

Apache Web Server

Tabla de Contenidos

5. Apache Web Server.....	2
5.1 Introducción.....	2
5.2 Instalación del Servidor Apache.....	3
5.3 Configuración.....	12
5.3.1 Section 1: Global Environment.....	12
5.3.2 Section 2: 'Main' server configuration.....	14
5.4 Section 3: Virtual Hosts.....	16
5.5 Directivas de control de recursos.....	18
5.6 Server Side Includes SSL.....	19
5.6.1 Algunos Comandos SSL.....	23
5.6.2 Variables SSL.....	23
5.6.3 Control de flujo.....	24
5.7 Configurando CGI.....	25
5.8 Autenticación.....	26
5.8.1 Autenticación basada en Host.....	27
5.8.2 Autenticación HTTP.....	29
5.9 SSL y Apache.....	34
5.10 Apéndice I del Fichero de Configuración httpd.conf.....	39
5.11 Apéndice II del fichero de configuración openssl.cnf.....	39



5. Apache Web Server

5.1 Introducción

Apache es posiblemente el servidor Web más utilizado en el mundo. Sus orígenes se remontan a 1995. Por esa época NCSA (National Center for Super Computing Applications) creó un servidor Web que se convirtió en el más usado. Cuando se abandona el proyecto de NCSA, los propios usuarios del mismo crearon un foro para poder compartir parches e información respecto al servidor. Surge el *Apache Group*. El servidor Apache se crea, entonces, a partir del código fuente del servidor de NCSA. La primera versión del servidor Apache surgió en Abril de 1995. Apache es un servidor flexible y simple que se ejecuta en varias plataformas: Linux, UNIX, Windows 95/98/NT/XP/2000.

La instalación necesaria depende del sistema operativo. Todas las distribuciones Linux cuentan con un servidor Apache integrado en la propia distribución por lo cual solamente hay que seleccionar la opción de instalar el servidor para que éste quede instalado y funcionando.

A no ser que se quiera utilizar el código fuente del servidor para modificarlo, lo recomendable y más sencillo es utilizar los binarios. Apache a partir de la versión 2 tiene un paquete completo de instalación en forma de ejecutable (.exe) o de instalador de Windows (.msi). El paquete de instalación puede descargarse desde:

- <http://www.apache.org/dist/httpd/binaries/win32/>

Si la instalación se realiza sobre Windows XP es necesario instalar el *Service Pack 1* mientras que si se instala sobre Windows 95 es necesario instalar Windows Socket 2 Update que puede descargarse desde el sitio de Microsoft (esto no es necesario si se tiene una versión más actual (98, Me, NT, 2000, XP)).



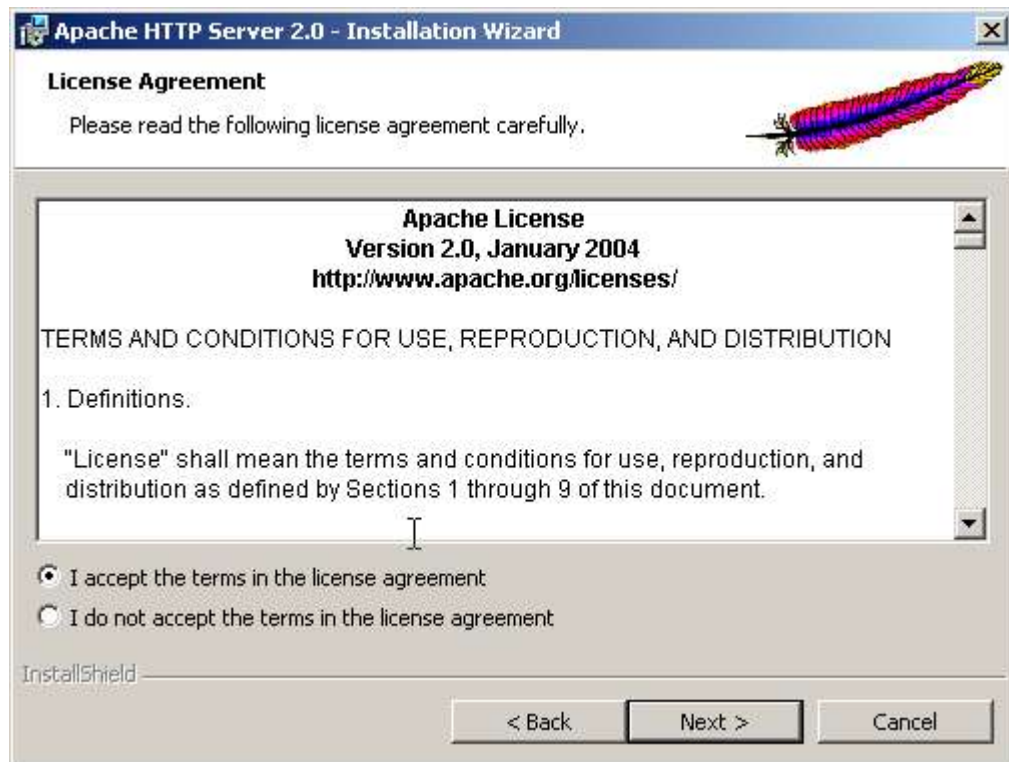
5.2 Instalación Del Servidor Apache

Una vez descargado el instalador se debe ejecutar y se presenta la pantalla de bienvenida:



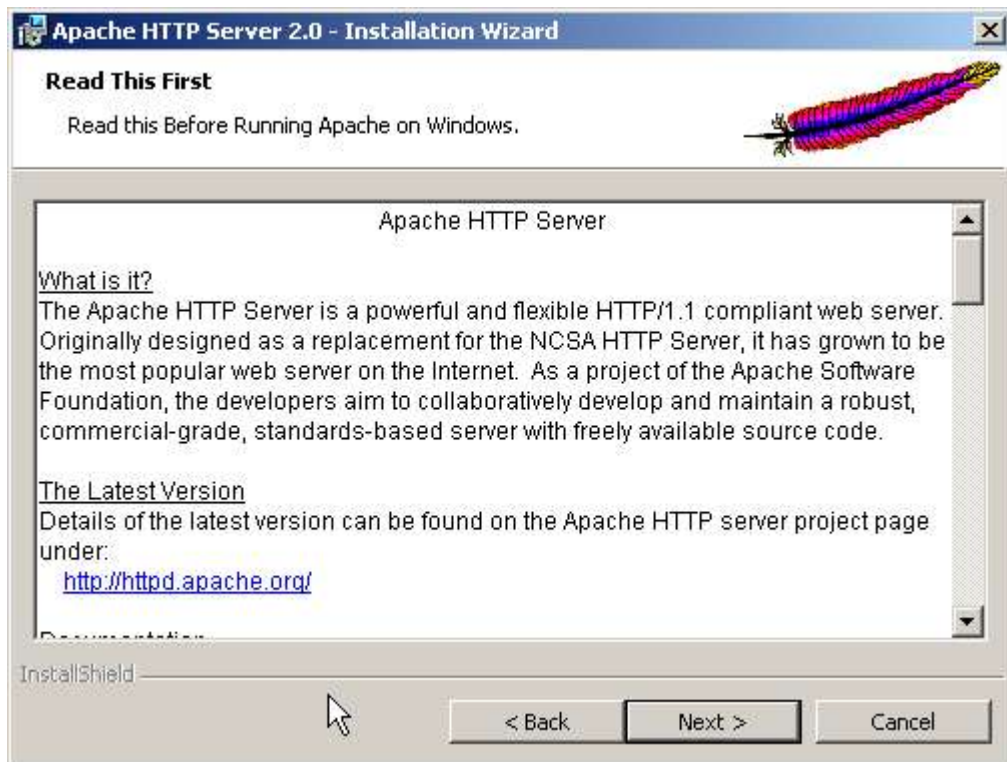
Se presiona sobre Next. Y en la siguiente ventana debe seleccionarse *I accept the terms in the license agreement* que indica que se acepta la licencia de Apache.





