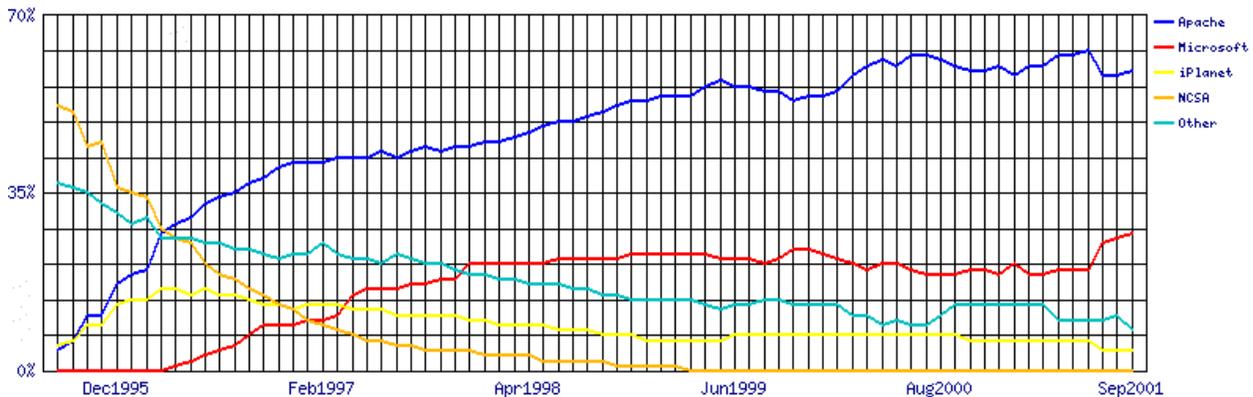


SERVIDOR WEB APACHE

Aileen Morrison
Conectiva Linux Chile
Aileen.Morrison@conectiva.cl

INTRODUCCIÓN

Apache es el servidor Web más utilizado hoy en día, superando todas las versiones comerciales que existen. Esto queda demostrado según estudios realizados por Netcraft, que han determinado que cerca del 60% de los sitios activos están soportados por Apache.



Apache deriva de un servidor HTTP desarrollado por Rob McCool de la NCSA (National Center for Supercomputing Applications). Posteriormente, tomando como base el HTTP de NCSA se desarrollaron extensiones y solucionaron bugs detectados teniendo como objetivo construir un servidor más rápido, robusto y eficiente. Fue así que la primera versión oficial de Apache fue liberada en abril de 1995.

Hoy, con un enorme equipo de voluntarios al rededor del mundo se han logrado estos objetivos logrando superar a grandes compañías como Microsoft y Netscape.

CÓMO FUNCIONA APACHE

En Linux, el proceso httpd corre como demonio que se ejecuta de forma continua en background escuchando peticiones de los clientes.

Si el Puerto especificado en el archivo de configuración es el puerto 80 o cualquier otro por debajo de 1024, es necesario tener privilegios de root para poder ejecutar Apache. Una vez que el servidor ha arrancado y completado algunas actividades, se crearan varios procesos *hijo* que harán el trabajo de escuchar y responder las peticiones de los clientes. El proceso principal httpd continua corriendo como usuario root, pero los procesos hijo se ejecutan como un usuario con menos privilegios.

Lo primero que hace httpd cuando es invocado es leer el archivo de configuración httpd.conf. La localización de este archivo se define en el momento de la compilación. En Conectiva Linux se encuentra en /etc/httpd/conf/httpd.conf, sin embargo, es posible definir una nueva ubicación al momento de la ejecución utilizando la opción -f en la línea de comandos de la siguiente forma:

```
/usr/sbin/httpd -f /usr/local/apache/conf/httpd.conf
```

Como alternativa a ejecutar httpd directamente, puede ser utilizado un script para controlar el proceso de inicialización del demonio con comandos simples como httpd start y httpd stop.

Si se quiere inicializar el servidor httpd al momento de subir el sistema, se deberá agregar una llamada al httpd desde los archivos de inicialización, *rc.local* o un archivo en el directorio *rc.N*. Con esto se arrancará Apache como root. Es importante asegurarse que el servidor está configurado correctamente y ofrece niveles de seguridad adecuados, de lo contrario se transformara en un punto vulnerable de la red.

CARACTERÍSTICAS DE APACHE

Apache es un servidor web flexible, rápido y eficiente, continuamente actualizado y adaptado a los nuevos protocolos (HTTP/1.1). Entre sus características destacan:

- Multiplataforma, ha sido desarrollado para plataforma WinXX, Netware, Unis y Linux
- Su desarrollo ha sido de acuerdo al protocolo HTTP/1.1 normalizado por el W3C (WWW Consortium)
- Modular, puede ser adaptado a diferentes entornos y necesidades, con los diferentes módulos de apoyo que proporciona, y con la API de programación de módulos, para el desarrollo de módulos específicos
- Open Source
- Extensible, gracias a ser modular se han desarrollado diversas extensiones entre las que destaca PHP, un lenguaje de programación del lado del servidor

EL PROTOCOLO HTTP

El Protocolo de Transferencia de HiperTexto (Hypertext Transfer Protocol) es un sencillo protocolo cliente-servidor que articula los intercambios de información entre los clientes Web y los servidores HTTP. La especificación completa del protocolo HTTP/1.0 está descrita en el RFC 1945, el cual fue propuesto por Tim Berners-Lee, atendiendo a las necesidades de un sistema global de distribución de información como el *World Wide Web*.

Desde el punto de vista de las comunicaciones, está soportado sobre los servicios de conexión TCP/IP, y funciona de la misma forma que el resto de los servicios comunes de los entornos UNIX: un proceso servidor escucha en un puerto de comunicaciones TCP (por defecto el 80), y espera las solicitudes de conexión de los clientes Web. Una vez que se establece la conexión, el protocolo TCP se encarga de mantener la comunicación y garantizar un intercambio de datos libre de errores.

