Modulo I

“Introducción al World Wide Web y Apache Web Server”

Antecedentes Históricos

La World Wide Web: su traducción más acertada podría ser un conjunto de documentos con referencias cruzadas. El concepto en sí no es nuevo. Las referencias a otros documentos, en forma de notas al margen, existían ya en los manuscritos medievales. La diferencia es que la Web es más global, más rápida, y más fácil de usar. Todo ello es posible gracias a los avances tecnológicos de finales del siglo pasado.

En 1945, el Director de la Oficina de Desarrollo e Investigación Científica (EE.UU.), el Doctor Vannevar Bush, escribió el artículo "As We May Think" para "The Athlantic Online", en que expresaba su preocupación por la ingente cantidad de información que existía y estaba siendo generada, y el poco tiempo y los ineficientes sistemas que había para encontrarla. Así, y basándose en la tecnología existente en aquel entonces, describió un dispositivo personal, al que llamó "memex", y que imaginaba como un suplemento íntimo a su memoria. Este aparato permitiría a cada individuo almacenar su información en microfilmes, consultarlos rápidamente y, lo que es más importante, crear vínculos entre unos documentos y otros, de modo que durante la lectura de un documento se recordara al lector qué documentos contenían información relacionada. Era una visión de lo que ocurriría sólo 45 años después.

En los años 60, Douglas Engelbart, mientras trabajaba en el Stanford Research Institute, propuso el NLS (oNLine System), un entorno de trabajo por computadora, con un sistema para almacenar publicaciones, con catálogos e índices para facilitar la búsqueda, y con reglas establecidas para citar documentos, de modo que fuera más fácil para los lectores acceder a los documentos referenciados. Era un entorno con teclado, pantalla, ratón e impresora, con posibilidad de teléconferencia y correo electrónico a través de una red de computadoras para una rápida comunicación entre los profesionales. Tenía las herramientas básicas de composición, estudio, organización y modificación de información. Los ficheros se guardaban jerárquicamente para su mejor organización. Se trabajaba con los documentos en modo multiventana, para ver varios documentos a la vez en ventanas diferentes, y se podían copiar objetos seleccionados de una ventana a otra.

El término "hipertexto" fue acuñado por Ted Nelson en 1965, en su artículo "A File Structure for the Complex, the Changing, and the Indeterminate", que leyó durante la vigésima conferencia anual de la Association of Computer Machinery (ACM). Ted Nelson ideó un modelo para la interconexión de documentos electrónicos. El proyecto Xanadu aún continúa luchando para conseguir un modelo de hipertexto superior al que trajo la World Wide Web.

La World Wide Web fue inventada en 1989 por un informático del CERN (Organización Europea de Investigación Nuclear) llamado Tim Berners-Lee. Era un sistema de hipertexto para compartir información basado en Internet, concebido originalmente para servir como herramienta de comunicación entre los científicos nucleares del CERN. Tim Berners-Lee había estado experimentando con hipertexto desde 1980, año en que programó Enquire, un programa para almacenar piezas de información y enlazarlas entre ellas. Enquire se ejecutaba en un entorno multiusuario y permitía acceder a varias personas a los mismos datos. Tim Berners-Lee entregó su propuesta al CERN en 1989, en septiembre de 1990 recibió el visto bueno y junto con Robert Cailliau comenzó a escribir el nuevo sistema de hipertexto. A finales de 1990 el primer browser de la historia, World Wide Web, ya tenía forma.

Los documentos necesitaban un formato que fuera adecuado para su misión. En aquella época casi todo el mundo utilizaba TeX y PostScript, pero éstos eran demasiado complicados teniendo en cuenta que debían ser leídos por todo tipo de computadoras, desde la terminales tontas hasta las estaciones de trabajo gráficas X-Windows. Así, tanto el lenguaje de intercambio (HTML), como el protocolo de red (HTTP) se diseñaron para ser realmente muy simples.

HTML son las siglas de "HyperText Mark-up Language". "Mark-up" es un término de imprenta que significa el conjunto de instrucciones estilísticas detalladas escritas en un manuscrito que debe ser tipografiado. Así, HTML podría ser traducido como "Lenguaje de Formato de Documentos para Hipertexto". HTML es una aplicación de SGML, un lenguaje muy general para definir lenguajes de formato de documentos.

A principios de 1993 había alrededor de 50 servidores. Existían básicamente dos tipos de browsers: el original, gráfico, pero sólo para plataformas NeXT, y el browser en modo de línea, preparado para cualquier plataforma pero muy limitado y muy poco atractivo. En Febrero se lanzó la primera versión alfa del navegador "Mosaic for X", desarrollado en el NCSA (National Center for Supercomputing Applications). Funcionaba en X Windows, que era una plataforma popular entre la comunidad científica. En Abril el tráfico de la WWW era el 0,1% del total de Internet. El CERN declaraba la WWW como tecnología de acceso gratuito. En septiembre ya había versiones de Mosaic para PC y Macintosh. El tráfico alcanzaba el 1% de todo el tráfico de Internet y había más de 500 servidores. Es el comienzo del crecimiento explosivo de la Web. A finales del 94 ya había más de 10.000 servidores y 10 millones de usuarios. En 1997, más de 650.000 servidores.

Hoy, en día, la Web es algo cotidiano para una gran parte de los más de 400 millones de usuarios de Internet que hay en todo el mundo. Sus utilidades son diversas, su impacto en la economía mundial es apreciable. No sólo hay documentos de texto: hay imágenes, vídeos, música, y a través de ella se pueden realizar infinidad de actividades.

Proyecto Apache.

El proyecto Apache es un esfuerzo de desarrollo colaborativo enfocado a la creación de un servidor HTTP de nivel comercial con diversas características y de código abierto. El proyecto es administrado por un grupo de voluntarios alrededor del mundo, usando la Internet y la misma Web para comunicar, planear y desarrollar el servidor y la documentación relacionada.

Estos voluntarios son conocidos como El Grupo Apache. Además de lo anterior, cientos de usuarios han contribuido con ideas, código y documentación del proyecto.

En Febrero de 1995, el software de servidor de Web mas popular en la Internet era HTTP Daemon desarrollado por Rob McCool en la National Center for Supercomputing Applications. Sin embargo, dicho desarrollo se estancó debido a la salida de McCool de la NCSA a mediados de 1994.

A partir de entonces varios administradores de sitios Web habían desarrollado sus propias extensiones y correcciones conforme las iban requiriendo. Un pequeño grupo de estos desarrolladores, contactados por medio de correo electrónico, se unieron con el propósito de coordinar sus cambios (a través de parches). Brian Behlendorf y Cliff Skolnick establecieron un servidor para el intercambio de información y código entre los desarrolladores de base del proyecto.

Para finales de Febrero, ocho desarrolladores fueron los fundadores del Grupo Apache original.

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

con contribuciones adicionales hechas por:

Eric Hagberg Frank Peters Nicolas Pioch

Utilizando NCSA httpd 1.3 como base, se agregaron todos las correcciones y mejoras acumuladas previamente, resultando la primera versión de Apache (0.6.2) en Abril de 1995.

Menos de un año después de que el grupo fue formado, el servidor Apache se convirtió en servidor de Web más utilizado en la Internet, según encuestas de Netcraft, hasta el día de hoy.

Funcionamiento del World Wide Web.

El World Wide Web, como se vislumbró en el apartado anterior, está compuesto de diversos componentes interrelacionados entre sí. Aunque el conocimiento total de cada uno de ellos bien podría ser cabalmente abarcado en cursos independientes; para efectos de comprensión se verá únicamente lo básico de cada uno de ellos.

Lenguaje HTML (Hipertexto)

El Lenguaje HTML, como ya se mencionó, tiene como objetivo “formatear” los documentos de hipertexto para facilitar su lectura. Los documentos HTML son documentos simples con distintas codificaciones de lenguaje (ASCII o Unicode, por ejemplo).

La mayoría de los documentos tienen estructuras comunes (títulos, párrafos, listas...) que van a ser definidas por este lenguaje mediante etiquetas. Cualquier cosa que no sea una etiqueta es parte del documento mismo.

Este lenguaje no describe la apariencia del diseño de un documento, sino que ofrece a cada plataforma darle formato según su capacidad y la de su navegador (tamaño de la pantalla, fuentes que tiene instaladas...). HTML tiene dos ventajas que lo hacen prácticamente imprescindible a la hora de diseñar un documento Web: Su compatibilidad y su facilidad de aprendizaje debido al reducido número de etiquetas que usa.

Básicamente, los documentos escritos en HTML constan del texto mismo del documento y las etiquetas que pueden llevar atributos. Esto llevado a la práctica, vendría a ser:

```
<etiqueta> texto afectado </etiqueta>
```

La etiqueta del principio activa la orden y la última (que será la del principio precedida del signo /) la desactiva. No todas las etiquetas tienen principio y final.

Tres son la etiquetas que describen la estructura general de un documento y dan una información sencilla sobre él. Estas etiquetas no afectan a la apariencia del documento y solo interpretan y filtran los archivos HTML.

<HTML>: Limita el documento e indica que se encuentra escrito en este lenguaje.

<HEAD>: Especifica el prólogo del resto del archivo. Son pocas las etiquetas que van dentro de ella, destacando la del titulo <TITLE> que será utilizado por los marcadores del navegador e identificará el contenido de la página. Sólo puede haber un título por documento, preferiblemente corto, aunque significativo, y no caben otras etiquetas dentro de él. En head no hay que colocar nada del texto del documento.

<BODY>: Encierra el resto del documento, el contenido. Vea el siguiente ejemplo.

```
<HTML>
<HEAD>
<TITLE>Ejemplo 1</TITLE>
</HEAD>
<BODY>
Hola mundo
</BODY>
</HTML>
```

Esta estructura, aunque básica, es la base para todo documento Web codificado en este lenguaje.

Visualizador HTML (Navegador)

De la misma forma que usted necesita Adobe Acrobat Reader para leer un documento de tipo PDF, también necesitará de un "visualizador" de documentos HTML. Este, conocido como navegador tiene la función de solicitar un documento a un servidor y de interpretar el lenguaje de codificación del documento.

Es importante aclarar que HTML no es el único lenguaje que un navegador puede interpretar. Los navegadores tienen la capacidad de ser "extensibles" por medio de conectores adicionales (conocidos comúnmente como plugins) que permiten la interpretación de otros lenguajes (Como JavaScript o Macromedia Flash).

Para poder acceder a los diferentes documentos que un servidor ofrece, los usuarios deberán utilizar una URL. A continuación de definirá el acrónimo, su forma de ulitización de algunos.

URL es el acrónimo de (Uniform Resource Locator), localizador uniforme de recursos y que permite localizar o acceder de forma sencilla cualquier recurso de la red desde el navegador de la WWW.

Con la WWW se pretende unificar el acceso a información de servicios que antes eran incompatibles entre sí, tratando de conseguir que todos los servicios de Internet sean accesibles a través de la WWW, de esta forma desde un mismo programa se puede tener acceso a todos los recursos de una forma uniforme y permite que los documentos HTML incluyan enlaces a otras fuentes de información en servicios como FTP, gopher, WAIS, etc.

Uso y Formato

Las **URL** se utilizarán para definir el documento de destino de los hiperenlaces, para referenciar los gráficos y cualquier otro fichero que se desee incluir dentro de un documento HTML. Cada elemento de Internet tendrá una **URL** que lo defina, ya se encuentre en un servidor de la WWW, FTP, gopher o las News.

El formato de una **URL** será:

```
servicio://maquina.dominio:puerto/camino/archivo
```

El **servicio** será alguno de los de Internet, estos pueden ser:

http: (HyperText Transport Protocol), es el protocolo utilizado para transmitir hipertexto. Todas las páginas HTML en servidores WWW deberán ser referenciadas mediante este servicio. Indicará conexión a un servidor de la WWW.

https: (HyperText Transport Protocol Secure), es el protocolo para la conexión a servidores de la WWW seguros. Estos servidores son normalmente de ámbito comercial y utilizan encriptación para evitar la intercepción de los datos enviados, usualmente números de tarjeta de crédito, datos personales, etc, realizará una conexión a un servidor de la WWW seguro.

ftp: (File Transfer Protocol), utilizará el protocolo FTP de transferencia de ficheros. Se utilizará cuando la información que se desee acceder se encuentre en un servidor de ftp. Por defecto se accederá a un servidor anónimo (anonymous), si se desea indicar el nombre de usuario se usará: ftp://maquina.dominio@usuario, y luego le pedirá la clave de acceso.

news: Accede al servicio de news, para ello el visualizador de la WWW debe ser capaz de presentar este servicio, no todos lo son. Se indicará el servidor de news y como camino el grupo de noticias al que se desea acceder: news://news.cica.es/uca.es.

telnet: Emulación de terminal remota, para conectarse a maquina multiusuario, se utiliza para acceder a cuentas públicas como por ejemplo la de biblioteca. Lo normal es llamar a una aplicación externa que realice la conexión. En este caso se indicará la maquina y el login: telnet://maquina.dominio@login.

mailto: Se utilizará para enviar correo electrónico, no todos los navegadores son capaces de hacerlo. En este caso sólo se indicará la dirección de correo electrónico del destino: mailto://alias. correo@domino.

La **maquina.dominio** indicará el servidor que nos proporciona el recurso, en este caso se utilizará el esquema IP para identificar la máquina, es decir el nombre de la maquina y el dominio. En el caso de nuestra Universidad el dominio siempre será **uca.es**. Por tanto un nombre válido de máquina será **www2.uca.es**.

Es muy importante indicar siempre el dominio, ya que debemos suponer que se conectarán a nuestras páginas desde servidores externos a nuestra red local por tanto si no indicamos el dominio las URL que especifiquemos no podrían ser seguidas por los navegadores externos. Si en vez de www2.uca.es utilizamos www2 será perfectamente accesible por cualquier máquina de nuestra red local pero si se referenciara desde una red con distinto dominio la máquina www2 será la máquina llamada así en el dominio remoto si existiera, que no es la que deseamos referenciar.

El **puerto TCP** es opcional y lo normal es no ponerlo si el puerto es el mismo que se utiliza normalmente por el servicio. Sólo se utilizará cuando el servidor utilice un puerto distinto al puerto por defecto. El **camino** será la ruta de directorios que hay que seguir para encontrar el documento que se desea referenciar. para separar los subdirectorios utilizaremos la barra de UNIX **/**, se usa por convenio al ser este tipo de máquinas las más usadas como servidores. El nombre de los subdirectorios y del fichero referenciado puede ser de más de ocho caracteres y se tendrá en cuenta la diferencia entre mayúsculas y minúsculas en el nombre.