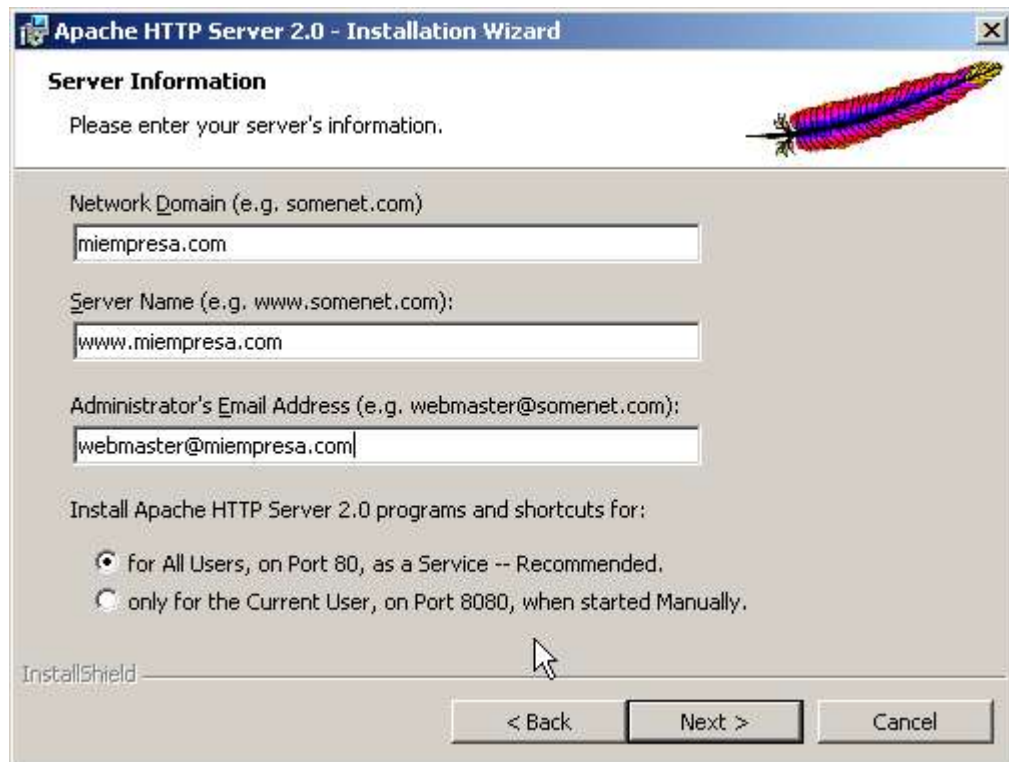
Un vez aceptada la licencia y presionando *Next*, aparece una ventana con el ReadMe (el documento introductorio con información del servidor (última versión, documentación, etc)).





La próxima pantalla permite ingresar la configuración del servidor. La misma, además de los nombres del dominio (atención si se quiere que funcione el nombre, debe ser uno válido) debe tener el tipo de instalación, ya sea como servicio, en cuyo caso se ejecutará escuchando el puerto 80 o para inicio manual sólo del usuario actual de Windows, en cuyo caso escuchará el puerto 8080. De todas formas los números de puerto pueden cambiarse, como se verá luego al tratar la configuración del servidor.





Apache HTTP Server 2.0 - Installation Wizard

Server Information

Please enter your server's information.

Network Domain (e.g. somenet.com):
miempresa.com

Server Name (e.g. www.somenet.com):
www.miempresa.com

Administrator's Email Address (e.g. webmaster@somenet.com):
webmaster@miempresa.com

Install Apache HTTP Server 2.0 programs and shortcuts for:

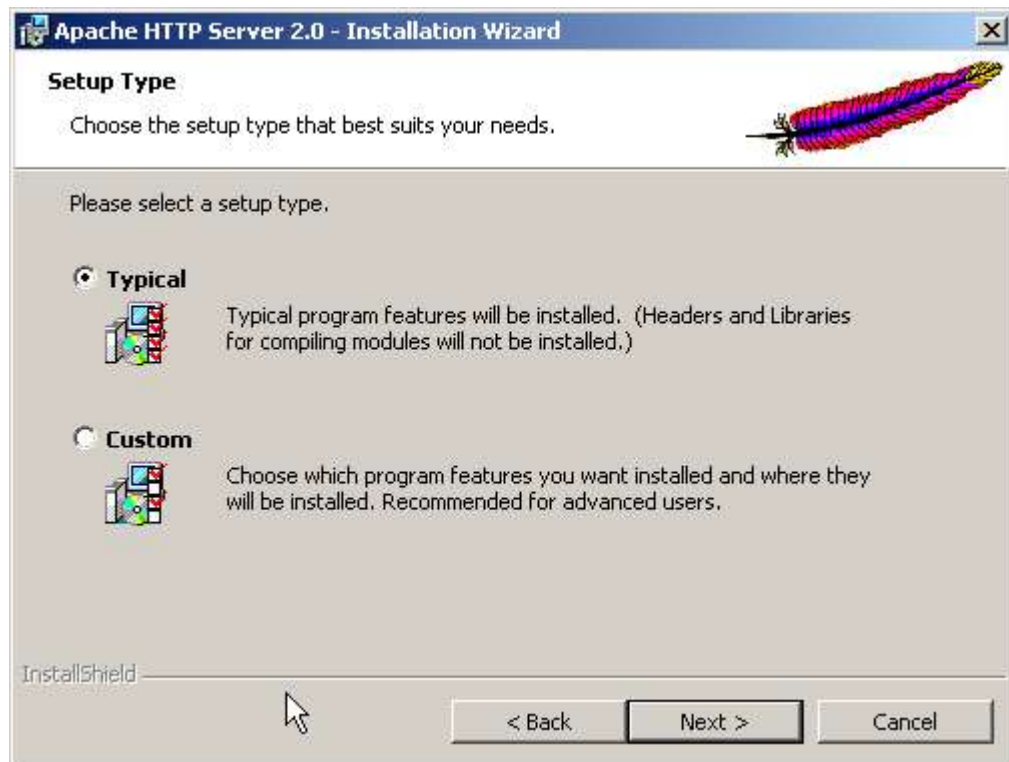
☒ for All Users, on Port 80, as a Service -- Recommended.

☐ only for the Current User, on Port 8080, when started Manually.

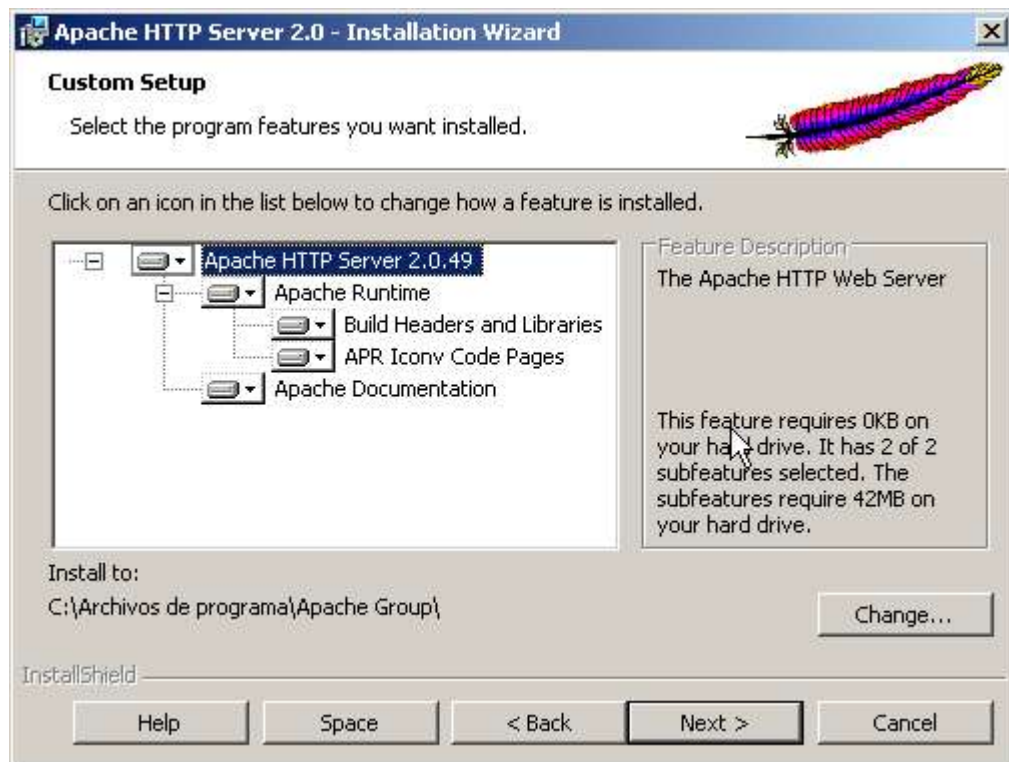
InstallShield

< Back Next > Cancel

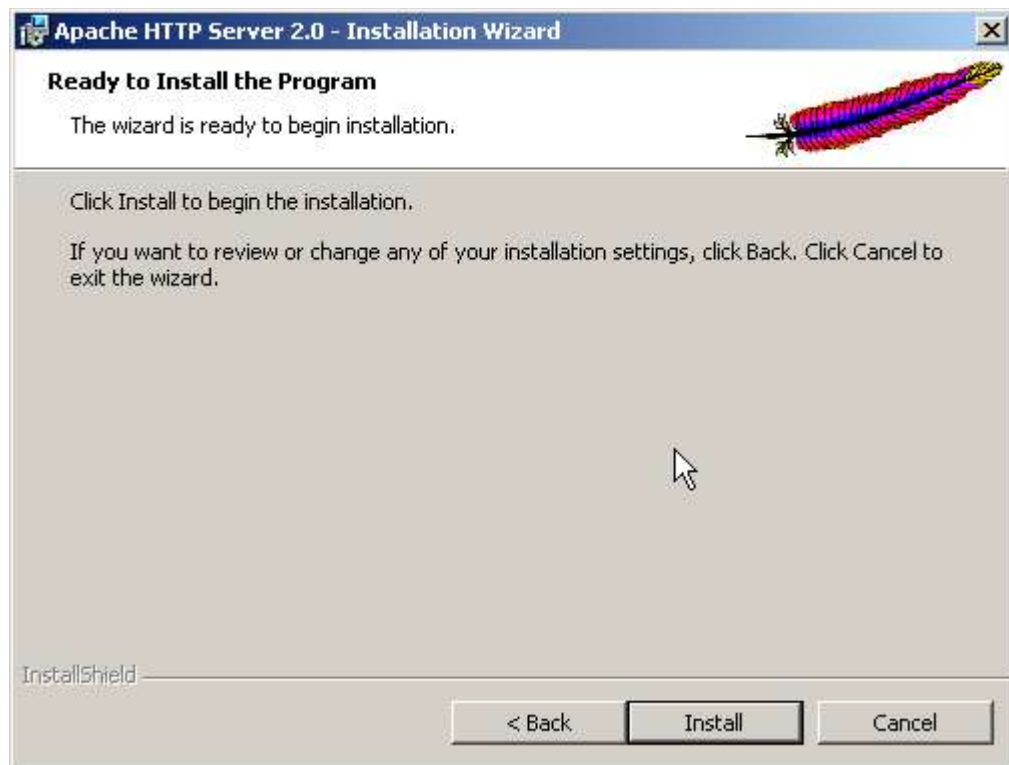
La ventana siguiente, permite elegir el tipo de instalación: personalizada o típica. Para la mayor parte de las necesidades la opción *Typical* (típica) es suficiente. En caso de querer compilar módulos adicionales, entonces se debe elegir la opción *Custom* (personalizada) ya que esta opción permite instalar las librerías y encabezados necesarios para hacerlo.