HTTP se basa en operaciones sencillas de solicitud/respuesta. Un cliente establece una conexión con un servidor y envía un mensaje con los datos de la solicitud. El servidor responde con un mensaje similar, que contiene el estado de la operación y su posible resultado. Todas las operaciones pueden adjuntar un objeto o recurso sobre el que actúan, como por ejemplo, un documento HTML, imágenes, sonido o una aplicación CGI, las que se conocen por medio de su URL.

Los recursos u objetos que actúan como entrada o salida de un comando HTTP están clasificados por su descripción MIME. De esta forma, el protocolo puede intercambiar cualquier tipo de dato, sin preocuparse de su contenido. La transferencia se realiza en modo binario, byte a byte, y la identificación MIME permitirá que el receptor trate adecuadamente los datos.

Las principales características del protocolo HTTP son:

- Toda la comunicación entre los clientes y servidores se realiza a partir de caracteres de 8 bits. De esta forma, se puede transmitir cualquier tipo de documento: texto, binario, etc., respetando su formato original.
- Permite la transferencia de objetos multimedia. El contenido de cada objeto intercambiado es identificado por su clasificación MIME.
- Existen tres verbos básicos que un cliente puede utilizar para dialogar con el servidor:
 - GET** : para recoger un objeto
 - POST** : para enviar información al servidor
 - HEAD** : para solicitar las características de un objeto como la fecha de modificación de un documento HTML
- Cada operación HTTP implica una conexión con el servidor, que es liberada al término de la misma. Es decir, en una operación se puede recoger un único objeto.
- No mantiene estado. Cada petición de un cliente a un servidor no es influida por las transacciones anteriores. El servidor trata cada petición como una operación totalmente independiente del resto.
- Cada objeto al que se aplican los verbos del protocolo está identificado a través de la información de situación del final de la URL.

HTTP se diseñó específicamente para el World Wide Web: es un protocolo rápido y sencillo que permite la transferencia de múltiples tipos de información de forma eficiente y rápida. Se puede comparar, por ejemplo, con FTP, que es también un protocolo de transferencia de archivos, pero tiene un conjunto muy amplio de comandos, y no se integra muy bien en las transferencias multimedia.

Etapas de una transacción HTTP

Para profundizar más en el funcionamiento de HTTP, veremos primero un caso particular de una transacción http. Cada vez que un cliente realiza una petición a un servidor, se ejecutan los siguientes pasos:

1. Un usuario solicita una URL.
2. El cliente Web decodifica la URL, separando sus diferentes partes. Así identifica el protocolo de acceso, la dirección IP del servidor, el posible puerto opcional (el valor por defecto es 80) y el objeto requerido del servidor.
3. Se abre una conexión TCP/IP con el servidor, llamando al puerto TCP correspondiente.
4. Se realiza la petición. Para ello, se envía el comando necesario (GET, POST, HEAD,...), la dirección del objeto requerido (el contenido de la URL que sigue a la dirección del servidor), la

versión del protocolo HTTP y un conjunto variable de información, que incluye datos sobre las capacidades del browser.

5. El servidor devuelve la respuesta al cliente. Consiste en un código de estado y el tipo de dato MIME de la información de retorno, seguido de la propia información.
6. Se cierra la conexión TCP.

Este proceso se repite en cada acceso al servidor HTTP. Por ejemplo, si se recoge un documento HTML en cuyo interior están insertas cuatro imágenes, el proceso anterior se repite cinco veces, una para el documento HTML y cuatro para las imágenes.

Veamos el proceso completo con un ejemplo:

1. Desde un cliente se solicita la URL <http://www.conectiva.cl/index.html>
2. Se abre una conexión TCP/IP con el puerto 80 del sistema www.unican.es.
3. El cliente realiza la solicitud, enviando algo similar a esto:

```
GET /index.html HTTP/1.0           Operación solicitada+objeto+versión de HTTP
Accept: text/plain                 Lista de tipos MIME que acepta o entiende
Accept: text/html                  El cliente
Accept: audio/*
Accept: video/mpeg

. . . . .
Accept: */*                         Indica que acepta otros posibles tipos MIME
User-Agent: Mozilla/3.0 (WinNT; I) Información sobre el tipo de cliente

Línea en blanco, indica el final de la petición
```

4. El servidor responde con la siguiente información:

```
HTTP/1.0 200 OK Status de la operación; en este caso, correcto
Date: Monday, 8-Oct-2001 18:00:00 Fecha de la operación
Server: NCSA 1.4 Tipo y versión del servidor
MIME-version: 1.0 Versión de MIME que maneja
Content-type: text/html Definición MIME del tipo de datos a devolver
Content-length: 254 Longitud de los datos que siguen
Last-modified: 8-Oct-2001:30:00 Fecha de modificación de los datos

Línea en blanco

<HTML> Comienzo de los datos
<HEAD><TITLE>Recursos de Linux en

    Conectiva</TITLE></HEAD>
<BODY>
. . . . .
. . . . .
</HTML>
```

5. Se cierra la conexión.

Estructura de los mensajes HTTP

El diálogo con los servidores HTTP se establece a través de mensajes formados por líneas de texto, cada una de las cuales contiene los diferentes comandos y opciones del protocolo. Sólo existen dos tipos de mensajes, uno para realizar peticiones y otro para devolver la correspondiente respuesta. La estructura general de los dos tipos de mensajes se puede ver en la siguiente tabla:

Mensaje de solicitud	Mensaje de respuesta
Comando HTTP + parámetros	Resultado de la solicitud
Cabeceras del requerimiento (línea en blanco)	Cabeceras de la respuesta (línea en blanco)
Información opcional	Información opcional

La primera línea del mensaje de solicitud contiene el comando que se solicita al servidor HTTP, mientras que en la respuesta contiene el resultado de la operación, un código numérico que permite conocer el éxito o fracaso de la operación. Después aparece, para ambos tipos de mensajes, un conjunto de cabeceras (unas obligatorias y otras opcionales), que condicionan y matizan el funcionamiento del protocolo.

La separación entre cada línea del mensaje se realiza con la secuencia CR-LF como fin de línea. El final de las cabeceras se indica con una línea en blanco, tras la cual se pueden incluir los datos transportados por el protocolo, por ejemplo, el documento HTML que devuelve un servidor o el contenido de un formulario que envía un cliente.

