

APACHE



Apache

- **Aim of this section**

- Introduce you to workings of the most common webservers
- Give you a little experience in role of webmaster
 - Configuring apache
 - Installing server
 - Setting access controls on files
 - etc

Not that much of an experience in 2008 (unless you install it on your own PC); in laboratory we will be using a pre-configured Apache based on the Zend simplified Apache/PHP/Oracle install package.

Apache

- Most common httpd daemon server
 - Just over 60% of market
 - Used as basis of some other products (e.g. IBM Websphere application servers)
- Versions exist for Unix, Linux, and Windows
 - Windows version does differ in implementation detail

Apache

- www.apache.org
- apachetoday.com
 - URL still valid, but now swallowed up by “ServerWatch”
 - Still has Apache tutorials, e.g.
 - <http://www.serverwatch.com/tutorials/article.php/3588671> “Apache Session Management Within Dynamic Sites”

Apache versions

- 1.3.xx (currently 1.3.39)
 - The classic Apache.
 - Still widely used
- 2.x
 - A modernized threaded server structure
 - 2.2.6

January Results

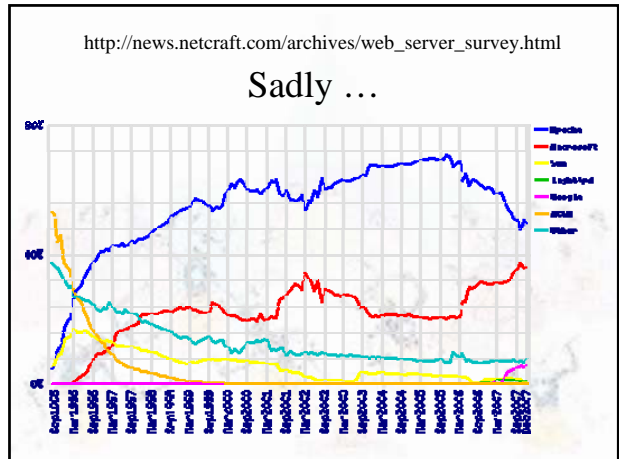
Server	No. of Sites	Market Share	No. of Sites With .com domains
Apache	50,502,840	67.38%	25,522,907
Microsoft-IIS	15,208,775	20.29%	8,371,612
Unknown	4,186,054	5.59%	3,115,393
Sun-ONE-Web-Server	1,551,930	2.07%	1,061,053
Zeus	561,524	0.75%	208,496
WebLogic	13,601	0.02%	7,846
...
Oracle	8,280	0.01%	2,637
...
NCSA HTTPd	1,717	0.00%	315

Jan 2006 counts of 79million sites surveyed by ServerWatch

October Results

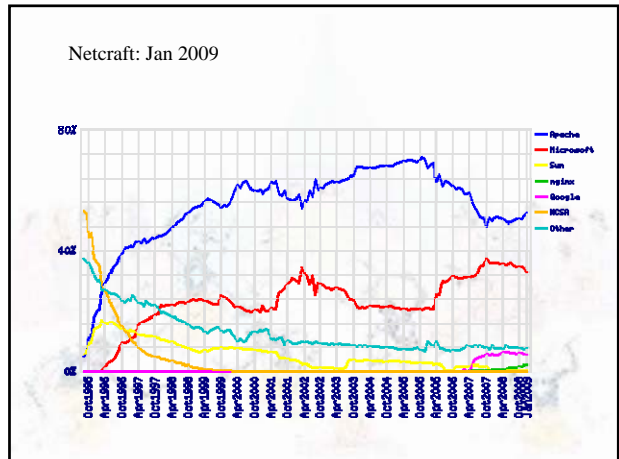
Server	No. of Sites	Market Share	No. of Sites With .com domains
Apache	60,166,642	61.44%	31,555,632
Microsoft-IIS	30,701,294	31.35%	19,116,192
Unknown	2,444,413	2.50%	1,819,993
Zeus	522,311	0.53%	167,252
Netscape-Enterprise	292,543	0.30%	134,883
tigershark	224,163	0.23%	169,569
RapidSite	192,644	0.20%	104,380
thttpd	171,541	0.18%	6,869
Lotus Domino	92,706	0.09%	30,231
Zope	46,276	0.05%	11,127

Oct 2006 counts of **97,932,447** sites surveyed by ServerWatch



Numbers

Developer	Nov-07	Percent	Dec-07	Percent	Change
Apache	76,028,287	50.76%	76,945,640	49.57%	-1.19
Microsoft	53,679,916	35.84%	55,509,223	35.76%	-0.08
Google	7,910,879	5.28%	8,558,256	5.51%	0.23
lighttpd	1,505,122	1.00%	1,521,250	0.98%	-0.02
Sun	619,262	0.41%	588,997	0.38%	-0.03



Developer	Dec-08	Share	Jan-09	Share	Change
Apache	35,125,901	48.38%	36,062,915	50.33%	1.95%
Microsoft	24,111,817	33.21%	22,645,049	31.61%	-1.60%
Google	7,874,043	10.84%	7,302,578	10.19%	-0.65%
nginx	1,978,479	2.72%	2,040,413	2.85%	0.12%
YTS	842,539	1.16%	864,975	1.21%	0.05%

A different view

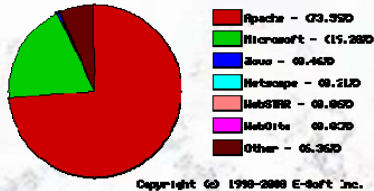
Total servers: **25,604,643**

Server ¹	December Count	December %	November Count	November %	Change
Apache	18,842,376	73.59%	18,935,171	73.64%	-0.05%
Microsoft	4,937,568	19.28%	4,902,108	19.06%	0.22%
Zeus	116,557	0.46%	118,311	0.46%	0.00%
Netscape	54,079	0.21%	55,660	0.22%	-0.01%
WebSTAR	16,536	0.06%	17,242	0.07%	-0.01%
WebSite	8,243	0.03%	8,424	0.03%	0.00%
Other	1,629,284	6.36%	1,677,762	6.52%	-0.16%

www.securityspace.com Securityspace excludes minor sites

Security space's numbers

Market Share for December 2007 - Across ALL Domains



Apache – things you should know ...