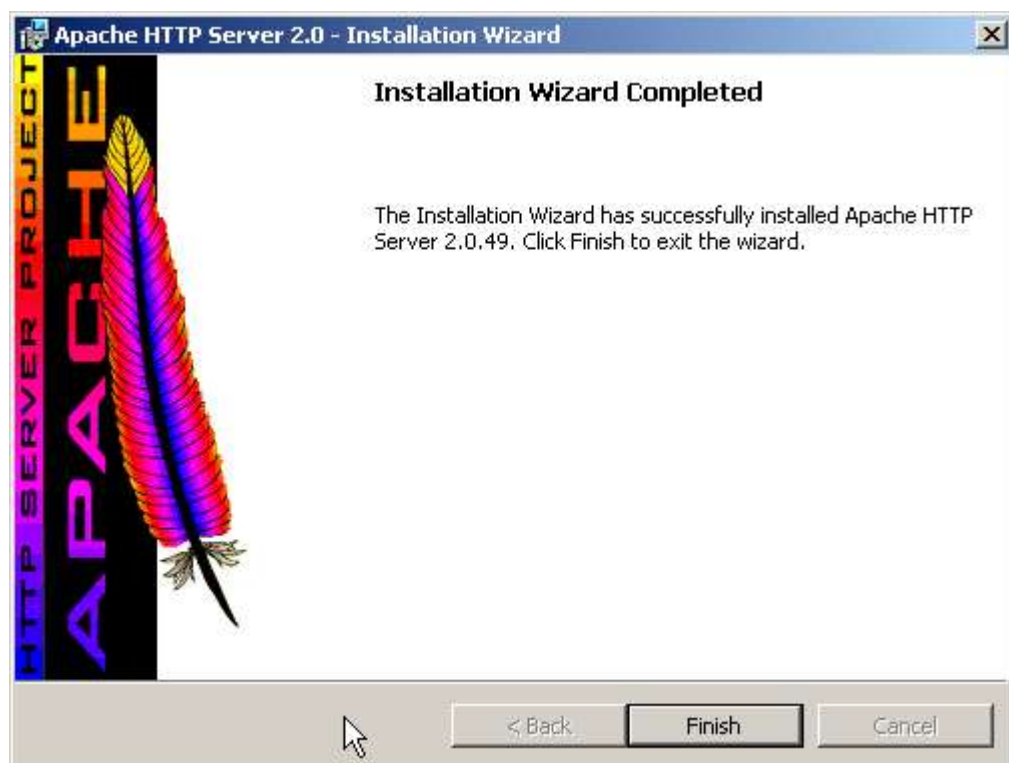
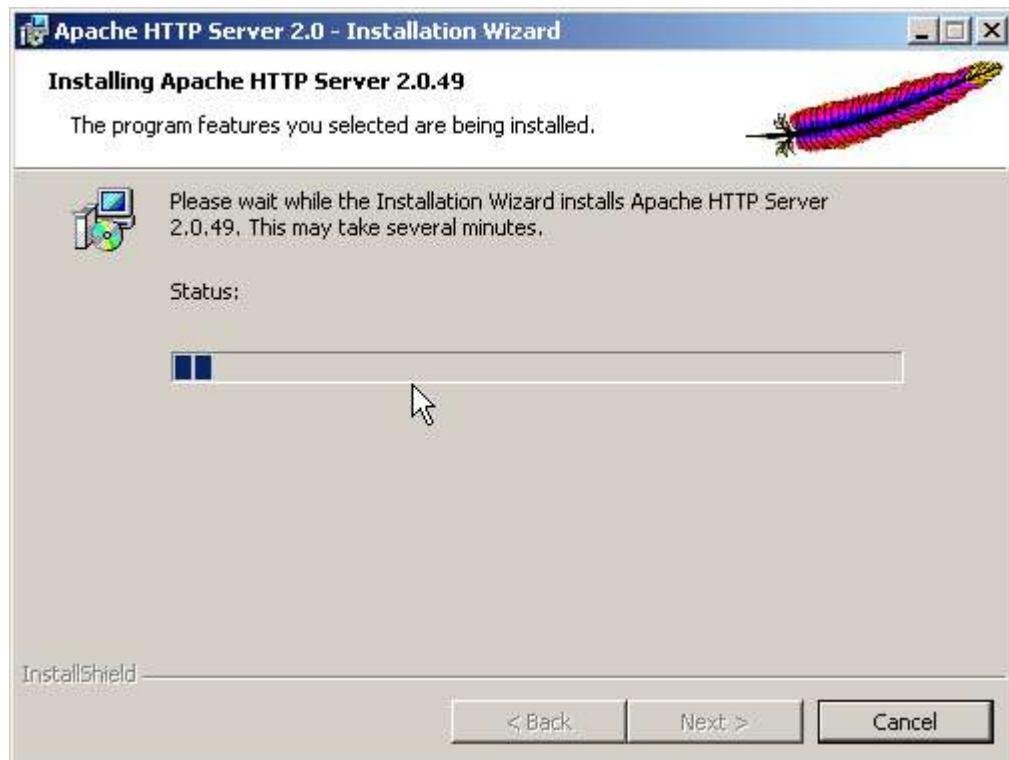
Se presiona el botón *Next*. En el caso de la opción *Custom* se mostrará la siguiente pantalla, donde se pueden elegir los componentes a instalar:



Si se eligió la opción *Typical*, entonces se pasa directamente a la última pantalla:



Al presionar el botón *Install* comienza la instalación.



Finalmente, al presionar el botón *Finish* se termina la instalación.

Controlar si la Instalación fue exitosa

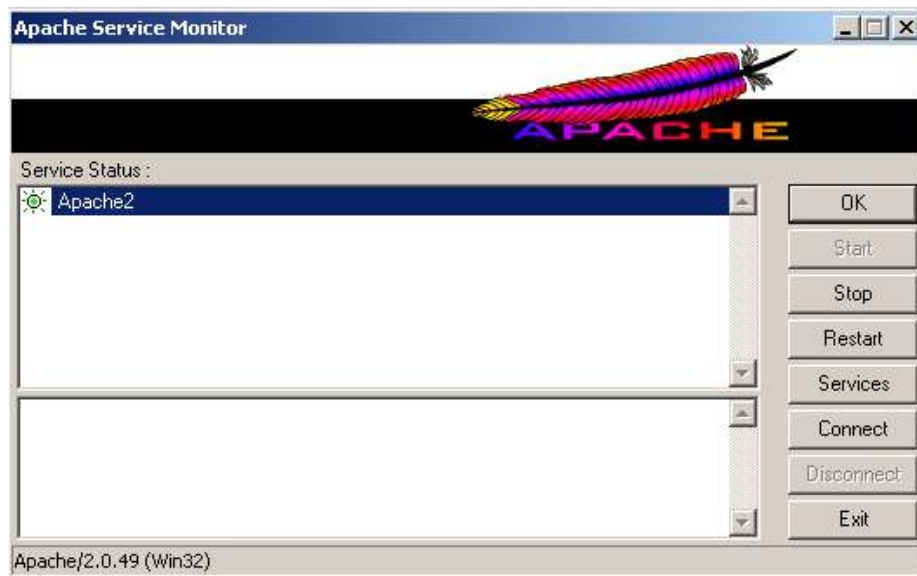
Si todo funcionó correctamente, entonces en la barra de tareas en la zona de *tray iconos* (a la derecha), debe aparecer un ícono con la plumita de apache y una flecha verde indicando que se está ejecutando el servidor.



Luego, es posible realizar la prueba tradicional, es decir abrir el Internet Explorer u otro browser, escribir <http://localhost> (o el nombre de la máquina en cuestión ó 127.0.0.1, o la IP de la máquina) y entonces se verá la famosa página indicando que todo está correcto: ¡Funcionó! ¡El Servidor de Red Apache ha sido instalado en ese sitio!