El conocer y utilizar los mensajes http es necesario cuando:

- Se diseñan módulos CGI, ya que es preciso construir una respuesta similar a las que el servidor HTTP proporciona al cliente.
- Para diseñar aplicaciones independientes que soliciten información a un servidor (automatizar la recuperación de documentos o construir robots de búsqueda) se debe construir una cabecera HTTP con la información de la petición al servidor.

Comandos del protocolo

Los comandos o verbos de HTTP representan las diferentes operaciones que se pueden solicitar a un servidor HTTP. El formato general de un comando es:

<i>Nombre del comando</i>	<i>Objeto sobre el que se aplica</i>	<i>Versión de HTTP utilizada</i>
-------------------------------	--	--------------------------------------

Cada comando actúa sobre un objeto del servidor, normalmente un archivo o aplicación, que se toma de la URL de activación. La última parte de esta URL, que representa la dirección de un objeto dentro de un servidor HTTP, es el parámetro sobre el que se aplica el comando. Se compone de una serie de nombres de directorios y archivos.

El estándar HTTP/1.0 recoge únicamente tres comandos, que representan las operaciones de recepción y envío de información y chequeo de estado:

GET

Se utiliza para recoger cualquier tipo de información del servidor. Se utiliza siempre que se pulsa sobre un enlace o se teclea directamente a una URL. Como resultado, el servidor HTTP envía el documento correspondiente a la URL seleccionada, o bien activa un módulo CGI, que generará a su vez la información de retorno.

HEAD

Solicita información sobre un objeto: tamaño, tipo, fecha de modificación... Es utilizado por los servidores proxy, para conocer cuándo es necesario actualizar la copia que se mantiene de un archivo.

POST

Sirve para enviar información al servidor, como por ejemplo, los datos contenidos en un formulario. El servidor pasará esta información a un proceso encargado de su tratamiento.

Un cliente Web selecciona automáticamente los comandos HTTP necesarios para recoger la información requerida por el usuario. Así, ante la activación de un enlace, siempre se ejecuta una operación GET para recoger el documento correspondiente. El envío del contenido de un formulario utiliza GET o POST, en función del atributo de <FORM METHOD="...">. Además, si el cliente Web tiene un caché de páginas recientemente visitadas, puede utilizar HEAD para comprobar la última fecha de modificación de un archivo, antes de traer una nueva copia del mismo.

En la versión HTTP/1.1 se agregan otras funcionalidades que se pueden utilizar, por ejemplo, para editar las páginas de un servidor Web trabajando en forma remota.

PUT

Actualiza información sobre un objeto del servidor. Es similar a POST, pero en este caso, la información enviada al servidor debe ser almacenada en la URL que acompaña al comando. Así se puede actualizar el contenido de un documento.

DELETE

Elimina el documento especificado del servidor.

LINK

Crea una relación entre documentos.

UNLINK

Elimina una relación existente entre documentos del servidor.

Las cabeceras

Son un conjunto de variables que se incluyen en los mensajes HTTP, para modificar su comportamiento o incluir información de interés. En función de su nombre, pueden aparecer en los requerimientos de un cliente, en las respuestas del servidor o en ambos tipos de mensajes. El formato general de una cabecera es:

Nombre de la variable : Cadena ASCII con su valor

Los nombres de variables se pueden escribir con cualquier combinación de mayúsculas y minúsculas. Además, se debe incluir un espacio en blanco entre el signo : y su valor. En caso de que el valor de una variable ocupe varias líneas, éstas deberán comenzar, al menos, con un espacio en blanco o un tabulador.

Cabeceras comunes para peticiones y respuestas

Content-Type

Descripción MIME de la información contenida en el mensaje. Es la referencia que utilizan las aplicaciones Web para dar el tratamiento correcto a los datos que reciben.

Content-Length

Longitud en bytes de los datos enviados, expresado en base decimal.

Content-Encoding

Formato de codificación de los datos enviados en el mensaje. Sirve, por ejemplo, para enviar datos comprimidos (x-gzip o x-compress) o encriptados.

Date

Fecha local de la operación. Las fechas deben incluir la zona horaria en que reside el sistema que genera la operación. Por ejemplo: Sunday, 16-Sep-2001 12:21:22 GMT+01. No existe un formato único en las fechas; incluso es posible encontrar casos en los que no se dispone de la zona horaria correspondiente, con los problemas de sincronización que esto produce. Los formatos de fecha a emplear están recogidos en los RFC 1036 y 1123.

Pragma

Permite incluir información variada relacionada con el protocolo HTTP en el requerimiento o respuesta que se está realizando. Por ejemplo, un cliente envía un Pragma: no-cache para informar de que desea una copia nueva del recurso especificado.

Cabeceras sólo para peticiones del cliente

Accept

Campo opcional que contiene una lista de tipos MIME aceptados por el cliente. Se pueden utilizar * para indicar rangos de tipos de datos; tipo/* indica todos los subtipos de un determinado medio, mientras que */* representa a cualquier tipo de dato disponible.

Authorization

Clave de acceso que envía un cliente para acceder a un recurso de uso protegido o limitado. La información incluye el formato de autorización empleado, seguido de la clave de acceso.

From

Campo opcional que contiene la dirección de correo electrónico del usuario del cliente Web que realiza el acceso.

If-Modified-Since

Permite realizar operaciones GET condicionales, en función de si la fecha de modificación del objeto requerido es anterior o posterior a la fecha proporcionada. Puede ser utilizada por los sistemas de almacenamiento temporal de páginas. Es equivalente a realizar un HEAD seguido de un GET normal.

Referer

Contiene la URL del documento desde donde se ha activado este enlace. De esta forma, un servidor puede informar al creador de ese documento de cambios o actualizaciones en los enlaces que contiene. No todos los clientes lo envían.