La extensión de los ficheros será también algo importante, ya que por ella sabe el servidor el tipo de documento que se accede e indica al cliente (navegador) el modo en que debe tratarse ese documento. Para definir los tipos de documentos se utiliza los tipos MIME. Las extensiones más normales con sus tipos correspondientes son:

Tipo MIME	**Extensión**	**Tipo de fichero**
text/html	html ó .htm,	documento HTML
text/plain	.txt	por defecto, texto plano
image/gif	.gif	imagen de formato GIF
image/jpeg	jpg ó .jpeg	imagen de formato JPEG

El navegador de la WWW, realiza una acción para cada tipo de fichero, sólo los que sean del tipo text/html serán mostrados como documentos HTML. En el caso de que el tipo no sea conocido por el cliente se considerará por defecto como un documento de texto normal.

Si no se indica un fichero y sólo referenciamos un directorio accederemos a la página html por defecto de ese directorio. En el servidor están definidos unos ficheros para ser usados si no se indica un fichero concreto, el nombre que debe tener este fichero es en nuestro caso default.htm ó default.html. Este fichero es la página inicial (home page) del servidor o del espacio Web.

Algunos ejemplos de URL podrían ser:

URL	**Definición**
http://www.uca.es	En este caso sólo se indica el servicio y la máquina y dominio. El resto de los parámetros se toman por defecto, el puerto 80, el directorio, el raíz del servidor y el documento por defecto de ese directorio.
http://www.uca.es/internet/internet.html	Esta URL está más completa, en este caso se accede al fichero internet.html que se encuentra en el directorio internet del servidor de la WWW, www.uca.es.
http://www2.uca.es/serv/sii	Se accederá al fichero por defecto del directorio /serv/sii del servidor de la WWW, www2.uca.es
ftp://ftp.uca.es/imagenes/globo.gif	En este caso se accederá a un servidor de FTP anónimo, ftp.uca.es por el protocolo FTP y se accederá al fichero globo.gif del directorio de imágenes.
news:uca.es	En este caso se accederá al grupo de news de la UCA en el servidor de news definido por defecto en el navegador de la WWW.
mailto://www-team@uca.es	Enviará un mail al equipo de la WWW de la UCA.

Protocolo de Transferencia de Hipertextos (HTTP)

Internet tiene su fundamento en base a protocolos estándares, sin los cuales no podría funcionar. Si bien el protocolo subyacente es el **TCP/IP**, para ciertas funciones particulares son necesarios otros protocolos, como en el caso específico de la Web, donde fue necesario crear un protocolo que resolviese los problemas planteados por un sistema hipermedial, y sobre todo distribuido en diferentes puntos de la Red.

Este protocolo se denominó **HTTP** (HyperText Transfer Protocol, o Protocolo de Transferencia de Hipertexto), y cada vez que se activa cumple con un proceso de cuatro etapas entre el browser y el servidor que consiste en lo siguiente:

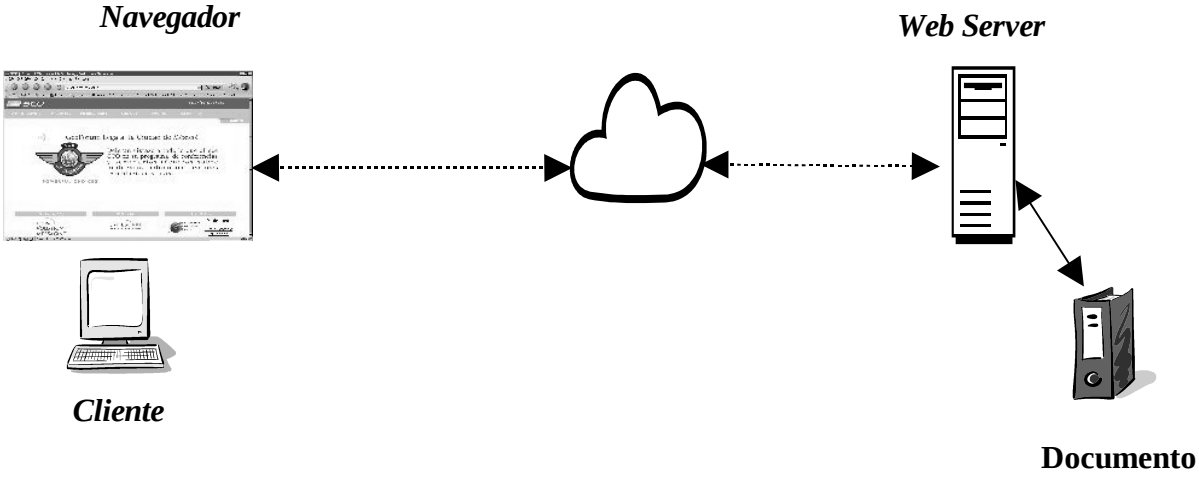
· **Conexión:** el browser busca el nombre de dominio o el número IP de la dirección indicada intentando hacer contacto con esa computadora.

· **Solicitud:** el browser envía una petición al servidor (generalmente un documento), incluyendo información sobre el método a utilizar, la versión del protocolo y algunas otras especificaciones.

· **Respuesta:** el servidor envía un mensaje de respuesta acerca de su petición mediante códigos de estado de tres dígitos.

· **Desconexión:** se puede iniciar por parte del usuario o bien por parte del servidor una vez transferido un archivo.

El siguiente diagrama nos puede facilitar la comprensión de lo anterior.



Modulo II

“Instalación y Configuración básica de Apache”

Una vez comprendidas la bases del llamado Web, procederemos a la instalación de Apache en nuestro servidor.

Si bien es cierto que los sistemas operativos Unix y derivativos como Linux incluyen el servidor Apache pre-configurado, muchas veces debido a que este no cumple con nuestras necesidades de publicación, mejorar el desempeño o simplemente tener mayor control con los servicios que el servidor ofrece, será necesario realizar una instalación desde cero; lo anterior será realizado por medio de la compilación de Apache y de los componentes adicionales que este requiera.

Antes de continuar es de suma importancia el conocer la estructura de funcionamiento de Apache, para así poder tomar las decisiones correctas al momento de habilitar o deshabilitar funciones de Apache para efectos de compilación.

El servidor Web Apache tiene dos ramas de desarrollo. Por un lado esta la rama “clásica”, es decir el código original de la NCSA más las modificaciones y adiciones del Apache Group. Esta rama es considerada la más estable y con mayor número de extensiones disponibles. Al momento de escribir este manual, la última versión estable es la 1.3.36.

Por el otro lado se encuentra la rama “nueva” de Apache, la cual empieza a partir de la versión 2.0. Esta se distingue por haber sido desarrollada completamente desde cero, por lo cual su desempeño es mayor a la rama “clásica”.

Los archivos de configuración y las entradas son casi iguales por lo cual un administrador que haya utilizado la rama “clásica” de Apache no tendrá problema alguno para configurar algún servidor de la rama “nueva”.

Apache puede ser compilado de dos formas:

a) Monolítica: En este caso, todo el software extra (conocido como “módulo”) será incluido en el binario principal de Apache (de nombre httpd). En este caso solamente al momento de compilar se podrá anexar los módulos.
b) Modular: Se habilitará Apache para soporte a software extra, el cual podrá anexarse posteriormente por medio del compilador de módulos de Apache.

En la práctica, lo más común es el segundo caso, ya que generalmente requerimos agregar funcionalidad a Apache.

Instalación de Apache vía compilación.

Para poder realizar este tipo de compilación será necesario tener el siguiente software previamente instalado.

- Compilador de C.
- Make.
- Librerias de desarrollo.

Para facilitarnos la tarea, los desarrolladores de Apache utilizan la utilería Autoconf, la cual genera dentro de la carpeta de código fuente un script de nombre "configure", el cual es ejecutado y en base a nuestro Sistema Operativo y al software de desarrollo instalado, nos genera el Archivo de instrucciones del Comando Make (Conocido como Makefile). En caso de que nos falte algún programa o librería o la versión que tengamos en nuestro Sistema Operativo no sea la adecuada, este script nos enviará un mensaje de error.

Al momento de instalar el software que se haya compilado, Apache creará su propia "rama" de directorios con la finalidad de evitar mezclarse con otros archivos. Por defecto esta es **/usr/local/apache**. Esto puede ser modificado por medio de instrucciones al script "configure".

Instrucciones de compilación.

Una vez descargado el código fuente de Apache (La lista de servidores espejos disponibles se encuentra en http://httpd.apache.org/download) deberá descomprimirlo:

```
# tar –xzvf apache-x.y.z.tar.gz
 (Donde x.y.z es la versión de Apache a instalar)
# tar –xjvf apache-x.y.z.tar.bz2
 (Donde x.y.z es la versión de Apache a instalar)
```

Observense las opciones del comando tar, estas varian según la extensión que tenga el archivo tar que contiene los fuentes, en el caso de extensiones .tgz o .tar.gz se emplea la opción z, si la extensión es bz2 la opción es j.

Una vez descomprimido el archivo, procederemos a acceder a la carpeta generada. Dentro de esta deberá estar el script de configuración. Para conocer las opciones disponibles, ejecutaremos lo siguiente:

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

La opción –-help nos mostrará la ayuda disponible del script. Para ayudarnos a personalizar nuestra instalación, los siguientes parámetros son a nuestro criterio los más importantes:

- --prefix. Este parámetro le indica al configurador el directorio de inicio del árbol de binarios de Apache. Si se omite, será el árbol por defecto mencionado en la sección anterior. Se sugiere que primero intente con un directorio inexistente y, después de realizadas las pruebas correspondientes, se migre al directorio definitivo.

- --enable-module. Apache tiene incluidos algunos módulos adicionales (Como el soporte a binarios de tipo CGI), Algunos de estos se encuentran habilitados por defecto.. Con esta instrucción se le indica al servidor que se desea adicionar ese código de forma monolítica.

- --enable-shared. Este parámetro le indica a Apache que agregue un módulo de forma independiente, es decir, de forma modular. Para poder hacer esto deberemos agregar el módulo "so" (Shared Object) de forma monolítica mediante la instrucción --enable-module=so.

- --add-module. Algunos desarrolladores de módulos de Apache permiten el anexado de módulos desde la compilación misma de Apache. Este parámetro le indica a Apache que considere dicho código fuente al momento de compilar.

- --activate-module. Además de ser adicionados, los módulos extras requieren ser activados. Utilice este parámetro para realizar esto.

- --with-perl. Este parámetro le indica a Apache donde se encuentra el binario de Perl. Esto es necesario (aunque no indispensable) si deseamos que Apache tenga la capacidad de ejecutar scripts de Perl (No confundir con el módulo de soporte nativo de Perl, conocido como mod_perl).

Nota importante: La opciones –-add-module y –-active-module serán necesarias dependiendo del módulo que se desea adicionar. Algunos módulos se deben adicionar de esta forma mientras que otros se adicionan post-instalación. Por favor remítase a la documentación del software para instrucciones detalladas.

Para efectos generales utilizaremos la siguiente línea de configuración.

```
#./configure --prefix=/opt/apache –-add-module=so –-enable-shared=status
--enable-shared=vhost_alias –-with-perl=/usr/bin/perl
```

Lo anterior hará que el Apache:

a) El árbol de archivos y directorios de Apache será instalado en la carpeta **/opt/apache**.
b) Adicionará los módulos por defecto, anexando a la lista el soporte a módulos externo.
c) Adicionará como módulos externos el módulo status (Para soporte al acceso del estatus del servidor de forma remota) y el módulo vhost (Para soporte a servicios virtuales).
d) Finalmente, indicamos la ubicación del intérprete de Perl para uso de Apache.

La salida de dicho comando será la siguiente (más o menos dependiendo del sistema operativo, compilador, etc.):

```
Configuring for Apache, Version 1.3.29
 + using installation path layout: Apache (config.layout)
Creating Makefile
Creating Configuration.apaci in src
```

```
Creating Makefile in src
 + configured for Linux platform
 + setting C compiler to gcc
 + setting C pre-processor to gcc –E
 + using "tr [a-z] [A-Z]" to uppercase
 + checking for system header files
 + adding selected modules
 + using system Expat
 + using -ldl for vendor DSO support
 + checking sizeof various data types
 + doing sanity check on compiler and options
Creating Makefile in src/support
Creating Makefile in src/regex
Creating Makefile in src/os/unix
Creating Makefile in src/ap
Creating Makefile in src/main
Creating Makefile in src/modules/standard
```

Posteriormente, procederemos a compilar dicho programa por medio del comando make:

```
#make
```

Make ejecutará las instrucciones del archivo Makefile (generado durante la configuración), entre ellas, la ejecución del compilador de c.

```
===> src
make[1]: Entering directory `/root/apache_1.3.29'
make[2]: Entering directory `/root/apache_1.3.29/src'
===> src/regex
gcc -I.   -I../os/unix -I../include    -DLINUX=22 -DUSE_HSREGEX `../apaci`
-DPOSIX_MISTAKE   -c -o regcomp.o regcomp.c
gcc -I.   -I../os/unix -I../include    -DLINUX=22 -DUSE_HSREGEX `../apaci`
-DPOSIX_MISTAKE   -c -o regexec.o regexec.c
gcc -I.   -I../os/unix -I../include    -DLINUX=22 -DUSE_HSREGEX `../apaci`
-DPOSIX_MISTAKE   -c -o regerror.o regerror.c
gcc -I.   -I../os/unix -I../include    -DLINUX=22 -DUSE_HSREGEX `../apaci`
-DPOSIX_MISTAKE   -c -o regfree.o regfree.c
rm -f libregex.a
...Contenido omitido ...
gcc   -DLINUX=22 -DUSE_HSREGEX `../apaci` -o logresolve     -L../os/unix
-L../ap logresolve.o -lm -lap -los  -lm -lcrypt -lexpat -ldl
gcc -c   -I../os/unix -I../include    -DLINUX=22 -DUSE_HSREGEX `../apaci`
ab.c
gcc   -DLINUX=22 -DUSE_HSREGEX `../apaci` -o ab     -L../os/unix -L../ap
ab.o -lm -lap -los  -lm -lcrypt -lexpat -ldl
sed <apxs.pl >apxs \
    -e 's%@TARGET@%httpd%g' \
    -e 's%@CC@%gcc%g' \
    -e 's%@CFLAGS@% -DLINUX=22 -DUSE_HSREGEX `../apaci`%g' \
    -e 's%@CFLAGS_SHLIB@%-fpic -DSHARED_MODULE%g' \
    -e 's%@LD_SHLIB@%gcc%g' \
    -e 's%@LDFLAGS_MOD_SHLIB@%-shared%g' \
    -e 's%@LIBS_SHLIB@%%g' && chmod a+x apxs
```

```
gcc -c  -I../os/unix -I../include   -DLINUX=22 -DUSE_HSREGEX `../apaci`
checkgid.c
gcc  -DLINUX=22 -DUSE_HSREGEX `../apaci` -o checkgid   -L../os/unix -L../
ap checkgid.o -lm -lap -los  -lm -lcrypt -lexpat -ldl
make[2]: Leaving directory `/root/apache_1.3.29/src/support'
<=== src/support
make[1]: Leaving directory `/root/apache_1.3.29'
<=== src
```