Es posible parar e iniciar el servicio de http con el Apache Service Monitor. Para acceder al mismo, se debe hacer doble click en el ícono de la barra de tareas. La figura siguiente muestra el monitor del servicio:

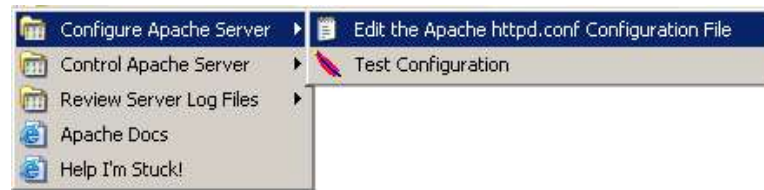


En la pantalla se indica con una luz verde que el servicio se encuentra ejecutándose. Si se presiona *Stop* el servicio se detendrá y se mostrará una luz roja. El botón de *Restart* permite reiniciar el servicio. El botón *Connect* permite conectarse a un servicio ejecutándose en otra computadora.

Otra forma de acceder al control del servicio, es desde el menú de *Inicio/Todos los programas/Apache http Server 2.0.49* (el número dependerá de la versión instalada, en este caso 2.0.49)



Entre las opciones, se encuentra el submenú de *Control Apache Server* desde el cual se puede iniciar, apagar o reiniciar el servicio. Además hay una opción para configurar que simplemente permite editar el archivo *httpd.conf*, el cual es el archivo de configuración principal del servidor Apache.



5.3 Configuración

La configuración de Apache se realiza en archivos de texto mediante directivas. El archivo de configuración principal del servidor Apache es el archivo *httpd.conf*. Este archivo cuenta con las directivas de configuración agrupadas en tres categorías:

- Directivas de control de operación (Section 1)
- Parámetros del servidor principal (Section 2)
- Configuración de *Host* virtuales (Section 3)

Las directivas de control de operación permiten controlar la forma de operación del servidor Apache en su conjunto. Las directivas del servidor principal son las que establecen el comportamiento de los requerimientos que no sean manejados por algún *host* virtual y que además actúan como valores por defecto para los *host* virtuales. La última categoría se encarga de la configuración de cada *host* virtual. Los *host* virtuales permiten que el mismo proceso servidor Apache atienda requerimientos dirigidos a diferentes direcciones IP o diferentes nombres de *host*. El apéndice I muestra el archivo de configuración de la instalación vista en el apartado anterior. Todo lo que se diga respecto de este archivo de configuración sirve para servidores bajo cualquier sistema operativo (Linux, Unix, etc). A continuación se analizarán las directivas más comunes separadas por sección.

5.3.1 Section 1: Global Environment

Primero se verá la configuración de las directivas para el ambiente global del servidor.

ServerRoot

La primera directiva que se verá es `ServerRoot`. En el archivo de configuración (ver apéndice) se ve la siguiente línea (tomar en cuenta que las líneas que comienzan con `#` son comentarios)

ServerRoot "C:/Archivos de programa/Apache Group/Apache2"

Esta directiva especifica la ubicación de la instalación del servidor, en donde se encuentran los archivos de configuración, error y registro.

Directiva Timeout

Esta directiva cuyo valor por defecto es:

`Timeout 300`

Permite establecer el tiempo transcurrido, medido en segundos, antes de que el servidor cierre la conexión. Este tiempo es aplicable a aspectos como el tiempo de espera hasta recibir una petición GET o el tiempo entre la recepción de los paquetes TCP correspondientes a una solicitud POST o PUT.

KeepAlive, MaxKeepAliveRequests, KeepAliveTimeout

Una característica importante de Apache es la posibilidad de mantener conexiones persistentes. Las conexiones persistentes permiten al servidor aprovechar una conexión TCP para realizar varias transacciones. Esto reduce el tiempo y la carga de abrir y cerrar conexiones. En caso de desactivar esta opción, se abrirá una conexión por cada petición http.

La directiva `KeepAlive` tiene como valores posibles `on` y `off` para activarla y desactivarla respectivamente

KeepAlive on

Además, existen las directivas `MaxKeepAliveRequests` y `KeepAliveTimeout`, para determinar por un lado el número máximo de peticiones que pueden establecerse por conexión y por otro, la cantidad de tiempo que el servidor Apache esperará por otra petición antes de cerrar una conexión. Naturalmente que ambas opciones tienen sentido solamente cuando `KeepAlive` se encuentra configurada `on`.

MaxKeepAliveRequests 100

KeepAliveTimeout 15

Listen

Esta directiva permite vincular el servidor Apache direcciones IP o puertos específicos.



Listen 80

Listen 10.20.218.12:8080

En el primer caso utilizara la IP por defecto y el puerto 80, mientras que en el segundo escuchará la dirección 10.20.218.12 en el puerto 8080.

LoadModule

Esta directiva permite la utilización de módulos. Los módulos son la forma de extender la funcionalidad del servidor Apache. Posteriormente se describirán los módulos más interesantes. La sintaxis es la siguiente:

```
LoadModule foo_module modules/mod_foo.s
```

La directiva anterior especifica que se debe cargar el módulo *foo_module* que se encuentra en el directorio *modules* con el nombre *mod_foo.so*. El subdirectorio *modules* se busca a partir del *ServerRoot* como fue especificado más arriba.

5.3.2 Section 2: 'Main' Server Configuration

Esta sección configura los valores del servidor principal y provee a su vez la configuración por defecto para los servidores virtuales.

ServerAdmin

Su valor es la dirección de e-mail del administrador del sistema y será mostrada en páginas de error generadas por el Server (si se produjera alguno)

```
ServerAdmin webmaster@miempresa.com
```

Este valor es el ingresado al instalar el servidor Apache en la ventana donde se pide la dirección de e-mail del administrador.

ServerName

Es el nombre y puerto que el servidor utiliza para poder identificarse a sí mismo. Debe ser un nombre DNS válido o una dirección IP

ServerName *www.miempresa.com:80*

Este valor también fue configurado al instalar el servidor pero como el anterior, es posible modificarlo desde aquí.

DocumentRoot

La directiva DocumentRoot indica el lugar donde por defecto el servidor buscará los documentos (páginas html en general).

DocumentRoot *"C:/Archivos de programa/Apache Group/Apache2/htdocs"*

Observar que la barra de separación de directorios utiliza el formato del sistema operativo Unix es decir / en lugar del formato de Windows \.

DirectoryIndex

Esta directiva indica el archivo que tomará el servidor Apache como defecto para el directorio solicitado. La sintaxis es la siguiente:

DirectoryIndex *archivo1, archivo2, archivo3.... archivoN*

Por ejemplo:

DirectoryIndex *index.html*

<Directory unDirectorio> </Directory>

Este par de directivas sirven para encerrar un grupo de directrices asociadas a los directorios. Ejemplo:

<Directory "C:/Archivos de programa/Apache Group/Apache2/htdocs">

Options *Indexes FollowSymLinks*

AllowOverride *None*

Order *allow, deny*

Allow from *all*

</Directory>



Options Indexes FollowSymLinks indica que el directorio se puede indexar. O sea que si el directorio no tiene un archivo por defecto, se creará un índice sobre la marcha. Este es el sentido de la opción *DirectoryIndex*, es decir, el índice de un directorio es un archivo especificado o es creado por el servidor mostrando una lista de los elementos del directorio y permitiendo acceder a él. Naturalmente que esta opción debería deshabilitarse para el acceso desde Internet logrando mayor seguridad.

Allow from all indica que es posible acceder a todo el directorio. *AllowOverride None* indica que si se especifica un archivo de control de acceso, éste no puede sobrescribir ninguna de las opciones.

AccessFileName

Es el nombre del archivo que el servidor Apache busca para obtener directivas de configuración adicional.

AccessFileName .htaccess

La directiva *AllowOverride None / All* afecta a esta directiva haciendo que el servidor aplique o no las directivas del *.htaccess*.

5.4 Section 3: Virtual Hosts

Esta sección permite mantener múltiples dominios en la misma máquina. Es posible mantener varios dominios sobre la misma dirección IP o diferentes direcciones IP. Las precauciones son las mismas que para IIS, es decir siempre es preferible tener múltiples direcciones IP ya que algunos navegadores antiguos y SSL no funcionan para *virtual hosting* basados en nombre.

Virtual hosting basado en Nombres

Supóngase que el servidor se encuentra atendiendo el dominio *www.miempresa.com* y se desea agregar un *virtual host* denominada *www.otraempresa.com* La configuración sería:

NameVirtualHost *:80




```
<VirtualHost *:80>  
    ServerName www.miempresa.com  
    ServerAlias miempresa.com *.miempresa.com  
    DocumentRoot /www/miempresa  
</VirtualHost>
```

```
<VirtualHost *:80>  
    ServerName www.otraempresa.com  
    DocumentRoot /www/otraempresa  
</VirtualHost>
```

La directiva *ServerAlias* permite que los *hosts* sean accedidos por más de un nombre. En el ejemplo, los requerimientos para todos los host en el dominio *miempresa.com* serán atendidos por *www.miempresa.com*. El *** es un comodín indicando cualquier secuencia de caracteres (válida). De todas formas, para que funcione, los DNS deben estar configurados para resolver correctamente el nombre.