User-agent

Cadena que identifica el tipo y versión del cliente que realiza la petición. Por ejemplo, los browsers de Netscape envían cadenas del tipo User-Agent: Mozilla/3.0 (WinNT; I)

Cabeceras sólo para respuestas del servidor HTTP

Allow

Informa de los comandos HTTP opcionales que se pueden aplicar sobre el objeto al que se refiere esta respuesta. Por ejemplo, Allow: GET, POST.

Expires

Fecha de expiración del objeto enviado. Los sistemas de cache deben descartar las posibles copias del objeto pasada esta fecha. Por ejemplo, Expires: Thu, 11 Oct 2001 00:00:00 GMT+1. No todos los sistemas lo envían. Puede cambiarse utilizando un <META EXPIRES> en el encabezado de cada documento.

Last-modified

Fecha local de modificación del objeto devuelto. Se puede corresponder con la fecha de modificación de un archivo en disco, o, para información generada dinámicamente desde una base de datos, con la fecha de modificación del registro de datos correspondiente.

Location

Informa sobre la dirección exacta del recurso al que se ha accedido. Cuando el servidor proporciona un código de respuesta de la serie **3xx** este parámetro contiene la URL necesaria para accesos posteriores a este recurso.

Server

Cadena que identifica el tipo y versión del servidor HTTP. Por ejemplo, Server: NCSA 1.5.

WWW-Authenticate

Cuando se accede a un recurso protegido o de acceso restringido, el servidor devuelve un código de estado 401, y utiliza este campo para informar de los modelos de autenticación válidos para acceder a este recurso.

Códigos de estado del servidor

Ante cada transacción con un servidor HTTP, éste devuelve un código numérico que informa sobre el resultado de la operación, como primera línea del mensaje de respuesta. Estos códigos aparecen en algunos casos en la pantalla del cliente, cuando se produce un error. El formato de la línea de estado es:

<i>Versión de protocolo HTTP utilizada</i>	<i>Código numérico de estado (tres dígitos)</i>	<i>Descripción del código numérico</i>
---	--	---

Existen cinco categorías de mensajes de estado, organizadas por el primer dígito del código numérico de la respuesta:

- | | |
|------------|---|
| 1xx | mensajes informativos |
| 2xx | mensajes asociados con operaciones realizadas correctamente. |
| 3xx | mensajes de redirección, que informan de operaciones complementarias que se deben realizar para finalizar la operación. |
| 4xx | errores del cliente; el requerimiento contiene algún error, o no puede ser realizado. |
| 5xx | errores del servidor, que no ha podido llevar a cabo una solicitud. |

Los más comunes se presentan a continuación:

Código	Comentario	Descripción
200	OK	Operación realizada satisfactoriamente.
201	Created	La operación ha sido realizada correctamente, y como resultado se ha creado un nuevo objeto, cuya URL de acceso se proporciona en el cuerpo de la respuesta. Este nuevo objeto ya está disponible. Puede ser utilizado en sistemas de edición de documentos.
202	Accepted	La operación ha sido realizada correctamente, y como resultado se ha creado un nuevo objeto, cuya URL de acceso se proporciona en el cuerpo de la respuesta. El nuevo objeto no está disponible por el momento. En el cuerpo de la respuesta se debe informar sobre la disponibilidad de la información.
204	No Content	La operación ha sido aceptada, pero no ha producido ningún resultado de interés. El cliente no deberá modificar el documento que está mostrando en este momento.
301	Moved Permanently	El objeto al que se accede ha sido movido a otro lugar de forma permanente. El servidor proporciona, además, la nueva URL en la variable Location de la respuesta. Algunos browsers acceden automáticamente a la nueva URL. En caso de tener capacidad, el cliente puede actualizar la URL incorrecta, por ejemplo, en la agenda de <i>bookmarks</i> .
302	Moved Temporarily	El objeto al que se accede ha sido movido a otro lugar de forma temporal. El servidor proporciona, además, la nueva URL en la variable Location de la respuesta. Algunos browsers acceden automáticamente a la nueva URL. El cliente no debe modificar ninguna de las referencias a la URL errónea
304	Not Modified	Cuando se hace un GET condicional, y el documento no ha sido modificado, se devuelve este código de estado.
400	Bad Request	La petición tiene un error de sintaxis y no es entendida por el servidor.
401	Unauthorized	La petición requiere una autorización especial, que normalmente consiste en un nombre y clave que el servidor verificará. El campo WWW-Authenticate informa de los protocolos de autenticación aceptados para este recurso.
403	Forbidden	Está prohibido el acceso a este recurso. No es posible utilizar una clave para modificar la protección.
404	Not Found	La URL solicitada no existe.
500	Internal Server Error	El servidor ha tenido un error interno, y no puede continuar con el procesamiento.
501	Not Implemented	El servidor no tiene capacidad, por su diseño interno, para llevar a cabo el requerimiento del cliente.

502	Bad Gateway	El servidor, que está actuando como proxy o gateway, ha encontrado un error al acceder al recurso que había solicitado el cliente.
503	Service Unavailable	El servidor está actualmente deshabilitado, y no es capaz de atender el requerimiento.

Magic Cookies

Las 'cookies' son pequeños archivos de texto que intercambian entre si los clientes y servidores HTTP, para solucionar la falta de información de estado entre dos transacciones. Fueron introducidas por Netscape, y han sido estandarizadas en el RFC 2109.

La primera vez que un usuario accede a un determinado documento de un servidor, éste proporciona una cookie que contiene datos que relacionarán posteriores operaciones. El cliente almacena la cookie en su sistema para usarla después. En los futuros accesos a este servidor, el browser podrá proporcionar la cookie original, que servirá de nexo entre este acceso y los anteriores. Todo este proceso se realiza automáticamente, sin intervención del usuario. El siguiente ejemplo muestra una cookie generada por la revista Rolling Stone.

```
EGSOFT_ID
191.46.211.13-655193640.29148285
www.rollingstone.com/
0
2867435528
30124157
4206779936
291478284
```

La información dentro de una cookie se organiza a partir de líneas de texto. El campo más interesante es la tercera línea. El cliente Web la compara cada URL a la que accede; si se produce una concordancia entre ambas, se enviará la cookie correspondiente. Es decir, una cookie asociada a www.rollingstone.com/shop sólo se enviará con accesos por debajo de la sección /shop, mientras que la cookie del ejemplo servirá para todo el servidor Rolling Stone. Las cookies pueden tener asociada una fecha de expiración, a partir de la cual el cliente Web tratará de obtener una nueva versión.