Aspects of Apache (1)

- Processes
 - How does it work
 - Administrator controls resource usage
- Modules
 - These determine functionality of particular Apache server
 - Administrator chooses the necessary modules
- Access control
 - Administrator can limit web-clients' access to different resources accessible via a server

Aspects of Apache (2)

- Logs
 - Useful data for identifying problems and analyzing usage patterns
- Virtual hosts
 - An apache server can pretend to be several servers belonging to different companies
- Mechanisms for dynamic page generation
 - CGI, SSI, ...
- How to configure

APACHE processes

Apache: Processes (Unix/Linux)

- Apache httpd CHIEF process monitoring port 80
- Child (tribesmen) processes:
 - Pool of pre-forked child processes each running Apache
 - Chief process supposed to keep track of tribe
 - Defined minimum (and maximum) numbers for "idle" children waiting for requests
 - Defined maximum number of child processes

Apache: Processes (Unix/Linux)

- Chief process
 - Use “select()” system call to detect clients attempting to establish TCP/IP connection
 - Check tribesmen:
 - If one is idle, allow that child to “accept()” connection on port 80 – child then continues handling new client
 - If no child idle, create some more children (subject to maximum number of child processes)
 - *If close to limit, print warnings in log files!*
 - At regular intervals, check number of idle children, terminate some if the number exceeds specified limit

Apache: Processes (Unix/Linux)

- Chief process/child processes
 - Share port 80
 - Monitored in master process
 - Used for accept() in child
 - Semaphores/mutexes used to control access to particular code segments
 - Chief locks code,
 - Child (finished with last client) comes to collect more work
 - Child blocks at lock

Apache: Processes (Unix/Linux)

- Chief process/child processes
 - “Scoreboard”
 - Usually a shared memory segment
 - On OSs that don’t support shared data segments for processes, Scoreboard is a file that gets read and written by all members of the tribe (subject to file locks)
 - Holds additional data for communicating status information amongst processes

Apache: Processes (Unix/Linux)

- **Child processes 1.3**
 - Each child acts like a “*serial server*”
 - Deal with one client connection at a time
 - When finished with client, close TCP/IP connection and seek more work (block at one of mutexes controlled by chief)
 - Each child has maximum number of connections that it is permitted to handle (handling one at a time); once it has dealt with this number it must terminate
 - (this is a protective device – possible resource leaks ...)

Apache: Processes (Unix/Linux)

- **Apache 2.**
 - Each child acts like a “*threaded server*”
 - Deal with a number of client connections at a time
- Can configure Apache 2 to run just the same as Apache 1 (single threaded children), usually have multithreaded children, need fewer processes

Apache: Processes (Unix/Linux)

- Child processes
 - Client connection typically has sequence of requests
 - Usually Apache configured to “keep alive” connections.
 - Client requests page (GET ../etc/hello.html HTTP1.1)
 - Connect to server
 - Selected by chief
 - Given to child process
 - Accepted by child
 - Handled by child
 - Page returned
 - Connection remains open
 - Client requests embedded gif, stylesheet, ...
 - Connection to Apache child still open
 - GET ../etc/hello.gif HTTP1.1 request

Server Breakdown Across All Domains of the 10 most popular servers. Server order is based on global market share.			
Server Name	Number Found	Percentage	
Apache	3,588,683	17.28%	
Microsoft-IIS/6.0	2,333,073	11.23%	
Microsoft-IIS/5.0	1,788,746	8.61%	
Apache/1.3.37	1,777,481	8.56%	
Apache/1.3.33	1,700,985	8.19%	
Apache/1.3.34	974,464	4.69%	
Apache/2.0.54	702,781	3.38%	
Apache/1.3.27	655,836	3.16%	
Apache/1.3.36	578,305	2.78%	
Apache/2.0.52	512,506	2.47%	

APACHE

Administrator's

roles:

performance

- ### Performance
- Administrator must set limits on processes etc to achieve acceptable performance from host computer
 - MaxClients
 - How many Apaches?
 - Memory limits
 - Also, swapping and caching (may do better with fewer processes if these can keep their code in memory, and preferably in cache)
 - MinSpareServers, MaxSpareServers
 - An idle server allows quick response to a client, but uses up a process slot and some memory

- ### Performance
- Administrator must select limits for actions of child processes
 - Keepalive, MaxKeepAliveRequests, KeepAliveTimeout
 - Do we even support "persistent connections"?
 - How many files can client request?
 - How long are we prepared to wait after last request from client before closing the connection anyway? (15 seconds)
 - Timeout
 - (another control for timing out clients – mainly relating to: opened connection no activity, multipart transfers that are not completed or acknowledged)
 - MaxRequestsPerChild

APACHE

modularity

- ### Apache modules
- Apache server:
 - Core component
 - Basically something for serving simple HTML files
 - Modules
 - Optional features and extension
 - Does this server permit "server side includes"?
 - Does this server support server side image maps?
 - Does this server apply security controls to some pages, and if so how?
 - Does this server ...