Cuando se recibe un requerimiento, el servidor Apache primero verifica si se está usando una dirección IP que coincida con *NameVirtualHost*. En el ejemplo anterior el *** indica todas las direcciones IP (naturalmente de la máquina en cuestión). Si coincide, entonces busca una sección de *VirtualHost* con una dirección IP que coincida (en el ejemplo, todos) y entonces prueba buscar uno donde el *ServerName* o *ServerAlias* coincida con el *hostname* requerido. Si no hay coincidencias, entonces el requerimiento es atendido por el *virtualhost* que tiene la dirección IP coincidente.

Una consecuencia de lo anterior, es el hecho que el primer *virtual host* listado en el archivo de configuración, es el *virtualhost por defecto*. O sea que el *main server* no se utilizará cuando una dirección IP coincide con la directiva *NameVirtualHost*.

Virtual Host basado en direcciones IP

Para utilizar diferentes direcciones IP por *virtual host* es necesario tener una dirección IP diferente por cada *virtual host* basado en dirección IP.

Una forma de lograr que Apache soporte múltiples hosts es tener un servicio (*daemon*) Apache por cada host, la otra es que el mismo servicio atienda todos los *virtual hosts*. En el primer caso es necesario tener múltiples instalaciones y configurar la directiva *Listen* para que atienda la dirección IP correspondiente. El segundo caso es el más común y la configuración sería como sigue:

```
<VirtualHost 210.78.232.24:80>
    ServerName www.miempresa.com
    ServerAlias miempresa.com *.miempresa.com
    DocumentRoot /www/miempresa
</VirtualHost>

<VirtualHost 210.78.232.23:80>
    ServerName www.otraempresa.com
    DocumentRoot /www/otraempresa
</VirtualHost>
```

En el ejemplo cada virtual host tiene asignada una dirección IP diferente.

Tener varias direcciones IP puede lograrse con varias tarjetas de red cada una con una dirección IP diferente o una tarjeta con soporte para múltiples direcciones IP. **Es** posible utilizar el *hostname* en el lugar de las direcciones IP pero no es lo recomendable.

5.5 Directivas De Control De Recursos

Existen una serie de directivas que permite administrar el uso del servidor por parte de Apache, estas directivas pueden establecerse a nivel servidor o a nivel virtualhost. Estas directivas confieren mucha flexibilidad al servidor. Hay que tomar en cuenta que muchos ataques a los servidores se basan en hacer que el servidor utilice todos los recursos del sistema provocando la caída del mismo.

RLimitCPU.

Esta directiva permite limitar la utilización de la CPU. Cuenta con dos parámetros, el primero limita la cantidad de recursos que utilizarán todos los procesos y el segundo limita el número máximo de recursos permitidos por el sistema operativo.

RLimitCPU segundos|max [segundos|max]



Los límites se establecen en segundos por proceso.

RLimitMEM

Esta directiva permite limitar la cantidad de memoria que utilicen los procesos del servidor Apache. Como la anterior, tiene dos parámetros. El primero define el límite para todos los procesos y el segundo indica el límite de recursos totales.

RLimitMEM bytes | max [bytes|max]

Cada parámetro puede ser un número (indicando la cantidad de bytes por proceso) o *max* indicando el máximo permitido por el sistema operativo.

RLimitPROC

Esta directiva permite limitar el número máximo de procesos simultáneos que puede haber por usuarios. Como los anteriores, cuenta con dos parámetros: uno para definir el límite de todos los procesos y el otro el límite de recursos.

RLimitPROC número | max [número | max]

Cada parámetro puede ser un número o *max* para indicar el máximo permitido por el sistema operativo.

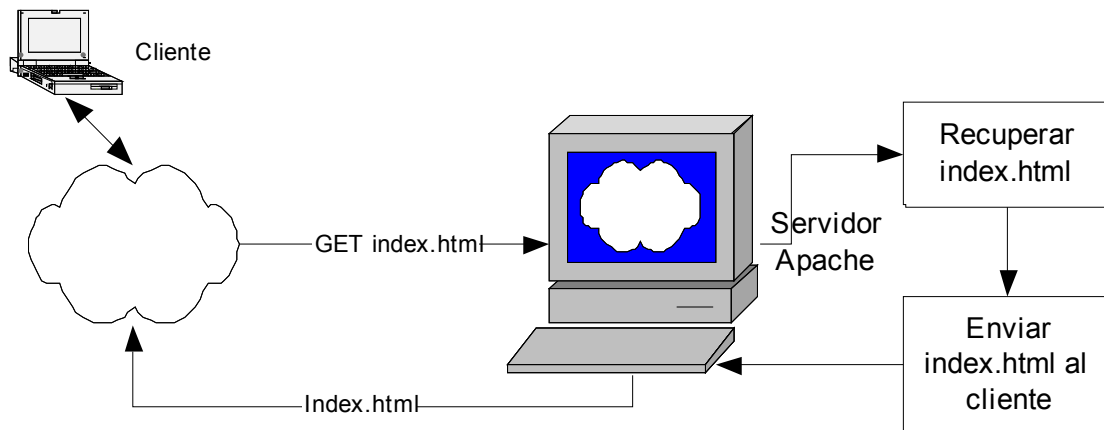
5.6 Server Side Includes SSI

SSI permite incorporar contenido dinámico básico a páginas html. Esto permite generar contenido dinámico sin necesidad de usar CGI, JSP, o tecnologías similares.

Claro está que no es el recurso para el desarrollo de aplicaciones Web, pero sí para resolver algunas necesidades en forma rápida y sencilla, fundamentalmente cuando se desea agregar pequeñas piezas de información. Por ejemplo, información de copyright al final de cada página.

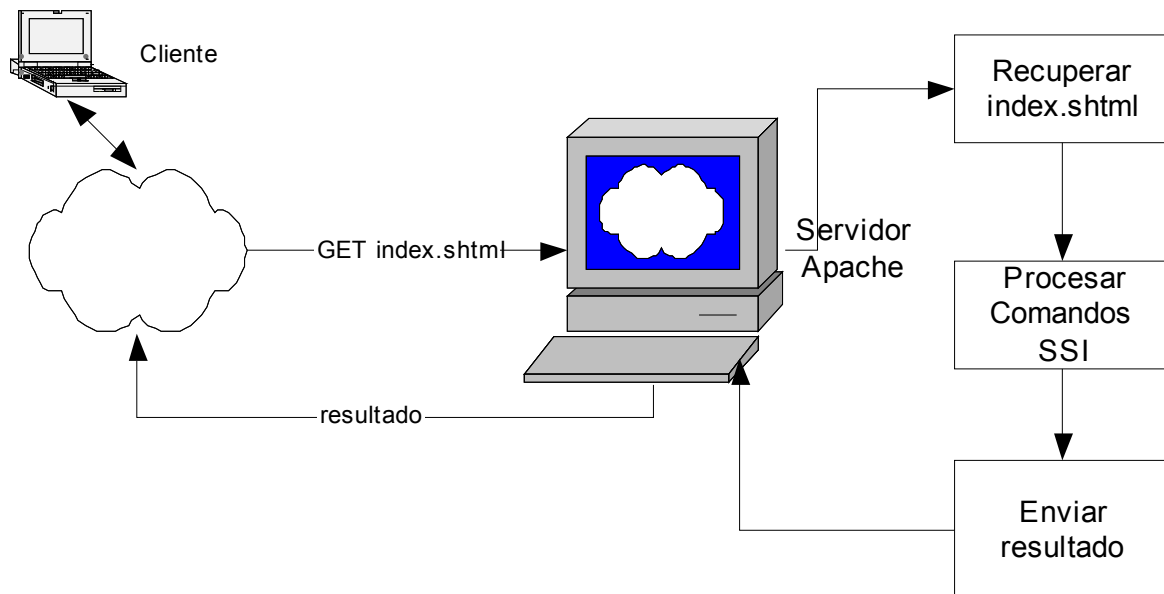
Para poder comprender el funcionamiento de SSI, se comparará con el procesamiento de páginas sin SSI. El

siguiente diagrama es un esquema del procesamiento sin SSI.



El navegador cliente realiza una petición GET para obtener la página (por ejemplo index.html), el servidor Apache busca en el directorio correspondiente (según su configuración) y devuelve la página en modo texto.