¿Para qué se pueden utilizar? Su aplicación más inmediata son los sistemas de comercio electrónico, ya que necesitan relacionar el contenido de un pedido con el cliente que lo ha solicitado. Sin cookies, la solución más sencilla es incluir dentro de los documentos HTML el contenido de las operaciones o pedidos previos, a través de variables ocultas. Con ellas es posible relacionar los sucesivos accesos al sistema con su origen y simplificar la gestión de la aplicación Web.

Otro uso muy interesante son los sistemas personalizados de recepción de información, en los que es posible construir una página a medida, con información procedente de fuentes muy diversas (ver por ejemplo my.yahoo.com/ o www.snap.com). En el primer acceso al sistema, se procede al registro; a partir de él, se genera una cookie que recibe el cliente. En accesos sucesivos, el cliente enviará la cookie, y el servidor podrá generar una página personalizada con las preferencias del usuario.

Por último, algunas organizaciones emplean las cookies para realizar un seguimiento de los accesos a sus servidores WWW, identificando las páginas más visitadas, la manera en que se pasa de una a otra sección, etc.

Por defecto, los clientes Web reciben y envían cookies sin solicitar confirmación al usuario, pero es posible recibir avisos de la recepción de cookies, para rechazarlas en caso de no ser de nuestro agrado.

¿Por qué rechazar las cookies?. Bueno, se pueden utilizar para seguir una pista del tipo de información que buscamos en Internet, y ofrecer información comercial en función de ello. Sin embargo, algunos servicios Web como tiendas on-line dependen de ellas para operar correctamente.

Las cookies tienen un tiempo de vida, que hace que sean descartadas pasado un cierto tiempo de actividad. Algunas expiran al salir del cliente Web, pero otras cookies tienen una duración mayor, por lo que el cliente Web las almacena en un archivo. Netscape utiliza el archivo cookies.txt de su directorio de instalación, mientras que Microsoft almacena cada uno en un archivo independiente del caché de páginas.

CONFIGURACIÓN DEL SERVIDOR

Los archivos de configuración de Apache se encuentran en el directorio `/etc/httpd/conf` y el script de inicialización se encuentra en `/etc/rc.d/init.d/httpd`, aun cuando esta ubicación puede variar dependiendo de la distribución de Linux utilizada. Dentro del directorio `/etc/httpd/conf` se encuentran los siguientes archivos:

- `httpd.conf` : principal de archivo de configuración de Apache
- `srm.conf` : archivo para definición del espacio de nombres que los usuarios ven desde el servidor web. En este archivo también se especifica donde se encuentran los cgi-bin, los iconos, el tipo de documento por defecto.
- `access.conf` : archivo de control de accesos
- `mime.types` : archivo de definición de los tipos MIME que son enviados al cliente en función de la extensión del archivo

Configuración del archivo `/etc/httpd/conf/httpd.conf`

El archivo "httpd.conf" es el archivo de configuración principal del servidor Apache. Existen muchas opciones para realizar diferentes configuraciones según nuestras necesidades.

La sintaxis de los archivos de configuración de Apache permite utilizar una directiva por línea. En las directivas dentro de los archivos de configuración no se hace distinción entre mayúsculas y minúsculas.

Las líneas que comiencen con el carácter "#" se consideran comentarios y son ignoradas. **No** se pueden incluir comentarios en una línea, después de una directiva de configuración. Los espacios y líneas en blanco antes de una directiva de configuración se ignoran, de manera que se puede dejar una sangría en las directivas para mayor claridad.

Para realizar un chequeo de la sintaxis de los archivos de configuración sin tener que reiniciar el servidor, se puede utilizar la opción `-t` de la línea de comandos.

A continuación se presenta un ejemplo del archivo `httpd.conf` con las configuraciones mínimas que se utilizan en la configuración de un servidor.

```
### Section 1: Global Environment
#
ServerType standalone
ServerRoot "/etc/httpd"
PidFile /var/run/httpd.pid
ResourceConfig /dev/null
AccessConfig /dev/null
Timeout 300
KeepAlive On
```

```

MaxKeepAliveRequests 0
KeepAliveTimeout 15
MinSpareServers 16
MaxSpareServers 64
StartServers 16
MaxClients 512
MaxRequestsPerChild 100000

### Section 2: 'Main' server configuration
#
Port 80
User www
Group www
ServerAdmin admin@midominio.cl
ServerName servidor.midominio.cl
DocumentRoot "/home/httpd/html"

<Directory />
    Options None
    AllowOverride None
    Order deny,allow
    Deny from all
</Directory>

<Directory "/home/httpd/html">
    Options None
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

<Files .pl>
    Options None
    AllowOverride None
    Order deny,allow
    Deny from all
</Files>

<IfModule mod_dir.c>
    DirectoryIndex index.html index.php index.php3 default.html index.cgi
</IfModule>

#<IfModule mod_include.c>
#Include conf/mmap.conf
#</IfModule>

UseCanonicalName On

<IfModule mod_mime.c>
    TypesConfig /etc/httpd/conf/mime.types
</IfModule>

DefaultType text/plain
HostnameLookups Off
ErrorLog /var/log/httpd/error_log
LogLevel warn
LogFormat "%h %l %u %t \"%r\" %>s %b \"%{Referer}i\" \"%{User-Agent}i\"" combined
SetEnvIf Request_URI \.gif$ gif-image
CustomLog /var/log/httpd/access_log combined env=!gif-image
ServerSignature Off

<IfModule mod_alias.c>
ScriptAlias /cgi-bin/ "/home/httpd/cgi-bin/"

```

```

<Directory "/home/httpd/cgi-bin">
    AllowOverride None
    Options None
    Order allow,deny
    Allow from all
</Directory>
</IfModule>

<IfModule mod_mime.c>
AddEncoding x-compress Z
AddEncoding x-gzip gz tgz

AddType application/x-tar .tgz
</IfModule>

ErrorDocument 500 "The server made a boo boo."
ErrorDocument 404 http://192.168.1.1/error.htm
ErrorDocument 403 "Access Forbidden -- Go away."

<IfModule mod_setenvif.c>
BrowserMatch "Mozilla/2" nokeepalive
BrowserMatch "MSIE 4\.0b2;" nokeepalive downgrade-1.0 force-response-1.0
BrowserMatch "RealPlayer 4\.0" force-response-1.0
BrowserMatch "Java/1\.0" force-response-1.0
BrowserMatch "JDK/1\.0" force-response-1.0
</IfModule>

```

Esta configuración le dice al servidor lo siguiente:

ServerType standalone

La opción "ServerType" define como operara en el sistema. Es posible ejecutar apache desde el servidor inetd o standalone para obtener mejor desempeño y velocidad.

ServerRoot "/etc/httpd"

La opción "ServerRoot" determina el directorio en el que estarán los archivos de configuración del servidor, los archivos de log y los de error.

PidFile /var/run/httpd.pid

La opción "PidFile" determina la ubicación del archivo donde el servidor grabara su número de proceso (PID) cuando este se inicialice. Esta opción es necesaria sólo cuando se configura Apache como standalone.

ResourceConfig /dev/null

La opción "ResourceConfig" define la ubicación del antiguo archivo "srm.conf" que Apache lee después de leer el archivo "httpd.conf". Cuando se define la localización como "/dev/null", Apache permite incluir el contenido de ese archivo en el archivo "httpd.conf" y de esa manera se maneja con mayor facilidad los parámetros de configuración.

AccessConfig /dev/null

La opción "AccessConfig" define la ubicación del antiguo archivo "access.conf" que Apache lee después de leer el archivo "srm.conf". Cuando se define la localización como "/dev/null", Apache permite incluir el contenido de ese archivo en el archivo "httpd.conf" y de esa manera tener un solo archivo de configuración que modificar.

Timeout 300

La opción "Timeout" especifica la cantidad de tiempo que Apache esperara por una solicitud GET, POST, PUT y ACKs en las transmisiones.

KeepAlive On

La opción "KeepAlive", si se define como "On", especifica que se habilitan las conexiones persistentes (HTTP/1.1) en el servidor Web. Para un mejor desempeño es recomendable definir esta opción en "On" y permitir mas de una solicitud por conexión.

MaxKeepAliveRequests 0

La opción "MaxKeepAliveRequests" determina el número de solicitudes permitidas por conexión cuando la opción anterior "KeepAlive" es definida como "On". Cuando el valor de esta opción se define en "0", entonces no existe limite en el número de solicitudes permitidas en el servidor.

KeepAliveTimeout 15

La opción "KeepAliveTimeout" determina cuánto tiempo, en segundos, esperara Apache por una siguiente solicitud antes de cerrar la conexión. El valor de "15" segundos es una buena opción para obtener un buen desempeño del servidor.

MinSpareServers 16

La opción "MinSpareServers" define el número mínimo de procesos hijo que será creado. Para una buena operación el valor de "16" es recomendado por varios benchmarks en Internet.

MaxSpareServers 64

La opción "MaxSpareServers" define el número máximo de procesos hijo que serán creados. Para una buena operación el valor de "64" es recomendado por varios benchmarks en Internet.

StartServers 16

La opción "StartServers" define el número de servidores (procesos) que serán creados cuando el servidor inicie. Para una buena operación el valor de "16" es recomendado por varios benchmarks en Internet.

MaxClients 512

La opción "MaxClients" define el número de solicitudes simultaneas que serán soportadas por Apache. Para una buena operación el valor debe ser relativamente alto, ya que una vez superado, ningún cliente más podrá acceder a la información.

MaxRequestsPerChild 100000

La opción "MaxRequestsPerChild" define el número de solicitudes que un proceso servidor podrá manejar. Esta también es un parámetro importante para lograra un mejor desempeño del servidor Apache.

Port 80

Puerto en el cual el servidor escuchará las solicitudes de los clientes. Un usuario sin privilegios solo puede utilizar puertos mayores que 1024.

User www

La opción "User" define el UID con que el servidor Apache se ejecutará. Es importante crear un usuario con privilegios mínimos en el sistema.

Group www

La opción "Group" define el GID con que el servidor Apache se ejecutará. Es importante crear un grupo con privilegios mínimos en el sistema.

ServerAdmin admin@midominio.cl

Dirección de correo de la persona responsable de la administración del servidor.

DocumentRoot /home/httpd/html

Directorio a partir del cual se publicarán los documentos.

DirectoryIndex index.htm index.html index.php index.php3 default.html index.cgi

La opción "DirectoryIndex" determina el archivo que Apache publicará como la página inicial.

HostnameLookups Off

La opción "HostnameLookups", si se define como "Off", determina que será deshabilitada las consultas al DNS. Se recomienda definirla como "Off" para disminuir el tráfico en la red y mejorar el desempeño del servidor Apache.

Virtual Host

Para habilitar la existencia de más de un sitio Web para ser publicado por el servidor Apache, es necesario modificar la configuración existente en la sección Virtual Host del archivo /etc/httpd/conf/httpd.conf

```
NameVirtualHost 192.168.117.X

VirtualHost 192.168.117.X>
    ServerAdmin webmaster@midominio.cl
    DocumentRoot /home/httpd/html/midominio
    ServerName www.midominio.cl
    ErrorLog logs/midominio.cl-error_log
    CustomLog logs/midominio.cl-access_log common
</VirtualHost>
```

El archivo mime.types

- **MIME.TYPES**

mime.types refleja la asociación entre tipos de documento y extensiones de archivo

El servidor Web debe anunciar cada objeto que transfiere con el tipo MIME adecuado: en algún momento deberá establecer la naturaleza del objeto.

El archivo *mime.types* establece la relación entre extensión de archivo y tipo MIME.

Es posible tener varias extensiones para un mismo tipo MIME. Adicionalmente, si se emplea el módulo `mod_mime_magic`, Apache intentará averiguar el tipo de los elementos de manera más elaborada.

```
application/EDI-Consent          image/cgm
application/EDI-X12              image/g3fax
application/EDIFACT              image/gif gif
application/activemessage        image/ief ief
...                               image/jpeg jpeg jpg jpe
application/octet-stream bin dms lha lzh
exe                               image/naplps
class                            image/png png
...                               ...
application/oda oda              text/css css
application/pdf pdf              text/directory
application/postscript ai eps ps  text/enriched
...                               text/html html htm
audio/32kadpcm                   text/plain asc txt
audio/basic au snd               ...
audio/midi mid midi kar          text/x-setext etx
audio/mpeg mpga mp2 mp3          text/xml xml
audio/vnd.qcelp                  video/mpeg mpeg mpg mpe
...                               video/quicktime qt mov
```

Archivos Instalados para el Servidor Apache en Conectiva Linux 7.0

```
/etc/httpd
/etc/httpd/conf
/etc/httpd/conf/access.conf
/etc/httpd/conf/apache-cnc.cnf
/etc/httpd/conf/httpd.conf
/etc/httpd/conf/magic
/etc/httpd/conf/srm.conf
/etc/httpd/conf/ssl.crt
/etc/httpd/conf/ssl.crt/ca-bundle.crt
/etc/httpd/conf/ssl.crt/server.crt
/etc/httpd/conf/ssl.csr
/etc/httpd/conf/ssl.csr/server.csr
/etc/httpd/conf/ssl.key
/etc/httpd/conf/ssl.key/server.key
/etc/httpd/logs
/etc/httpd/modules
/etc/logrotate.d/apache
/etc/rc.d/init.d/httpd
/home/httpd
/home/httpd/cgi-bin
/home/httpd/html
/home/httpd/html/english.html
/home/httpd/html/espanol.html
/home/httpd/html/imgs
/home/httpd/html/imgs/apache_pb.gif
/home/httpd/html/imgs/logo-cnc.gif
/home/httpd/html/imgs/mod_ssl_sb.gif
/home/httpd/html/imgs/openssl.gif
/home/httpd/html/index.html
/home/httpd/html/wap
/home/httpd/html/wap/index.wml
/home/httpd/icons
/home/httpd/icons/README
/home/httpd/icons/a.gif
/home/httpd/icons/alert.black.gif
/home/httpd/icons/alert.red.gif
/home/httpd/icons/apache_pb.gif
/home/httpd/icons/back.gif
/home/httpd/icons/ball.gray.gif
/home/httpd/icons/ball.red.gif
/home/httpd/icons/binary.gif
/home/httpd/icons/binhex.gif
/home/httpd/icons/blank.gif
/home/httpd/icons/bomb.gif
/home/httpd/icons/box1.gif
/home/httpd/icons/box2.gif
/home/httpd/icons/broken.gif
/home/httpd/icons/burst.gif
/home/httpd/icons/c.gif
/home/httpd/icons/comp.blue.gif
/home/httpd/icons/comp.gray.gif
/home/httpd/icons/compressed.gif
/home/httpd/icons/continued.gif
/home/httpd/icons/dir.gif
/home/httpd/icons/down.gif
/home/httpd/icons/dvi.gif
/home/httpd/icons/f.gif
/home/httpd/icons/folder.gif
/home/httpd/icons/folder.open.gif
/home/httpd/icons/folder.sec.gif
/home/httpd/icons/forward.gif
/home/httpd/icons/generic.gif
/home/httpd/icons/generic.red.gif
/home/httpd/icons/generic.sec.gif
/home/httpd/icons/hand.right.gif
/home/httpd/icons/hand.up.gif
/home/httpd/icons/icon.sheet.gif
/home/httpd/icons/image1.gif
/home/httpd/icons/image2.gif
/home/httpd/icons/image3.gif
/home/httpd/icons/index.gif
/home/httpd/icons/layout.gif
/home/httpd/icons/left.gif
/home/httpd/icons/link.gif
/home/httpd/icons/movie.gif
/home/httpd/icons/p.gif
/home/httpd/icons/patch.gif
/home/httpd/icons/pdf.gif
/home/httpd/icons/pie0.gif
/home/httpd/icons/pie1.gif
/home/httpd/icons/pie2.gif
/home/httpd/icons/pie3.gif
/home/httpd/icons/pie4.gif
/home/httpd/icons/pie5.gif
/home/httpd/icons/pie6.gif
/home/httpd/icons/pie7.gif
/home/httpd/icons/pie8.gif
/home/httpd/icons/portal.gif
/home/httpd/icons/ps.gif
/home/httpd/icons/quill.gif
/home/httpd/icons/right.gif
/home/httpd/icons/screw1.gif
/home/httpd/icons/screw2.gif
/home/httpd/icons/script.gif
/home/httpd/icons/small
/home/httpd/icons/small/README.txt
/home/httpd/icons/small/back.gif
/home/httpd/icons/small/binary.gif
/home/httpd/icons/small/binhex.gif
/home/httpd/icons/small/blank.gif
/home/httpd/icons/small/broken.gif
/home/httpd/icons/small/burst.gif
/home/httpd/icons/small/comp1.gif
/home/httpd/icons/small/comp2.gif
/home/httpd/icons/small/compressed.gif
/home/httpd/icons/small/continued.gif
/home/httpd/icons/small/dir.gif
/home/httpd/icons/small/dir2.gif
/home/httpd/icons/small/doc.gif
/home/httpd/icons/small/forward.gif
/home/httpd/icons/small/generic.gif
/home/httpd/icons/small/generic2.gif
/home/httpd/icons/small/generic3.gif
/home/httpd/icons/small/image.gif
/home/httpd/icons/small/image2.gif
/home/httpd/icons/small/index.gif
```

```

/home/httpd/icons/small/key.gif
/home/httpd/icons/small/movie.gif
/home/httpd/icons/small/patch.gif
/home/httpd/icons/small/ps.gif
/home/httpd/icons/small/rainbow.gif
/home/httpd/icons/small/sound.gif
/home/httpd/icons/small/sound2.gif
/home/httpd/icons/small/tar.gif
/home/httpd/icons/small/text.gif
/home/httpd/icons/small/transfer.gif
/home/httpd/icons/small/unknown.gif
/home/httpd/icons/small/uu.gif
/home/httpd/icons/sound1.gif
/home/httpd/icons/sound2.gif
/home/httpd/icons/sphere1.gif
/home/httpd/icons/sphere2.gif
/home/httpd/icons/tar.gif
/home/httpd/icons/tex.gif
/home/httpd/icons/text.gif
/home/httpd/icons/transfer.gif
/home/httpd/icons/unknown.gif
/home/httpd/icons/up.gif
/home/httpd/icons/uu.gif
/home/httpd/icons/uuencoded.gif
/home/httpd/icons/world1.gif
/home/httpd/icons/world2.gif
/usr/bin/dbmmanage
/usr/bin/htdigest
/usr/bin/htpasswd
/usr/bin/make-hash-symlinks
/usr/bin/make-test-certificate
/usr/bin/thawte-create
/usr/lib/apache
/usr/lib/apache/httpd.exp
/usr/lib/apache/libproxy.so
/usr/lib/apache/libssl.so
/usr/lib/apache/mod_access.so
/usr/lib/apache/mod_actions.so
/usr/lib/apache/mod_alias.so
/usr/lib/apache/mod_asis.so
/usr/lib/apache/mod_auth.so
/usr/lib/apache/mod_auth_anon.so
/usr/lib/apache/mod_auth_db.so
/usr/lib/apache/mod_auth_dbm.so
/usr/lib/apache/mod_autoindex.so
/usr/lib/apache/mod_cern_meta.so
/usr/lib/apache/mod_cgi.so
/usr/lib/apache/mod_define.so
/usr/lib/apache/mod_digest.so
/usr/lib/apache/mod_dir.so
/usr/lib/apache/mod_env.so
/usr/lib/apache/mod_example.so
/usr/lib/apache/mod_expires.so
/usr/lib/apache/mod_headers.so
/usr/lib/apache/mod_imap.so
/usr/lib/apache/mod_include.so
/usr/lib/apache/mod_info.so
/usr/lib/apache/mod_log_agent.so
/usr/lib/apache/mod_log_config.so
/usr/lib/apache/mod_log_referer.so
/usr/lib/apache/mod_mime.so
/usr/lib/apache/mod_mime_magic.so
/usr/lib/apache/mod_mmap_static.so
/usr/lib/apache/mod_negotiation.so
/usr/lib/apache/mod_rewrite.so
/usr/lib/apache/mod_setenvif.so
/usr/lib/apache/mod_speling.so
/usr/lib/apache/mod_status.so
/usr/lib/apache/mod_unique_id.so
/usr/lib/apache/mod_userdir.so
/usr/lib/apache/mod_usertrack.so
/usr/lib/apache/mod_vhost_alias.so
/usr/sbin/ab
/usr/sbin/apachectl
/usr/sbin/httpd
/usr/sbin/logresolve
/usr/sbin/rotatelogs
/usr/share/doc/apache-1.3.19
/usr/share/doc/apache-1.3.19/ABOUT_APACHE
/usr/share/doc/apache-1.3.19/LICENSE
/usr/share/doc/apache-1.3.19/LICENSE.SSL
/usr/share/doc/apache-1.3.19/README
/usr/share/doc/apache-1.3.19/README.CRT
/usr/share/doc/apache-1.3.19/README.CSR
/usr/share/doc/apache-1.3.19/README.KEY
/usr/share/doc/apache-1.3.19/README.SSL
/usr/share/doc/apache-1.3.19/mod_bandwidth.txt
/usr/share/doc/apache-1.3.19/mod_gzip-
commands.txt
/usr/share/doc/apache-1.3.19/mod_gzip-
samples.txt
/usr/share/doc/apache-1.3.19/pkg.contrib
/usr/share/doc/apache-1.3.19/pkg.contrib/README
/usr/share/doc/apache-1.3.19/pkg.contrib/cca.sh
/usr/share/doc/apache-1.3.19/pkg.contrib/gid-
mkcert.sh
/usr/share/doc/apache-1.3.19/pkg.contrib/gid-
tagcert.c
/usr/share/doc/apache-
1.3.19/pkg.contrib/loadcacert.cgi
/usr/share/doc/apache-1.3.19/pkg.contrib/rse.pgp
/usr/share/doc/apache-1.3.19/pkg.contrib/sign.sh
/usr/share/doc/apache-1.3.19/pkg.contrib/sxnet.tar
/usr/share/doc/apache-
1.3.19/pkg.contrib/truerand.c
/usr/share/doc/apache-1.3.19/snakeoil-ca-dsa.crt
/usr/share/doc/apache-1.3.19/snakeoil-ca-rsa.crt
/usr/share/doc/apache-1.3.19/snakeoil-dsa.crt
/usr/share/doc/apache-1.3.19/snakeoil-rsa.crt
/usr/share/man/man1/dbmmanage.1.gz
/usr/share/man/man1/htdigest.1.gz
/usr/share/man/man1/htpasswd.1.gz
/usr/share/man/man8/ab.8.gz
/usr/share/man/man8/apachectl.8.gz
/usr/share/man/man8/httpd.8.gz
/usr/share/man/man8/logresolve.8.gz
/usr/share/man/man8/rotatelogs.8.gz
/var/cache/httpd
/var/log/httpd

```

REFERENCIAS WEB

Netcraft Web Server Survey

<http://www.netcraft.com/survey>

The Apache Software Foundation

<http://www.apache.org>

HTTP - Hypertext Transfer Protocol

<http://www.w3.org/pub/WWW/Protocols/>

Apache Server Survival Guide

http://docs.online.bg/NETWORKING/Apache_Server_Survival_Guide_Table_of_Contents/index.htm

Apache: The Definitive Guide

<http://www.oreilly.com/catalog/apache2/chapter/ch13.html>

Cookie Central

<http://www.cookiecentral.com>