you want to install `mod_dav` on Apache 1.3, it is recommended that you use `Apache Toolbox` or similar in order to get it installed correctly.

22.2 mod_dav on 2.0

To enable `mod_dav` on Apache 2.0, run `configure` with these additional arguments:

```
./configure --enable-dav --enable-dav-fs
```

The `dav-fs` argument tells dav to use the file system as the document repository. You can also use a database, or a versioning system such as Subversion for your document repository, but for our purposes, using the file system seems to make a lot more sense.

To enable `DAV` in a particular directory, you need only do the follwing:

```
<Directory /usr/local/apache/htdocs>
    DAV On
</Directory>
```

Note that the files in that directory will need to be owned and/or writeable by the web user.

22.3 DAV clients

There are a number of DAV-enabled clients. Among these are:

- Windows XP (The entire OS)
- DreamWeaver
- Cadaver

We'll be using `Cadaver` for our demonstration.

Obtain `cadaver` from `http://webdav.org/cadaver/` and install it. Once installed, you should be able to access your server via:

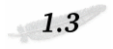
```
% cadaver localhost:90/
dav:/> ls
Listing collection `/': succeeded.
Coll:  images                                0  Apr 11  2002
      *header.html                        2474  Aug 27 21:50
      *index.html                         7914  Dec  4 19:24
dav:/> edit index.html

... etc
```

With this in place, you can enable HTTP authentication on the particular directory, and use this as an FTP replacement. There are clients available for most operating systems - see `http://webdav.org/projects/` for a listing of some of the clients available.

Section 23

mod_proxy

mod_proxy implements a simple caching proxy server on top of Apache. We'll consider caching, and proxying, and then talk more generally about how this can be used.



There's a significant difference between mod_proxy on Apache 1.3 and 2.0. On 1.3, proxying and caching are both provided by mod_proxy, but in 2.0 these are separate modules. Additionally, the <Proxy> directive in 2.0 replaces the rather less intuitive <Directory proxy:*> syntax in 1.3. If you are going to do proxying on 1.3, you'll need to consult the documentation for other nuances.

23.1 Caching

```
ProxyRequests On
CacheRoot /usr/local/apache/proxy
```

Make sure that the specified directory is owned/writable by nobody.

In your browser, you will need to configure your proxy server to point to the new caching proxy server. When you make requests, they will get logged in the Apache access log, and retrieved files will be cached in the specified directory.

S

ecurity concerns regarding cache poisoning. Talk about the naming convention for files, and how files could be subverted.

23.2 Proxying

The proxy part lets you hide other servers behind your server. This lets you do load balancing, or simply serve content from another site for some reason.

```
ProxyPass         /mirror/foo/ http://foo.com/
ProxyPassReverse  /mirror/foo/ http://foo.com/
```

Requests to any URL starting with `/mirror/foo/` will be proxied through to requests on `http://foo.com/`. The `ProxyPassReverse` intercepts redirects coming from `foo.com` and adjusts them to contain the `/mirror/foo/` URL before passing them on to the client.

Note that absolute URLs in HTML returned from `foo.com` are likely to break this entire arrangement.

23.3 Rewrite and proxying

23.4 mod_proxy_html

23.5 General comments

These two sets of functionality are, of course, closely related, but they are divisible. In 2.0, they have been split into two separate modules in order to modularize things a little better.

One frequent use of this module is to split the load of a particular server onto several other servers, and have a front-end server farm out the content to those back end servers. Similarly, it could be used to expose machines that would otherwise be unreachable from the outside world, by making them available via a special URL on the proxy server.

Index

Andreessen, Marc, 2
Apache Software Foundation, The, 3
apxs, 124
Architecture, 13
As We May Think, 1
ASF, 3

Behlendorf, Brian, 2
Berners-Lee, Tim, 1
Boot
    starting at, 19
Bush, Vannevar, 1

CERN, 1

Fielding, Roy, 2

History, 1

MaxSpareServers, 13
McCool, Rob, 2
MinSpareServers, 13
mod_proxy, 145
mod_security, 124
Mosaic, 2
MPM, 14
    perchild, 16
    prefork, 14
    win32, 16
    worker, 14, 15

NCSA, 2
Netscape, 2

Perchild MPM, 16
Prefork, 13
Prefork MPM, 14
Processes, 13
Proxy, 145

Shambala, 3
Starting at boot, 19

UIUC, 2

Win32 MPM, 16
Worker MPM, 14, 15
World Wide Web, 1
WWW, 1