Posteriormente, instalaremos el software compilado en la carpeta previamente indicada mediante la siguiente instrucción:

```
#make install
```

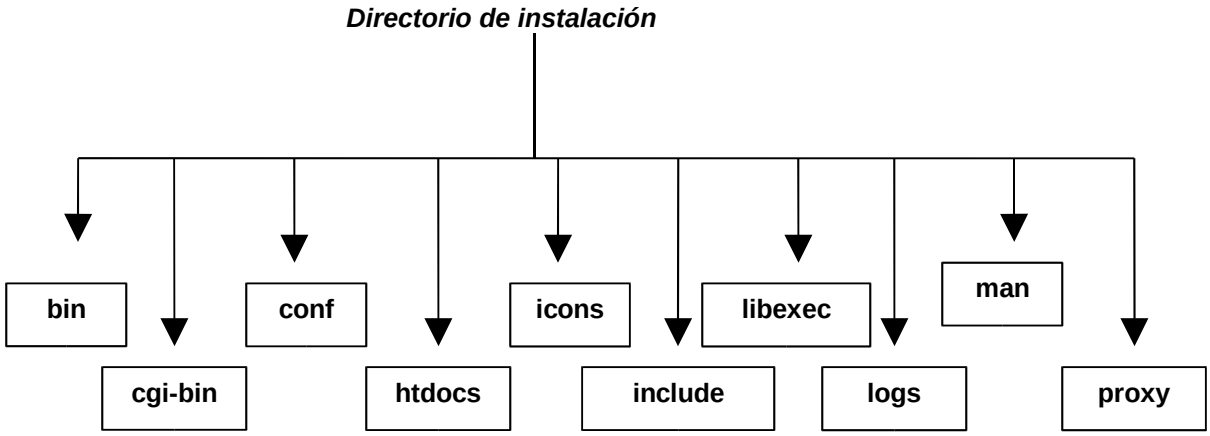
Una forma común de compilar apache es con las opciones:

```
#./configure  –prefix=/usr/local/apache  --enable-module=so  –enable-
module=most
```

Lo que permite esto es que sea todo instalado bajo un directorio dado, se active el módulo de carga dinámica y los módulos más comunes sean compilados estáticamente en el ejecutable de apache.

Archivos, directorios y programas de Apache Web Server

Una vez compilado e instalado el servidor Apache, nos moveremos a la carpeta previamente designada. El árbol de directorios quedará de la siguiente manera:



Los directorios son los siguientes:

bin: En este directorio se encuentran los archivos binarios de nuestro servidor Web entre los que se encuentran:
httpd: Programa del servicio de http.
apachectl: Script para el control de httpd (Arrancar, Parar, Reiniciar, etc.)
ab: Herramienta para benchmarking para nuestro servidor Apache.
htpasswd: Herramienta para la generación de cuentas de acceso.

htdigest: Herramienta similar a la anterior para ser utilizada en autenticaciones seguras.
Entre otros…

cgi-bin: Este directorio se utiliza para la colocación de binarios CGI.

conf: En este directorio se encuentran los archivos de configuración principales. El más importante de todos es **httpd.conf**, sin el cual Apache dejaría de funcionar.

htdocs: Este es el directorio por defecto donde serán colocados los documentos a publicar.

icons: Como su nombre lo indica, este directorio se utiliza para la colocación de los íconos de las páginas de ejemplo que el servidor Apache muestra.

include: Contiene los archivos de cabecera de lenguaje C que algunos módulos de terceros requieren para poder ser compilados.

libexec: Este directorio contiene los módulos compilados externos de Apache.

logs: Como su nombre lo indica, este directorio contiene las bitácoras de acceso y de error del servidor para futura referencia.

man: Directorio que contiene las páginas de manual de Apache. Refiérase a la documentación de su sistema operativo para ver como anexarlas a las ya existentes.

Algunos módulos generan sus propios directorios para usos distintos. En el caso de nuestra configuración, fue generado el directorio **proxy** para la utilización del módulo de proxy de Apache. Refiérase a la documentación del módulo para conocer su configuración y funcionamiento.

Parámetros de configuración principales del archivo de configuración principal.

Como se mencionó en la sección anterior el archivo de configuración principal se encuentra en la carpeta **conf** de nombre **httpd.conf**. Este archivo controla casi en su totalidad el comportamiento del servidor. A pesar de que este pueda parecer intimidante debido a la gran cantidad de líneas que posee, en realidad es muy sencillo. A continuación procederemos a analizar los parámetros principales del archivo. Para mayor comprensión hemos omitido los comentarios y los parámetros que no son significativos para el funcionamiento más común del servidor.

El archivo httpd.conf está dividido en tres secciones. La primera establece el entorno general de operación que comprende la forma en que nuestro servidor se comportará.

Section 1: Global Environment

ServerType (Opciones: standalone ó inetd)

El parámetro ServerType le indica al servidor si será él mismo el que contestará las peticiones de los clientes o será llamado bajo demanda por medio del servicio inetd (Internet Daemon). Por lo general, salvo casos muy raros la opción a colocar aquí no será diferente a standalone.

ServerRoot "/opt/apache"

ServerRoot, como su nombre lo indica, es el directorio en el cual Apache localizará los directorios de configuración y bitácora. Este valor rara vez es modificado aunque bien podría modificarse este valor al momento de migrar este archivo de configuración a otra carpeta de trabajo.

PidFile logs/httpd.pid

Por medio de este parámetro le indicamos a nuestro servidor en que directorio y en que archivo será escrito el número de identificador del proceso padre de Apache. Este archivo puede ser muy útil al momento de escribir scripts de arranque del servidor.

#ResourceConfig conf/srm.conf
#AccessConfig conf/access.conf

Como se mencionó al principio de esta sección, hoy en día Apache utiliza casi siempre el archivo httpd.conf. Sin embargo, esto no siempre fue así. En las primeras versiones de Apache, este utilizaba 3 archivos.

a) httpd.conf para la configuración general.
b) access.conf para la restricción de los usuarios a los recursos del sistema (Restricción por dirección o en base a contraseñas).
c) srm.conf para la configuración de los recursos del sistema (Como el manejo de alias).

Si de todos modos por alguna razón desea utilizar estos archivos, descomente estas líneas. Para mayor información con respecto al manejo del conjunto de archivos, diríjase a la dirección Web: http://httpd.apache.org/info/three-config-files.html

Timeout 300

Como su nombre lo indica, este parámetro estable el tiempo máximo de espera para una petición valuado en segundos.

KeepAlive On

Por defecto, el protocolo HTTP abre una conexión hacia el cliente **por cada elemento de cada página**. Es decir, si un documento contiene 2 imágenes, se abrirán tres conexiones. Para evitar esto, Apache ha implementado la función keepalive, que mantiene la primera conexión que se abre para enviar todos los elementos de una página; esto con la finalidad de mejorar el rendimiento del servidor. Este parámetro está habilitado por defecto.

```
MaxKeepAliveRequests 100
```

Este parámetro esta íntimamente ligado al parámetro keepalive. Su función es establecer el número máximo de peticiones que se pueden hacer dentro de una misma conexión. Si su valor equivale a cero no existen restricciones.

```
KeepAliveTimeout 15
```

Como su nombre lo indica, este valor establece el tiempo de espera máximo entre peticiones dentro de una misma conexión.

```
StartServers 5
MaxClients 150
```

Por medio de este parámetro le indicamos a nuestro servidor el número de servicios que levantará al principio y el máximo de conexiones simultáneas. La combinación de estos valores junto con los 5 anteriores ayudarán a mejorar el desempeño del servidor.

```
MinSpareServers 5
MaxSpareServers 10
```

Apache, de forma dinámica, va levantando servicios conforme aumentan las peticiones. Con estos valores indicamos los rangos de aumento de los servicios. Apache irá aumentando el número de servicios dentro de estas bandas límite.

```
#Listen 3000
#Listen 12.34.56.78:80
```

Este parámetro indica los puertos adicionales en los cuales nuestro servidor Apache contesta. Como puede observar, puede indicarse combinaciones Dirección ip:Puerto

```
#BindAddress *
```

Utilice BindAddress para indicar la o las direcciones IP en las cuales nuestro servidor escuchará. Este parámetro hoy en día esta en vías de extinción ya que el comando Listen lo sustituye.

```
LoadModule vhost_alias_module libexec/mod_vhost_alias.so
LoadModule status_module libexec/mod_status.so
```

El comando LoadModule es uno de los más importantes, ya que le indica a nuestro servidor que módulos disponibles bajo el directorio libexec serán cargados al arranque de este. La palabra siguiente después del parámetro indica la descripción genérica del módulo y posteriormente sigue la ubicación del modulo pudiendo ser relativa al árbol de directorios del servidor o una ruta absoluta en caso de que se encuentre el módulo fuera de dicho árbol.

```
ClearModuleList
AddModule mod_vhost_alias.c
AddModule mod_status.c
```

Por medio de este comando activamos los módulos internos (aquellos que compilamos dentro del binario de Apache) y externos (Siempre y cuando hayan sido cargados previamente mediante el parámetro LoadModule). Claro está que por razones de performance es mejor desactivar aquellos módulos que no se estén empleando.

```
#ExtendedStatus On
```

Este parámetro será activado siempre y cuando hallamos habilitado el modulo de status y tengamos habilitado el acceso a la página de estatus del servidor, lo cual será visto más adelante. Al habilitarlo, la información de estatus será más detallada.

Section 2: 'Main' server configuration
```
Port 80
```

Este parámetro indica el puerto principal para recibir peticiones de Apache. Sustitúyase preferentemente por el valor Listen explicado previamente arriba.

```
User nobody
Group nobody
```

Por seguridad Apache **NUNCA** deberá ser utilizado públicamente por el usuario de root. Para ello deberemos indicar por medio de los parámetros User y Group el usuario sin privilegios por medio del cual se controlará nuestro servidor.

Nota importante: El primer proceso de Apache tiene como dueño root, sin embargo, este proceso **JAMAS** es ofrecido al exterior del servidor. Observe la siguiente lista de procesos:

```
# ps -ef|grep httpd
root      1644     1  0 14:58 ?        00:00:00 /opt/apache/bin/httpd
nobody    1645  1644  0 14:58 ?        00:00:00 /opt/apache/bin/httpd
nobody    1646  1644  0 14:58 ?        00:00:00 /opt/apache/bin/httpd
nobody    1647  1644  0 14:58 ?        00:00:00 /opt/apache/bin/httpd
nobody    1648  1644  0 14:58 ?        00:00:00 /opt/apache/bin/httpd
nobody    1649  1644  0 14:58 ?        00:00:00 /opt/apache/bin/httpd
root      1657  1496  0 14:58 pts/0    00:00:00 grep httpd
```

El proceso con el PID 1644 es el proceso padre de todos los servicios ofrecidos de nuestro servidor.

ServerAdmin superfunkylistic@yahoo.com

Cuando el servidor envía un mensaje de error a un cliente (p.e. 404 Document not found) a través de una página html, anexa la dirección de correo del administrador. Este parámetro simplemente indica dicho correo de contacto.

Nota: Si desea indicar una página distinta a la por defecto para mostrar en caso de error, remitasé al parámetro **ErrorDocument** en el mismo archivo de configuración.

Configuración del nombre completo del servidor.

ServerName xipe.orbis.org.mx

Este valor es muy importante pues indica con que nombre el servidor Apache se publicará a la red. Si este valor **no es resuelto o conocido por los clientes, NO SE TENDRÁ ACCESO A DOCUMENTOS PUBLICADOS VÍA ALÍAS.**

Configuración del directorio de documentos a publicar.

```
DocumentRoot "/opt/apache/htdocs"


<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>

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

Las etiquetas <Directory nombre del directorio> y </Directory>, similares a etiquetas HTML, son utilizadas para establecer valores de presentación y restricción de directorios para su acceso vía Apache. El parámetro Options nos sirve para establecer criterios de uso del directorio como puede ser:

FollowSymlinks: Permite al servidor seguir ligas simbólicas de documentos y archivos.
Multiviews: En el caso que el módulo de negociación (mod_negociation) esté cargado, permitirá el uso de multipáginas para soporte a diferentes idiomas.
Indexes: En caso de no existir un archivo de índice de directorio (ver más abajo), muestra el contenido de la carpeta.
Includes: Páginas con Server Side Includes son permitidos.
ExecCGI: Permite la ejecución de binarios CGI que estén colocadas en dicha carpeta.

```
<IfModule mod_userdir.c>
    UserDir public_html
</IfModule>
```

Las etiquetas <IfModule> y </IfModule> serán utilizadas para definir parámetros particulares de los distintos módulos a utilizar. En la parte superior, se define el parámetro a utilizar en caso de que el módulo mod_userdir sea cargado y activado.

Este parámetro (UserDir) es para indicar el directorio en el cual los usuarios deberán colocar sus archivos html para ser publicados. Para acceder a las páginas de usuario utilice la URL:

http://servidor.dominio/~nombre_del_usuario/
(Es importante la última diagonal)

```
<IfModule mod_dir.c>
    DirectoryIndex index.pl index.html
</IfModule>
```

En este caso el valor DirectoryIndex indica el nombre del archivo de índice de directorio. Este archivo será el que el servidor Apache publicará primero. Si se indican más de un archivo, se buscaran en el orden que se indique. Para mejorar el desempeño es recomendable poner primero las opciones más utilizadas.

```
AccessFileName .htaccess
```

Para evitar que los desarrolladores web tengan alguna injerencia con el servidor Apache y que, a su vez, les permita establecer los valores de los directorios con los que trabajan; podemos por medio del parámetro AccessFileName establecer el nombre de un archivo donde el dueño de la carpeta puede colocar sus opciones y parámetros de restricción. Por defecto este archivo tiene el nombre de .htaccess.

```
<Files ~ "^\.ht">
    Order allow,deny
    Deny from all
    Satisfy All
</Files>
```

Las etiquetas <Files> y </Files> son utilizadas para establecer parámetros de configuración y restricción de archivos. Para la definición de estos archivos, pueden utilizarse expresiones regulares colocando previamente el carácter “~”. En el caso descrito arriba, todos los archivos que comiencen (indicado por medio de carácter “^”) con un punto (la \ deshabilita la interpretación del punto) y a continuación tengan las letras HT, serán ocultos para los navegadores de los clientes.

