HTTP is a stateless protocol. Cookies provide a mechanism to "maintain state".

- **Cookie Central: The Unofficial Cookie FAQ**
  http://www.cookiecentral.com/faq/
  http://www.cookiecentral.com/

**Maintaining State with Cookies**

- **HTTP State Management Mechanism**
- **Cookie Central: The Unofficial Cookie FAQ**
  http://www.cookiecentral.com/faq/
- **Persistent Client State HTTP Cookies**
  http://www.netscape.com/newsref/std/cookie_spec.html
Cookie Example

- Server returns cookie to HTTP client ("Set-Cookie" response header)
- HTTP client returns cookie to server ("Cookie" request header)

ESPN Cookies

- SWID=C8F9AF31-F170-42BF-9471-50A95DA24C17
  - path=/
  - expires=2027-04-10
  - domain=.go.com

- DE2=dXNhO21hO2NhbWJyaWRnZTt0MTs1OzQ7NDs1MDY7MDQyLjM4MDstMDcxLjEzN
  - path=/
  - expires=2007-04-17
  - domain=.go.com

Cookie Properties/Attributes

- name
- expires
- domain
- path
- secure

HTTP State Management Mechanism, RFC 2965

RFC 2109, February 1997
RFC 2965, October 2000

- name
- comment
- comment URL
- discard
- domain
- max-age
- path
- port
- secure
- version

Additional Cookie Notes

- Client: 300 total cookies
- 4 kb per cookie
- 20 cookies per server or domain
Cookie Example: Server Sets a Cookie

Form that will set a Cookie:
http://cscie12.dce.harvard.edu/http/cookie.cgi

Set-Cookie HTTP Response Header:
Set-Cookie: YourName=David P. Heitmeyer; domain=cscie12.dce.harvard.edu; path=/http/; expires=Fri, 13-May-2005 18:05:04 GMT

---

Cookie Example: Returning a Cookie

Form that will set a Cookie:
http://cscie12.dce.harvard.edu/http/cookie.cgi

Set-Cookie HTTP Response Header:
Set-Cookie: YourName=David P. Heitmeyer; domain=cscie12.dce.harvard.edu; path=/http/; expires=Fri, 13-May-2005 18:05:04 GMT
Your Cookies

Firefox Webdeveloper Toolbar has a "Cookies" section.

screenshot

Mozilla Cookie Manager

Cookies and Session IDs

A UserID or SessionID (a long character/number string that is uniquely assigned) is often stored in cookie. The SessionID is used as the key or identifier when storing information about the user or session.

For example, a user logs in to a site. If the username and password match, the server sets a cookie ("Set-Cookie") in the browser that contains a session id; the server also makes an entry in website database that maps the session id to the username. When the cookie is returned, the session id is read and the username is looked up in the database.
Google Cookie Example

Using Google’s “Preference” page and setting:

- Search Language preference to: English, French, German
- SafeSearch Filtering: Strict Filtering
- Number of Results: 50

The Cookie name is: PREF
The Value is:

This cookie contains a session id as well as the values of certain preferences in a colon-separated data structure.

Cookies and Ad Tracking

Cookies are used by websites to store information about your visit. Ad tracking involves cookies being used to track your online activity. Advertisers use cookies to show you relevant ads. Blocking cookies can help reduce ad tracking.
Method: POST

Form that will set a Cookie:
http://cscie12.dce.harvard.edu/http/cookie.cgi

WebDAV: an extension of HTTP

Web-based Distributed Authoring and Versioning

- WebDAV Resources
  http://www.webdav.org/
- From the WebDAV Resources:

  WebDAV stands for "Web-based Distributed Authoring and Versioning". It is a set of extensions to the HTTP protocol which allows users to collaboratively edit and manage files on remote web servers.
HTTP Resources

- W3C HTTP
  http://www.w3.org/Protocols/
- HTTP Pocket Reference
  http://www.oreilly.com/catalog/htppr/
  by Clinton Wong (O'Reilly).
- Illustrated Guide to HTTP
  http://www.manning.com/hethmon/
  by Paul Hethmon (Manning Publications; ISBN 0138582262)
  see sample chapters and resources online.

Other Readings:

- W3C Recommendations Reduce 'World Wide Wait'
  http://www.w3.org/Protocols/NL-PerfNote.html
- Apache Week: HTTP version 1.1
  http://www.apacheweek.com/features/http11
- WebTechniques: HTTP 1.1: What's in it for Me?
- Cookie Central: The Unofficial Cookie FAQ
  http://www.cookiecentral.com/faq/
  http://www.cookiecentral.com/

Apache HTTP Server

- Apache Software Foundation
- Apache HTTP Server Project
  - Apache 1.3
  - Apache 2.x
- Apache Modules
  - PHP
  - Perl
  - Python
  - many, many others
Apache: The Most Widely Used Web Server on the Public Internet

In this Unit: Configuring Apache with .htaccess files
- Custom Error Documents
- Redirect
- Rewrite
- Directory Index
- Setting HTTP Headers
  - Expires
  - Headers
- Access Control
- Requiring a Secure Connection (SSL)
Apache Configuration Overview

- **Server Configuration** ([httpd.conf](http://localhost:8080/cocoon/projects/cscie12/slides/20070417/handout.html))
  Unless you are the server administrator, you generally will not have access to this account. On the DCE systems, you do not have read or write access to this file. Server configuration is read at server start or restart.

- **Per Directory** (.htaccess)
  Certain configuration directives for Apache can be placed within *per-directory*. **.htaccess** files. **.htaccess** file is read on a per request basis.

---

**.htaccess File Example**

```
document root: /home/e12/htdocs
filename: .htaccess
location: /home/e12/htdocs/apache/.htaccess
contents:

```1. ErrorDocument 404 status404.html
```

- **filename:** status404.html
- **location:** /home/e12/htdocs/apache/status404.html

- **http://cscie12.dce.harvard.edu/apache/ZZZ.html**
Scope of .htaccess files

Directives within .htaccess files apply to the directory that contains the .htaccess file and all its descendants.

Directives within the file, /home/e12/htdocs/.htaccess would apply to all files within and "under" the public_html directory for the user cscie12.

Directives within the file, /home/e12/htdocs/assignments/.htaccess would apply to all files within and "under" the public_html/assignments directory for the user cscie12.

Problems You Will Have with .htaccess files

- Internal Server Error
- Can't "see" the file
- Incorrect Permissions
Problems You will encounter when using .htaccess files

500 Internal Server Error
If you see begin seeing 500 Internal Server Error responses from the server after you have created or edited an .htaccess file, the most likely cause of the problem is incorrect permissions and/or an error in the directive syntax.

- Permissions on the .htaccess file are not set correctly. Just like HTML and image files, the server must be able to read the .htaccess file. The simplest way to allow that is to make your .htaccess file readable by "other".

- Syntax Error. An error in the syntax of a directive the .htaccess file will result in a 500 Internal Server Error. In addition, correct usage of a directive that is not allowed in the .htaccess file will result in a 500 status code. Whether or not a directive is allowed depends upon the server configuration file (httpd.conf; AllowOverride) and the directive itself.

You can't "see" your .htaccess file.

- HTTP
  The web server is typically configured to deny requests for .htaccess files. For example, the file corresponding to the URL http://cscie12.dce.harvard.edu/.htaccess exists and is readable by the Web server, but if we try to follow the link, we get a 403 Forbidden response.

- UNIX
  The ls command will not list files or directories that begin with a '.' (dot). In order to see the .htaccess file when you do a directory listing, use the -a (all) option:

- SFTP
  Sometimes your SFTP program will hide the "dot" files unless explicitly told to show them.
Apache Configuration Sections

Configuration directives can be limited by using "sections", such as

- Directory
- Location
- Files
- VirtualHost
- DirectoryMatch
- LocationMatch
- FilesMatch

Within .htaccess

Note that only Files and FilesMatch can be used within .htaccess files.

Examples:

1. <Files .htaccess>
2. Order allow,deny
3. Deny from all
4. </Files>

Examples:

1. # deny access to any .txt backup files
2. <Files *~>
3. Order allow,deny
4. Deny from all
5. </Files>

Configuring Apache with .htaccess files

- Custom Error Documents
- Redirect
- Rewrite
- Directory Index
- Setting HTTP Headers
  - Expires
  - Headers
- Access Control
Custom Error Documents

.htaccess

1. ErrorDocument 403 /apache/status403.html
2. ErrorDocument 404 /apache/status404.html
3. ErrorDocument 405 /apache/status405.html

- ErrorDocument directive
  http://www.apache.org/docs/2.0/mod/core.html#errordocument
- Custom Error Responses
  http://www.apache.org/docs/2.0/custom-error.html

HTTP Redirect

Fight Linkrot!


- Redirect
- Rewrite
- Meta http-equiv refresh

Redirecting Requests

HTTP Status Codes:
301 Moved permanently
302 Moved temporarily

Redirecting client requests can be very useful:

- URL moves to a new location
  - resource removed
  - site structure is reorganized
- Provide "friendly" or additional URLs to access a resource
Redirect

- Redirect directive
  http://www.apache.org/docs/2.0/mod/mod_alias.html#redirect
  .htaccess

Try it:
- http://cscie12.dce.harvard.edu/apache/church_st

Rewrite

- mod_rewrite
  http://www.apache.org/docs/2.0/mod/mod_rewrite.html
- A Users Guide to URL Rewriting with the Apache Webserver
  http://www.engelschall.com/pw/apache/rewriteguide/

Rewrite uses regular expressions to match on a pattern and rewrite to a new location. For example, the Derek Bok Center site used to be a "user" account and had the "~bok_cen/" base. When moved to its own virtual host, all of the "~bok_cen/" requests could be rewritten to the new site with a single rewrite rule.

```plaintext
# rewrite for Bok Center
RewriteRule ^/~?bok_cen(.*) http://bokcenter.fas.harvard.edu$1 [R=301]
```

- Old URL: http://www.fas.harvard.edu/~bok_cen/tf/resources.html
  - (*) matches on: /tf/resources.html
- New URL: http://bokcenter.fas.harvard.edu/tf/resources.html

```plaintext
GET http://bokcenter.fas.harvard.edu/tf/resources.html --> 302 Found
GET http://www.fas.harvard.edu/~bok_cen/tf/resources.html --> 200 OK
```
Examples of Rewrite Uses

Provide a standard mechanism to access course Web sites within Harvard College.

- http://www.courses.harvard.edu/<4 digit catalog number>

For example, Chemistry 7 has a catalog number of 5118, so the URL for the course Web site can be reached through:

- http://www.courses.harvard.edu/5118

The "real" location of the site is:

- http://my.harvard.edu/icb/icb.do?course=fas-chem7

HASCS Site Restructure

Dozens of rewrite directives were put in place when the HASCS site was restructured so that links to documents within the previous site would get redirected to the appropriate page in the new site.

Rewrite: Can be conditional

Rewrite rules can conditional (match against almost any environment variable).

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>RewriteRule on</td>
</tr>
<tr>
<td>2.</td>
<td>RewriteCond %{HTTP_USER_AGENT} ^Lynx</td>
</tr>
<tr>
<td>3.</td>
<td>RewriteRule ^(index.html)?$ text/ [R=302]</td>
</tr>
</tbody>
</table>

Here we match on host and user-agent to deliver an error page explaining that their browser is not supported.

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td># rewrite rule to catch IE Mac browsers since</td>
</tr>
<tr>
<td>2.</td>
<td># the PIN Service does not support them as of 10/16/2006</td>
</tr>
<tr>
<td>3.</td>
<td>RewriteCond %{HTTP_HOST} ^login.icommons.harvard.edu$</td>
</tr>
<tr>
<td>4.</td>
<td>RewriteCond %{HTTP_USER_AGENT} &quot;MSIE 5.*; Mac_PowerPC&quot;</td>
</tr>
<tr>
<td>5.</td>
<td>RewriteRule ^/pinproxy.* /pin_error_ie_mac.html [R,L]</td>
</tr>
</tbody>
</table>
An aside: Text-only sites and "link"

Meta-information can be used to describe alternate content.

- W3C Web Content Accessibility Guidelines: alternate pages
  http://www.w3.org/TR/WAI-WEBCONTENT-TECHS/#alt-pages

In ~/cscie12/public_html/index2.html

Lint view of index2.html provides the text-only version as a link:

1. <link title="Text-only version" rel="alternate" href="http://cscie12.dce.harvard.edu/text/index.html" media="aural, braille, tty"/>

Meta Refresh

Note: redirection may also be achieved on some browsers by using the http-equiv attribute of the <meta> element. More information and examples are provided at http://www.fas.harvard.edu/~web/tutorial/meta/refresh/ . The recommended method is to do it at the server level.

```
<!-- in head -->
<!-- will redirect in 10 seconds -->
<meta http-equiv="Refresh" content="10; URL=http://www.harvard.edu/"/>
```
Directory Index and Listings

Note: Remember the difference between a directory having rwx------ and rwx--r-x permissions?

- **DirectoryIndex**
  [http://www.apache.org/docs/2.0/mod/mod_dir.html](http://www.apache.org/docs/2.0/mod/mod_dir.html)
  Would you prefer main.html or overview.html to be the default files returned when a directory is requested?

- **mod_autoindex**
  [http://www.apache.org/docs/2.0/mod/mod_autoindex.html](http://www.apache.org/docs/2.0/mod/mod_autoindex.html)
  Provides for automatic indexing of a directory.

### DirectoryIndex

<table>
<thead>
<tr>
<th>#</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>main.html</td>
<td>Clients and Servers</td>
</tr>
<tr>
<td>2</td>
<td>overview.html</td>
<td>Simple HTIP Server Overview</td>
</tr>
<tr>
<td>3</td>
<td>slide1.html</td>
<td>HTIP Server Resources</td>
</tr>
</tbody>
</table>

More Control over Directory Listings

**mod_autoindex**

### Basic

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>main.html</td>
<td>Clients and Servers</td>
</tr>
<tr>
<td>overview.html</td>
<td>Simple HTIP Server Overview</td>
</tr>
<tr>
<td>slide1.html</td>
<td>HTIP Server Resources</td>
</tr>
</tbody>
</table>

The details:

**CSCIE12: AutoIndexing**

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>slide1.html</td>
<td>Clients and Servers</td>
</tr>
<tr>
<td>slide2.html</td>
<td>Simple HTIP Server Overview</td>
</tr>
<tr>
<td>slide3.html</td>
<td>HTIP Server Resources</td>
</tr>
</tbody>
</table>

david_haas@fas.harvard.edu
Setting HTTP Headers

- Expires
- Headers
Expires

- Module mod_expires
  http://www.apache.org/docs/2.0/mod/mod_expires.html

.htaccess

1. ExpiresActive On
2. ExpiresByType text/html A3600
3. # HTML expires in 1 hour
4. ExpiresByType image/gif A2592000
5. # GIF expires in 30 days
6. ExpiresByType image/jpeg A2592000
7. # JPEG expires in 30 days
8. ExpiresByType image/png A2592000
9. # PNG expires in 30 days
10. # types not specified
11. ExpiresDefault "now plus 1 day"
12. # expires in 1 day

Do not cache

If you do not want your page cached, set these HTTP response headers:

- Cache-control: no-cache
- Pragma: no-cache
- Expires: <set to now>

In .htaccess in Apache, this would translate to:

1. ExpiresDefault "now"
2. Header set Pragma "no-cache"

Or, expire based upon modification time of document:

1. ExpiresActive On
2. ExpiresByType text/html M86400
3. # HTML expires 1 day after it was last modified
4. ExpiresDefault M86400

From the Apache mod_expires documentation:

This module controls the setting of the Expires HTTP header in server responses. The expiration date can set to be relative to either the time the source file was last modified, or to the time of the client access.

The Expires HTTP header is an instruction to the client about the document's validity and persistence. If cached, the document may be fetched from the cache rather than from the source until this time has passed. After that, the cache copy is considered "expired" and invalid, and a new copy must be obtained from the source.
Headers

- mod_headers
  http://www.apache.org/docs/2.0/mod/mod_headers.html

The optional headers module allows for the customization of HTTP response headers. Headers can be merged, replaced or removed. The server will always add a "Server" and "Date" header to the HTTP response.

Usertrack with Cookies

- mod_usertrack

```
<IfModule mod_usertrack>  # View plain text

# .htaccess file:

1. CookieTracking on
2. CookieStyle RFC2965
3. CookieName MyCookie
4. CookieExpires "1 month 3 days 2 hours"
5. CookieDomain .dce.harvard.edu
</IfModule>
```

```
# View plain text

view plain print ?
1. Header set Author "David P. Heitmeyer"
```
WWW Access Control

You can implement access control on all or part of your Web site so that:

- users must provide a username and password (Basic Authentication);
- users' computers must be within a particular domain.

Basic Authentication: Warning

Basic Authentication alone does not provide the security and privacy to adequately protect truly confidential or personal information.

Basic Authentication is analogous to simply "closing a door" to parts of your Web site. It will prevent the casual or polite users from "opening the door", but will not prevent someone mildly determined to walking in.

Two issues that contribute to the lack of security and privacy are:

- the content is transmitted over the network in plaintext
- the usernames and passwords (submitted with each HTTP request) is transmitted over the network in plaintext.
HTTP: Authenticate

<table>
<thead>
<tr>
<th>line</th>
<th>text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Minerva% telnet 140.247.197.240 80</td>
</tr>
<tr>
<td>2.</td>
<td>Trying 140.247.197.240...</td>
</tr>
<tr>
<td>3.</td>
<td>Connected to 140.247.197.240.</td>
</tr>
<tr>
<td>4.</td>
<td>Escape character is: &quot;&quot;.</td>
</tr>
<tr>
<td>5.</td>
<td>HEAD /apache/access/example1/ HTTP/1.1</td>
</tr>
<tr>
<td>6.</td>
<td>Host: cse112.dce.harvard.edu</td>
</tr>
<tr>
<td>7.</td>
<td>HTTP/1.1 401 Authorization Required</td>
</tr>
<tr>
<td>8.</td>
<td>Connection: close</td>
</tr>
<tr>
<td>10.</td>
<td>Server: Apache/2.0.49 (Fedora)</td>
</tr>
<tr>
<td>11.</td>
<td>WWW-Authenticate: Basic realm=&quot;Basic Authentication Tutorial 1&quot;,</td>
</tr>
<tr>
<td>12.</td>
<td>Content-Type: text/html; charset=iso-8859-1</td>
</tr>
<tr>
<td>13.</td>
<td>Client-Date: Wed, 13 Apr 2005 20:44:39 GMT</td>
</tr>
<tr>
<td>15.</td>
<td>Client-Response-Num: 1</td>
</tr>
</tbody>
</table>

HTTP: Authentication/Authorization

The username:password is sent MIME BASE 64 encoded (not encrypted).

<table>
<thead>
<tr>
<th>line</th>
<th>text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Minerva% telnet 140.247.197.240 80</td>
</tr>
<tr>
<td>2.</td>
<td>Trying 140.247.197.240...</td>
</tr>
<tr>
<td>3.</td>
<td>Connected to 140.247.197.240.</td>
</tr>
<tr>
<td>4.</td>
<td>Escape character is: &quot;&quot;.</td>
</tr>
<tr>
<td>5.</td>
<td>HEAD /apache/access/example1/ HTTP/1.1</td>
</tr>
<tr>
<td>6.</td>
<td>Host: cse112.dce.harvard.edu</td>
</tr>
<tr>
<td>7.</td>
<td>Authorization: Basic Z3Vlc3Q6Z3Vlc3Q=</td>
</tr>
<tr>
<td>8.</td>
<td>Server: Apache/2.0.49 (Fedora)</td>
</tr>
<tr>
<td>9.</td>
<td>Content-Length: 492</td>
</tr>
<tr>
<td>10.</td>
<td>Content-Type: text/html; charset=iso-8859-1</td>
</tr>
<tr>
<td>11.</td>
<td>Client-Date: Wed, 13 Apr 2005 20:44:39 GMT</td>
</tr>
<tr>
<td>12.</td>
<td>Client-Peer: 140.247.197.240:80</td>
</tr>
<tr>
<td>13.</td>
<td>Client-Response-Num: 1</td>
</tr>
</tbody>
</table>
### Access Control Documentation

**Apache**

- Apache FAQ has a section on user authentication.
- Using User Authentication from Apache Week
- Relevant Apache Module and Directive Documentation
  - mod_access module
  - mod_auth module
  - require directive
  - satisfy directive

### Implementing Access Control

To implement access control, you must create a file name ".htaccess" that contains the proper configuration instructions. You may also need to create a ".htpasswd" file using the utility "htpasswd" and a ".htgroup" file.

- htpasswd program
- .htaccess file
- htpasswd file
- htgroup file
htpasswd file

This file contains usernames and encrypted passwords (username:enc_passwd). It is created and managed with the utility, "htpasswd", which can be run from the command line.

This file should not lie within your public_html. It should reside at the root level of your home directory (for example, /home/courses/j/h/jharvard/.htpasswd)

This file needs to be readable by the Web Server.

Sample content:

```
minerva% which htpasswd
```

```
minerva% htpasswd
```

```
Usage: htpasswd [-c] passwordfile username
```

```
minerva% more ~e12/.htpasswd.demo
```

```
guest:79WeSn3vYGsKQ
guest2:wGcgIYLtHNIpM
guest3:j9VzpSX/C8Kr2
guest4:CjHmW1PWNFwXM
```

htgroup file

This file contains group definitions (group_name:member1 member2 ...).

This file should not lie within your public_html. It should reside at the root level of your home directory (for example, /home/courses/j/h/jharvard/.htgroup)

This file needs to be readable by the Web Server.
Access Control Examples

For the examples given, the user "cscie12" is used. You should substitute your username and home directory appropriately.

The following .htpasswd.demo and .htgroup.demo files are used:

/home/e12/.htpasswd.demo
The .htpasswd.demo was generated by using the utility "htpasswd"

Password for "guest" (and all other entries) is "guest". Entries for guest2, guest3, and guest4 are created without the "-c" flag, since the .htpasswd.demo file already exists.

Contents of file:

```
1. guest:79WeSn3vYGsKQ
2. guest2:PR4APgA.4CKO.
3. guest3:5DbCMPbSDstj2
4. guest4:htPnr8jT4bI5E
```

.htgroup.demo
Contents of file:

```
1. VIP: guest guest4
   2. guest:guest4
```

Access Control Example 1

Any valid user in .htpasswd.demo is allowed access

The "AuthName" is the description that is displayed by the browser in the Basic Authentication dialog box.

Contents of sample .htaccess file:

```
<IfModule mod_auth_cass>$2y$10$SdO4uF5eo.iX79WZwXb

AuthName "Basic Authentication Tutorial 1"
AuthType Basic
AuthFile /home/e12/.htpasswd.demo
require valid-user
```

Demonstration of Example 1
You may login as any of the following users (username:password):
guest:guest
guest2:guest
guest3:guest
guest4:guest
Access Control Example 2

Only certain users in .htpasswd.demo are allowed access

Contents of sample .htaccess file:

1. AuthName "Basic Authentication Tutorial 2"
2. AuthType Basic
3. AuthFile /home/e12/.htpasswd.demo
4. require user guest2 guest3

Demonstration of Example 2
Only guest2 and guest3 are authorized:

- guest2:guest
- guest3:guest

Unauthorized:

- guest:guest
- guest4:guest

Access Control Example 3

Only members of a particular group are allowed access

Contents of .htaccess file:

1. AuthName "Basic Authentication Tutorial 3"
2. AuthType Basic
3. AuthFile /home/e12/.htpasswd.demo
4. AuthGroupFile /home/e12/.htgroup.demo
5. require group VIP

Contents of .htgroup.demo file:

VIP: guest, guest4

Demonstration of Example 3
Only members of the group "VIP" (as defined by /home/e12/.htgroup.demo) are authorized (guest and guest4):

- guest:guest
- guest4:guest

Unauthorized:

- guest2:guest
- guest3:guest
**Access Control Example 4**

Only certain computers are allowed access

Contents of sample .htaccess file:

1. `order deny, allow`
2. `deny from all`
3. `allow from 140.247`
4. `allow from 128.103`
5. `allow from .harvard.edu`

**Demonstration of Example 4**

Computers that are on the Harvard network (computers with hostnames ending in .harvard.edu or with IP addresses beginning with 128.103 or 140.247) will have access, others will be denied.

**Access Control Example 5**

Only certain computers are denied access

Contents of sample .htaccess file:

1. `order allow, deny`
2. `allow from all`
3. `deny from .fas.harvard.edu`

**Demonstration of Example 5**

Connections from within the domain 'fas.harvard.edu' will be denied.
**Access Control Example 6**

Certain computers are allowed in; others must provide a username and password.

Contents of sample .htaccess file:

1. `order deny, allow`  
2. `deny from all`  
3. `allow from .yale.edu`  
4. `AuthType Basic`  
5. `AuthUserFile /home/e12/.htpasswd.demo`  
6. `AuthName "Basic Authentication Tutorial 6"`  
7. `require valid-user`  
8. `satisfy any`  

Demonstration of Example 6

Connection from within ".yale.edu" will be allowed; others must provide a valid username and password.

---

**Access Control Example 7**

Only certain computers are allowed in and users must provide a valid username and password.

Contents of sample .htaccess file:

1. `order deny, allow`  
2. `deny from all`  
3. `allow from .harvard.edu`  
4. `AuthType Basic`  
5. `AuthUserFile /home/e12/.htpasswd.demo`  
6. `AuthName "Basic Authentication Tutorial 7"`  
7. `require valid-user`  
8. `satisfy all`  

Demonstration of Example 7

and satisfy all
Requiring SSL (https://)

SSL (Secure Socket Layer) is a protocol that encrypts data between the client and the server. https is HTTP over SSL. More details in our last lecture on Security and Privacy.

Contents of sample .htaccess file:

```
SSLRequireSSL
```

- **Allowed:** https://www.people.fas.harvard.edu/~heitmey/secure/index.html
- **Forbidden:** http://www.people.fas.harvard.edu/~heitmey/secure/index.html

Details about enabling .htaccess and allowed directives

- Context: can these directives be in .htaccess files?
- AllowOverride: is the server configured to allow this group of directives to be overridden in this location?
- Is the required module loaded?
Legal Directives I: Context

Certain Apache directives are legal within .htaccess files. Some are not. See the Apache Documentation for details. Specifically, look at the Context line that is given for the directive in question.

- Apache Core Features
  http://www.apache.org/docs/2.0/mod/core.html
- Apache Module List
  http://www.apache.org/docs/2.0/mod/
- standard Apache Directives
  http://www.apache.org/docs/2.0/mod/directives.html

The following is an excerpt from the Apache HTTP Server Version 1.3 documentation

### ErrorDocument

**Syntax:** ErrorDocument error-code document  
**Context:** server config, virtual host, directory, .htaccess  
**Status:** core  
**Override:** FileInfo  
**Compatibility:** The directory and .htaccess contexts are only available in Apache 1.1 and later.

Also, the "a" indicator on the Apache Quick Reference Card indicates that the directive is valid within an .htaccess file.

Legal Directives II: AllowOverride

Users are allowed to override certain aspects of the main server configuration. The main server configuration file (httpd.conf) contains an AllowOverride directive that determines which directives within .htaccess files Apache will process. The Override line that is given for each directive in the Apache documentation indicates which configuration directive must be active in order to use that directive with an .htaccess file.

For the FAS system, the main server configuration file has the following directive in place for users' public_html directories:

```
<Directory />
  AllowOverride FileInfo AuthConfig Limit Indexes Options
</Directory>
```

The following is an excerpt from the Apache HTTP Server Version 1.3 documentation

### ErrorDocument

**Syntax:** ErrorDocument error-code document  
**Context:** server config, virtual host, directory, .htaccess  
**Status:** core  
**Override:** FileInfo  
**Compatibility:** The directory and .htaccess contexts are only available in Apache 1.1 and later.
Legal Directives III: Apache Modules

Apache is distributed with several modules. These modules may or may not be active within the Apache server with which you are working. The Core features will always be available.

For example, if the Rewrite Module (mod_rewrite) has not been activated, none of the Rewrite directives will be available to use.

Refer to the Status and Module lines in the documentation for each directive and to the documentation for the specific Apache installation you are using.
Webmaster Tools

- Site Icons (‘favicon.ico’)
- Web Robots
  - Link Checking
  - Search Robots
- Other Webmaster Tools
  - HTML/CSS Validation
  - Accessibility Compliance
  - Web Site Mirroring
  - Converting HTML to other formats
  - HTTP Server Performance
- Log Analysis

Site Icons

- favicon.ico at root of web site
- or “link” element in “head” element of XHTML/HTML document

MSIE uses http://www.somesite.com/favicon.ico for icons in the bookmark list.

Firefox uses favicon.ico or link element, rel="icon", in the location bar, bookmark list and tab display.

The code in the ‘head’ of the XHTML would look something like:

```
1. <link rel="icon" href="images/mozilla-16.png" type="image/png"/>
2. <link rel="shortcut icon" href="images/mozilla.ico" type="image/x-icon"/>
```
SEO: Search Engine Optimization

Make your site ready for search engines

- well-formed (and hopefully valid) HTML/XHTML
- use mark-up language for headings and lists
- titles that stand on their own
- "meta" keywords and description

An example using O'Reilly OnLamp.com

In "head" element of page:

```
<meta name="keywords" content="ONLamp.com,O'Reilly Network,oreillynet,
onlamp.com,lamp,lampp,linux,apache,mysql,perl,python,
php,linux,bsd,web development,server development reference,
technical information,open source" />
<meta name="description" content="Welcome to ONLamp.com, the high performance web development site from the O'Reilly Network offering comprehensive Lamp developer information and resources. O'Reilly Network's ONLamp site features original articles, news and commentary." />
```

Firefox as a Web Development Tool

- Web Developer Extension
- Firefox Extension - Live HTTP Headers
- Firefox Extension - Firebug
xurl and churl

**xurl** A simple Perl script that extract the links for a single page. Adapted from *The Perl Cookbook*.

**churl** A simple Perl script that will check the links for a single page. Adapted from *The Perl Cookbook*.
Page Weight

Page weight of http://www.harvard.edu/

Firefox Web Developer Tool Bar

Welcome to Harvard University - Mozilla Firefox

Device Information Console and Grid

Welcome to: Harvard University (Main)

Page Weight

Page weight of http://www.harvard.edu/

Firefox Web Developer Tool Bar

Welcome to Harvard University - Mozilla Firefox

Device Information Console and Grid

Welcome to: Harvard University (Main)

Page Weight

Page weight of http://www.harvard.edu/

Firefox Web Developer Tool Bar

Welcome to Harvard University - Mozilla Firefox

Device Information Console and Grid

Welcome to: Harvard University (Main)
timefetch examples

timefetch is in need of updating (does not parse CSS and will not get images referenced in CSS), but can still be a useful tool.

timefetch will show the actual download time (often not useful) and the total kilobytes downloaded (often useful). Warning: timefetch will not execute JavaScript, nor does it fetch images included by CSS.

Compare to:

Document Size
- http://isites.harvard.edu/icb/icb.do?keyword=k12622

- Documents (16 files): 21112
- Images (11 files): 3312
- Objects (6 files): 0
- Scripts (8 files): 16212
- Stylesheets (17 files): 6112
- Total: 30712
Web Robots

Robots, Spider, Crawlers

As they "spider" a site, the robots can perform various actions, such as:

- Gathering content for search engines or a website mirror
- Validating, checking, or processing content

Spidering Behavior: an example with Lynx

After lynx is done, here's a look at the files we have:

- The "lnkNNNNNNNN.dat" files contain the text dump of the pages lynx retrieved
- The "traverse.dat" files contain the list of link that lynx retrieved
- The "reject.dat" files contain a list of URLs that lynx did not fetch (due to the fact that they are outside the "realm" as specified on the command line).

```bash
minerva% lynx -traversal 
minerva% ls -l
```
Robots Exclusion Standard (RES)

- The Web Robots Pages
  http://www.robotstxt.org/wc/robots.html
- A Standard for Robot Exclusion
  http://www.robotstxt.org/wc/norobots.html
- The Web Robots FAQ

RES provides two mechanisms to instruct robots that visit your site:

1. `robots.txt` file
2. robots `meta` tag

Two directives:
- User-Agent
- Disallow

Note: `robots.txt` must lie at the root level of the server.

Examples of `robots.txt` files:
- http://www.fas.harvard.edu/robots.txt
- http://www.foxnews.com/robots.txt
- find them at a couple of your favorite web sites

Why won't the following `robots.txt` files do anything useful? (they aren't at the root level of server)

- http://www.people.fas.harvard.edu/~jharvard/robots.txt
- http://www.fas.harvard.edu/computing/robots.txt
robots.txt Examples

http://www.npr.org/robots.txt

1. Disallow all robots from certain areas:
2. User-agent: *
3. Disallow /cgi-bin
4. Disallow /content/
5. Disallow /*.xml
6. Disallow /*.asp
7. Disallow /*.ram
8. Disallow /*.ram
9. Disallow /*.js
10. Disallow /*.au
11. Disallow /stations/force/localization.php?
12. Disallow /rundowns/segment.php?

http://www.foxnews.com/robots.txt

1. User-agent: *
2. Disallow /printer_friendly_story
3. User-agent: fusionbot
4. User-agent: Googlebot
5. Disallow /printer_friendly_story
6. User-agent: Mediapartners-Google*
7. Disallow /printer_friendly_story
8. User-agent: Teoma
9. Disallow /printer_friendly_story
10. User-agent: yahoo-newscrawler
11. Disallow /printer_friendly_story
12. User-agent: Yahoo! Slurp
13. Disallow /printer_friendly_story
15. Disallow /printer_friendly_story
16. User-agent: gsa-crawler
17. Disallow /printer_friendly_story

Robots meta element

- name="robots"
- content
  - index or noindex
  - follow or nofollow

The Robots meta element can be used on a per document basis.

OK to index page; OK to follow links on page

```html
<meta name="robots" content="index,follow" />
```

OK to index page; Don't follow links on page

```html
<meta name="robots" content="index,nofollow" />
```

Don't index page; OK to follow links on page

```html
<meta name="robots" content="noindex,follow" />
```

Don't index page; Don't follow links on page

```html
<meta name="robots" content="noindex,nofollow" />
```
Link Checking Robots

- Check the links on a single page; or on an entire site.
- If following links, will do a get request, otherwise it should do a head request.

Examples of Link Checking Robots

- curl
- checklink
- webbot
- checkbot
- webcheck
- lynx
### checklink

**W3C Link Checker**
http://validator.w3.org/docs/checklink

- **Use Online**
  http://validator.w3.org/checklink
- **Use command line:**

```
Perl, Free
```

---

<table>
<thead>
<tr>
<th>view plain</th>
<th>print</th>
<th>?</th>
</tr>
</thead>
<tbody>
<tr>
<td>minerva% checklink URL</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Usage:**
```
checklink <options> <uris>
```

**Options:**
- `-s`/`--summary`: Result summary only.
- `-b`/`--broken`: Show only the broken links, not the redirects.
- `-d`/`--directory`: Hide directory redirects, for example
  `http://www.w3.org/TR -> http://www.w3.org/TR/`
- `-r`/`--recursive`: Check the documents linked from the first one.
- `-D`/`--depth n`: Check the documents linked from the first one to depth n (implies `--recursive`).
- `-l`/`--location uri`: Scope of the documents checked in recursive mode.
  By default, for example for
  `http://www.w3.org/TR/html4/Overview.html`
  it would be `http://www.w3.org/TR/HTML`.
- `-n`/`--noaccept-language`: Do not send an Accept-Language header.
- `-t`/`--timeout value`: Timeout for HTTP requests.
- `-y`/`--proxy proxy`: Specify an HTTP proxy server.
- `-V`/`--version`: Output version information.
- `-?`/`--help`: Show this message.

**Notes:**
- See "perldoc Net::FTP" for information about various environment variables affecting FTP connections and "perldoc Net::NNTP" for setting a default NNTP server for news: URIs.
- The `W3C_CHECKLINK_CFG` environment variable can be used to set the configuration file to use. See details in the full manual page, it can be displayed with:
  `perl doc /usr/local/bin/checklink`
- More documentation at:
  [http://www.w3.org/2000/07/checklink](http://www.w3.org/2000/07/checklink)
- Please send bug reports and comments to the www-validator mailing list:
  [www-validator@w3.org](mailto:www-validator@w3.org) (with 'checklink' in the subject)
- Archives are at:
  [http://lists.w3.org/Archives/Public/www-validator/](http://lists.w3.org/Archives/Public/www-validator/)
checklink

```
checklink -r -D 0 -s -b
```

webbot

webbot is part of the W3C Libwww package. [http://www.w3.org/Robot/](http://www.w3.org/Robot/)

```
webbot 5.4.0
using the W3C libwww library version 5.4.0.
See "http://www.w3.org/Robot/User/CommandLine" for help
See "http://www.w3.org/Robot/User/" for user information
See "http://www.w3.org/Robot/" for general information
Please send feedback to the <www-lib@w3.org> mailing list,
see "http://www.w3.org/Library/#Forums" for details
```
webbot example

```
webbot example

minerva% webbot -img 
> -depth 99 
> -prefix http://cscie12.dce.harvard.edu/lecture_notes/20060131/ 
> -include http://cscie12.dce.harvard.edu/lecture_notes/20060131/ 
> -404 404.log 
> -l clf.log 
> -referer referer.log 
> -reject reject.log 
> http://cscie12.dce.harvard.edu/lecture_notes/20060131/ 
...content removed... 
Robot....... Received element 0, attribute 5 with anchor 0x8073700 
Robot....... Found `http://cscie12.dce.harvard.edu/' - 
............ Already checked 
Robot....... Received element 0, attribute 5 with anchor 0x8073688 
Robot....... Found 
`http://cscie12.dce.harvard.edu/lecture_notes/20060131/slide1.html' -  
............ Already checked 
Robot....... done with http://cscie12.dce.harvard.edu/lecture_notes/20060131/slide0.html  
2 outstanding requests 
Robot....... done with http://cscie12.dce.harvard.edu/lecture_notes/20060131/images/veri 
1 outstanding request  
Robot....... done with http://cscie12.dce.harvard.edu/lecture_notes/20060131/images/KUSe 
everything is finished...

Accessed 62 documents in 2.61 seconds (23.79 requests pr sec) 
Did a GET on 53 document(s) and downloaded 182K bytes of document bodies (71396.8 
Did a HEAD on 9 document(s) with a total of 49K bytes 
Raw Log files: 
Logged 62 entries in general log file 'clf.log' 
Logged 63 entries in referer log file 'referer.log' 
Logged 54 entries in rejected log file 'reject.log' 
Logged 0 entries in not found log file '404.log'
```

checkbot

Checkbot
http://degraaff.org/checkbot/

Checkbot Example Output
Checkbot Example

<table>
<thead>
<tr>
<th>View</th>
<th>Plain</th>
<th>Print</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td><code>minerva% checkbot --help</code></td>
<td></td>
</tr>
<tr>
<td>2.</td>
<td>Checkbot 1.75 command line options:</td>
<td></td>
</tr>
<tr>
<td>3.</td>
<td><code>--cookies</code> Accept cookies from the server.</td>
<td></td>
</tr>
<tr>
<td>4.</td>
<td><code>--debug</code> Debugging mode: No pauses, stop after 25 links.</td>
<td></td>
</tr>
<tr>
<td>5.</td>
<td><code>--mailto address</code> Mail brief synopsis to address when done.</td>
<td></td>
</tr>
<tr>
<td>6.</td>
<td><code>--noproxy domains</code> Do not proxy requests to given domains.</td>
<td></td>
</tr>
<tr>
<td>7.</td>
<td><code>--verbose</code> Verbose mode: display many messages about progress.</td>
<td></td>
</tr>
<tr>
<td>8.</td>
<td><code>--url url</code> Start URL.</td>
<td></td>
</tr>
<tr>
<td>9.</td>
<td><code>--match match</code> Check pages only if URL matches 'match'.</td>
<td></td>
</tr>
<tr>
<td>10.</td>
<td><code>--exclude exclude</code> Exclude pages if the URL matches 'exclude'.</td>
<td></td>
</tr>
<tr>
<td>11.</td>
<td><code>--filter regexp</code> Run regexp on each URL found.</td>
<td></td>
</tr>
<tr>
<td>12.</td>
<td><code>--ignore ignore</code> Ignore URLs matching 'ignore'.</td>
<td></td>
</tr>
<tr>
<td>13.</td>
<td><code>--exclude exclude</code> Exclude pages if the URL matches 'exclude'.</td>
<td></td>
</tr>
<tr>
<td>14.</td>
<td><code>--proxy URL</code> URL of proxy server for HTTP and FTP requests.</td>
<td></td>
</tr>
<tr>
<td>15.</td>
<td><code>--sleep seconds</code> Sleep this many seconds between requests (default 0).</td>
<td></td>
</tr>
<tr>
<td>16.</td>
<td><code>--style url</code> Reference the style sheet at this URL.</td>
<td></td>
</tr>
<tr>
<td>17.</td>
<td><code>--timeout seconds</code> Timeout for HTTP requests in seconds (default 120).</td>
<td></td>
</tr>
<tr>
<td>18.</td>
<td><code>--interval seconds</code> Maximum time interval between updates (default 10800).</td>
<td></td>
</tr>
<tr>
<td>19.</td>
<td><code>--enable-virtual</code> Use only virtual names, not IP numbers for servers.</td>
<td></td>
</tr>
<tr>
<td>21.</td>
<td>Options <code>--match</code>, <code>--exclude</code>, and <code>--ignore</code> can take a perl regular expression.</td>
<td></td>
</tr>
<tr>
<td>22.</td>
<td>as their argument.</td>
<td></td>
</tr>
<tr>
<td>23.</td>
<td>Use <code>perldoc checkbot</code> for more verbose documentation.</td>
<td></td>
</tr>
<tr>
<td>25.</td>
<td>Mail bugs and problems: <a href="mailto:checkbot@degraaff.org">checkbot@degraaff.org</a></td>
<td></td>
</tr>
</tbody>
</table>

Results are given in an HTML page: checkbot.html
For the Programmer: Writing Your Own

- The Perl modules, LWP and WWW::Robot make writing robots almost trivial.
- Examples in Perl Cookbook, published by O'Reilly
- Perl and LWP by Sean Burke, published by O'Reilly

Other Webmaster Tools

- Checking HTML Pages
- Web Site Mirroring
- Document Version Control
- Monitor HTTP Server Performance
Checking HTML Pages

- W3C HTML Validation, http://validator.w3.org/
- W3C CSS Validation, http://jigsaw.w3.org/css-validator/
- WebXact (WAI and Section 508 Compliance), http://webxact.watchfire.com/
- Watchfire.

Web Site Mirroring

- GNU wget
  http://www.gnu.org/software/wget/wget.html
- w3mir
  http://langfeldt.net/w3mir/
**GNU wget**

This is a command-line tool for downloading files from the web. It supports HTTP and FTP protocols. Some of its features include:

- **Download large files in the background.**
- **Resume interrupted downloads.**
- **Download from multiple URLs on a single command.**
- **Download files from FTP sites.**
- **Support for HTTP authentication, cookies, and forms.**

**Usage:**

```
$ wget --help
```

**Example:**

```
$ wget http://example.com/file.txt
```

**Link:**

[GNU wget](http://www.gnu.org/software/wget/wget.html)

---

**w3mir**

W3mir is a all purpose HTTP copying and mirroring tool. The main focus of w3mir is to create and maintain a browseable copy of one, or several, remote WWW site(s). Used to the max w3mir can retrieve the contents of several related sites and leave the mirror browseable via a local web server, or from a filesystem, such as directly from a CDROM.

**Usage:**

```
w3mir
```

**Example:**

```
w3mir http://example.com/
w3mir
```

**Link:**

[langfeldt.net/w3mir/](http://langfeldt.net/w3mir/)
### HTTP Server Stress Test

- **ApacheBench (ab)** is part of the Apache HTTP Server Distribution
  

- **Apache JMeter**
  

---

#### Apache Bench (ab)

```
minerva% /usr/bin/ab -n 10000 -c 10 http://cscie12.dce.harvard.edu/lecture_notes/
```

1. Completed 1000 requests
2. Completed 2000 requests
3. Completed 3000 requests
4. Completed 4000 requests
5. Completed 5000 requests
6. Completed 6000 requests
7. Completed 7000 requests
8. Completed 8000 requests
9. Completed 9000 requests
10. Finished 10000 requests

Server Software:        Apache/2.0.51
Server Hostname:        cscie12.dce.harvard.edu
Server Port:            80
Document Path:          /lecture_notes/
Document Length:        1040 bytes
Concurrency Level:      10
Time taken for tests:   23.653070 seconds
Complete requests:      10000
Failed requests:        0
Write errors:           0
Total transferred:      12097254 bytes
HTML transferred:       10406240 bytes
Requests per second:    422.78 [#/sec] (mean)
Time per request:       23.653 [ms] (mean)
Time per request:       2.365 [ms] (mean, across all concurrent requests)
Transfer rate:          499.43 [Kbytes/sec] received

Connection Times (ms)
  min  mean [+/-sd] median   max
Connect:        1    8   4.2      9      39
Processing:     6   14   4.3     14      44
Waiting:        0    8   4.2      8      36
Total:         18   23   4.9     21      47

Percentage of the requests served within a certain time (ms)
 50%     21
66%     23
75%     22
80%     23
90%     31
95%     38
98%     39
99%     40
100%     47 (longest request)
### Apache Bench (ab)

```
<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>server software</td>
<td>Apache/2.0.49</td>
</tr>
<tr>
<td>2</td>
<td>server hostname</td>
<td>cscie12.dce.harvard.edu</td>
</tr>
<tr>
<td>3</td>
<td>server port</td>
<td>80</td>
</tr>
<tr>
<td>4</td>
<td>document path</td>
<td>/tools/webcube.cgi</td>
</tr>
<tr>
<td>5</td>
<td>document length</td>
<td>58163 bytes</td>
</tr>
<tr>
<td>6</td>
<td>concurrency level</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>time taken for tests</td>
<td>98.986587 seconds</td>
</tr>
<tr>
<td>8</td>
<td>complete requests</td>
<td>1000</td>
</tr>
<tr>
<td>9</td>
<td>failed requests</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>write errors</td>
<td>0</td>
</tr>
<tr>
<td>11</td>
<td>total transferred</td>
<td>58323402 bytes</td>
</tr>
<tr>
<td>12</td>
<td>HTML transferred</td>
<td>58171098 bytes</td>
</tr>
<tr>
<td>13</td>
<td>requests per second</td>
<td>10.10 [#/sec] (mean)</td>
</tr>
<tr>
<td>14</td>
<td>time per request</td>
<td>989.866 [ms] (mean)</td>
</tr>
<tr>
<td>15</td>
<td>transfer rate</td>
<td>575.39 [Kbytes/sec] received</td>
</tr>
</tbody>
</table>
```

### HTTP Server Monitoring

- **Nagios**
  - [http://www.nagios.org/](http://www.nagios.org/)
- **Cricket**

**Monitoring Apache with Cricket**

[Graph showing Apache performance metrics]
Web Logs

Common Tools
- Analog
- Analog + Report Magic
- AWStats
- WebTrends

HTTP Server Logs

- access log
- error log
- referer log (no longer common)
- user-agent log (no longer common)

error log:

[Thu Dec 2 11:26:08 1999] [notice] Apache/1.3.9 (Unix) configured -- resuming normal operation
[Thu Dec 2 11:27:19 1999] [notice] caught SIGTERM, shutting down
[Mon Dec 6 19:15:04 1999] [notice] Apache/1.3.9 (Unix) configured -- resuming normal operation
[Mon Dec 6 19:27:33 1999] [notice] caught SIGTERM, shutting down

access log:

is03.fas.harvard.edu -- [02/Dec/1999:11:26:42 -0500] "GET /server-status HTTP/1.0" 200 154
140.247.30.103 -- [02/Dec/1999:11:26:48 -0500] "GET / HTTP/1.0" 200 162
140.247.30.104 -- [02/Dec/1999:11:26:56 -0500] "GET /server-info HTTP/1.0" 200 456
140.247.30.104 -- [02/Dec/1999:11:26:56 -0500] "GET / HTTP/1.0" 200 162
is03.fas.harvard.edu -- [06/Dec/1999:19:16:58 -0500] "GET /server-status HTTP/1.0" 200 154
140.247.27.63 -- [06/Dec/1999:19:17:08 -0500] "GET / HTTP/1.1" 200 162
140.247.27.63 -- [06/Dec/1999:19:17:08 -0500] "GET / server-info HTTP/1.1" 200 162
is04.fas.harvard.edu -- [06/Dec/1999:19:18:32 -0500] "GET /server-status HTTP/1.0" 200 154
Data in Access Logs

http://www.apache.org/docs/mod/mod_log_config.html

Typical Data:
- Time
- IP address / Hostname
- Username (if under Authentication)
- Request
- User-Agent
- Referrer URL
- Response Status
- Bytes returned

Possible Data:
- The contents of a specified environment variable
- Filename
- The request protocol
- The contents of specified HTTP request headers
- The contents of specified HTTP response headers
- Remote logname (from identd, if supplied)
- The request method
- The canonical Port of the server serving the request.
- The process ID of the child that serviced the request.
- The query string
- First line of request
- The time taken to serve the request.
- The URL path requested.
- The canonical ServerName of the server serving the request.
- The server name according to the UseCanonicalName setting.

Log Formats

- Common Log Format (CLF)
  - host ident auth_user date request status bytes
- User-Agent Log
  - date user-agent
- Referrer Log
  - date referrer-url request-url
- Combined Log Format
  - host ident authuser date request status bytes referrer user-agent

Custom Log Formats in Apache
Combined Log Format

damhab.103.60.61 - - [14/Sep/1999:11:37:46 -0400] "GET /~cscie12/syllabus/workshops.html HTTP/1.0" 404 2

damhab.103.60.61 - - [14/Sep/1999:11:37:46 -0400] "GET /~cscie12/discussion/ HTTP/1.0" 404 2


vicfux.115.63.27 - - [14/Sep/1999:11:38:56 -0400] "GET /~cscie12/images/dce2.gif HTTP/1.0" 200 0

vicfux.115.63.27 - - [14/Sep/1999:11:38:58 -0400] "GET /~cscie12/images/dot.gif HTTP/1.0" 200 0


Web Server Logs: Two perspectives

- Server Administrator
- Content Provider
Web Server Logs: What we would like to know and what we can know

1. What is the busiest time?  
2. How long do they stay?  
3. How long did it take to fulfill a request?  
4. How many requests were there for a specific resource?  
5. Where are the users coming from?  
6. What browsers are people using?  
7. What pages have they been to?  
8. How many were looking versus buying?  
9. What requests resulted in errors (status 404, etc)?  
10. Where do the users go when they leave the site?  
11. Do they come back?

Complicating Issues

- HTTP is a stateless protocol
- Local Cache
- Proxy Cache
- Proxy Servers
- Shared Computers
Log Rotation

- approximately 200 to 250 bytes per line (request)
- For example, 1,000,000 requests per day (12 requests per second)
- log grows at 2.8 kb per second
- 238 Mb for 1,000,000 requests
- compressed (gzip'ed) logs are 7 to 10% of original size!

Tools for Log Analysis

- Analog
  http://www.analog.cx/
  Stephen Turner
  UNIX, Windows, MacOS, others
  Free!!
- Report Magic
- WebTrends Log Analyzer
  http://www.webtrends.com/