Modules

- An Apache installation is built with a set of included modules chosen by web administrator.
- “Out of the box” configuration defined in a configuration file
- Initial configuration program allows administrator to modify these defaults.
- Config program produces **makefile**.

“Standard” modules

- **Core functionality**
 - mod_cgi (*support CGI*)
 - mod_env (*environment variables passed to CGI scripts*),
 - mod_log_config (*logging*),
 - mod_mime (*determine Mime type from file extension*),
 - mod_negotiation (*deals with things like “prefer jpeg, will accept gif”*),

“Standard” modules

- **Information**
 - mod_status (*display of server status on web*)
 - mod_dir (*display of directory data*),
 - mod_autoindex (*fancier directory listings*),

“Standard” modules

- **Access control,**
 - mod_access (*control availability of files/subdirectories of a directory; access is determined by IP address of requestor.*)
 - mod_auth (*requires some form of user supplied authentication data before permitting access to pages*),
 - mod_userdir (*pages in user directories etc*),
 - mod_alias (*maps URL directory names onto actual names*)

“Standard” modules

- **Extra processing**
 - mod_imap (*image maps.*)
 - mod_include (*server side includes*),
 - mod_actions (*cgi scripts associated with file types*),

“Optional” modules

- **Apache features that you may/may not want**
 - mod_proxy (*configure server as proxy server*)
 - mod_rewrite (*change URLs as return pages, it is used as part of one way of recording client state data*),
 - mod_speling (*tries “near miss” matches of URL pathnames/filenames against available resources*),

Add-on modules

- **Finally, there are extensions to the server**
 - mod_perl (*perl interpreter as part of Apache*),
 - mod_php (*php interpreter as part of Apache*),
 - SSL extensions,
 - Others such as a mod that allows a Unix/Linux Apache server to deal with Microsoft Active Server Pages (*well, some ASP pages*)

APACHE

Administrator's

roles:

modularity

Modularity

- Administrator must select the modules that are to be incorporated
 - Security issues ...
 - Some options, eg Server Side Includes, are known to have security risks if not well managed
 - Functionality ...
 - Does your installation need php?
 - Does your installation need perl integrated into server instead of running as a CGI?
 - ...

Modularity

- Administrator must then
 - rerun the set up program specifying modules,
 - rebuild Apache,
 - reinstall Apache
- Once Apache installed its range of capabilities are fixed.¶
- Still have to set control parameters in a configuration file (*e.g. you may have specified that you want mod-status included, but you still have to set a Location parameter that says who can read the status data*)

¶If build with "dynamic shared objects" on Unix then can add functionality after original build; on windows, can add dll.

Configuring apache

- Can change some of configuration parameters after installation
 - Shut down (or suspend) server
 - Edit configuration files
 - Restart server
- (Changes that you make to a config file will "survive" a reinstallation of Apache)

APACHE

access

control

Access controls

- Apache set up must secure file-space against inappropriate web-access.
 - General setup:
 - Where are files located?
 - Are other parts of file-space blocked off
 - Specific setup for individual directories
 - Access controls
 - Authorisation

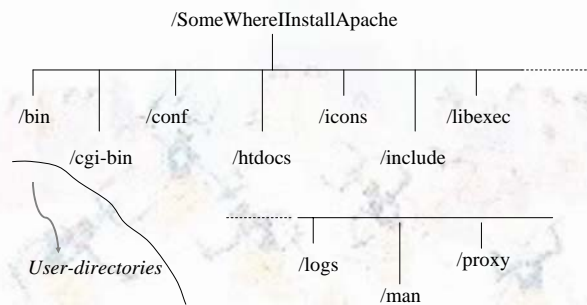
Apache's directory, user

- Standard web server uses port 80
 - Means that it has to be started by “root” (though should switch immediately to different identity such as www)
 - Installation directory typically something like /opt, or /usr/local
- If you are not Sys. Admin, merely running your own Apache set up
 - Use chosen port (or default 8080)
 - Install where you have permissions

Own Linux apache ...

- Create new user and group entries
 - User www or apache or ...
 - Group web or ... (members: www and ids of users allowed to modify apache set up)
- User entry in /etc/passwd:
 - User name
 - Probably no password (can not login as www)
 - uid
 - Gid
 - Real user name: you or your web administrator
 - Directory where apache installed
 - Shell as /bin/false

Typical Apache file set up



Apache file set up

- bin
 - http (the real web server),
 - plus helper programs:
 - apachectl (complex script supporting simple commands for starting/stopping server),
 - rotatelogs,
 - htpasswd, dbmanage
- cgi-bin,
- htdocs
- conf
- logs

Apache in assignment 1

- Laboratory Ubuntu (Linux) systems
 - Actually two apaches –
 - one installed standard for Ubuntu
 - One installed for CSCI399 (Zend-Oracle special configuration downloaded from Zend)
 - Set up to use user directory (

Access control

- Basic controls (that come with mod_access) allow some restrictions based on IP address included in request
 - Can make a page generally available
 - Can prohibit access from range of IP addresses but allow access from everywhere else
 - More useful, can prohibit general access and then specify IP addresses of those that are permitted access.

Access controls

- General policies are specified in configuration file (via <Directory...> sections)
- Maybe allow overrides in individual directories
 - Overrides in form of .htaccess files with individualized rules
- Checking for .htaccess files is costly (before returning a page, server must check for .htaccess file in specified directory *and in parent directories*); many installations disallow .htaccess controls because of cost (and possibility of errors)

```
<Directory "@@ServerRoot@@/htdocs">  
Options Indexes FollowSymLinks MultiViews  
AllowOverride None  
Order allow,deny  
Allow from all  
</Directory>
```

This is a template for the default setting. Controls main htdocs and all subdirectories.
Permits browsers to get directory listings (do you want this?).
Disallows .htaccess changes in subdirectories. Allows access by world.
MultiViews option relates to support for negotiations for different Language versions of pages etc. Used to be standard – but not with most Apache-2s

Authorization

- Mentioned authorization in http.
 - Page in a directory that requires authorization
 - Response to an http request is a 401 error demanding authorization (*TCP/IP link left open so that client can respond*)
 - Browser puts up dialog requesting name, and password
 - User enters name, password data; browser returns
 - Webserver checks data against some form of record
 - Text file
 - dbm database
 - sql database

Authorization

- Browser does some state maintenance here
 - Keeps record of name, password, and “realm” of controlled resource
 - Includes “Authorization” when needed in subsequent requests
- *Note that server*
 - First has to send the request for authorization
 - Waits for restated request with Authorization header
 - Looks up name/password in its records

Authorization mechanisms

- Basic:
 - Name/password file as text file (password encrypted using standard crypt() function)
 - Apache server must read entire file searching for match
 - Not advisable if have hundreds of name/password accounts
 - Do not attempt any tie in with main /etc/passwd passwords.
- DBM
 - Unix, use dbm (db) “database” package

Basic authorization

- Can:
 - List allowed users by name
 - Simply require that user specify their password, any username/password combination in password file is OK
 - Define groups of users and require group membership

APACHE

Administrator's

roles:

access

Administrator: access

- Edit main controls into httpd.conf file
 - Default controls defined for general file space, htdocs, and cgi
 - Augment with overriding controls that apply to chosen subdirectories; examples
 - A single subdirectory where you allow server side include files?
 - Access controls (IP/Domain/Password for a chosen directory and its subdirectories)
 - Permission for CGI programs somewhere other than cgi-bin
 - Support for Multiviews (multiple versions of same content, selected by negotiation with user)
 - ...

Note: Apache 2.2; httpd.conf file split into main file & include files

httpd.conf or .htaccess

- Can specify controls for a directory in httpd.conf, or can have entry in httpd.conf which basically says “look for an .htaccess” file and use it”; then put detailed controls in .htaccess file
- Probably advantageous to put controls in httpd.conf:
 - Only web admin can make changes
 - All access data summarized in the one place.

APACHE

logs

Logging

- Two logs are normally maintained:
 - Access
 - Who asked for what? When?
 - Errors
 - Output from dieing cgi programs etc
 - Files not found
- Custom logs also available.
- Logs grow rapidly in size at a busy site
 - Rotate logs
 - Analyze logs

Access log

- 130.130.189.103 - - [28/May/2001:14:37:17 +1000] "GET /~yz13/links.htm HTTP/1.0" 200 1011
- 208.219.77.29 - - [28/May/2001:14:37:26 +1000] "GET /robots.txt HTTP/1.1" 404 216
- 130.130.189.103 - - [28/May/2001:14:38:18 +1000] "GET /~yz13/image/tb.gif HTTP/1.0" 200 94496
- 130.130.64.188 - yag [25/May/2001:11:39:49 +1000] "GET /controlled/printenv.pl HTTP/1.1" 401 486

Access log : content

- 130.130.189.103 - - [28/May/2001:14:37:17 +1000] "GET /~yz13/links.htm HTTP/1.0" 200 1011

- 130.130.189.103 IP address requestor
 - - (well it would be email address as per identd)
 - - (Decoded name from Authorization header)
- Date & time (and offset relative to UTC)
- The request
- Result code
- Bytes returned

Someone successfully read links.htm from yz13/links.htm.
Came back for a gif about 1 minute later

Access log : error recorded

- 208.219.77.29 - - [28/May/2001:14:37:26 +1000] "GET /robots.txt HTTP/1.1" 404 216
- For reasons unknown, someone at marvin.northernlight.com was running a web spider to investigate banshee:2000 server (and isn't obeying the rules, spiders should leave email address)
 - Request for robots.txt is part of spider initial probe
 - Got a 404 "Not Found" response

host -a 208.219.77.29

Access log : authorizations

- 130.130.64.188 - yag [25/May/2001:11:39:49 +1000] "GET /controlled/printenv.pl HTTP/1.1" 401 486
- Access to a resource that is in a password controlled area.
- Authorization name was included

Host id

- Default is log contains IP address
- Can have web server:
 - Use gethostbyaddr() function (or equivalent) to take address and lookup hostname,
 - Record hostname in log
 - Costly.
 - Cheaper to get the hostnames when analyze the log (as then would lookup each name only once)

Error logs

- Configuration parameter sets level for logging:
 - debug
 - info,
 - notice,
 - warn,
 - **error,**
 - **crit,**
 - **alert,**
 - **emerg.**

Errors log

- [Thu May 24 13:27:55 2001] [error] [client 202.129.93.44] File does not exist: /packages/csci213-www/documents/ma61
- [Thu May 24 14:00:30 2001] [error] [client 130.130.66.60] (13)Permission denied: file permissions deny server access: /packages/csci213-www/documents/cgi-bin/sp15/ass4/myApplet2.html

Errors log

- [Sun May 27 20:49:14 2001] [error] [client 130.130.64.1] **malformed header from script**. Bad header=6: /packages/csci213-www/documents/cgi-bin/yz13/cgi/cou/counter.cgi
- [Fri May 25 10:31:34 2001] [error] [client 130.130.64.33] user Aladdin not found: /controlled/test.html

Other logs

- Can log:
 - Agent (browser)
 - Referrer
 - Page from whence link was followed to get to your page
 - Example:
 - Many special interest groups have web sites that have links back to Amazon noting books there that are of interest to group members; Amazon can count customers arriving from site and pay some fraction of a cent for each one.

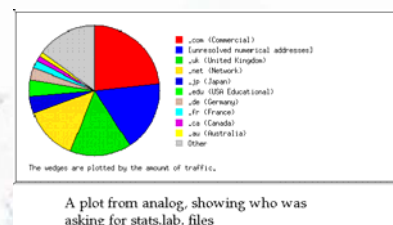
Logs : use

- Debugging:
 - Watch the access and error logs as you test your new Perl script CGI program
- Site maintenance
 - Pick up broken links where something has been moved without a redirect page left at original location
 - Find incorrect permissions set on files.
- Security
 - Spot amateur attacks on web site

Logs : analysis

- Various tools (some free, some not) that run through the log files and produce reports (e.g. www.statslab.cam.ac.uk/~sret1/analog , well that link is now dead but try <http://www.amslog.cx/>)
- Information:
 - Histograms of traffic (pages & bytes) showing monthly/daily/hourly traffic
 - Summaries on origins of all requests
 - Identities of sites with highest number of requests
 - Result codes
 - Files most requested

Log analysis data



APACHE

virtual hosts

Virtual hosts

- Mainly of interest to those who want to run a small Internet Service Provider.
 - Your customers want web sites
 - Don't want addresses like:
 - www.smallisp.com/~FlightyFashion
 - www.smallisp.com/~RecordStore
 - Want addresses like:
 - www.FlightyFashion.com
 - www.RecordStore.com

Virtual hosts

- Most of work here is in the Domain Name Services area
 - Must get those host names:
 - Created
 - Registered in appropriate enclosing domain (.com for those examples)
 - Registered address is IP address of host running your Apache server
 - (Commercial companies eg www.networksolutions.com can help you register, and pay for, a domain name – all the obvious ones have now been taken!)

Virtual hosts: Apache side

- Configuration file must identify the different hosts
- Each host would have separate file space area (*can't put all the files into one htdocs directory, there are bound to be conflicts; also you might be hosting competitive sites and wouldn't want to have situation where they could possibly interfere with one another*).

Virtual hosts : knowing which one!

- HTTP 1.1 compliant browsers include a header with name of Host
- This becomes environment variable, so server can use value of environment variable to determine whether Sales.html is RecordStore/Sales.html or FlightyFashion/Sales.html
- You have HTTP 1.0 clients!
 - Tough.

Virtual hosts: configuring Apache

- Directives:
 - NameVirtualHost
 - Identifies IP address being used for virtual hosting
 - VirtualHost directives
 - One for each host
 - Gives server name www.FlightyFashion.com
 - Gives corresponding DocumentRoot directory

IP-based multiple hosts

- If your computer has multiple network connections (and multiple network cards) you will have multiple IP addresses
- Can configure your Apache to listen at all these IP addresses
- Can then set up a hosting scheme where different virtual hosts correspond to the different IP addresses
- Clumsy – but it does work with older browsers

Problems setting up virtual hosts etc

- The `httpd.conf` file has to contain details of host names and IP addresses.
 - `BindAddress` directive
 - `VirtualHost` directive
- Easier to create things correctly if use hostnames – but this has negative impact on performance; once you've got it sorted out switch to numeric IP addresses.

APACHE

dynamic pages

Dynamic pages

- Dynamic pages:
 - CGI
 - Server Side Includes
 - `mod_perl`, `php` etc
- Best if all mechanisms for creating dynamic pages are limited to use in a few specified directories.
 - Every script, ssi page etc represents a security risk, a point where hacker may disrupt or break into your site.

Server Side Includes

- Server must be configured to
 - Return pages that had included ssi directives as html pages
 - Know which pages must be parsed by server prior to return
- Configuration should be done on a directory basis (and shouldn't delegate responsibility to users who have directories available via Web)

Server Side Includes

- Levels:
 - Pretty harmless (*some performance loss*)
 - `Config`, `flastmod`, `fsize`, `set`, `printenv`
 - "Ify" ...
 - `include`
 - Potentially dangerous
 - `Exec`
- Controls
 - None, no `exec` (and no includes that cause anything to be executed), all

Server Side Includes

```
<!--#exec cgi="/cgi-bin/aprogram.pl" -->
<!--#exec cmd="cat /etc/passwd" -->

<!--#exec cmd="touch users" -->
<!--#exec cmd="Count.sh" -->
```

A counter

```
Val=`cat counter`
NewVal=`expr $Val + 1`
echo $NewVal > counter
echo $NewVal

Counter
0
```

Microsofties and SSI

- <http://support.microsoft.com/kb/203064>
- Batch (.bat) coding even more arcane than shell-scripting.
- Difficult to implement scripts
 - Also discouraged as viewed as particularly high risk on Windows

CGI

- Default directory where CGI programs expected to be located.
- Can add:
 - Permission to use CGI programs in another directory
 - General directives to identify particular file types to be treated as executable in all contexts

Handlers

- Associated with directory (or URL)
- Identify an Apache module that should process a file
 - Example, would have a handler specified for all mod_perl perl scripts.

Administrator responsibilities

- Compose the httpd.conf script so that executables are only in directories where you want them.

What user id?

- Apaches run under specified user id (given in config file, should not be 'root'). Typically "nobody" or "www"
- Consequently:
 - Processes launched by Apaches (CGI processes) run as "nobody" or "www"
 - Processes can still do things like access databases (as code can contain database log-in or password)
 - Processes have problems with files.

Changing the user id for CGI process

- Suppose have CGI program that is to update a file
 - It needs write access
 - File probably does not belong to "nobody" or "www", it belongs to particular user with pages hosted on your machine (eg FlightyFashion = ff001)
 - www, nobody etc can't normally write to ff01's files.
 - ff01 should NOT make order file "writeable by other" (your site also hosts CheapCheerfulClothes, ccc01; user ccc01 might find compelling commercial reasons to accidentally overwrite ff01/orderfile.dat)
 - CGI programs run for FlightyFashion must run as user ff01.

Set user id

- First option: a "set user id" file system.
- Files that are executable by group and other can have directory entry marked as "set user id".
- When Unix does exec() on such a file, it changes effective user id to the user id of the file owner.
- So, ff01 can have CGI programs that are tagged set user id
- When run, these run as if ff01 had launched them.

Set user id

- Unix has to be configured to honor set user id file tags
 - It is done on a file system basis, can configure parts of file system (directory and its subdirectories) as supporting set user id
 - Normally restricted to small number of directories
- Set user id regarded as risky:
 - Gives control to users who put programs in their private web directories
 - Users who make lots of mistakes!

SU-exec

- Optional extension to Apache system
 - Acts as if intermediary in process of launching a CGI program
 - Runs as root (?)
 - Performs a large number of safety checks then sets user id by system call prior to execing required CGI program.
 - (documentation refers to it as the "wrapper")

SU-exec checks

- Checks are intended primarily to:
 - Prevent anyone from sneakily getting a program to run with user-id = "root"
 - Prevent anyone from trying to run a script or executable that might have been changed by someone other than official owner

checks

- Was the wrapper called with the proper number of arguments?
- Is the user executing this wrapper a valid user of this system?
- Is this valid user allowed to run the wrapper?
- Does the target program have an unsafe hierarchical reference?
- Is the target user name valid?
- Is the target group name valid?
- Is the target user *NOT* superuser?
- Is the target userid *ABOVE* the minimum ID number?
- Is the target group *NOT* the superuser group?
- Is the target groupid *ABOVE* the minimum ID number?
- Can the wrapper successfully become the target user and group?
- Does the directory in which the program resides exist?
- Is the directory *NOT* writable by anyone else?
- Does the target program exist?
- Is the target program *NOT* writable by anyone else?
- Is the target program *NOT* setuid or setgid?
- Is the target user/group the same as the program's user/group?
- Can we successfully clean the process environment to ensure safe operations?
- Can we successfully become the target program and execute?

APACHE

out of the box

Apache (Unix)

- Download a tar, gzip version of server
- Decompress
- Untar
- You should now have subdirectory (apache_1.3.xx, apache_2.yy) containing:
 - README, License, similar files
 - Subdirectories for
 - cgi-bin, conf, htdocs (containing some default pages for MultiViews, and directory with manual), icons, logs, src

Apache installing

- These are your master copies.
- When you build and install, you copy most of this stuff into a separate directory.
- *(You should keep only the tar.gz file; otherwise you will probably be exceeding your file quota.)*

config

- Config script in directory:
- Run `./configure --help`
- This provides information about the version of Apache that will be built,
 - lists those modules that will be included
 - Lists names of parameters that can be set
- (configure takes complex command line arguments, mostly as name=value parameter pairs)

modules

```
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         spelling=no
status=yes      unique_id=no   userdir=yes
usertrack=no    vhost_alias=no
```

Run config

- At minimum
 - Specify base directory where your Apache camps.
`./configure --prefix=/spare/abc123/Apache`
- Usually
 - Specify “standard modules” that are to be disabled
 - Specify “non standard modules” that are to be enabled.
- Optionally
 - Define locations for log files, icons, man pages etc if for some reason you don't want standard set up.
 - Define non-standard port number, user id, ..
 - Define su-exec set up

Configure example

- Don't feel like having user directories with this tribe.
- Thinking of playing with a dbm database for passwords.
- Want to build in /local/Amerindian
`./configure --prefix=/local/Amerindian \
--disable-userdir \
--enable-auth_dbm`

Form of ./configure arguments varies slightly with Apache version

make

- Next step is to run the generated makefile
make
- Then do the install into the selected target directory
make install

On banshee, work in a directory created within /packages/tmp;
make may take 20 minutes, make install 3 minutes

Your installed Apache

- You should now have a ready to run Apache.
 - cgi-bin
 - Two demonstration scripts (perl and sh) that echo environment variables
 - htdocs
 - A welcome page in several European languages
 - Subdirectory with Apache manual
 - conf
 - Default configuration
 - bin
 - apachectl script and helper programs

Check installation

- cd to your Amerindian/bin directory
 - apachectl configtest
 - Should respond Syntax OK
 - **apachectl start**
- aim a browser at your server –
 - `http://localhost:8080`
 - (Note, problems with proxy-server configurations for browsers might mean you have to use the actual machine name rather than “localhost”; can get machine name with Unix `uname` command)

Response

- “It works”
 - Apache 2.2 style response
- Apache Welcome page in any of a dozen (mostly European) languages – earlier Apaches.

PC install

- You get a .exe file that does the install.
- It will add an Apache entry to your programs in Start control (or might make it a PC service – but you probably don't want that)
- It don't work?
 - You have to edit the httpd.conf file to set yourservername.
 - Then it will work.
- Installer should fix this but sometimes theservername is left undefined

Check installation

- Look at your processes:
`ps -ef | fgrep httpd`
You will find something like 10 copies running (the chief and 9 little indians)
- Try getting server status or info
 - `http://localhost:8080/server-status`
- You will be disappointed; you have included the required modules but default configuration file leaves them switched off.

Check installation

- Try the provided CGI scripts
 - `http://localhost:8080/cgi-bin/test-cgi`
- They may not work
 - Anomaly?
 - I had to change permissions adding x for owner group and other before scripts ran.
 - I had expected set up scripts to set the correct permissions when the files were installed.
- Shutdown your Apaches
 - `apachectl stop`

APACHE

conf

Configuration files

- Conf subdirectory contains configuration files for your Apaches.
- Traditionally had httpd.conf, srm.conf, and access.conf;
later mostly combine all data into the httpd.conf file
but with Apache-2 again get multiple files!
- You can edit a conf file while keeping your Apaches running and arrange graceful transition to using new file.

Configuration files

- Contain directives
- Some set general properties of server
- Others affect specific directories (or files, or URLs, or Virtual hosts)
- Some provision for overrides via control files in other directories – avoid where possible, best to have all configuration data in one place under web master control

Apache 2.2 httpd.conf

- Lots of stuff moved to “extras” subdirectory
 - Apache manual
 - Multilanguage demo
 - server-status and server-info
 - ...
- http.conf file has commented-out “includes” on things in extras subdirectory
- You re-instate include
- You edit things like the server-info conf file in the extras subdirectory

httpd.conf

- ServerType
 - inetd, or standalone
- ServerRoot
 - “/spare/Amerindian”
- LockFile, PidFile, ScoreBoardFile
 - PidFile holds process id of Apache chief; (useful if you want to send a signal to the entire tribe)

httpd.conf

- Timeout
- KeepAlive, MaxKeepAliveRequests, KeepAliveTimeout
- MinSpareServers, MaxSpareServers, StartServers, MaxClients
 - These defaults are for a real web server, decrease them for a toy test site!
- MaxRequestsPerChild

httpd.conf

- BindAddress, and Listen directives
 - Control over port numbers and IP addresses that are monitored
- Loadable modules?
 - Dynamic Shared Objects in Unix/Linux, dlls for Windows
 - May have list of extra modules that can add in via dynamic linking
- ExtendedStatus
 - Extra information (at some cost) on status of individual tribesmen

httpd.conf

- Port
- User, Group
 - Only meaningful for a server run initially by root
- ServerAdmin
 - Email address, default is generated by script
- ServerName
 - Must set this for Windows version to work
- DocumentRoot

httpd.conf

- Directory controls (also Location) – *later*.
- Logging:
 - ErrorLog
 - LogLevel
 - LogFormat
 - Log file location
- Module specific configuration (conditional includes)
 - e.g. if you have mod_autoindex, (fancy directory listings), then there is a config section specifying the icons used for different file types etc

httpd.conf

- Alias
 - Allows mappings from directory names used in the URLs in documents etc to actual physical directories

CGI

- ScriptAlias
 - Specifies where CGI programs normally kept
 - ScriptAlias /cgi-bin/ "/spare/Amerindian/cgi-bin/"
- Other CGI programs
 - Directory must be marked as having CGI programs
 - Apache must have some way of recognizing the executables in that directory

CGI

- *Letting Apache know which files are programs:*
 - Have to relate file type (e.g. ".cgi") to a handler
 - AddHandler cgi-script .cgi**
 - Might also want to specify other types (e.g. .pl)
- *Alternatively*
 - You can use mime types to achieve much the same
 - AddType application/x-httpd-cgi .pl**

"magic mime types" are sort of deprecated; AddHandler would be preferred.

CGI

- Authorizing use of CGI on a per-directory basis ...

Why didn't server-status work?

- Mod_status was included but request for server-status failed.
- "server-status" is a special "Location" (not a file or a directory)
- A request aimed at server-status should be handled by the mod_status code in the Apache server.
- So, somewhere in the config file there will be definition of the server-status handler.

Server-status

```
#<Location /server-status>  
#   SetHandler server-status  
#   Order deny,allow  
#   Deny from all  
#   Allow from .your_domain.com  
#</Location>
```

- It is commented out.
- You have to specify who can read your server's status.
- This is done by specifying domain.
- Similar for server-info

Changing httpd.conf

- Virtual hosts:
 - NameVirtualHost 128.129.130.131
 - (specify IP address to which DNS maps the host names)
 - <VirtualHost a.b.c.d>
 - Control
 - Control
 - ...
 - </VirtualHost>

Note: may need to change some OS controls on your TCP/IP connection to get virtual hosts to work

Virtual host controls

- Essential:
 - ServerName *virtual host name*
 - DocumentRoot *separate the files of different web sites*
- Optional
 - Error logs separated
 - Different names for web administrators
 - ... (can have other controls – eg language negotiation, handlers for data types – specified for a virtual host instead of being done globally as is the default)

Virtual hosts

- Care needed
- What about a default host?

Adding negotiation ...

- mod_mime
 - Can add languages and charsets
 - Add language specifies a mapping from a language code (specified by user in browser) and a file extension; can use with MultiViews to provide language support
 - AddLanguage es .es

Other changes to mime types

- Other AddType directives are likely to be needed
- Specify mime-type, file extension
 - Can be for synonyms
 - AddType image/gif GIF
 - Can be to specify type to be returned in http response header for given file type
 - AddType audio/x-midi .mid .midi .kar

A global control on file access ...

- <Files ~ "\.ht">
- Order allow,deny
- Deny from all
- </Files>
- Uses regular expression matching on file names
 - ^ at start of name
 - \. Look for a period (backslash needed as "." has meaning in regular expression)
 - Then look for characters starting with ht
- Match .htaccess .htpasswd etc

Controlling resources

- Directory (also, sometimes Location) tag
- `<Directory directory_name>`
 - Control
 - Control
 - Control
- `</Directory>`

*Directory name is usually full pathname of directory,
Can have "wild card" strings where same directory directive applies to
several directories*

Standard controls ... "root"

- ```
<Directory />
Options FollowSymLinks
AllowOverride None
</Directory>
```
- Root directory for your Apache tribe
  - And, unless changed by a subsequent Directory directive it is policy for all subdirectories
  - Will use Unix "links" to files in other directories
  - Will not do anything much else!

## Standard controls ... "htdocs"

- ```
<Directory "/local/Apache/htdocs">  
Options Indexes FollowSymLinks MultiViews  
AllowOverride None  
Order allow,deny  
Allow from all  
</Directory>
```
- Support language negotiation (MultiViews), directory listing (Indexes)
 - Don't allow .htaccess overrides in subdirectories
 - Make everything accessible on web

Controls ...

- mod_access controls
 - Limits access to resource (e.g. contents of directory, file, ...)
 - Uses domain names or IP addresses of client
 - If domain name specified, Apache does sanity check
 - Easy for hacker to put wrong domain name in http header,
 - Not quite so easy for hacker to fake the IP address if really wanting a response
 - So check domain name, do reverse look-up from domain name to IP address, recheck IP address
 - Can identify clients that are denied access, clients that are allowed access

Mod-access

- Order
- Allow
- Deny
- mutual-failure
- Order allow, deny || Order deny, allow
 - If mostly allowing, then order allow, deny
- Mutual-failure: check is in the allow group and is not in the deny group

Mod-access Examples

```
<Directory "/local/Apache/htdocs/onCampus">  
order deny, allow  
deny from all  
Allow from .uow.edu.au  
</Directory>  
<Directory /local/Apache/htdocs/notForTheFrench>  
order allow, deny  
allow from all  
deny from .fr  
</Directory>
```

mod-auth

- Directory must now specify:
 - AuthName
 - AuthType
 - Either
 - Require valid-user
 - Require user tom dick harry anna
 - Require group csci321g321

Similar controls for other “auth” systems

mod-auth and related

- AuthName
 - Authorization name
 - Name for controlled area
 - It appears in dialogs put up by browsers when asking would-be user to enter credentials
- AuthType
 - Basic
 - Simple name/password combinations in a text file
 - Need names of password and “group files”

Passwords

- Create a password file
 - Probably have a directory in your Apache root directory where hold password files
- htpasswd
 - Utility program in /bin
 - htpasswd -c passwordfile firstuser
 - htpasswd passwordfile anotheruser
 - Prompts for entry of password

Password files

- Obviously can have a different password file, with user names and passwords, for each AuthName resource; but that means that
 - Either your users may have to remember several passwords
 - Or you have work keeping passwords consistent in different files
- Or can work with single password file, but then if have different resources with different user populations you can't rely on simple “require valid-user”

Group files

- If have resources only to be viewed by subset of users, can list names
 - require user tom dick harry anna
- Bit tiresome if several hundred users
- Can supplement password file with a group file:
 - Text file, lines define groups (group name, colon, names of members)
 - BridgePlayers: anne david carol phillip peter jon james

Mod-auth example

```
<Directory notices>
  AuthName "Private Departmental Notices"
  AuthType Basic
  AuthUserFile /spare/Amerindian/pwrds/passes
  AuthGroupFile /spare/Amerindian/pwrds/grps
  Require valid-user
</Directory>
```

Combination checking

- Can require
 - Restrictions on client IP address OR password
 - Save people from entering password while on campus, require password for use from outside local domain
 - Restrictions on client IP address AND password
 - Slightly more security

Combined mod-access, mod-auth

```
<Directory /some/path/hotstuff>
  Order deny, allow
  Deny from all
  Allow from cs.uow.edu.au
  AuthName ...
  ...
  Require group staff
  Satisfy all
</Directory>
```

- Alternatively,
 - Satisfy any

Adding privileges

- Things you might want to add to a directory:
 - Option for an .htaccess file in the directory that specifies controls (*almost anything that can go in a <Directory> directive of httpd.conf can instead be defined in an .htaccess file*)
 - Add/Remove features
 - Server Side Includes
 - CGI execution
 - Multiviews
 - Indexing
 - File types, languages, mime types, ...
 - (Access and Auth controls)

AllowOverride

- Permit .htaccess overrides
- But can specify what aspects can be overridden:
 - All, None
 - AuthConfig - password related controls
 - FileInfo - adding languages, types, ...
 - Indexes - ability to show directory listing and controls on its format
 - Limit - access controls: allow, deny, order
 - Options - add/remove CGI, SSI etc
- AllowOverride None (inherited from main htdocs directory) is best – least cost, least risk.

Options

- Use in <Directory> directive
- List options required, or add/remove inherited options (inherited from parent directory).
 - Options IncludesNoExec FollowSymLinks Indexes
 - Options +ExecCGI
 - Options -Indexes

.htaccess

- Want ExecCGI in this directory (and have permission to change options)?
- Then simply have a ".htaccess" file containing:
Options +ExecCGI

Options

- All all (except MultiViews which must be requested explicitly)
- ExecCGI
- FollowSymLinks
- Includes
- IncludesNoEXEC
- Indexes
- MultiViews
- SymLinksIfOwnersMatch

Changing httpd.conf

- After section where it defines the two defaults (global and htdocs), add Directory directives for each directory where it is appropriate to specify different controls.

```
<Directory /long/path/name/active>  
    Options +Includes  
</Directory>
```

apachectl restart

- Working tribesmen finish
- Tribe terminates
- New tribe created – members read updated config file and start obeying new rules.