```
<IfModule mod_mime.c>
    TypesConfig conf/mime.types
</IfModule>
```

En este caso, si el módulo mod_mime es cargado y activado se leerá el archivo mime.types el cual indicará el tipo MIME (Multipurpose Internet Mail Extensión) que el servidor reconocerá. También se pueden anexar nuevas extensiones en el archivo de configuración general mediante el parámetro Addtype.

```
DefaultType text/plain
```

Indica el tipo de archivo que nuestro servidor utilizará por defecto.

```
HostnameLookups Off
```

Este parámetro le indica a nuestro servidor si deseamos que resuelva las direcciones que le solicitan peticiones para efecto de escritura de bitácoras. El valor por defecto es Off. El tener encendida está opción involucra una carga extra para el servidor, ya se tinen que hacer una búsqueda del nombre del cliente en cada petición.

```
ErrorLog logs/error.log
```

Este parámetro le indica a nuestro servidor donde serán almacenadas las bitácoras. Las bitácoras que Apache maneja son dos:

error_log: En esta bitácora se muestran los mensajes de arranque del servidor así como los errores que se generen.
access_log: En esta bitácora se muestran los accesos a los diferentes documentos de nuestro servidor.

Nota: Los archivos también pueden llamarse error.log y access.log, o bien como decida el administrador.

```
LogLevel warn
```

Indica el nivel de información proporcionada por la bitácora. Los posibles niveles son:

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

Para mayor información consulte el manual de su servidor Apache.

Configuración de un alias de documentos.

```
<IfModule mod_alias.c>

    Alias /icons/ "/opt/apache/icons"

    Alias /manual/ "/opt/apache/htdocs/manual/"

    ScriptAlias /cgi-bin/ "/opt/apache/cgi-bin/"
```

Los ejemplos anteriores nos muestran la forma de crear un “alias”. Los alías son redirecciones de carpetas de documentos a través de la URL. Estas serán utilizadas cuando.

a) La carpeta de documentos está fuera de la raíz de documentos.
b) La ruta de la carpeta de documentos es muy larga y deseamos simplificarla.

En los ejemplos, las dos primeras líneas indican a que carpeta deberá el servidor enviarnos en caso de querer acceder a las URL’s http://servidor/icons/ y http://servidor/manual/.

El tercer caso es un poco distinto. La palabra ScriptAlias además de indicar el alías también indica que en ese servidor se encuentran CGI's. Es el equivalente a la entrada ExecCGI en la configuración de parámetros de directorio.

Configuración de una redirección de documentos.

Es probable que alguna vez necesitemos crear alías de directorios de documentos que se encuentran en otro servidor. Para realizar esto, utilizaremos el parámetro Redirect de la siguiente manera:

```
Redirect /buscador http://www.google.com.mx
```

Es importante el hacer notar que es **INDISPENSABLE** la escritura de la **URL Completa**.

```
<IfModule mod_mime.c>

    #
    # AddType allows you to tweak mime.types without actually editing it,
      or to
    # make certain files to be certain types.
    #
    AddType application/x-tar .tgz
    AddType application/x-httpd-php .php
    #
    AddEncoding x-compress .Z
   AddEncoding x-gzip .gz .tgz
   </IfModule>
```

En caso de que el módulo de soporte a MIME esté cargado, además del archivo de configuración antes declarado. Podemos indicar entradas aquí.

```
  AddLanguage da .dk
    AddLanguage nl .nl
    AddLanguage en .en
    AddLanguage et .ee
    AddLanguage fr .fr
    AddLanguage de .de
    AddLanguage el .el
    AddLanguage he .he
    AddCharset ISO-8859-8 .iso8859-8
    AddLanguage it .it
    AddLanguage ja .ja
    AddCharset ISO-2022-JP .jis
    AddLanguage kr .kr
    AddCharset ISO-2022-KR .iso-kr
    AddLanguage nn .nn
    AddLanguage no .no
    AddLanguage pl .po
    AddCharset ISO-8859-2 .iso-pl
```

```
    AddLanguage pt .pt
    AddLanguage pt-br .pt-br
    AddLanguage es .es
    AddLanguage sv .sv
    AddLanguage cs .cz .cs
    AddLanguage ru .ru
    AddLanguage zh-TW .zh-tw
```

Por medio de los parámetros AddLanguage y AddCharset podemos indicarle a nuestro servidor los lenguajes y codificaciones de carácter a soportar. Recuerde que para hacerlos válidos tiene que configurar la opción Multiviews en la carpeta de documentos que desee hacerlo. Si no requiere ver documentos en distintos idiomas se pueden comentar las líneas de los lenguajes no requeridos y poner en orden de prioridad los idiomas más vistos en la opción LanguajePriority.

```
  <IfModule mod_negotiation.c>
        LanguagePriority es en fr de it pt
    </IfModule>
```

Si el módulo de negociación de contenido está cargado y activado, por medio del parámetro LanguagePriority indicamos el orden de lenguaje que ofreceremos al cliente. Recuerde que este orden no aplicará si el navegador solicita sus documentos en un idioma en particular que nuestro servidor soporte y tenga los documentos en dicha lengua.

```
    AddHandler cgi-script .cgi .pl
    AddType text/html .shtml
    AddHandler server-parsed .shtml
```

Estas entradas, que se encuentran por defecto comentadas, deberán descomentarse en caso de que se desee soporte de CGI's fuera de un directorio indicado para este tipo de archivos así como archivos de lenguaje **Server-Side includes (SSI)**, que permite la creación de documentos html a partir de archivos de texto simple.

Para mayor información sobre SSI escriba la siguiente URL en su navegador:

http://www.bignosebird.com/ssi.shtml

```
<Location /server-status>
    SetHandler server-status
    Order deny,allow
    Deny from all
    Allow from localdomain
</Location>
```

Las etiquetas <Location> y </Location> se utilizan de manera similar a las etiquetas <Directory> y </Directory>. La diferencia radica en que la etiqueta <Location> solo aplica a nivel de URL mientras que la etiqueta </Directory> aplica para un directorio y sus respectivos directorios.

El recurso /server-status es una página “virtual” que nos muestra en tiempo real el estatus de nuestro servidor Apache. Esta página se obtiene cuando el módulo de status de Apache está cargado y activado. En el ejemplo, se ha restringido el acceso a esta página únicamente al propio servidor. Por seguridad en un servidor de producción activo no se recomienda tener activada esta opción.

Configuración de servidores virtuales.

Debido a la capacidad de Apache para crear y manejar múltiples instancias, este no solo tiene la capacidad de manejar peticiones simultáneas dirigidas a un mismo servidor, sino además de atenderlas de forma diferenciada en base a la dirección IP que recibe la solicitud o al nombre de la URL a la cual fue solicitada el documento. Esto último es especialmente ventajoso ya que podemos ofrecer a través de la misma dirección IP peticiones a nombres de servidor diferentes. A continuación veremos ambas formas de configuración.

a) Por dirección.

Esta es la más sencilla forma de configuración, ya que simplemente tendremos que indicar el nombre completo de cada una de las direcciones IP, ya sea vía la función resolv() o por medio de un DNS.

Supongamos que tenemos dos direcciones. Una de ellas de tipo homologada con el número **200.33.26.40** y una dirección de tipo no homologada con el número **192.168.1.200**. Si deseamos ofrecer un contenido a la red interna y otra al público en general tendremos que hacer lo siguiente.

1. Asignar un nombre a cada una de las direcciones por los métodos antes señalados. Para este caso le daremos a la dirección homologada el nombre www.apache-server.com y a la dirección no homologada el nombre www.intranet-apache.com

2. En nuestro servidor Apache pondremos la siguiente configuración:

```
<VirtualHost www.apache-server.com>
    ServerAdmin admin@apache-server.com
    DocumentRoot /home/public/html
    ServerName www.apache-server.com
    ErrorLog logs/www.apache-server.com_error.log
    CustomLog logs/www.apache-server.com_access.log
</VirtualHost>


<VirtualHost www.intranet-apache.com>
    ServerAdmin admin@intranet-apache.com
    DocumentRoot /home/intranet/html
    ServerName www.intranet-apache.com
    ErrorLog logs/www.intranet-apache.com_error.log
    CustomLog logs/www.intranet-apache.com_access.log
```

```
</VirtualHost>
```

Como puede observar, para realizar esto sólo necesitamos especificar los valores particulares de cada servidor virtual por medio de las etiquetas <Virtualhost> y </Virtualhost>. Para este ejemplo, lo más importante es el valor que se encuentra en la primera etiqueta <Virtualhost> en la cual se indica el nombre que nuestro servidor deberá distinguir. Recuerde que este nombre deberá estar "casado" con alguna de las direcciones con la que queramos utilizar. Los demás valores tienen el mismo significado que en la configuración general del servidor. Es importante denotar que el valor de ServerName sea el mismo que el nombre declarado en la primera etiqueta de Virtualhost.

b) Por nombre.

Para realizar la configuración con una sola dirección IP, será necesario que le indiquemos a nuestro servidor en que dirección IP deberá de estar atento a las cabeceras de las peticiones, con la finalidad de distinguir las peticiones por medio de los distintos nombres que el servidor ofrezca al exterior. Esto será posible a través del parámetro NameVirtualHost. Para mayor comprensión, veamos el siguiente ejemplo:

```
NameVirtualHost *:80
```

En este caso, indicamos a nuestro servidor que analice las peticiones provenientes de cualquier dirección IP con destino al puerto 80.

```
NameVirtualHost 10.10.10.2:510
```

En este otro caso, nuestro servidor analizará las peticiones provenientes únicamente de la dirección 10.10.10.2 con destino al puerto 510.

Una vez realizado esto, la configuración será realizada de forma muy similar a la hecha en el apartado anterior:

```
<VirtualHost *:80>
    ServerAdmin admin@apache-server.com
    DocumentRoot /home/public/html
    ServerName www.apache-server.com
    ErrorLog logs/www.apache-server.com_error.log
    CustomLog logs/www.apache-server.com_access.log
</VirtualHost>


<VirtualHost *:80>
    ServerAdmin admin@intranet-apache.com
    DocumentRoot /home/intranet/html
    ServerName www.intranet-apache.com
    ServerAlias www.intranet.com
    ErrorLog logs/www.intranet-apache.com_error.log
    CustomLog logs/www.intranet-apache.com_access.log
</VirtualHost
```

Como puede observar, la configuración es igual salvo el la etiqueta inicial <VirtualHost>. Simplemente se ha indicado "*:80" debido a que ha sido el valor que en el parámetro NameVirtualHost se ha indicado previamente. La distinción de que documento se ofrecerá será realizado por medio del parámetro ServerName por lo cual, todos los nombres que se desean atender deberán ser resueltos a la misma dirección. Si se desea entregar los mismos documentos a dos diferentes nombres, puede utilizar el parámetro ServerAlias, como aparece en el segundo ejemplo en la parte superior de este texto, donde el servidor entregará los mismos documentos tanto con el nombre www.intranet-apache.com y www.intranet.com. Recuerde, sin importar el método que utilice, todos los nombres que se utilicen deberán de ser resueltos de alguna forma.

ServerSignature Off

Este parámetro devuelve información sobre el servidor apache, versión módulos cargados, etc. Para un servidor en estado de pruebas está bien tener esta opción en On, pero no para uno en producción. Los valores que puede tomar son On | Off | EMail

ServerTokens Prod

Este parámetro se utiliza para devolver información sobre la architectura sobre la cual se está corriendo el servidor. Si el servidor está en producción se recomienda tener con un valor Prod, apra devolver la menor información a un posible atacante. Los valores que puede tomar son min, devuelve sólo la versión de apache; os devuelve información general sobre el sistema operativo y apache; full devuelve información sobre módulos, apache y sistema operativo; prod sólo devuelve apache.

Algunas recomendaciones extras son:

Un servidor web depende mucho de la memoría disponible, por tanto nunca debe de swapear, si esto comenzara a ocurrir es necesario afinar el parámetro MaxClients a un número menor.

La opción de directorio SymLinksIfOwnerMatch hace llamadas al sistema extras, una por cada componente de archivo. Por tanto no se recomienda activar esta opción desde el directorio raíz, si es necesario se recomienda una sintaxis como la siguiente:

```
documentRoot /www/htdocs
<Directory />
Options FollowSymLinks
</Directory>

<Directory /www/htdocs/checa>
Options –FollowSymLinks +SymLinksIfOwnerMatch
</Directory>
```

La opción AllowOverride hace que intente buscar un archivo .htaccess por cada componente del nombre de archivo. Por ejemplo:

```
documentRoot /www/htdocs
<Directory />
allowOverride all
</Directory>
```

Hace que el servidor al recibir la petición URI/index.html intente abrir los archivos /.htaccess, /www/.htaccess y /www/htdocs/.htaccess. La solución es semejante a la propuesta anteriormente, se recomienda sólo utilizar esta opción en los directorios determinados y si es posible utilizar un mecanismo más robusto de autenticación en caso de que sólo se utilice para esto.

Se recomienda de vez en cuando actualizar la versión de Apache con la que se este trabajando, ya que cada nueva versión corrige errores de las anteriores y son mejores en desempeño.

No se recomienda servir páginas o escribir bitácoras sobre NFS, ya que esto acarrea tráfico extra en la red y retardo por las operaciones I/O sobre el NFS.

Finalmente para saber con que módulos internos está compilado nuestro apache podemos utilizar:

```
root@webfs1i # /usr/local/apache-perl/bin/httpd -l
Compiled-in modules:
  http_core.c
  mod_vhost_alias.c
  mod_env.c
  mod_log_config.c
  mod_mime_magic.c
  mod_mime.c
  mod_negotiation.c
  mod_status.c
  mod_info.c
  mod_include.c
  mod_autoindex.c
  mod_dir.c
  mod_cgi.c
  mod_asis.c
  mod_imap.c
  mod_actions.c
  mod_speling.c
  mod_userdir.c
  mod_alias.c
  mod_rewrite.c
  mod_access.c
  mod_auth.c
  mod_auth_anon.c
  mod_auth_dbm.c
  mod_digest.c
  mod_proxy.c
  mod_cern_meta.c
  mod_expires.c
  mod_headers.c
  mod_usertrack.c
  mod_log_forensic.c
  mod_unique_id.c
  mod_setenvif.c
```

mod_perl.c

El módulo http_core.c es el corazón de los módulos, está ligado estaticamente con el kernel de Apache el provee las funciones básicas que deben encontrarse en cada servidor web Apache. Este módulo es requerido, cualquier otro es opcional.

El módulo mod_so.c da soporte en tiempo de ejcuación para los módulos DSO. Este es requerido si se planea ligar dinámicamente con otros módulos en tiempò de ejecución. Si los módulos son cargados a partir del archivo httpd.conf este módulo debe estar instalado en Apache para soportar la carga dinámica.

Modulo III

"Seguridad en Apache"

Criptología Básica.

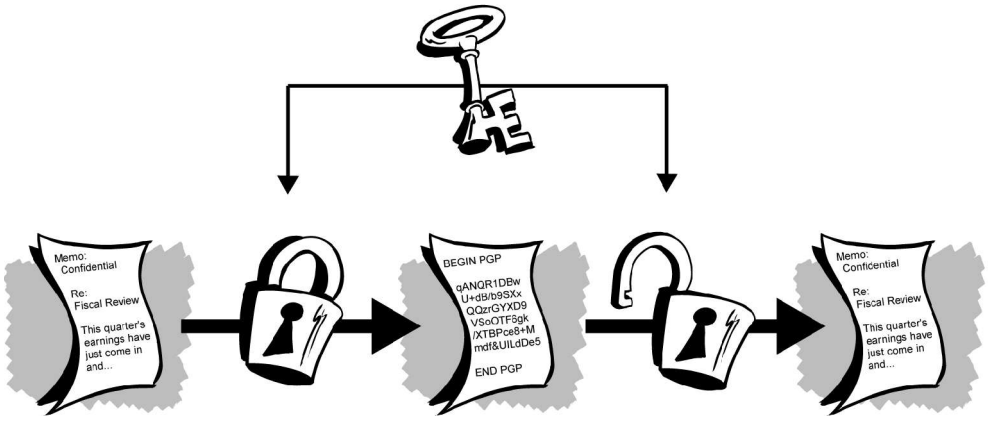
Antes de poder iniciar la configuración de nuestro servidor para soporte de envió de documentos cifrados, es de suma importancia entender los conceptos subyacentes a esto.

La Criptología es el proceso completo de generar texto cifrado y regenerar posteriormente el texto original.

Este proceso requiere de un elemento llamado llave, que nos permite crear el texto cifrado (conocido como criptotexto) y en descifrarlo. La utilización de esta llave está hoy en día regida bajo dos modelos que serán explicados a continuación.

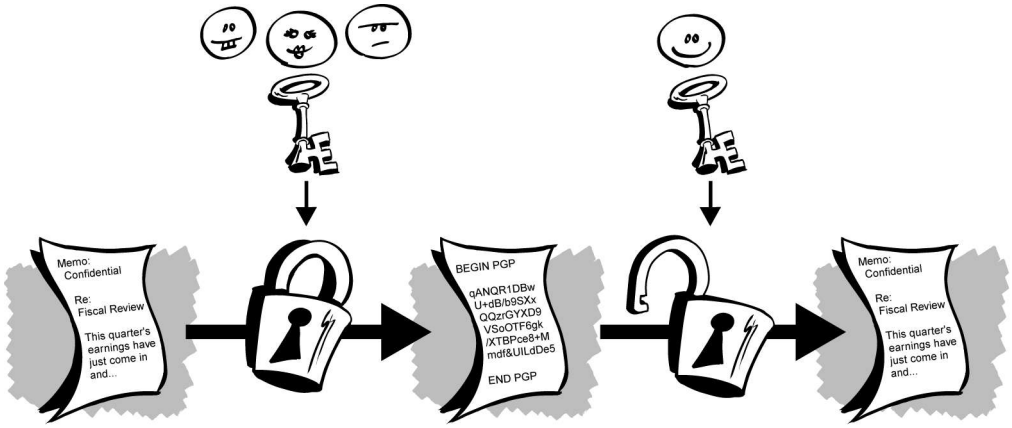
Modelo de Llave Simétrica

En este modelo, la llave realiza la doble función antes descrita: Crea el criptotexto a partir de un texto abierto (conocido como texto plano) y también restaura dicho criptotexto a su forma original.



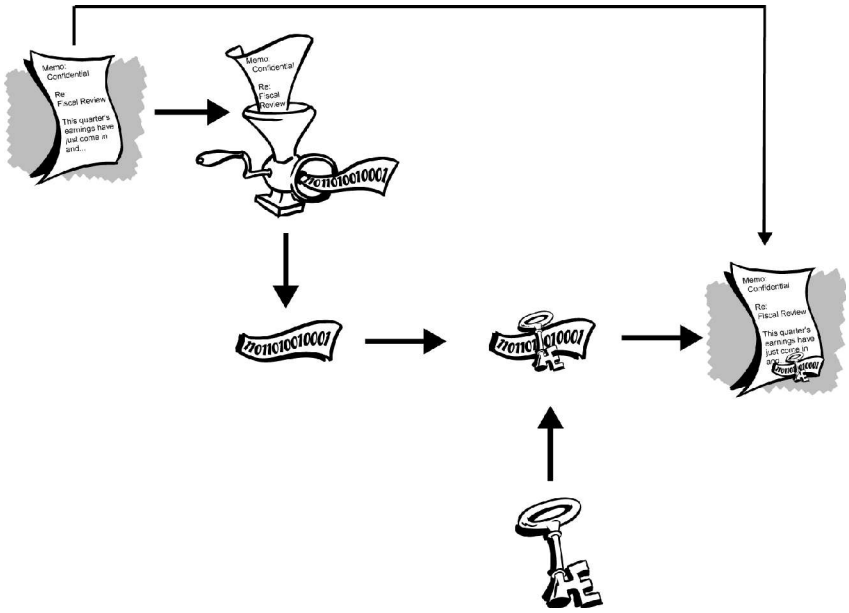
Modelo de Llave Asimétrica

Este otro modelo, pensado cuando el medio de transmisión de la llave es poco confiable, establece la utilización de dos llaves, una designada a crear el criptotexto (conocida como llave pública) y otra para la restauración del mensaje original (conocida como llave privada). Estas son independientes y no pueden obtenerse una de la otra. Para asegurar la confidencialidad, la llave privada quedará sólo en manos del receptor de los mensajes cifrados mientras que la privada se enviará a aquellas personas con quien se desea establecer una comunicación segura.



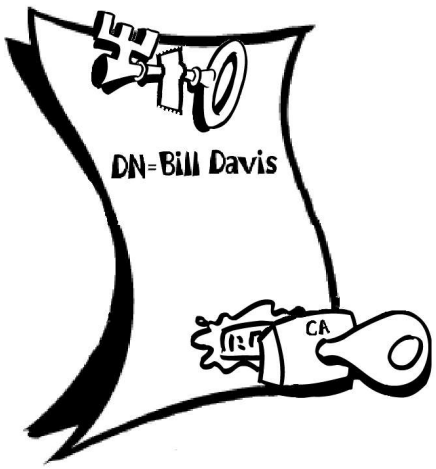
Concepto de Firma Digital

Además de asegurar la transmisión segura de un documento, también requerimos asegurarnos de la integridad de este. Para lograrlo se desarrolló la función hash que, en base a una serie de reglas (conocidas como algoritmo), las cuales obtienen una serie de caracteres que son alterados al mínimo cambio. Lo anterior permite la generación de una Firma Digital que permite corroborar la autenticidad de un documento.



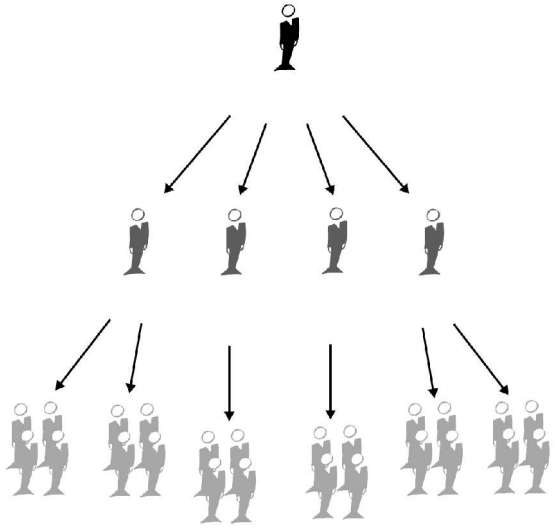
Concepto de Certificado

Un certificado en realidad es una llave pública a la cual se le ha anexado información del dueño de la llave además de la firma digital de un tercero el cual certifica (De ahí su nombre) la autenticidad de la llave anexada.



Concepto de Autoridad Certificadora

Como se mencionó en el punto anterior, todo certificado requiere la firma digital de alguien. Pero si se puede desconfiar de la integridad del mensaje y la llave del emisor, también se puede desconfiar de aquel que firma la llave. Para esto, se han creado empresas que han sido reconocidas por alguna comunidad como fiables.



Una vez comprendido lo anterior, podemos conocer el protocolo que se encuentra detrás de las comunicaciones seguras en Web.

Protocolo Secure Socket Layer (SSL)

El protocolo SSL fue desarrollado por Netscape para permitir confidencialidad y autenticación en Internet. SSL opera como una capa adicional entre Internet y las aplicaciones, esto permite que el protocolo sea independiente de la aplicación, siendo posible utilizar FTP, Telnet y otras aplicaciones además de HTTP.

Para establecer una comunicación segura utilizando SSL se tienen que seguir una serie de pasos. Primero se debe hacer una solicitud de seguridad (Al solicitar en la URL el protocolo de hipertexto seguro, es decir, **https://servidor/documento**). Después de haberla hecho, se deben establecer los parámetros que se utilizarán para SSL. Esta parte se conoce como **SSL Handshake**. Una vez se haya establecido una comunicación segura, se deben hacer verificaciones periódicas para garantizar que la comunicación sigue siendo segura a medida que se transmiten datos. Luego que la transacción ha sido completada, se termina SSL.

Antes de que se establezca SSL, se debe hacer una solicitud. Típicamente esto implica un cliente haciendo una solicitud de una URL a un servidor que soporte SSL. SSL acepta solicitudes por un puerto diferente al utilizado normalmente para ese servicio (El puerto es el número 443).

Una vez se ha hecho la solicitud, el cliente y el servidor empiezan a negociar la conexión SSL, es decir, hacen el SSL Handshake.

SSL Handshake:

Durante el hanshake (reconocimiento mutuo entre servidor y cliente) se cumplen varios propósitos. Se hace autenticación del servidor y opcionalmente del cliente, se determina que algoritmos de criptografía serán utilizados y se genera una llave secreta para ser utilizada durante el intercambio de mensajes subsiguientes durante la comunicación SSL.

Los pasos que se siguen son los siguientes:

- Client Hello : El "saludo de cliente" tiene por objetivo informar al servidor que algoritmos de criptografía puede utilizar y solicita una verificación de la identidad del servidor. El cliente envía el conjunto de algoritmos de criptografía y compresión que soporta y un número aleatorio. El propósito del número aleatorio es para que en caso de que el servidor no posea un certificado para comprobar su identidad, aún se pueda establecer una comunicación segura utilizando un conjunto distinto de algoritmos. Dentro de los protocolos de criptografía hay un protocolo de intercambio de llave que define como cliente y servidor van a intercambiar la información, los algoritmos de llave secreta que definen que métodos pueden utilizar y un algoritmo de hash de una sola vía. Hasta ahora no se ha intercambiado información secreta, sólo una lista de opciones.

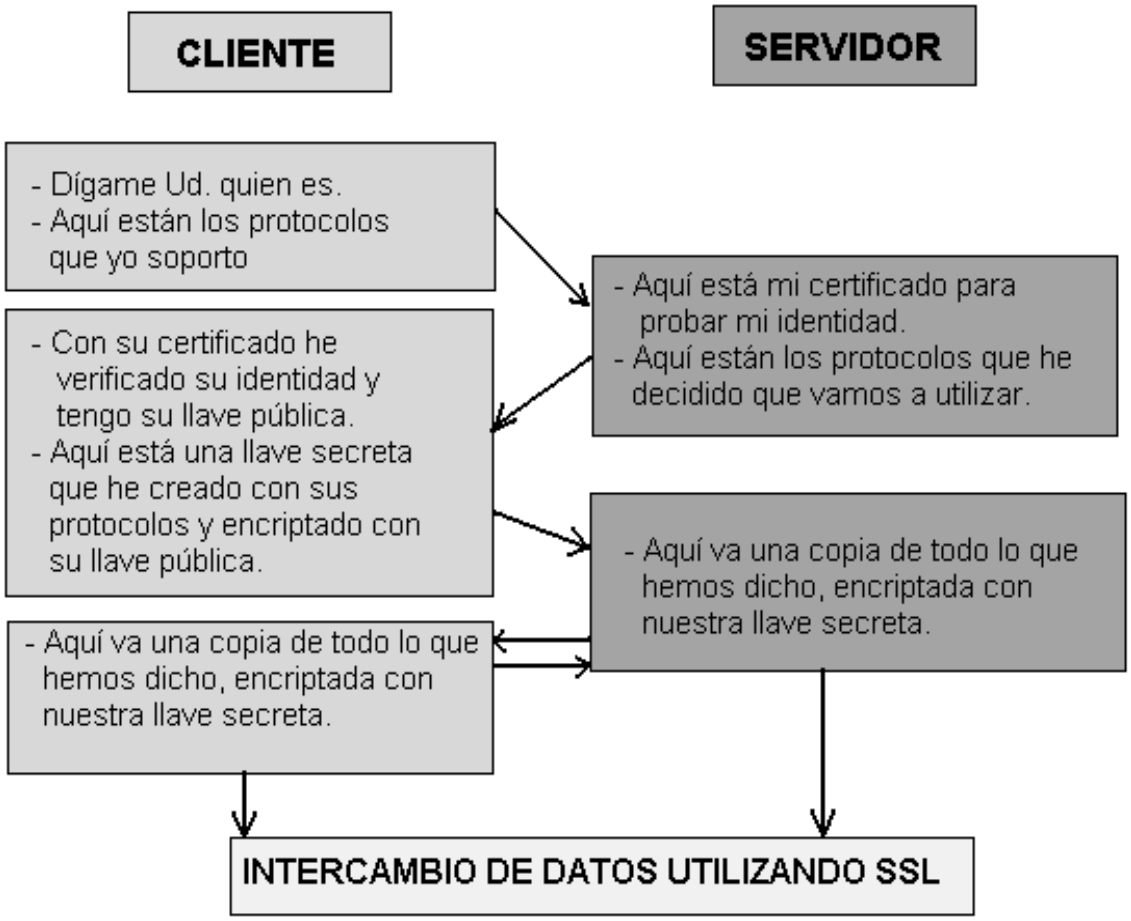
- Server Hello : El servidor responde enviando su identificador digital el cual incluye su llave pública, el conjunto de algoritmos criptográficos y de compresión y otro número aleatorio. La decisión de que algoritmos serán utilizados está basada en el más fuerte que tanto cliente como servidor soporten. En algunas situaciones el servidor también puede solicitar al cliente que se identifique solicitando un identificador digital.

- Aprobación del Cliente: El cliente verifica la validez del identificador digital o certificado enviado por el servidor. Esto se lleva a cabo descifrando el certificado utilizando la llave pública del emisor y determinando si este proviene de una entidad certificadora de confianza. Después se hace una serie de verificaciones sobre el certificado, tales como fecha, URL del servidor, etc. Una vez se ha verificado la autenticidad de la identidad del servidor. El cliente genera una llave aleatoria y la encripta utilizando la llave pública del servidor y el algoritmo criptográfico y de compresión seleccionado anteriormente. Esta llave se le envía al servidor y en caso de que el handshake tenga éxito será utilizada en el envío de futuros mensajes durante la sesión.

- Verificación: En este punto ambas partes conocen la llave secreta, el cliente por que la generó y el servidor por que le fue enviada utilizando su llave pública, siendo la única forma posible de descifrarla utilizando la llave privada del servidor. Se hace una última verificación para comprobar si la información transmitida hasta el momento no ha sido alterada. Ambas partes se envían una copia de las anteriores transacciones cifradas con la llave secreta. Si ambas partes confirman la validez de las transacciones, el handshake se completa, de otra forma se reinicia el proceso.

Ahora ambas partes están listas para intercambiar información de manera segura utilizando la llave secreta acordada y los algoritmos criptográficos y de compresión. El handshake se realiza sólo una vez y se utiliza una llave secreta por sesión.

En la figura se ilustra el proceso de handshake:



Intercambio de datos:

Ahora que se ha establecido un canal de transmisión seguro SSL, es posible el intercambio de datos. Cuando el servidor o el cliente desea enviar un mensaje al otro, se genera un digest (utilizando un algoritmo de hash de una vía acordado durante el handshake), encriptan el mensaje y el digest y se envía, cada mensaje es verificado utilizando el digest.

Terminación de una sesión SSL:

Cuando el cliente deja una sesión SSL, generalmente la aplicación presenta un mensaje advirtiendo que la comunicación no es segura y confirma que el cliente efectivamente desea abandonar la sesión SSL.

Instalación de OpenSSL

Para poder utilizar nuestro servidor Apache con soporte SSL tendremos muchas veces que instalar también OpenSSL, la versión libre de SSL.

La instalación es sumamente sencilla. He aquí los pasos a seguir:

1. Descargue la última versión de OpenSSL de la URL http://www.openssl.org

2. Descomprima el software y colóquese dentro del directorio.
   `# tar -xzvf openssl-x.y.z.tar.gz`

3. Configure por medio del script Configure.
   `./Configure --openssldir <directorio de destino final de OpenSSL> <arquitectura y tipo de binario a utilizar>`

4. Una vez configurado el software, compile e instale mediante de los comandos:
   `#make`
   `#make install`

5. Opcionalmente puede ejecutar el comando:
   `#make test`
   Antes de la instalación con la finalidad de verificar la integridad del software.

Instalación de Apache con soporte SSL vía compilación

Por desgracia, el soporte de SSL para Apache, conocido como mod-ssl requiere la recompilación del mismo. Este software básicamente modifica el código fuente de Apache (operación conocida en argot técnico como parcheo), por lo cual es importante recalcar que cada versión de mod-ssl está **emparentada** con una versión específica de Apache. Tenga mucho en cuenta esto.

El procedimiento a seguir es el siguiente:

1. Descargue el software de la URL http://www.modssl.org.

2. Descomprima es software de la misma carpeta donde se encuentra el código fuente de Apache que previamente instaló.

`# tar -xzvf mod_ssl-2.8.16-1.3.29.tar.gz`

Como puede observar, la versión de OpenSSL 2.8.16 está **emparentada** con la versión 1.3.29 de Apache.

3. Colóquese dentro de la carpeta de código fuente de Apache y ejecute el siguiente comando:

   `#make clean`

Lo cual limpiará cualquier configuración previa existente. Si así lo desea, saque una copia de respaldo de algún archivo Makefile existente.

4. Colóquese dentro de la carpeta de mod-ssl y ejecute lo siguiente:

```
#./configure --with-apache=../apache_1.3.29 --with-
ssl=../openssl-0.9.7c  --enable-module=so --enable-shared=status
--enable-shared=vhost_alias --enable-shared=ssl --with-
perl=/usr/bin/perl --prefix=/opt/apachessl
```

Como puede observar, debido a que limpiamos la configuración original de nuestro servidor debemos de volver a reconfigurar la opciones originales que habíamos seleccionado. También hemos escogido el directorio **/opt/apachessl** como destino de nuestro software. Este comando realizará el parchado y la configuración del software a ser compilado:

```
Configuring mod_ssl/2.8.16 for Apache/1.3.29
 + Apache location: ../apache_1.3.29 (Version 1.3.29)
 + OpenSSL location: ../openssl-0.9.7c
 + Auxiliary patch tool: ./etc/patch/patch (local)
 + Applying packages to Apache source tree:
   o Extended API (EAPI)
   o Distribution Documents
   o SSL Module Source
   o SSL Support
   o SSL Configuration Additions
   o SSL Module Documentation
   o Addons
Done: source extension and patches successfully applied.

Configuring for Apache, Version 1.3.29
 + using installation path layout: Apache (config.layout)
Creating Makefile
Creating Configuration.apaci in src
Creating Makefile in src
 + configured for Linux platform
 + setting C compiler to gcc
 + setting C pre-processor to gcc -E
 + using "tr [a-z] [A-Z]" to uppercase
 + checking for system header files
 + adding selected modules
    o ssl_module uses ConfigStart/End
      + SSL interface: mod_ssl/2.8.16
      + SSL interface build type: OBJ
      + SSL interface compatibility: enabled
      + SSL interface experimental code: disabled
      + SSL interface conservative code: disabled
      + SSL interface vendor extensions: disabled
      + SSL interface plugin: Built-in SDBM
      + SSL library path: /root/openssl-0.9.7c
      + SSL library version: OpenSSL 0.9.7c 30 Sep 2003
      + SSL library type: source tree only (stand-alone)
 + enabling Extended API (EAPI)
 + using system Expat
```

```
 + using -ldl for vendor DSO support
 + checking sizeof various data types
 + doing sanity check on compiler and options
Creating Makefile in src/support
Creating Makefile in src/regex
Creating Makefile in src/os/unix
Creating Makefile in src/ap
Creating Makefile in src/main
Creating Makefile in src/modules/standard
Creating Makefile in src/modules/ssl
```

5. Una vez realizada la configuración y la modificación del código fuente, procederemos a la compilación de nuestro nuevo servidor Apache con soporte a SSL. Esto se hará en la carpeta del código fuente de apache.

```
# cd ../apache_1.3.29
#make
```

6. Una vez realizada la compilación, se solicitará la creación de un certificado. Para efecto de prueba, generaremos un certificado sin validez alguna (Conocido como comúnmente como tipo DUMMY)

```
#make certificate type=dummy
```

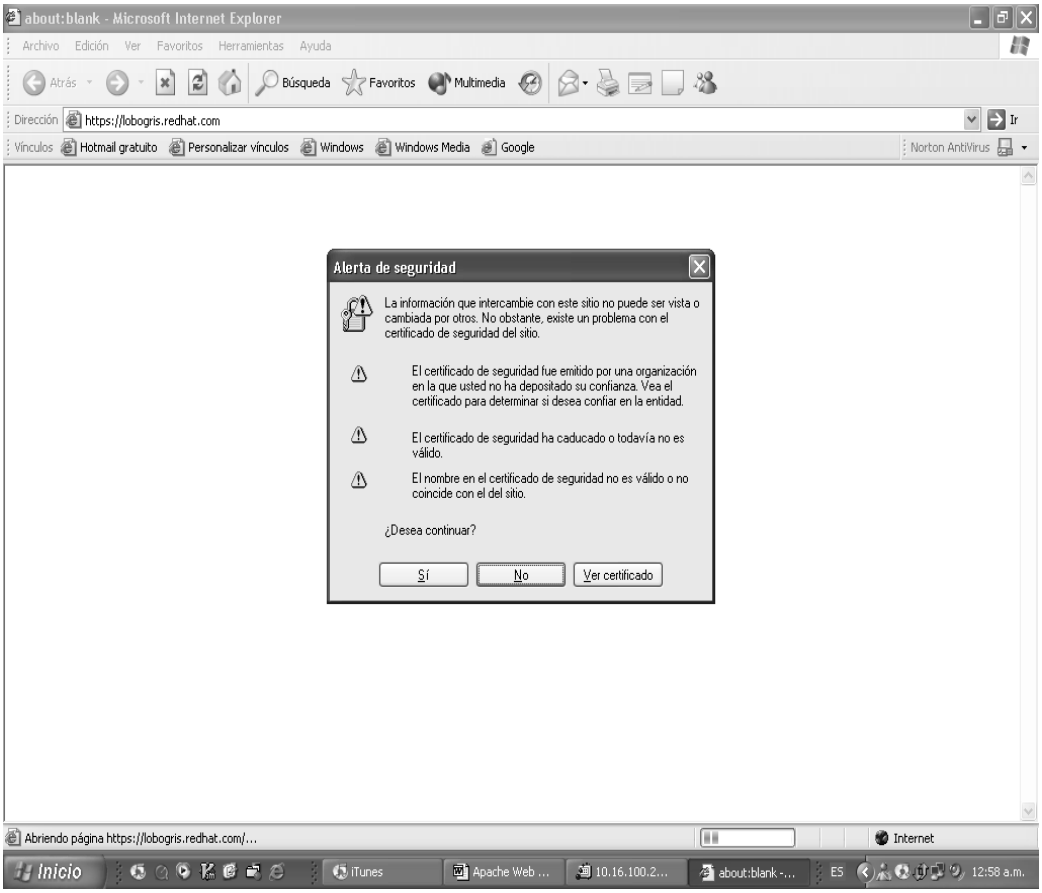
7. Finalmente, instalaremos todo el software con los certificados recientemente creados por medio del comando:

```
#make install
```

8. Una vez realizado lo anterior, podemos iniciar nuestro servidor con el comando:

```
#/opt/apachessl/bin/apachectl startssl
```

9. Verificaremos el funcionamiento de nuestro servidor recién instalado utilizando un navegador con soporte a SSL.



Generación de una llave privada

Si bien el procedimiento anterior nos sirve para poder generar un certificado de prueba, al momento de sustituir este certificado por uno real, requeriremos de otros procedimientos adicionales. OpenSSL, además de incluir las librerías antes mencionadas, también posee una interfaz de texto para la generación de llaves, solicitudes de certificado (CSR) y certificados de auto-firma. Para crear una llave privada necesaria para crear una solicitud de certificado realice lo siguiente una vez colocado el directorio de OpenSSL (En nuestro caso, el directorio **/opt/openssl**):

```
#cd bin
#openssl genrsa -des3 -rand –out llave1.key 1024
```

Lo cual generará la siguiente salida:

```
Generating RSA private key, 1024 bit long modulus
...................++++++
.............................++++++
e is 65537 (0x10001)
Enter PEM pass phrase:
Verifying password - Enter PEM pass phrase:
```

Una vez creado aparecerá un archivo de texto[1] con el nombre designado (llave1.key). El contenido del archivo es el siguiente:

```
#cat llave1.key

-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: DES-EDE3-CBC,6F17F46057748981
9inqDQQDkrgSS7+PO+QCT5AqTuF2MtODv3az0ZQ4U3annTircFf/zj7qM9js+4Qv
a2LpSXkdqjgfft4mJQVp45AZjWNecf6XJBUxKirae/kEcJUrDC1JGRA701ZyMnWl
8d3vwmCSieAWFoTuczY0JlFJnL+QEV2bZi0BWXYG+f0+Bngy5MsaFJ2co1fDyNmR
ymW+mo/zPmUET9oTCZ7JzzT9SqelIMomLJPGM4tyZorQByix9wSifMW1jUrDKSrn
MwVPe8MYbXgHjU3KpBZ32EHji2k4FVVQehiWqR1UwZfOoGP9JjKcN8bWHXswYiJT
N3iLw+QpMGwgD0ZDmS5AiY4crslUZE9jaKUScwFogZLMdbKOSYOibyBMfFiAEymK
+dMSrp6uG+vRG6fY5LNPL02OBRYIerACUB6Ao4DHjyRdlg2AsxG7CibVM+TMeJ8V
a26Mpu8xN1BHSOib+zpE6PV7H18k6p4opD0wDuOmCcbRPwlH6FJJDQKVHypC1zqy
qXuNAKE66Lazm/P8i4AGZ1aqvN3ilZdoGGf9k/U1/N2YKvS/lJ/Ae1ftTyunfHKb
+VtsKTJpVKzXAFDq8v3BtA0Wrp4k6zPPVYGuSHbhheHsW9aFIvnqeuTLhMTN1VMD
CZh0KI01xEKLbTHbDk7wvNrjJL/FoDk8UhNagKZCxz+9sw3+N84kOpvD33GI2gtK
S7TsT/XjUwce5GQWUlmvm6srPDyldjH8Qik83uCrr95RtRR2Z9tJxrlQa1JjOSU+
gMxIxePeh2VMDyZ9ZBs12eJ+FnUzFANOSCtd7UUmwHlpx998HQa9Fg==
-----END RSA PRIVATE KEY-----
```

Generación de una solicitud de certificado (CSR)

Para que una autoridad certificadora pueda generarnos un certificado es necesario generar una solicitud que posteriormente será enviada (generalmente vía Web) a un servidor que nos regresará nuestro certificado. Para generar el CSR realizaremos lo siguiente:

```
#openssl req -new -key llave1.key –out precert.csr


Using configuration from /etc/ssl/openssl.cnf
Enter PEM pass phrase:
You are about to be asked to enter information that will be
incorporated
into your certificate request.
What you are about to enter is what is called a Distinguished Name or
a DN.
There are quite a few fields but you can leave some blank
For some fields there will be a default value,
If you enter '.', the field will be left blank.
-----
Country Name (2 letter code) [AU]:MX
State or Province Name (full name) [Some-State]:DF
Locality Name (eg, city) []:Mexico
```

[1] En realidad el archivo resultante NO ES totalmente un archivo de texto. Este se encuentra en un formato especial conocido como PEM - que contiene información confidencial en un formato de texto cifrado .

```
Organization Name (eg, company) [Internet Widgits Pty Ltd]:ILCE
Organizational Unit Name (eg, section) []:Desarrollo
Common Name (eg, YOUR name) []:Antonio Galindo Castro
Email Address []:superfunkylistic@yahoo.com
Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:
An optional company name []:
```

Como se puede observar, el programa realiza diversas preguntas para anexar la información correspondiente al certificado. Al igual que en el punto anterior, el programa crea un archivo de texto con la información de la solicitud de certificado:

```
#cat precert.csr

-----BEGIN CERTIFICATE REQUEST-----
MIIByzCCATQCAQAwgYoxCzAJBgNVBAYTAk1YMQswCQYDVQQIEwJERjEPMA0GA1UE
BxMGTWV4aWNvMQwwCgYDVQQKEwNTQ08xEDAOBgNVBAsTB1NvcG9ydGUxGjAYBgNV
BAMTEUp1YW4gUmFmYWVsIEdvbWV6MSEwHwYJKoZIhvcNAQkBFhJqcmdvbWV6QHNj
by5jb20ubXgwgZ8wDQYJKoZIhvcNAQEBBQADgY0AMIGJAoGBAOCo3iKK1E6K4bjw
DrGnEHL7HyZoF85ExEF/9cgq6Bybrta5iyjmTeNDm5gXmDhZtaqzho+Y1+xed6xa
diSA6VATIhwWSG0yccO6ptmQK0r0tiBMFosyL20p7vAXNsIK7+M/8jKgNP/sURse
8mA6aLjtlKVNGw2XxtWkHLLvdNpLAgMBAAGgADANBgkqhkiG9w0BAQQFAAOBgQCC
fDbHqkMYUXuevZBWre0ZxHaJ0W5hmHv8j4KBS44f/mwyKRdbs27UOPzgxoPHPuHK
OyYXbZD4SoChocxPmCoV4qYJ+Rezuk7SuRrMmeX1U9SXO8e4SIKx4okcihyzcFoc
/JwWZFQy4ah/ykI5+bqK+SLaMJkYMx3UOpc6PZrpnQ==
-----END CERTIFICATE REQUEST-----
```

Finalmente la información es enviada a la Entidad Certificadora que nos remitirá un certificado.

Generación de un certificado de auto-firma

OpenSSL tiene la capacidad de poder crear certificados de “Auto Firma” para efectos de prueba; es decir, un certificado firmado con la misma llave privada de la solicitud. Esto nos permite poder hacer pruebas con programas que utilicen certificados sin necesidad de adquirir uno.

Para emitir un certificado de este tipo ejecutaremos lo siguiente:

```
#openssl x509 -req -days 30 -in precert.csr -signkey llave1.key -out
certificate.cert

Signature ok
subject=/C=MX/ST=DF/L=Mexico/O=ILCE/OU=Desarrollo/CN=Antonio Galindo
Castro/Email=superfunkylistic@yahoo.com
Getting Private key
Enter PEM pass phrase:
```

Podemos visualizar el certificado creado:

```
#cat certificate.cert

-----BEGIN CERTIFICATE-----
MIIChTCCAe4CAQAwDQYJKoZIhvcNAQEEBQAwgYoxCzAJBgNVBAYTAk1YMQswCQYD
VQQIEwJERjEPMA0GA1UEBxMGTWV4aWNvMQwwCgYDVQQKEwNTQ08xEDAOBgNVBAsT
B1NvcG9ydGUxGjAYBgNVBAMTEUp1YW4gUmFmYWVsIEdvbWV6MSEwHwYJKoZIhvcN
AQkBFhJqcmdvbWV6QHNjby5jb20ubXgwHhcNMDMwNTMwMTYwMTU4WhcNMDMwNjI5
MTYwMTU4WjCBijELMAkGA1UEBhMCTVgxCzAJBgNVBAgTAkRGMQ8wDQYDVQQHEwZN
ZXhpY28xDDAKBgNVBAoTA1NDTzEQMA4GA1UECxMHU29wb3J0ZTEaMBgGA1UEAxMR
SnVhbiBSYWZhZWwgR29tZXoxITAfBgkqhkiG9w0BCQEWEmpyZ29tZXpAc2NvLmNv
bS5teDCBnzANBgkqhkiG9w0BAQEFAAOBjQAwgYkCgYEA4KjeIorUTorhuPAOsacQ
cvsfJmgXzkTEQX/1yCroHJuu1rmLKOZN40ObmBeYOFm1qrOGj5jX7F53rFp2JIDp
UBMiHBZIbTJxw7qm2ZArSvS2IEwWizIvbSnu8Bc2wgrv4z/yMqA0/+xRGx7yYDpo
uO2UpU0bDZfG1aQcsu902ksCAwEAATANBgkqhkiG9w0BAQQFAAOBgQBWZyvs79Tc
UtMAZjjDi8iYVPdJjTNzydtiLZ9Qqv51cevYosn/+QADLnRrjq6/W6ZWYrm5sbVg
fgHsNIP9iRCDVbPXV0XtKXebD8r2SsIpeMD92b1UMs71ksQ5IgP9IRDzfQ0m2XHZ
b7ZOZMvQtFn3igRlVXCAzvLus/mVysUyFw==
-----END CERTIFICATE-----
```

Detengámonos por un momento y analicemos lo que hemos hecho:

- Al momento de configurar el código fuente de Apache, le indicamos la opción

--enable-shared=ssl, por lo cual deberá existir el módulo correspondiente en la carpeta **/opt/apachessl/libexec**:

```
#ls /opt/apachessl/libexec
httpd.exp  libssl.so  mod_status.so
```

- Dentro de la carpeta conf de nuestro árbol de directorios de Apache con soporte aparecerán unos directorios que contienen los certificados de la entidad falsa configurada durante la instalación:

```
#ls /opt/apachessl/conf

access.conf             httpd.conf.default      srm.conf.default
access.conf.default     magicssl.crl            ssl.csr
httpd.conf              mime.types              ssl.key
httpd.conf.default      mime.types.default
```

- En el archivo de configuración general httpd.conf tenemos nuevos módulos a cargar y nuevas entradas <IfDefine> :

```
<IfDefine SSL>
LoadModule ssl_module         libexec/libssl.so
</IfDefine>
```

```
      <IfDefine SSL>
      AddModule mod_ssl.c
      </IfDefine>

      <IfDefine SSL>
      Listen 80
      Listen 443
      </IfDefine>

#</VirtualHost>

#<VirtualHost _default_:*>
#</VirtualHost>

<IfDefine SSL>
AddType application/x-x509-ca-cert .crt
AddType application/x-pkcs7-crl    .crl
</IfDefine>

<IfModule mod_ssl.c>

#SSLSessionCache        none
#SSLSessionCache        shmht:/opt/apachessl/logs/ssl_scache(512000)
#SSLSessionCache        shmcb:/opt/apachessl/logs/ssl_scache(512000)
SSLSessionCache         dbm:/opt/apachessl/logs/ssl_scache
SSLSessionCacheTimeout  300

SSLMutex  file:/opt/apachessl/logs/ssl_mutex

SSLRandomSeed startup builtin
SSLRandomSeed connect builtin
#SSLRandomSeed startup file:/dev/random  512
#SSLRandomSeed startup file:/dev/urandom 512
#SSLRandomSeed connect file:/dev/random  512
#SSLRandomSeed connect file:/dev/urandom 512

SSLLog      /opt/apachessl/logs/ssl_engine_log
SSLLogLevel info


</IfModule>

<IfDefine SSL>


<VirtualHost _default_:443>
DocumentRoot "/opt/apachessl/htdocs"
ServerName xipe.orbis.org.mx
ServerAdmin root@xipe.orbis.org.mx
ErrorLog /opt/apachessl/logs/error_log
TransferLog /opt/apachessl/logs/access_log

SSLEngine on
```

```
SSLCipherSuite
ALL:!ADH:!EXPORT56:RC4+RSA:+HIGH:+MEDIUM:+LOW:+SSLv2:+EXP:+eNULL

SSLCertificateFile /opt/apachessl/conf/ssl.crt/server.crt
#SSLCertificateFile /opt/apachessl/conf/ssl.crt/server-dsa.crt

SSLCertificateKeyFile /opt/apachessl/conf/ssl.key/server.key
#SSLCertificateKeyFile /opt/apachessl/conf/ssl.key/server-dsa.key

#SSLCertificateChainFile /opt/apachessl/conf/ssl.crt/ca.crt

#SSLCACertificatePath /opt/apachessl/conf/ssl.crt
#SSLCACertificateFile /opt/apachessl/conf/ssl.crt/ca-bundle.crt

#SSLCARevocationPath /opt/apachessl/conf/ssl.crl
#SSLCARevocationFile /opt/apachessl/conf/ssl.crl/ca-bundle.crl

#   Client Authentication (Type):
#   Client certificate verification type and depth.  Types are
#   none, optional, require and optional_no_ca.  Depth is a
#   number which specifies how deeply to verify the certificate
#   issuer chain before deciding the certificate is not valid.
#SSLVerifyClient require
#SSLVerifyDepth  10

#<Location />
#SSLRequire (    %{SSL_CIPHER} !~ m/^(EXP|NULL)/ \
#            and %{SSL_CLIENT_S_DN_O} eq "Snake Oil, Ltd." \
#            and %{SSL_CLIENT_S_DN_OU} in {"Staff", "CA", "Dev"} \
#            and %{TIME_WDAY} >= 1 and %{TIME_WDAY} <= 5 \
#            and %{TIME_HOUR} >= 8 and %{TIME_HOUR} <= 20       ) \
#           or %{REMOTE_ADDR} =~ m/^192\.76\.162\.[0-9]+$/
#</Location>

#SSLOptions +FakeBasicAuth +ExportCertData +CompatEnvVars +StrictRequire
<Files ~ "\.(cgi|shtml|phtml|php3?)$">
    SSLOptions +StdEnvVars
</Files>
<Directory "/opt/apachessl/cgi-bin">
    SSLOptions +StdEnvVars
</Directory>

SetEnvIf User-Agent ".*MSIE.*" \
         nokeepalive ssl-unclean-shutdown \
         downgrade-1.0 force-response-1.0

CustomLog /opt/apachessl/logs/ssl_request_log \
          "%t %h %{SSL_PROTOCOL}x %{SSL_CIPHER}x \"%r\" %b"

</VirtualHost>

</IfDefine>
```

- Se abrió el puerto 443 para recibir y enviar por medio de este peticiones SSL.
- Se generaron diversas entradas <IfDefine> para la carga del módulo **libssl**.
- Se agregó el reconocimiento de los archivos con extensión .crt como certificados de tipo X.509 y pkcs7-crl
- Se generó una entrada para servidor virtual de manera que la entrega de documentos por canal seguro no interfiriera con operaciones no cifradas.
- Se creo una entrada para la generación de una bitácora de las solicitudes ssl en el archivo **/opt/apachessl/logs/ssl_request_log**.
- Se establecieron condiciones de ambiente en casos especiales, como la solicitud de certificado SSL por parte de un navegador Internet Explorer.

Además de lo anterior, los parámetros para sustitución de certificados que nos será necesario conocer son los siguientes:

- SSLCertificateFile: Este parámetro indica la ruta del archivo de certificado.
- SSLCertificateKeyFile: Este parámetro indica la llave privada necesaria para la generación de llaves públicas de sesión SSL.

Autenticación básica por usuarios.

Con la finalidad de establecer criterios de acceso, la información publicada por Apache puede ser restringida. Esta restricción puede ser dividida en dos tipos:

De forma global: Es decir, el acceso o negación a uno o más equipos en una red de forma completa.

De forma personalizada: El acceso a los documentos es por medio de listas de usuarios independientes a los usuarios de sistema.

Para configurar ambas formas es necesario crear un apartado de configuración por cada directorio de documentos o ejecutables que se quiera restringir. La sintaxis es similar a la utilizada en HTML. Observemos el siguiente ejemplo:

```
<Directory "/home/httpd/htdocs">
Options Indexes FollowSymLinks
AllowOverride None
Order allow,deny
Allow from all
</Directory>
```

Como puede notarse, todos los valores están “encerrados” por la etiqueta <Directory> , siendo la única diferencia la diagonal invertida, indicando el cierre de dicha etiqueta. El primer valor importante es “AllowOverride”, el cual nos indica si el servidor Web Apache deberá o no hacer caso omiso a archivos .htaccess que se encuentren en el directorio declarado por la etiqueta <Directory>.

También se pueden declarar diversas opciones que nos facilitan la configuración de las carpetas, por ejemplo la entrada Option Indexes FollowSymLinks le indica a Apache que siga las ligas simbólicas y que, en caso de que no exista algún tipo de archivo declarado en la directiva DirectoryIndex, muestre el contenido del directorio en cuestión. Para la negación de servicio a nivel de hosts. la última entrada en el ejemplo es la que nos interesa, la palabra Order nos indica que la carpeta manejará criterios de acceso a ella, como opciones pueden utilizase las palabras:

allow,deny: Indica que se leerán primero la entradas allow antes que las deny. El acceso es denegado por defecto; es decir, aquellos casos que no coincidan con ninguno de los presentados, no podrán acceder a la carpeta. Cualquier usuario que coincida con la directiva deny o no coincida con la directiva allow, le será denegado el acceso.

deny,allow: Al contrario del punto anterior, las entradas deny serán consideradas antes que las allow, El acceso es permitido por defecto. Cualquier usuario que coincida con la directiva allow y no coincida con la directiva deny, le será permitido el acceso.

Mutual-failure: Solo aquellos equipos que aparezca en Allow y no aparezcan en deny se les permitirá el acceso a la carpeta.

En el ejemplo previamente presentado permite acceder a cualquier persona a la carpeta raíz de documentos. Si, por ejemplo, no deseamos que cualquier usuario de la red 10.16.101.0/255.255.255.0 acceda a la carpeta **home/documentos/html/ultrasecreto**, deberemos realizar lo siguiente:

```
<Directory /home/documentos/html/ultrasecreto>
AllowOverride None
Order Allow,Deny
Deny from 10.16.101.0/255.255.255.0
Allow from all
</Directory>
```

Para realizar la negación de acceso a ciertas partes de nuestro web en base a listas de acceso, es necesario realizar lo siguiente:

Crear una entrada <Directory> con las siguientes entradas:

```
<Directory /home/documentos/html/ultrasecreto>
AllowOverride None
AuthType Basic
AuthName “Acceso Restringido”
AuthUserFile /etc/cuentas-apache
</Directory>
```

La primera entrada ya fue explicada en el punto anterior.

La entrada AuthType indica el tipo de autenticación a manejar. En este caso la palabra Basic indica que la autenticación será de forma individual. La entrada AuthName le indica a Apache que título tendrá la ventana de díalogo solicitando el nombre y la contraseña de usuario válida. La entrada AuthUserFile indica el archivo que contiene las cuentas de usuario. Este archivo es creado por el programa htpasswd (incluido con la suite de programas de Apache) con la siguiente sintaxis:

```
#htpasswd –c /etc/cuentas-apache superfunkylistic
New password:******
New password:******
Adding password for user superfunkylistic.
```

En el archivo resultante aparece el siguiente contenido:

```
superfunkylistic:7NN3YAnPrFRL.
```

A partir del segundo usuario se omite la opción –c del comando htpasswd.

La entrada require indica el o los usuarios válidos para acceder a la información. Las opciones pueden ser:

Valid-user: Cualquier usuario que este declarado en el archivo indicado en **AuthUserFile** puede acceder.
User: El o los usuarios indicados después de esta palabra.

Una vez configurado habrá que reiniciar Apache para que los cambios sean tomados.

Autenticación básica por grupos.

En algunas versiones antiguas de Apache, el poner después de la palabra user más de un usuario no funciona, por lo cual se le debe indicar a Apache que se utilizarán grupos de usuarios validos para acceder a la información deseada.
La configuración quedaría de la siguiente forma:

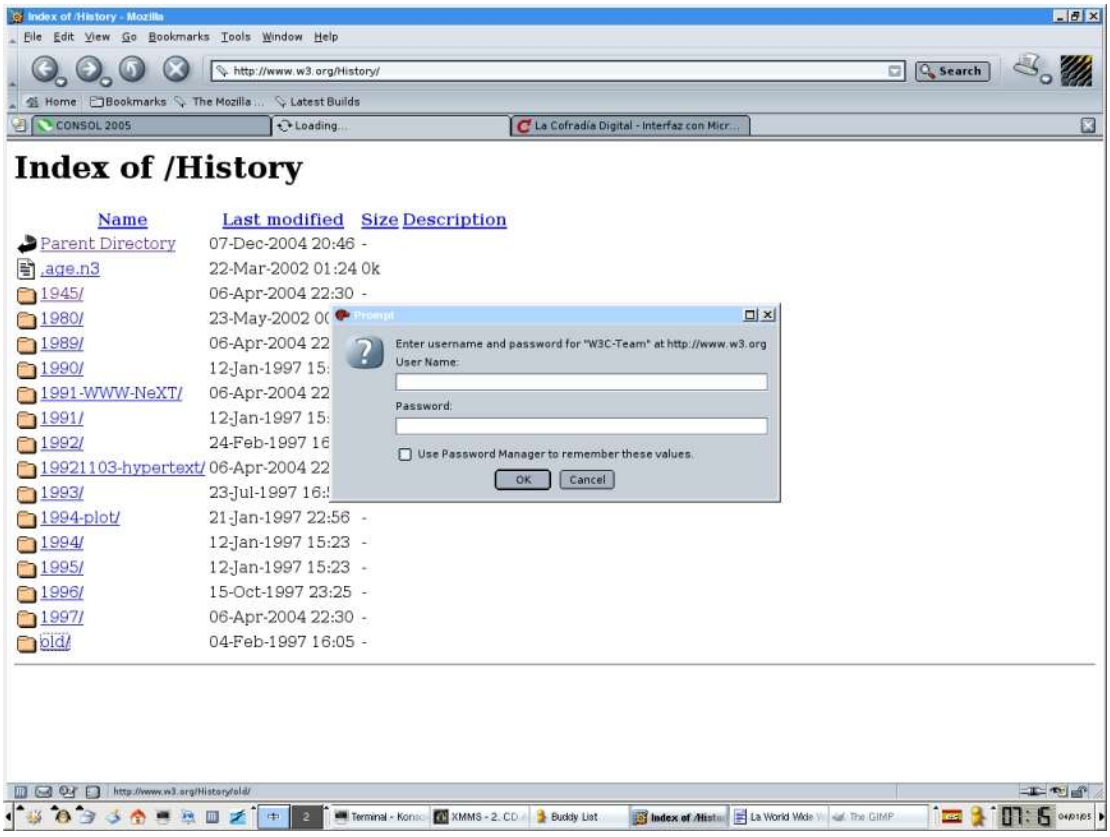
```
<Directory /home/documentos/html/ultrasecreto>
AllowOverride None
AuthType Basic
AuthName “Acceso Restringido”
AuthUserFile /etc/cuentas-apache
AuthGroupFile /etc/grupos-apache
require group ventas
</Directory>
```

Como se puede observar, la entrada AuthGroupFile indica la ubicación de un archivo y, en la entrada require en lugar de indicar un usuario se coloca un grupo y el nombre relacionado (ventas).

El archivo indicado por AuthGroupFile tiene el siguiente formato:

```
ventas: superfunkylistic jose
```

Independientemente del tipo de configuración que seleccione deberá reiniciar el servidor (kill –HUP número de proceso de httpd o por medio del script de arranque, generalmente ubicado en /etc/rc.d/init.d ó /etc/rc.d/rc3.d) y verifique su funcionamiento.



Modulo IV

"Apache como servidor de aplicaciones Web y contenido dinámico"

Definición de Common Gateway Interface (CGI)

Además de la publicación de documentos, Apache tiene la capacidad de ejecutar programas por medio de la interfaz Web gracias a los CGI's. Estos a grandes rasgos son programas realizados un cualquier lenguaje comprendido por el sistema operativo (C, C++, Perl, Shell, etc.), cuya salida del programa es en lenguaje HTML u otro comprendido por el servidor.

Es importante aclarar que, para que esto funcione, es necesario que Apache pueda distinguir entre directorios que contienen CGI's, y aquellos que contienen documentos. Por lo cual, **NO** podemos colocar documentos y CGI's juntos ya que para Apache le sería imposible distinguir entre ellos. Al hacer dicha distinción, Apache **primero** ejecuta el CGI y posteriormente publica la salida de este.

Configuración de Apache para soporte de CGI

Como se mencionó en el módulo que analiza el archivo de configuración, hay dos formas de indicar a nuestro servidor Apache la existencia de un directorio de ejecutables CGI

a) Por medio de la sentencia ScriptAlias:

```
ScriptAlias /misproductos/cgi-bin   /programasweb
```

b) Por medio del parámetro ExecCGI dentro de una declaración de directorio o un archivo .htaccess o su equivalente:

```
<Directory /misproductos/cgi-bin>
Options AllowOverride None
ExecCGI
</Directory>
```

Programación Básica de CGI

Con la finalidad de que comprenda mejor el concepto de CGI, escriba un script de nombre “prueba.cgi” con el código que se muestra a continuación y colóquelo en un directorio previamente declarado como directorio de ejecutables CGI:

```
----------CORTE AQUÍ----------
    #!/bin/bash
    #
    # Cgi de prueba hecho con shell scripting.
    #
    echo Content-type: text/html
    echo
    echo "<HTML>\n<HEAD>”
    echo "<title>CGI de prueba</title>"
    echo “</HEAD>\n<BODY bgcolor=\”#ffffff\”>\n”
    echo "<h3>Hola mundo, la hora es: </h3>\n"
    echo $(/bin/time)
    echo "</BODY>\n</HTML>”
----------CORTE AQUÍ----------
```

Posteriormente, acceda al ejecutable vía un navegador y vea el resultado.

Es importante notar la línea **echo Content-type: text/html** ya que esta es la que le indica que el documento es precisamente una página web escrita en HTML.

Introducción a Hypertext Preprocessor (PHP)

A diferencia de un CGI, PHP es un lenguaje para generación de contenido dinámico que va incrustado **dentro** de una hoja HTML, Esto nos da la ventaja que no tenemos que programar nada por fuera de la hoja (como sucede con los CGI’s) y, en caso de error, se puede observar y descubrir el código erróneo.

Lo que distingue a PHP de la tecnología JavaScript, la cual se ejecuta en la máquina cliente, es que el código PHP es ejecutado en el servidor. Si tuviésemos un script similar al de nuestro ejemplo en nuestro servidor, el cliente solamente recibiría el resultado de su ejecución en el servidor, sin ninguna posibilidad de determinar que código ha producido el resultado recibido. El servidor Web puede ser incluso configurado para que procese todos los ficheros HTML con PHP.

Lo mejor de usar PHP es que es extremadamente simple para el principiante, pero a su vez, ofrece muchas características avanzadas para los programadores profesionales, junto con una gran variedad de funciones que facilitan el trabajo.

Instalación de PHP vía compilación

La instalación de PHP como módulo de Apache es sumamente sencilla. Para lograrlo, nos apoyaremos del compilador de módulos del mismo Apache de nombre apxs. Realice el siguiente procedimiento:

1. Descargue el código fuente de PHP desde la URL http://www.php.net y colóquelo, si así lo desea, dentro de la carpeta con la que ha estado trabajando hasta el momento.

2. Descomprima el código y colóquese dentro de la carpeta resultante.

3. Configure el código fuente por medio del script configure:

 # ./configure --with-apxs=/opt/apachessl/bin/apxs --with-mysql --with-openssl=/opt/openssl/ --with-calendar --prefix=/opt/php

Como puede observar, hemos indicado al script de configuración la ruta de ubicación de apxs, que tenga soporte a conexión de la base de datos MySQL, que anexe soporte para Calendario Juliano y OpenSSL además de que sea instalado en la carpeta **/ opt/php**,

4. Una vez configurado, procederemos a compilar por medio del comando make.

#make

5. Podemos, si así lo deseamos verificar la integridad de nuestro software recién compilado mediante la instrucción:

#make test

6. Finalmente, podemos proceder a la instalación de PHP por medio del comando:

#make install.

Si todo fue realizado correctamente, en la carpeta libexec de nuestro servidor Apache, aparecerá el módulo de PHP (en este caso libphp4.so) y nuestro archivo de configuración habrá sido sustituido por una copia con las respectivas adiciones para la carga del módulo de PHP.

Modificación de Apache para soporte a php

Para la utilización de PHP en Apache, es necesario agregar un nuevo tipo MIME para indicarle al servidor cuando llamará al intérprete PHP. Simplemente agregue las siguientes líneas en su archivo httpd.conf.

AddType application/x-httpd-php .php

Y, si así se requiere, la búsqueda de Apache de índices de documentos en formato PHP:

DirectoryIndex index.html index.cgi index.php Entrada anexada.

Una vez realizado esto y, habiendo reiniciado nuestro servidor, podemos comenzar a realizar algo de programación en este interesante lenguaje.

Página de Prueba

Tal vez usted se preguntará: ¿Cómo puedo corroborar si mi servidor Apache esta realmente entendiendo lenguaje PHP? ¿Cómo puedo saber que es lo que soporte mi instalación de PHP?

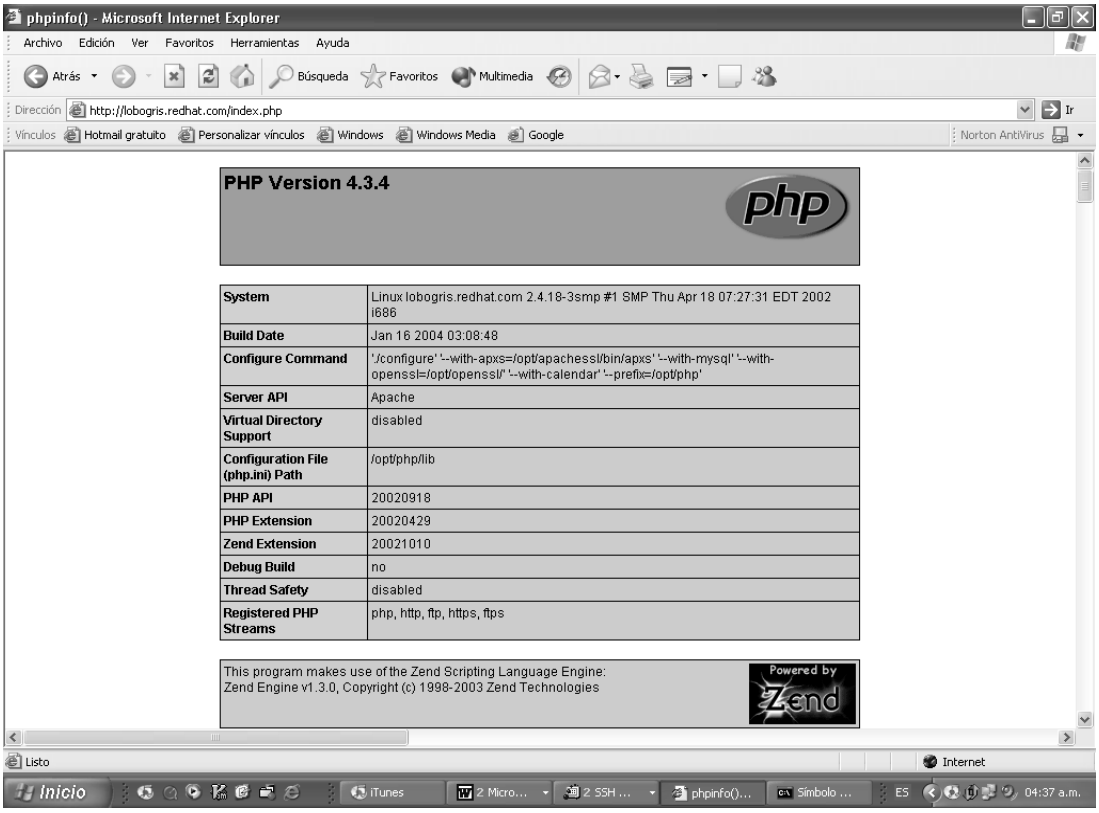
La forma más sencilla para realizarlo es apoyándonos de una función integrada en PHP de nombre phpinfo() que, como su nombre lo indica, nos muestra la configuración por defecto de nuestro servidor. Suponiendo que ha agregado el soporte a páginas de índice escritas en PHP escriba el siguiente script y colóquelo dentro de la carpeta indicada como raíz de documentos:

```
<?php
        phpinfo();
?>
```

Guarde el documento como **index.php**, abra un navegador y escriba la siguiente URL:

http://servidor/index.php

Lo cual le deberá mostrar la siguiente salida:



El comportamiento de PHP está limitado por un archivo de configuración de nombre **php.ini**. Este archivo no se instala por defecto. Deberá ser copiado desde el directorio de código fuente a la ubicación indicada por la salida de phpinfo(). Para ello, tenemos dos opciones:

1. Utilizar el archivo php.ini-dist el cual carece de restricción alguna.
2. Utilizar el archivo php.ini-recommended que tiene deshabilitadas algunas funciones que pueden ser peligrosas en entornos de producción.

Para mayor información al respecto a la configuración de este archivo, remítase a la documentación de PHP.

Perl y apache

En cuanto a programación de CGI con Perl tenemos varias opciones, estas van desde contruir todo lo que se sirve a mano pasando las cabeceras del documento HTML hasta utilizar módulos que automaticen esto por nosostros.

```
----------vegetales.pl ----------
 #!/usr/bin/perl
 # Script: vegetales.pl
 use CGI ':standard';
 print header,
    start_html('Vegetales'),
    h1('Come vegetales'),
    ol(
       li(['frijoles',
           'brocoli',
           'clabazas',
           'chiles' .
           ul(['rojos','amarillos','verdes']),
           'espinacas',
           'r&aacute;banos'
       ),
    hr,
    end_html;

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

En el ejemplo anterior utilizamos el módulo CGI para que genere algunas etiquetas ya conocidas por nosotros y que envie las cabeceras del documento.