Cuando se trabaja con extensiones SSI, el servidor Apache, luego de recuperar el documento, realiza el análisis y procesamiento de los comandos SSI y devuelve la página resultado. El siguiente diagrama muestra el proceso:



Para que el servidor Apache pueda trabajar con SSI, es necesario configurarlo para que lo haga. En primer lugar, se debe verificar que se cargue el módulo encargado del procesamiento SSI. Ello puede verse en el archivo de configuración con la directiva LoadModule. La línea siguiente debe estar habilitada (lo está por defecto):

LoadModule include_module modules/mod_include.so

Otra directiva a configurar es Options, la sintaxis es:

Options +Include

Esta directiva indica a Apache que procese ficheros SSI. De todas maneras Apache no procesa cualquier documento. Es preciso indicarle al servidor Apache que archivos debe procesar, por ejemplo documentos con la extensión *shtml* (que es la más usada). Esto se logra con las siguientes directivas:

AddType text/html .shtml

AddHandler server-parsed .shtml

La primer directiva le indica a Apache que tipo de información tiene que indicarle al navegador que le está enviando (el tipo MIME). La segunda le indica que debe procesar documentos con extensión *shtml*. Estas directivas pueden establecerse a nivel directorio. Por ejemplo, supónganse que se desea que el directorio ssi_demo procese documentos shtml como SSI. Si el directorio se encuentra en un *virtual host* que se encuentra escuchando en la dirección IP 210.78.232.23 la configuración será la siguiente:

<VirtualHost 210.78.232.23:80>

ServerName www.miempresa.com

ServerAlias miempresa.com *.miempresa.com

DocumentRoot /www/miempresa

<Directory /www/miempresa/ssi_demo >

Addhandler server-parsed .shtml

AddType text/html .shtml

Options +include

</Direcotry>



</VirtualHost>

Es posible que no se quiera/pueda/convenga usar distintas extensiones para los archivos con directivas SSI. Es decir, puede quererse que se procesen archivos .html e incluir en ellos directivas SSI. En este caso, habría que indicarle al servidor que procese archivos html. Pero esto es peligroso, ya que si se pusieran directivas como:

Addhandler server-parsed html

AddType text/html html

El servidor Apache procesaría todas las líneas de todos los documentos .html aunque éstos no contuvieran directivas SSI.

Si se encuentra en un ambiente Linux\Unix, entonces la opción recomendada es usar la directiva XBitHack, que como su nombre lo indica, intenta analizar el *bit* de permiso de ejecución del archivo (naturalmente esto no tiene sentido en un ambiente Windows).

Esta opción tiene la siguiente sintaxis:

XBitHack on | off |full

Afecta a los archivos asociados con el tipo MIME text/html (generalmente .html o .htm) y configura al servidor Apache para que procese aquellos que tengan permiso de ejecución (el bit X prendido). La opción *on* indica que cualquier archivo text/html que tenga permiso de ejecución se considerará SSI y por lo tanto será procesado. La opción *full*, además, analiza el bit de grupo y en caso de estar prendido devuelve la fecha de última modificación permitiendo a los servidores proxy mantener un cache de la página.

El comando de Unix/Linux que permite asignar permiso de ejecución a un documento es el siguiente:

chmod +x index.html

En este caso, se le está dando permiso de ejecución al documento index.html.



5.6.1 Algunos Comandos SSI

Los comandos SSI se incluyen en páginas HTML de la siguiente forma:

```
<!--#comando argumento=valor ....>
```

La cantidad de argumentos dependerá del comando en cuestión.

include

Este comando permite incluir texto de un archivo en el documento que se está procesando

```
<!--#include file= "empresa.html -->
```

5.6.2 Variables SSI

Existen un conjunto de variables que pueden utilizarse en SSI.

- DATE_GMT (fecha actual del meridian de Greenwich)
- DATE_LOCAL (fecha de la zona horaria local).
- DOCUMENT_NAME (nombre del archivo).
- DOCUMENT_URI (El URL del documento)
- LAST_MODIFIED (Fecha de la última modificación)

Ejemplo:

```
<!--#echo var="DATE_LOCAL" -->
```



El ejemplo anterior, tiene como efecto que se mostrará la fecha local.

5.6.3 Control De Flujo

El control de flujo puede realizarse mediante el comando *if*. La siguiente es la sintaxis del mismo:

```
<!--#if exp.="expresión" -->
<!--#endif -->
```

Si el resultado de expresión es verdadero entonces se incluye el contenido entre el *if* y el *endif*. También puede usarse:

```
<!--#if exp.="expresión" -->
<!--#else -->
<!--#endif -->
```

Que actúa de la siguiente manera: si la expresión es evaluada en verdadero, se mostrará el contenido entre el *if* y el *else*. En cambio, si es evaluada falso, se mostrará el contenido entre el *else* y el *endif*. Es decir, que sólo un bloque será mostrado dependiendo del valor de verdad de la expresión.

La expresión puede ser una cadena de caracteres (que siempre evalúa en verdadero) o *cadena operador cadena* donde operador es un operador de comparación (=,!=, <,>,<= o >=)

Ejemplo

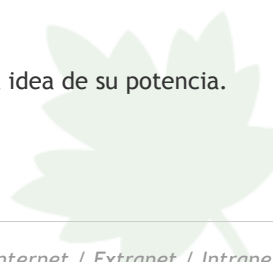
```
<!--#set var="variable" value= "si" -->
<!--#if exp= " $variable= si " -->
```

La variable tenía Si

```
<!--#endif -->
```

El comando set permite asignar valor a una variable.

La sintaxis completa de SSI escapa al contenido del curso pero con lo dicho se tiene una idea de su potencia.



5.7 Configurando CGI

Como se mencionó en la sección de IIS, CGI define una interfaz para que el servidor Web pueda interactuar con programas que generen el contenido automáticamente. A estos programas muchas veces se los denomina programas CGI, siendo Perl y PHP dos de los más populares lenguajes para programas CGI.

El primer paso en la configuración de CGI es establecer un directorio donde se ubicarán los programas CGI. Idealmente los directorios donde se ubiquen programas CGI no deben estar bajo el DocumentRoot para evitar fugas de seguridad. La primera directiva a configurar es ScriptAlias

```
ScriptAlias /cgi-bin/ "C:/Archivos de programa/Apache Group/Apache2/cgi-bin/"
```

O en Linux/Unix

```
ScriptAlias /cgi-bin/ /www/empresa/cgidirectorio
```

En caso de querer configurar alias para cada servidor virtual (*virtual host*) deberá agregarse la línea en la configuración de cada servidor virtual. Por ejemplo:

```
<VirtualHost 210.78.232.23:80>  
    ServerName www.otraempresa.com  
    DocumentRoot /www/otraempresa  
    ScriptAlias /cgi-bin/ /www/empresa/cgidirectorio  
</VirtualHost>
```

En este caso, el acceso a la ejecución del programa CGI se realiza mediante la URL:

```
www.otraempresa.com/cgi-bin/programa.cgi
```

Si existiera el programa programa.cgi

Por razones de seguridad, solamente se ejecutan programas residentes en los directorios establecidos en ScriptAlias (y que además tengan el permiso de ejecución adecuado). No obstante, es posible asignar permiso de ejecución de scripts cgi a directorios particulares con la opción:

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



En este caso, la directiva `Options +ExecCGI` permite la ejecución de scripts en el directorio *directorioespecial*.

Además, es preciso decirle al servidor Apache cuales archivos son los scripts. Eso se logra con la directiva `AddHandler`. El siguiente ejemplo, muestra la asignación de archivos con extensión `cgi` y `pl` como archivos CGI:

AddHandler cgi-script cgi pl

Como cuando el servidor Apache intenta acceder a un directorio, busca el archivo `.htaccess` para aplicar las directivas que estén permitidas. Es posible utilizarlo para configurar la ejecución de programas CGI. Primeramente, es necesario permitir que `.htaccess` pueda sobrescribir la directiva `Options`. Para ello, en el archivo de configuración debe ubicarse la siguiente línea:

AllowOverride Options

Que nos indica que se permite sobrescribir la directiva `Options`, naturalmente que también funciona `AllowOverride All`.

Luego, en el archivo `.htaccess` se debe indicar que se autoriza la ejecución con la directiva: `Options +ExecCGI`.

Si se tiene PERL instalado entonces los scripts CGI programados en PERL en Linux/Unix deben comenzar con la sentencia:

#!/usr/bin/perl

Indicando donde se encuentra ubicado el PERL en el sistema. No debe olvidarse de asignar permiso de ejecución con el comando `chmod`.

5.8 Autenticación

Existen varias formas de lograr la autenticación de los usuarios para permitir o restringir el acceso a determinadas páginas. Como `http` es un protocolo sin estado, es decir que no se mantiene la conexión entre cliente y servidor a través de las diversas peticiones, es necesario tener formas de autenticación diferentes a las de un sistema cliente/servidor tradicional. Las directivas de autenticación pueden ir tanto en el archivo principal de configuración `httpd.conf` en una sección `<Directory > ..</Directory >` o en un archivo `.htaccess` (o el que se haya especificado) en el directorio en cuestión.

5.8.1 Autenticación Basada En Host

Este mecanismo utiliza la dirección IP o el nombre de host del cliente para realizar la autenticación. Básicamente al realizarse un pedido el servidor Apache, comprueba si el host que realiza el pedido tiene autorización para acceder al recurso. El módulo que permite el control de acceso a partir de la dirección IP del cliente o el nombre de host, es el módulo *mod_access*. Se encuentra activo por defecto.

En el archivo de configuración *httpd.conf* se encuentra la siguiente línea:

LoadModule access_module modules/mod_access.so

Que indica que el modulo *mod_access* es cargado cuando se inicia el servicio.

Para este tipo de autenticación existen las directivas *allow*, *deny* y *order*.

Allow

Permite definir una lista de direcciones IP o nombres de Host que contarán con permiso para acceder al directorio. Las alternativas para establecer la lista son varias:

allow from all

Especifica que todos pueden acceder al directorio. Generalmente se combina con la opción *deny* como se verá más adelante.

allow from www.direccion.com

Aquí se está especificando que el host de nombre *www.direccion.com* puede acceder.

allow from 192.12.34.5

Aquí se está especificando que el host de dirección IP *192.12.34.5* puede acceder.

allow from 234.45.76



En este caso no se encuentra completa la dirección IP, entonces se permitirá el acceso a clientes cuya dirección IP coincida en sus primeros bytes con los especificados. En el ejemplo todas las direcciones IP que comiencen con 234.45.76 sin importar el último byte.

allow from 234.56.78.0/255.255.255.0

En este caso se especifica el par dirección IP/máscara de subred. En el ejemplo se permitirá acceso a clientes cuya dirección IP sean desde 234.56.78.1 hasta 234.56.78.255. Aunque parece similar a la anterior, es más flexible ya que la máscara de subred no tiene porque limitarse a uno, dos o tres bytes y puede establecer rangos más flexibles.

allow from 234.56.78.0/24

Aquí es una especificación CIDR (direccionamiento interdominio sin clases) y afecta a todas las direcciones generadas por la máscara de subred correspondiente.

deny

Esta directiva es la opuesta de la anterior y permite establecer una lista de hosts o direcciones IP a las que se les denegará el acceso. Las opciones son las mismas de allow. Por ejemplo:

deny from all

Deniega el acceso a todos los clientes.

order

Esta directiva es la que permite combinar las anteriores dando un flexible mecanismo de control

La sintaxis de la misma es:

order deny, allow | allow, deny | mutual-failure

Es decir, que establece el orden en el que se analizarán las directivas deny y allow. La lista establecerá el orden de preferencia. Por ejemplo:

order deny, allow

deny from alguien.compania.com

allow form all



En este caso, se niega el acceso a un host llamado alguien.compania.com y se le permite el acceso al todo el resto (la preferencia es deny). Mientras que si se ubica:

order allow, deny

deny from all

allow form alguien.compania.com

Se estará permitiendo el acceso solamente a un host llamado alguien.compania.com

La otra posibilidad es utilizar mutual-failure, que estaría indicando que se les denegará el acceso a los host que se encuentran en la lista deny y se les permitirá a los que se encuentran en la lista allow.

Además, es posible que allow o deny tomen valores de variables de entorno, esto se logra con la directiva *allow from env=variable* o *deny from env=variable*

5.8.2 Autenticación HTTP

En caso de utilizar .htaccess no hay que olvidarse de permitir la configuración mediante la directiva siguiente en el archivo de configuración:

AllowOverride AuthConfig

De esta forma, se podrá trabajar con el archivo .htaccess a nivel directorio.

El funcionamiento de la autenticación HTTP es el siguiente: al solicitar el acceso a un documento que se encuentre protegido (luego se verá como se logra proteger un documento), el servidor Apache envía una cabecera con el estado 401 y otra para la respuesta de autenticación. La cabecera contiene el sistema de autenticación (HTTP básica) y el nombre de dominio.

El navegador muestra entonces un cuadro de diálogo que le pide al usuario su nombre y contraseña. Este ingresa la información, que es enviada al servidor Apache que su vez comprueba la validez de la misma. Si es inválida, vuelve a responder con 401 y la cabecera de autenticación. Si es válida, entonces retorna el documento solicitado. Posteriormente, el servidor no requerirá la autenticación para cada página del directorio, ya que el explorador volverá a enviar la información de usuario/contraseña sin necesidad de pedirla al usuario. La información no viaja en

forma encriptada, sino utilizando una codificación conocida, lo cual hace que sea potencialmente vulnerable.

El módulo básico que se encarga de esta tarea es el módulo `mod_auth`, que por defecto está habilitado, según la siguiente línea del archivo de configuración:

`LoadModule auth_module modules/mod_auth.so`

Lo primero que se necesita, es un archivo con los usuarios/contraseñas. Para ello, se utiliza el comando `htpasswd` que se encuentra en el directorio de los binarios de Apache. Este comando, la primera vez, debe ejecutarse con la opción `-c` que indica que se debe crear el archivo y como parámetro se le debe indicar la ubicación y nombre. En la siguiente figura se muestra la creación de un archivo llamado *secreto* con el *usuario1* como usuario y ubicado en el directorio `c:\archivos de programa\apache group\apache2\passwd\`

```
C:\Archivos de programa\Apache Group\Apache2\bin>htpasswd -c "c:\archivos de programa\apache group\apache2\passwd\secreto" usuario1
Automatically using MD5 format.
New password: ****
Re-type new password: ****
Adding password for user usuario1
```

Como se ve en la figura, una vez ingresado el comando se pide la *password* y luego se reingresa para confirmar. Como resultado, se obtiene el archivo *secreto* con el usuario *usuario1* y la contraseña que hayamos ingresado. Es importante que el archivo de usuario/contraseñas no se encuentre en el árbol del sitio sino fuera de él para que no pueda ser accedido desde Internet.

El archivo tendrá información como:

usuario1:\$apr1\$/Z2.....\$GZcMTQI8lLm6OAsCYJs4.

Es decir que la contraseña se encuentra encriptada. En Windows por defecto se utiliza MD5, en Linux hay que utilizar la opción `-m` para que se utilice MD5. Si se trabaja en Unix/Linux, puede ser una buena opción que el nombre del archivo de contraseñas comience con un punto.

Una vez que se tiene el usuario/contraseña es posible agregar más usuarios de la misma forma pero sin usar la opción `-c` como se ve en la siguiente figura:

```
C:\Archivos de programa\Apache Group\Apache2\bin>htpasswd "c:\archivos de programa\apache group\apache2\passwd\secreto" usuario2
Automatically using MD5 format.
New password: ****
Re-type new password: ****
Adding password for user usuario2
```

Ahora se está en condiciones de crear el archivo `.htaccess` o utilizar el archivo de configuración principal `httpd.conf` para establecer el control de acceso. Las directivas que se deben usar son las siguientes:

AuthType Basic

AuthName "Restricted Files"

AuthUserFile "c:/archivos de programa/apache group/apache2/passw/secreto"

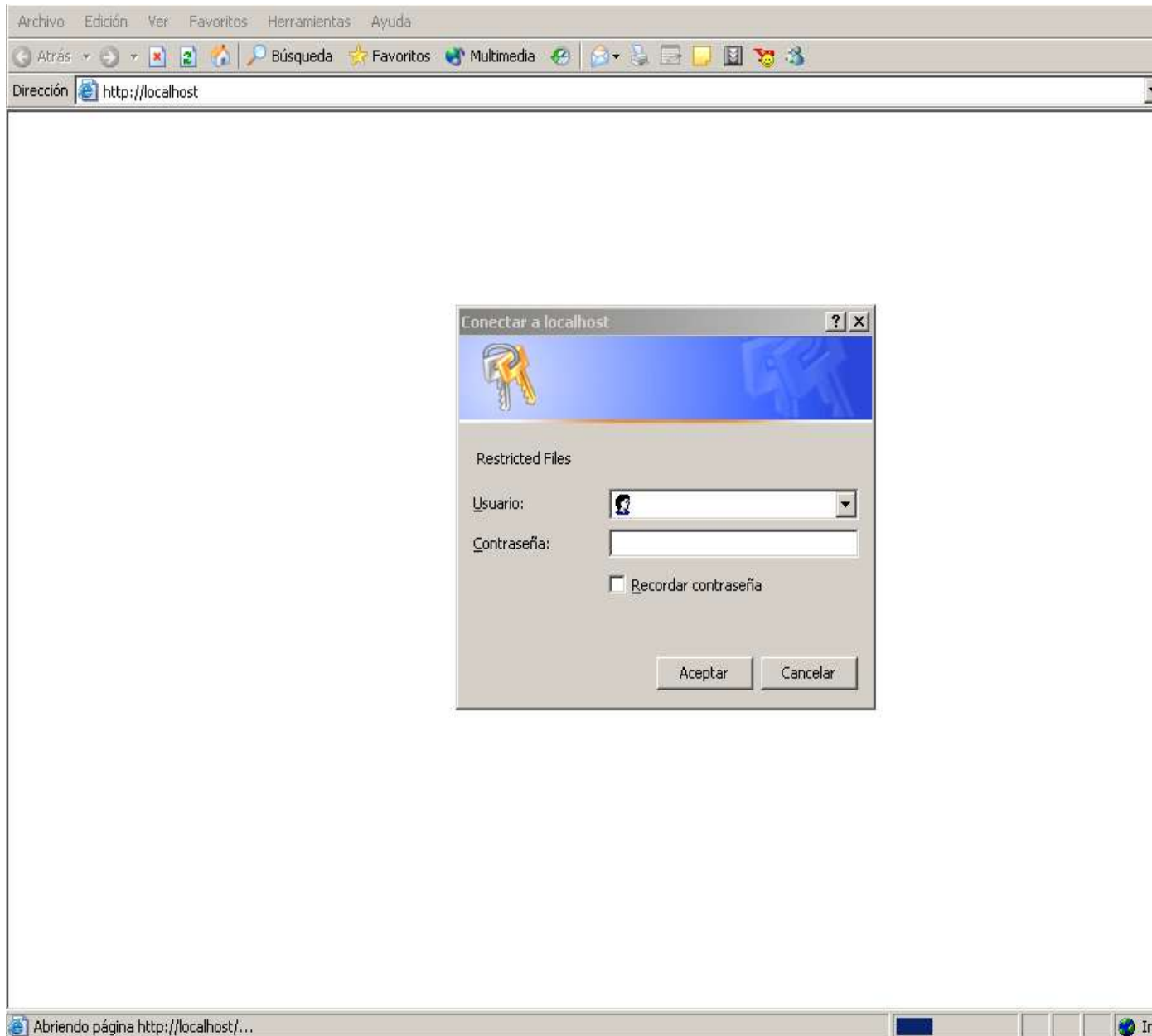
Require user usuario1

En la primera línea se indica el tipo de autenticación, en este caso Basic. La segunda, indica el área de identificación. El área de identificación sirve para, por un lado, que el navegador le muestre al cliente la información como parte del diálogo de usuario/contraseña y por otro, para que pueda identificar las señas (usuario/contraseña) para el área correspondiente. De esta forma, evita tener que preguntar al usuario cada vez que se accede el área. Es una forma de marcar un dominio de validez para las contraseñas. La otra directiva `AuthUserFile`, indica el lugar donde se encuentra el archivo de contraseñas y por último *Requiere user usuario1* indica que el usuario1 puede acceder a el directorio.

Actividad: Configure el servidor para que se permite utilizar `.htaccess` en directorios. (Ver `AllowOverride`) Luego cree un archivo de contraseñas y un archivo `.htaccess` con directivas como las vistas en el directorio raíz del servidor. Verifique que pasa al solicitar en el navegador `http://localhost`.

Las siguientes figuras muestran lo que debería resultar:





En primer lugar el navegador pide usuario y contraseña. Se debe ingresar usuario1 que es el autorizado y la contraseña que se haya especificado al crear el archivo de contraseñas. En el diálogo del navegador se muestra *Restricted Files* que es el valor que especificamos en la directiva *AuthName*



Y si todo funciona correctamente se verá la página inicial de Apache.



De esta forma se ha protegido todo el sitio utilizando el archivo `.htaccess`. Aunque en este caso hubiera sido más fácil modificar directamente el archivo `httpd.conf`.

Es posible también controlar el acceso a nivel grupo. O sea, se definen grupos de usuarios y en lugar de permitir el acceso a cada uno en particular, se autoriza al grupo. De esta forma, se simplifica notoriamente la administración de usuarios y accesos usando la autenticación HTTP básica.

Habiendo creado los dos usuarios *usuario1* y *usuario2*, lo que se necesita es crear un archivo de grupos. Para ello, se utiliza un editor de textos (vi en linux, notepad en windows) como se usó para crear el archivo *.htaccess*.

Se crea un archivo *.htgroup* (u otro similar) en el directorio de donde está ubicado el archivo de contraseñas con el siguiente contenido:

jefes: usuario1 usuario2

La primer columna indica el nombre del grupo, en este caso: *jefes*, luego se ubica una lista de usuarios separados por espacios que son los usuarios del grupo. Con la estructura anterior se creó un grupo *jefes* compuesto por los usuarios *usuario1* y *usuario2*.

Para utilizar los grupos se debe agregar la directiva *AuthGroupFile* como se muestra a continuación:

AuthType Basic

AuthName "Restricted Files"

AuthUserFile "c:/archivos de programa/apache group/apache2/passw/secreto"

AuthUserFile "c:/archivos de programa/apache group/apache2/passw/.htgroup"

Require group jefes

De esta forma, tanto el usuario *usuario1* como *usuario2* tendrán acceso al directorio. En caso de querer agregar más usuarios con permiso sólo deben agregarse al grupo correspondiente sin necesidad de modificar el archivo *.htaccess*.

5.9 SSL Y Apache

El soporte de SSL (Secure Socket Layer) en Apache puede hacerse de dos formas. Por un lado utilizando Apache-SSL que es desarrollado por la propia fundación Apache y es algo así como una versión de Apache corregida con

soporte SSL, o un parche para Apache. La otra forma es utilizar *mod_ssl* que es un módulo que puede ser instalado como cualquier otro. Se verá el segundo método que es el más cómodo para trabajar.

El módulo *mod_ssl* es una interfaz entre Apache y OpenSSL que es una implementación de código abierto de SSL. Por lo tanto es necesario descargar del sitio de OpenSSL los archivos correspondientes (el sitio de OpenSSL es <http://www.openssl.org>).

Obteniendo los binarios de OpenSSL se tienen tres archivos *openssl.exe*, *libeay32.dll* y *ssleay32.dll* las DLL deberían ubicarse en el directorio Win32.

En el caso Linux es necesario instalarlo. Para ello, una vez descargados los ficheros se descomprimen:

```
# tar xvzf openssl-0.9.5a.tar.gz
```

```
# cd openssl-0.9.5a
```

Una vez compilado hay que instalarlo como cualquier programa Linux:

```
# ./config --prefix=/usr/local/ssl
```

```
# make
```

```
# make test
```

```
# make install
```

```
# cd ..
```

Con *--prefix* indicamos donde queremos instalarlo. En este caso en */usr/local/ssl*. La configuración e instalación de *mod_ssl* se realiza en forma similar:

```
# tar xvzf mod_ssl-2.6.4-1.3.12.tar.gz
```

```
# cd mod_ssl-2.6.4-1.3.12
```

```
# ./configure --with-apache=../apache_1.3.12
```

```
# cd ..
```



Claro que si se obtuvieron los binarios no es necesario compilar nada. El paso siguiente en la instalación es instalar el servidor Apache con soporte para SSL. Con los siguientes comandos:

```
SSL_BASE=../openssl-0.9.5a ./configure --prefix=/usr/local/apache \
--enable-module=ssl --enable-shared=ssl

# make

# make certificate TYPE=custom

# make install
```

Luego de `make certificate TYPE=custom` se preguntarán una serie de datos sobre la empresa. Que deben completarse.

En el archivo de configuración hay que habilitar el puerto de SSL, generalmente el 443.

```
<IfModule mod_ssl.c>
```

```
Listen 443
```

```
Listen 80
```

```
</IfModule>
```

Se debe también dejar habilitado:

```
LoadModule ssl_module modules/mod_ssl.so
```

Para que el servidor Apache cargue el módulo `mod_ssl`.

Luego, bajo una directiva *virtual host*, hay que configurar el uso de SSL:

```
<VirtualHost *:443>
```

```
SSLEngine on
```



SSLProtocol all -SSLv3

SSLCertificateFile camino/server.crt

SSLCertificateKeyFile camino/server.key

</ VirtualHost >

Donde camino es el *path* al directorio donde se ubicarán las claves. Debe reiniciar el servidor Apache.

Para crear el certificado en Windows, podemos usar openssl.exe. Al ejecutarlo, se muestra la pantalla siguiente:



O sea OpenSSL espera por comandos. El primer comando a ingresar es:

req -config openssl.cnf -new -out servidor.csr

Este comando hace que OpenSSL pida una serie de datos como se muestra en la siguiente figura (es importante no olvidarse la frase de contraseña que se pide ingresar dos veces para confirmación). Es necesario contar con un archivo de configuración openssl.cnf para que funcione. El Apéndice II cuenta con uno como ejemplo, pero que puede usarse perfectamente, sólo debe ser un archivo de texto.



```
OpenSSL> req -config openssl.cnf -new -out servidor.csr
Loading 'screen' into random state - done
Generating a 1024 bit RSA private key
.....+++++
.....+++++
writing new private key to 'privkey.pem'
Enter PEM pass phrase:
Verifying - 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) []:ES
State or Province Name (full name) []:sa
Locality Name (eg, city) []:Aca
Organization Name (eg, company) []:TC
Organizational Unit Name (eg, section) []:Seccion23
Common Name (eg, your websites domain name) []:www.miempresa.com
Email Address []:webmaster@miempresa.com

Please enter the following 'extra' attributes
to be sent with your certificate request
A challenge password []:nombre mascota
OpenSSL>
```

Luego generamos la clave como se ve en la siguiente figura:

```
OpenSSL> rsa -in privkey.pem -out servidor.key
Enter pass phrase for privkey.pem:
writing RSA key
OpenSSL>
```

La clave se creó usando RSA. Ahora sólo resta crear el certificado y luego con *quit* salir del OpenSSL.

```
OpenSSL> x509 -in servidor.csr -out servidor.cert -req -signkey
servidor.key -days 365
Loading 'screen' into random state - done
Signature ok
subject=/C=ES/ST=sa/L=Aca/O=TC/OU=Seccion23/CN=www.miempresa.com/emailAddress=webmaster@miempresa.com
Getting Private key
OpenSSL> quit
```

Si todo fue hecho correctamente, se tendrán en el directorio cuatro archivos:





Este certificado no tiene valor para desarrollos comerciales ya que en ese caso se debe ponerse en contacto con una Autoridad de Certificación, como *Thawte* o *Verisign*, para que firme nuestro certificado. De todas formas, para propósitos de desarrollo y prueba es suficiente. