






Servidores Web



Mario Muñoz Organero
Departamento de Ingeniería Telemática
<http://www.it.uc3m.es/mario>



Servidores Web

- **Bibliografía:** “Servidor Apache al Descubierta”, Rich Bowen y Ken Coar. Biblioteca: L/S 004.738.52 Bow.
- **Temario**
 - El protocolo HTTP.
 - Diferentes Servidores Web: Apache
 - Instalación de Apache
 - Arranque y Parada del Servidor.
 - El fichero de configuración http.conf.
 - Configuración descentralizada.
 - Ámbitos de Configuración.
 - Ficheros de Log

Mario Muñoz Organero. Aplicaciones Avanzadas de Telemática 2

Servidores Web

■ Temario (Cont)

- Modificación del espacio de URIs
- Directorios de usuario
- Redirección de Peticiones
- Hosts Virtuales basados en Nombres y en IPs
- Contenido Dinámico: CGIs

El protocolo HTTP



Recordatorio sobre HTTP



- HyperText Transfer Protocol
 - Aunque no sólo transfiere hipertexto
- Arquitectura cliente / servidor
 - navegador / httpd
- Sobre TCP (puerto 80)
- Orientado a transacciones
 - petición / respuesta
 - una conexión por objeto transferido (1.0)
- Peticiones y respuestas ASCII

Recordatorio sobre HTTP



- Sin estado
 - cada transacción se trata independiente
 - simplicidad, menos recursos
 - solución: cookies, campos ocultos en formularios, parámetros en URL...
- Intercambio de metadatos (información relativa a un recurso pero que no es parte del recurso)
 - Las interacciones incluyen usualmente información relativa a los recursos (cabeceras de entidad)
 - Por ejemplo: tamaño, tipo de contenido, última modificación,...
- Versiones:
 - HTTP/0.9
 - HTTP/1.0 (RFC 1945).
 - HTTP/1.1 (RFC 2616)

HTTP/1.0



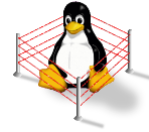
- Hereda funcionamiento de 0.9:
 - Una transacción (petición, respuesta) por conexión.
 - Petición y respuesta ASCII.
 - Líneas terminadas en <CR><LF>.
- Compatible hacia atrás con la versión 0.9:
 - HTTP-message = Simple-Request ; HTTP/0.9 messages
 - | Simple-Response
 - | Full-Request ; HTTP/1.0 messages
 - | Full-Response

HTTP/1.0



- Full-Request = Request-Line
 - *(General-Header
 - | Request-Header
 - | Entity-Header)
 - CRLF
 - [Entity-Body]
- Full-Response = Status-Line
 - *(General-Header
 - | Response-Header
 - | Entity-Header)
 - CRLF
 - [Entity-Body]

HTTP/1.1



- Multitud de nuevas cabeceras (de 16 en HTTP/1.0 a 46 en HTTP/1.1).
- Los servidores deben ser compatibles con peticiones HTTP/1.0.
- Deben soportar al menos los métodos GET y HEAD.
- Nuevos métodos de petición opcionales:
 - OPTIONS: para determinar capacidades del servidor.
 - TRACE: para trazar reenvío de HTTP a través de proxys, túneles...
 - CONNECT: reservado para usarse con proxys que puedan conmutar a túnel (por ejemplo SSL).
- Cambios más importantes:
 - Cabecera **Date**: obligatoria en respuestas.
 - Cabecera **Host**: obligatoria en peticiones.
 - Conexiones persistentes y cabecera **Connection: close**.
 - Respuesta **100 Continue**.
 - Datos troceados (*chunked*).

Mario Muñoz Organero.

Redes de Ordenadores

9



Diferentes Servidores Web: Apache

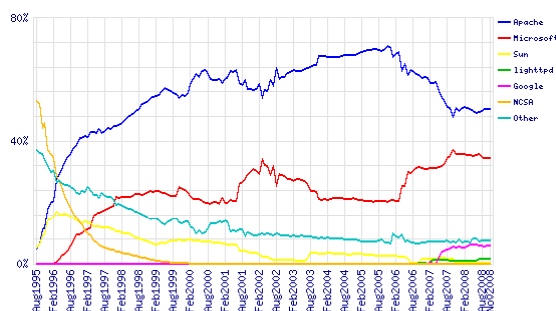


Servidores Web: Apache

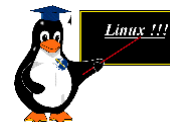
- El servidor Web es el encargado de responder las peticiones HTTP **enviadas por los clientes**.
- Es **una aplicación más** en el conjunto que ejecuta en un determinado equipo.
- Existen **múltiples servidores web**, cada uno con sus ventajas e inconvenientes.
- **Para nuestro estudio:** Servidor Apache. Es de libre distribución. Está presente en un número elevado de equipos servidores web.
- Toda la información referente a esta herramienta está disponible en www.apache.org.

Diferentes Servidores

- La cuota de mercado de Apache supera ampliamente al resto



http://news.netcraft.com/archives/web_server_survey.html



El Programa Servidor

- Una vez obtenido el programa, compilado e instalado, el binario se llama **httpd (o apache)**.
- Veremos que podemos arrancar Apache de varias formas entre las que destacan:
 - Con httpd
 - Dos opciones principales:
 - -d *serverroot*
 - Por defecto /usr/local/etc/httpd.
 - -f *config*
 - Fichero de configuración relativo al serverroot. Por defecto: conf/httpd.conf.
 - Con apachectl start
 - Mediante inetd

Mario Muñoz Organero.

Aplicaciones Avanzadas de Telemática

13



The Apache Software Foundation
<http://www.apache.org/>

Obtener e instalar Apache en Linux



Consiguiendo Apache

- Múltiples distribuciones de Linux traen ya Apache con la distribución:
 - Gentoo, Debian, Red Hat, S.u.S.E...
 - Para una instalación de Linux sin Apache, utilizar vuestro gestor de paquetes favorito para su instalación:
 - Con yum (fedora):
yum install httpd
 - Con apt-get (debian) para instalar Apache2 y PHP 4:
apt-get install apache2 libapache2-mod-php4

Compilando Apache




- Descargar el código fuente de apache.org:
 - <http://httpd.apache.org/download.cgi>
- 3 pasos:
 - ./configure --prefix=/usr/local/apache-2.0.48 --enable-so
 - make
 - sudo make install
- Si no ponemos opciones a configure se instala en /usr/local/apache2

Opciones adicionales de ./configure



- **--enable-auth-digest**
- --enable-echo
- **--enable-mem-cache**
- --enable-case-filter
- --enable-auth-ldap
- --enable-usertrack
- --enable-unique-id
- **--enable-proxy**
- --enable-info
- --enable-vhost-alias
- --enable-speling
- --enable-rewrite

Layout de ficheros en Apache

- Los directorios de instalación son:
 - bin/**
 - cgi-bin/**
 - error/
 - icons/
 - lib/
 - man/
 - modules/**
 - build/
 - conf/**
 - htdocs/**
 - include/
 - logs/**
 - manual/



Arrancar y Detener Apache



Arrancar y Detener Apache

- El arranque y parada del proceso servidor es **complejo** y por tanto se suele realizar a través de un script.
- El servidor mantiene **varios hilos de ejecución en paralelo** para servir las peticiones recibidas.
- El script **apachectl** que se incluye en la distribución del servidor, se utiliza para arrancar/parar el servidor con un conjunto de opciones.
- Opciones adicionales de apachectl:
 - **start/stop**: Arranca y para el servidor.
 - **restart/graceful**: Re-arranca los procesos leyendo de nuevo los ficheros de configuración. En el caso de **graceful** no pierde las conexiones abiertas.
 - **status/fullstatus**: Muestra el estado interno del servidor.
 - **configtest**: Únicamente comprueba la sintaxis del fichero de configuración e informa de posibles errores.

Mario Muñoz Organero. Aplicaciones Avanzadas de Telemática 20

Bajo inetd

- Inetd es un demonio en Linux que escucha varios puertos de red, les da ciertos servicios como el control de conexiones, y delega la ejecución a un segundo demonio dependiendo del servicio. La comunicación entre inetd y el demonio se hace por stdin y stdout.
- Ventaja: reutilización y eficiencia se servicios
- Para configurar Apache bajo inetd hay que poner en el httpd.conf (lo veremos ya mismo):
 - ServerType inetd
- Y en el /etc/inetd.conf:
 - `httpd stream tcp nowait root /usr/local/apache/sbin/httpd -f /usr/local/apache/etc/httpd.conf`
- Y en /etc/services:
 - `httpd 80/tcp httpd`

El fichero de configuración

httpd.conf



Configuración

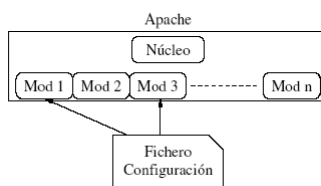
- Los servidores web tienen **infinidad** de aspectos de su ejecución que pueden ser configurados.
- Todas estas opciones son **imposibles** de incluir en la línea de comando.
- Se utilizan **ficheros de configuración** con diferentes componentes.
- En el caso de apache, la configuración del servidor se almacena en **httpd.conf**.
- Este fichero contiene **directivas de control** para el núcleo del servidor y para **los módulos adicionales**.

Fichero httpd.conf

- Fichero de texto que contiene un conjunto de **directivas**.
- Cada directiva representa **una orden** a aplicar en un contexto o ámbito determinado.
- El fichero de configuración puede ser **muy extenso**. La directiva **Include** se utiliza para fragmentar su contenido en ficheros que pueden ser incluidos.
- Las directivas se aplican dentro de un ámbito. Si una directiva no aparece dentro de un ámbito aplica a todo el servidor Apache:
 - Fuera de ámbito:
 - **Ejemplo:** Include conf.d/*.conf
 - Dentro de un ámbito: El ámbito se enmarca mediante una sintaxis **similar** a XML.
 - **Ejemplo:**
<Directory /usr/local/httpd/htdocs>
Options Indexes FollowSymLinks
</Directory>

Módulos de Funcionalidad

- En Apache, la funcionalidad está dividida en **módulos**.
- Cada módulo contiene un conjunto de funciones relativas a un **aspecto concreto del servidor**.
- El fichero binario **httpd** contiene **un conjunto de módulos que han sido compilados**.
- La funcionalidad de estos módulos puede ser **activada o desactivada** al arrancar el servidor.



Módulos de Funcionalidad

- Se pueden desarrollar módulos nuevos que **extiendan la funcionalidad del servidor**.
- Existe la posibilidad de que el servidor **cargue módulos de forma dinámica**.
- Mientras se ejecuta el fichero binario se **cargan porciones de código adicionales** que contiene más funcionalidad.

Configuración de Módulos

- Cada módulo tiene asignadas **sus propias directivas de configuración**.
- Para cargar un módulo se utiliza la directiva LoadModule.
- Para configurar los módulos dependiendo de su presencia se utiliza la directiva <IfModule> que permite interpretar su contenido si un módulo está instalado y activado.
- **Ejemplo:**

```
<IfModule mod_mime_magic.c>
MIMEMagicFile conf/magic
</IfModule>
```
- La lista de módulos de los que consta el servidor se pueden ver ejecutando httpd -l.

Directivas de Configuración

- Apache admite, en su versión actual, alrededor de **340** directivas diferentes.
- La principal dificultad de configuración del servidor es **la heterogeneidad de sus contenidos** y la necesidad de tener una granularidad muy fina **en el control de dichos contenidos**.
- Toda directiva que aparezca en el fichero de configuración afecta la **aplicación servidor** excepto si está incluida en la definición de un **ámbito**.
- Los ámbitos posibles que permite Apache son:

```
<Directory>      <DirectoryMatch>
<Files>         <FilesMatch>
<Location>     <LocationMatch>
<Proxy>        <ProxyMatch>
<IfDefine>     <IfModule>
<VirtualHost>
```
- Estas directivas limitan el ámbito de aplicación a un conjunto de **ficheros o URIs**.
- Estas declaraciones se pueden **anidar** lo que proporciona un control muy fino de los recursos del servidor.
- Existen reglas que estipulan **qué directivas** se pueden utilizar en qué contextos.



Configuración Descentralizada

- Conforme el tamaño de los datos de un servidor es más grande, **las tareas de configuración se complican.**
- Si la configuración reside **en un único fichero** su tamaño puede llegar a ser inmenso.
- Apache permite la configuración descentralizada mediante la presencia de **ficheros de configuración en los directorios de documentos.**
- Las directivas locales se almacenan en el fichero **.htaccess** (se puede cambiar este nombre con la directiva `AccessFileName`) en el propio directorio de datos.
- La sintaxis de estos ficheros es la misma que para **httpd.conf.**

Configuración Descentralizada

- Sólo un subconjunto de directivas pueden incluirse en el fichero .htaccess:
 - La directiva AllowOverride permite especificar en la configuración del httpd.conf qué directivas pueden aparecer en el .htaccess
 - Ej: AllowOverride Options
 - que permite poner en el .htaccess:
 - Options +ExecCGI
 - AllowOverride dentro de un ámbito de directorio:

```
<Directory "C:/Program Files/Apache Group/Apache/htdocs">
  AllowOverride None
</Directory>
```

Configuración Descentralizada

- **AllowOverride permite activar o desactivar 5 categorías de directivas:**
 - **AuthConfig**
 - Permite directivas de seguridad como AuthName, Satisfy, y Require
 - **FileInfo**
 - Controlan como procesar los ficheros
 - **Indexes**
 - Afecta a los listados e incluye directivas como IndexOptions, AddDescription, y DirectoryIndex.
 - **Limit**
 - Similar a AuthConfig pues las directivas que cubre son también de seguridad. Incluyen directivas Order, Allow, y Deny.
 - **Options**
 - Para permitir opciones como ContentDigest, XBitHack, y Options.

El fichero .htaccess

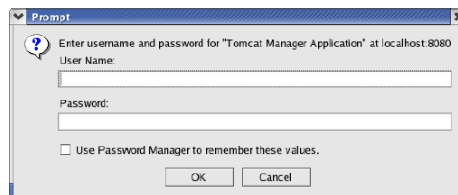
- **Ejemplo (si AllowOverride AuthConfig y Limit):**

```
AuthUserFile /www/it2/config/.htpasswd
AuthName "Autenticación de Usuario"
AuthType Basic
Order deny,allow
Deny from all
Require valid-user
Allow from 163.117.139
```

- Se protege el contenido del directorio con un **mecanismo de autenticación**.
- El fichero que contiene **los usuarios y sus claves** se especifica por la directiva AuthUserFile.
- Se permite acceder únicamente a usuarios **debidamente autenticados** y cuyas peticiones procedan de máquinas en la red **163.117.139.***

Ejemplo de Acceso con Autenticación

- Cuando **.htaccess** requiere autenticación, el cliente se encarga de ofrecer al usuario el interfaz para obtenerla.



- Una vez obtenidos los datos por el navegador, se envían al servidor **para su comprobación**, y en caso de ser correctos se envía el documento requerido.
- Los ficheros de autenticación es particular para ese directorio y **pueden estar en cualquier lugar del sistema de ficheros**

Ejemplo: permitiendo SSI

- SSI: Server Side Includes
 - Ej: `<!--#echo var="DATE_LOCAL" -->`
- AllowOverride Options y AllowOverride FileInfo
- Ejemplo **.htaccess**
 - Options +Includes
 - AddType text/html shtml
 - AddHandler server-parsed shtml

Ámbitos o contextos de configuración



Ámbitos o contextos de Configuración

- Las directivas de definición de ámbito `<IfModule>` y `<IfDefine>` se evalúan **únicamente cuando arranca el servidor**.
- **Ejemplos:**
`<IfDefine ClosedForNow>`
`Redirect / http://otherserver.example.com/`
`</IfDefine>`
- Las constantes se pueden definir al arrancar el servidor mediante la opción **-DClosedForNow**
- Las condiciones se pueden **negar** anteponiendo el prefijo "!".
- El resto de directivas de ámbito se evalúan **para cada petición que recibe el servidor**.

Ámbitos en el Sistema de Ficheros

- Las directivas `<Directory>` y `<Files>` delimitan su ámbito de aplicación a un conjunto de ficheros **en el sistema de ficheros visible por el servidor**.
- La directiva `<Directory>` afecta a todo el contenido **por debajo del directorio especificado**.
`<Directory /var/web/dir1>`
`Options +Indexes`
`DirectoryIndex index.html`
`</Directory>`
- Se aplica al acceso a todos aquellos directorios por debajo de `/var/web/dir1`. En este caso si la URL hace referencia a un directorio y no existe el fichero especificado por la directiva `DirectoryIndex` se mostrará el contenido del directorio automáticamente.
- La directiva `<Files>` afecta a los ficheros con el nombre dado, independientemente del directorio en el que se encuentren.
- Estas dos directivas se pueden combinar mediante su definición anidada:
`<Directory /var/web/dir1>`
`<Files private.html>`
`Order allow,deny`
`Deny from all`
`</Files>`
`</Directory>`
- Se impide el acceso a los ficheros con nombre `private.html` por debajo del directorio `/var/web/dir1`.

Ámbito en el espacio de URIs

- La jerarquía en una URI corresponde con el sistema de ficheros **excepto cuando se especifica un mapeo alternativo**.
 - DocumentRoot /usr/local/apache/htdocs
- Las URIs se interpretan **mediante las reglas en el fichero de configuración**, y si no se modifican, se accede al fichero pertinente.
- La directiva <Location> sirve para redireccionar una petición de una URI.

```
<Location /private>
Order Allow, deny
Deny from all
</Location>
```
- Hace que las URIs que comiencen por /private no sean servidas.

Expresiones Regulares en Definiciones Ámbitos

- Las directivas <Directory>, <Files> y <Location> tienen versiones similares que permiten la utilización de **expresiones regulares**.
- Contienen la palabra Match añadida al final del nombre.
- Permiten referirse a **un conjunto de recursos**.

```
<DirectoryMatch /home/*/public_html>
Options Indexes
</DirectoryMatch>
<FilesMatch \.(gif|jpe?g|png)$>
Order allow,deny
Deny from all
</FilesMatch>
```
- El primer ámbito permite el acceso al índice de los directorios con nombre public.html.
- El segundo ámbito prohíbe el acceso a los ficheros que con extensiones .png, .jpeg, .jpg o .gif

Orden de Procesado de Ámbitos

- Los ámbitos definidos se procesan en determinado orden de preferencia:
 - 1. <Directory> y ficheros .htaccess.
 - 2. <DirectoryMatch>.
 - 3. <Files> y <FilesMatch> simultáneamente.
 - 4. <Location> y <LocationMatch> simultáneamente.
- Ámbitos definidos posteriormente anulan a los anteriores excepto para <Directory> que se procesa en orden creciente de tamaño de directorio.

Configuración Global del Servidor

- Las siguientes directivas aplican al **comportamiento global del servidor**

Tipo	Directivas
Identificación	ServerName, ServerAdmin, ServerSignature, ServerTokens, UseCanonicalName
Ficheros Administración	CoreDumpDirectory, DocumentRoot, ErrorLog, LockFile, PidFile, ScoreBoardFile, ServerRoot
Límites	LimitRequestBody, LimitRequestFields, LimitRequestFieldSize, LimitRequestLine, RLimitCPU, RLimitMEM, RLimitNPROC, ThreadStackSize

- El primer tipo de directivas son para controlar la identidad del servidor de cara al exterior. Se puede ofrecer al exterior un nombre diferente al de la máquina que alberga el servidor.
- Las directivas de administración definen dónde se almacenan los datos que genera el servidor al funcionar.
- Las directivas de tipo límite restringen el uso de los recursos de la máquina para evitar problemas de seguridad y escalabilidad.



Ficheros de Log

- Todo servidor debe dejar constancia de la **actividad que realiza**.
- Estos ficheros no sólo se utilizan para diagnóstico de anomalías sino también para **obtener información sobre el tráfico que procesa el servidor**.
- Se recomienda no permitir el acceso a los directorios donde se almacenan los ficheros de log **excepto** al usuario con el que se ejecuta el servidor.
- Apache incluye directivas para la gestión de mensajes de error, y mensajes de acceso.
- Las directivas para controlar los mensajes de error son: <ErrorLog> y <LogLevel>.
- La directiva <ErrorLog> especifica el fichero en el que se depositan los mensajes de error producidos por el servidor.
- **Ejemplo:**
[Wed Oct 11 14:32:52 2000] [error] [client 127.0.0.1]
client denied by server configuration: /export/home/live/ap/htdocs/test
- La directiva <LogLevel> especifica el nivel de errores que se escriben en el fichero.
- El fichero de errores se puede controlar continuamente mediante el comando: tail -f error.log.

Ficheros de Log para Accesos

- Las directivas que controlan esta funcionalidad son: <CustomLog>, <LogFormat> y <SetEnvIf>.
- El servidor genera un mensaje **por cada petición HTTP que es procesada**.
- El formato del mensaje de log que se almacena es altamente configurable.
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log common
- La primera directiva define un formato de mensaje.
- La segunda directiva almacena los mensajes en ese formato en el fichero dado.
- **Ejemplo:**
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700]
"GET /logo.gif HTTP/1.0" 200 2326
- Este formato se conoce con el nombre de "Common Log Format" y existen un gran número de herramientas de **minería de datos** capaces de analizar estos ficheros.
- Estos ficheros han adquirido un valor estratégico **muy elevado** para las empresas.

Formato de Log de Acceso Combinado

- Además del formato CLF definido anteriormente, existe un segundo formato que se utiliza para accesos "referidos".
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \\\n\"%{User-agent}i\" combined
CustomLog log/access_log combined
- Con este formato se obtienen mensajes tales como:
127.0.0.1 - frank [10/Oct/2000:13:55:36 -0700]
"GET /apache pb.gif HTTP/1.0" 200 2326
"http://www.example.com/start.html"
"Mozilla/4.08 [en] (Win98; I ;Nav)"
- El campo Referer incluye la información que el cliente envía como URL que contiene la referencia al recurso pedido.
- El campo User-agent contiene la información que el cliente envía para identificarse.
- Alrededor de este tipo de logs se han organizado **modelos de negocio** referente a publicidad, así como estudios de porcentajes de accesos.

Otros tipos de Logs

- Se pueden utilizar múltiples instancias de la directiva CustomLog para clasificar diferentes mensajes.
LogFormat "%h %l %u %t \"%r\" %>s %b" common
CustomLog logs/access_log common
CustomLog logs/referer_log "%{Referer}i -> %U"
CustomLog logs/agent_log "%{User-agent}i"
- Debido al gran tamaño que pueden alcanzar los ficheros de log se necesita un **mecanismo de filtrado**.

Filtrado de los logs

- La directiva SetEnvIf permite definir una condición sobre los campos de una petición para posteriormente ser filtrada mediante la directiva CustomLog.
SetEnvIf Remote_Addr "127.0.0.1" dontlog
SetEnvIf Request_URI "/robots.txt\$" dontlog
CustomLog logs/access_log common env=!dontlog
- Las tres últimas directivas excluye aquellos mensajes de log que son producidos por accesos locales, o que pretenden acceder al fichero robots.txt.
- SetEnvIf admite:
 - Remote_Host – Nombre del cliente
 - Remote_Addr – IP del cliente
 - Request_Method – método usado en la petición (GET, POST...)
 - Request_Protocol – versión HTTP (e.g., "HTTP/0.9", "HTTP/1.1", etc.)
 - Request_URI- recurso solicitado (según aparece en la línea de petición)
- Se pueden usar expresiones regulares:
 - SetEnvIf Request_URI "\.gif\$" object_is_image=gif

Controlando el tamaño de los Ficheros de Log

- Los ficheros de log de un servidor crece a un ritmo de 1 megabyte por cada 10.000 peticiones.
- En servidores con tráfico moderado, estos ficheros deben ser comprimidos y almacenados en otras plataformas con cierta frecuencia.
- ¿Cómo se puede reemplazar los ficheros de log sin parar el servidor?
- Para este cometido se provee la opción **graceful** en el script de arranque parada.

```
mv access_log access_log.old
mv error_log error_log.old
apachectl graceful
sleep 600
gzip access_log.old error_log.old
```
- Tras rearrancar el servidor se debe esperar unos segundos para que las peticiones que estaban pendientes terminen y se escriban sus logs.

Modificación del espacio de URIs



Modificación del Espacio de URIs

- La correspondencia entre **URIs** y **documentos** puede ser modificada.
- Por defecto Apache traduce el sufijo de la URI a un **camino en el sistema de ficheros**.
- ¿Cómo se puede servir un documento **fuera** del sistema de ficheros del servidor?
- El sistema Unix permite la utilización de **enlaces simbólicos**.
- Apache **no interpreta estos enlaces** a no ser que se especifique mediante la directiva Options FollowSymLinks
- La directiva Alias fuente destino hace que toda URL que contenga la palabra fuente se reemplace por destino y el resultado se interprete como un camino a fichero.
- **Ejemplo:** Alias /docs /var/web. Al recibir el servidor la petición `http://www.example.com/docs/dir/file.html` internamente el servidor accede al fichero en `/var/web/dir/file.html`
- La directiva **AliasMatch** acepta una expresión regular para aplicar a la modificación.

Directorios de usuario



Directorios de Usuario

- En Unix, cada usuario tiene un directorio asociado considerado su **directorio raíz**.
- En Apache existe el mismo concepto pero aplicado al **servicio de documentos**.
- La directiva Userdir directorio define el directorio en el que buscar documentos cuando la URI tiene el formato `http://www.example.com/~user`
- Por razones de seguridad, **no se debe servir el directorio raíz de un usuario**.
- La notación `~` es **específica para Apache**.
- Esta notación se puede evitar mediante la utilización de **AliasMatch**.
`AliasMatch ~/upages/([a-zA-Z0-9]*)/?(.*) /home/$1/public_html/$2`

Redirección de Peticiones



Redirección de Peticiones

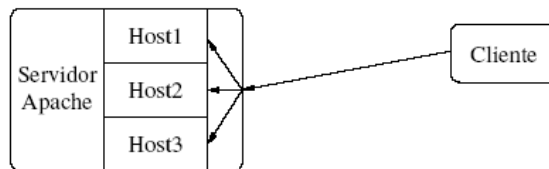
- A veces la URI recibida debe ser traducida a otra **que se sirve en otra URL**.
- La directiva **Redirect [status] uri1-path uri2** permite redirigir las peticiones de **uri1-path** a **uri2**.
- La nueva petición no tiene por qué ser servida **por el mismo servidor**.
- **Ejemplo:**
`Redirect permanent /viejo/
http://www.example.com/nuevo/`
- Toda petición que contenga el texto “/viejo” será redirigida a la URI obtenida reemplazando el texto por “http://www.example.com/nuevo/”
- De nuevo esta directiva cuenta con su versión **RedirectMatch** en la que se permite el uso de expresiones regulares.

Hosts Virtuales basados en Nombres y en IP



Hosts Virtuales

- El servidor web **responde a peticiones** mediante el envío de información interna.
- ¿Se pueden servir peticiones **de más de una máquina**?
- **Ejemplo:** Las direcciones **www.example.com** y **mail.example.com** deben ser gestionadas por el mismo servidor.
- La solución que ofrece apache está basada en el concepto de **host virtual**.
- El servidor responde a las peticiones recibidas no sólo por el servidor por defecto sino también **por cada uno de los hosts virtuales**.



Directivas para Hosts Virtuales

- A efectos de configuración, un host virtual se configura **exactamente igual que uno normal**.
- Las directivas de configuración se incluyen en el elemento **<VirtualHost>**
- La mayoría de directivas de configuración global se **pueden incluir** en el contexto de un host virtual.
- **Ejemplo:**

```
<VirtualHost *>
  ServerName www.domain.tld
  ServerAlias domain.tld *.domain.tld
  DocumentRoot /www/domain
</VirtualHost>
```
- Apache permite **dos modalidades de host virtuales:** basados en nombre y basados en dirección IP.

Host Virtuales Basados en Nombre

- Dos nombres de dominio **tienen idéntica dirección de IP**, pero deben ser tratados como dos **hosts virtuales**.
- Apache canaliza las peticiones **dependiendo del nombre de la máquina en la URL**.
- Cuando se utilizan hosts virtuales, el propio host por defecto de apache **pasa a ser un host virtual**.
- **Ventajas:** Se ahorran direcciones IP para hosts que únicamente se utilizan para referenciar documentos en URIs.
- **Inconveniente:** Ciertas aplicaciones de seguridad, identifican los recursos por su dirección de IP y no por su nombre.

Configuración de un Host Virtual Basado en Nombre

- Para utilizar hosts virtuales basados en nombre es preciso especificar **la dirección de IP y puerto** por el que se reciben las peticiones.
- La directiva **NameVirtualHost ip** se utiliza para definir este parámetro.
- A continuación se debe crear un bloque con la directiva **<VirtualHost>** para cada uno de los hosts que se quieran albergar.
- El argumento de **<VirtualHost>** debe ser **idéntico** al incluido en la directiva **NameVirtualHost**.
- La definición del host virtual debe incluir **como mínimo** las directivas **ServerName** y **DocumentRoot** especificando, respectivamente el nombre y localización de los documentos del servidor.
- Como el host por defecto desaparece, la creación de un host virtual requiere la definición de **dos** elementos VirtualHost.

Ejemplo de Virtual Host Basado en Nombre

- Supongamos que se quieren servir documentos para los hosts: `www.dominio.com` y `www.otrodominio.com`.

```
NameVirtualHost *
<VirtualHost *>
  ServerName www.domain.com
  ServerAlias domain.com *.domain.com
  DocumentRoot /www/domain
</VirtualHost>
<VirtualHost *>
  ServerName www.otherdomain.com
  DocumentRoot /www/otherdomain
</VirtualHost>
```
- El parámetro "*" en la primera directiva indica que se deben procesar todas las peticiones recibidas en el servidor (puede tener más de una dirección de IP).
- El primer host virtual sirve todas las peticiones recibidas en los dominios acabados en **domain.com**.
- El segundo host virtual procesa sólo aquellas peticiones dirigidas a **www.otherdomain.com**.
- La directiva **ServerAlias** permite que el servidor sea referido mediante más de un nombre.
- Esta configuración depende de que el **DNS tenga la información correcta**.

Múltiples sitios con Diferentes Puertos

- Supongamos que tenemos **múltiples dominios redirigidos a la misma IP**.
- Se quiere servir diferente información por diferentes puertos.
- La directiva **Listen** permite considerar más de un puerto.

```
Listen 80
Listen 8080
NameVirtualHost 167.10.20.30:80
NameVirtualHost 167.10.20.30:8080
<VirtualHost 167.10.20.30:80>
  ServerName www.servidor1.com
  DocumentRoot /home/servidor1/html
  ErrorLog /logs/httpd/error_log1
  TransferLog /logs/httpd/access_log1
</VirtualHost>
<VirtualHost 167.10.20.30:8080>
  ServerName www.servidor2.com
  DocumentRoot /home/servidor2/html
  ErrorLog /logs/httpd/error_log2
  TransferLog /logs/httpd/access_log2
</VirtualHost>
<VirtualHost 167.10.20.30:80>
  ServerName www.servidor3.com
  DocumentRoot /home/servidor3/html
  ErrorLog /logs/httpd/error_log3
  TransferLog /logs/httpd/access_log3
</VirtualHost>
<VirtualHost 167.10.20.30:8080>
  ServerName www.servidor4.com
  DocumentRoot /home/servidor4/html
  ErrorLog /logs/httpd/error_log4
  TransferLog /logs/httpd/access_log4
</VirtualHost>
```

Virtual Hosts Basados en Direcciones IP

- Otra posible configuración de un servidor es que disponga **de más de una dirección de IP**.
- Esto implica que se dispone de **más de una tarjeta de red** o que el sistema operativo **gestiona más de una interfaz al exterior**.
- Herramientas tales como **ifconfig** o técnicas como **ip aliases** permiten implementar esta configuración.
- Los paquetes se dirigen a una de las posibles direcciones de IP, pero **se procesan por la misma máquina**.
- **Ventajas:** A todos los efectos, al exterior se ofrecen varias direcciones de IP.
- **Inconvenientes:** Si el número de hosts es muy elevado, se puede tener problemas con el número de IPs disponibles.

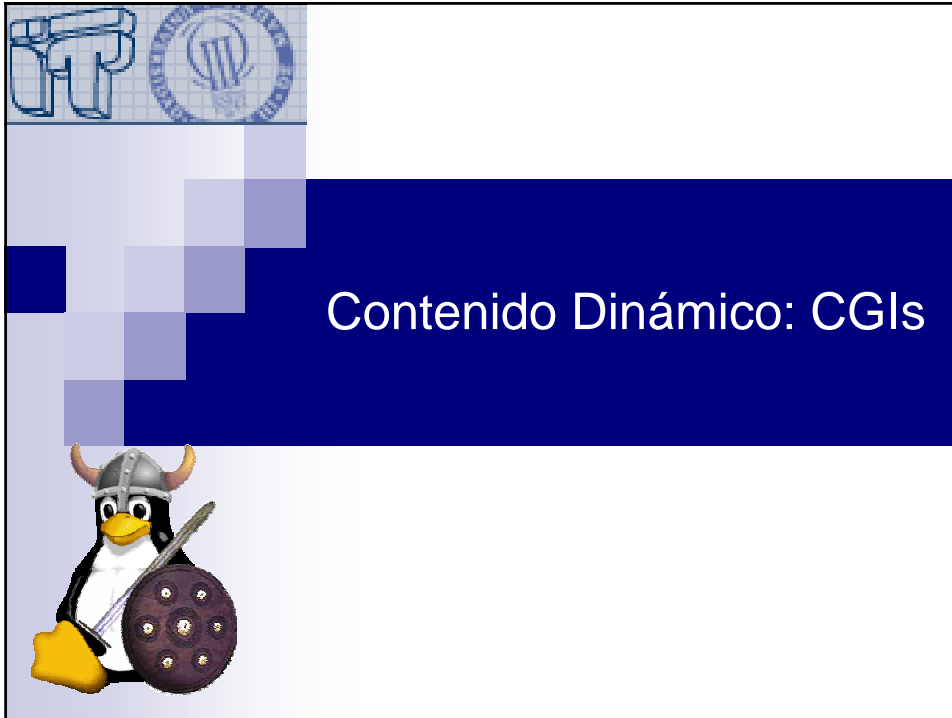
Configuración de Hosts Virtuales Basados en IPs

- En esta configuración no es preciso utilizar la directiva **NameVirtualHost**.

```
<VirtualHost 192.168.1.150>
  ServerAdmin wb@mimaquina.com
  DocumentRoot /home/servidor1/html
  ServerName www.servidor1.com
  ErrorLog /logs/httpd/error_log1
  TransferLog /logs/httpd/access_log1
</VirtualHost>

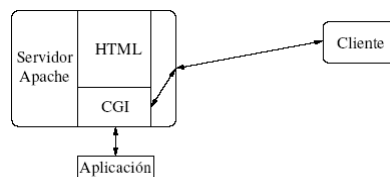
<VirtualHost 192.168.1.151>
  ServerAdmin wb@mimaquina.com
  DocumentRoot /home/servidor2/html
  ServerName www.servidor2.com
  ErrorLog /logs/httpd/error_log2
  TransferLog /logs/httpd/access_log2
</VirtualHost>
```

- La directiva **<VirtualHost>** puede especificar un **nombre de máquina** en lugar de una dirección de IP.
- Se recomienda utilizar una dirección de IP e incluir la directiva **ServerName** para garantizar el funcionamiento independientemente del servicio de **DNS**.
- Apache permite combinar de forma arbitraria **hosts virtuales basados en nombres, basados en IPs y discriminación por número de puerto**.



Contenido Dinámico: CGI

- CGI (Common Gateway Interface) es una especificación para la **comunicación de datos genéricos** entre el servidor Web y **una aplicación genérica** que se ejecuta en la misma máquina.
- El objetivo es permitir que el servidor responda con contenido **no previamente creado, sino obtenido dinámicamente**.
- La generación de contenidos dinámicos, y en general el procesado de datos tiene **serias repercusiones en la seguridad del servidor**.
- Las directivas de configuración de CGI están diseñadas para **reducir** el ámbito de ejecución de programas lo máximo posible.



Configuración de CGI

- La directiva de configuración para **permitir** la ejecución de programas CGI es **ScriptAlias nombre dir**.
- **Ejemplo:** `ScriptAlias /cgi-bin/ /usr/local/apache/cgi-bin/`
- Toda URI que contenga el texto **/cgi-bin/** se redirige a los programas contenidos en el directorio dado.
- Los programas de procesamiento de peticiones se suelen almacenar en lugares **externos al sistema de ficheros servido**.
- Además de esta directiva, la opción **ExecCGI** permite dar este privilegio a un determinado directorio.
- **Ejemplo:**

```
<Directory /usr/local/apache/htdocs/somedir>  
Options +ExecCGI  
</Directory>
```

Configuración de CGI

- La directiva `AddHandler` permite especificar las extensiones asociadas a los CGI:
 - `AddHandler cgi-script cgi pl`

Ejemplo 1 CGI

- Usemos sh:

```
#!/bin/sh
echo Content-type: text/html
echo
echo
echo "<HTML>"
echo "<HEAD>"
echo "</HEAD>"
echo "<BODY>"
echo "<H2>Users logged in are:</H2>"
echo "<PRE>"
who
echo "</PRE>"
echo "</BODY>"
echo "</HTML>"
```

Ejemplo 2 CGI (I)

- Formulario:

```
<form          action="http://our-server/cgi-bin/mailer"
  method="post">
<dl>
<dt> Your Email <dd> <input name="email" size="50">
<dt> Filename <dd> <input name="file" size="50">
</dl>
<input type="submit" value="Submit">
</form>
```

Ejemplo 2 CGI (I)

- El script:

```
#!/bin/sh
eval `proccgi.sh $*`
mail $FORM_email < $FORM_file
echo Content-type: text/plain
echo
echo done
```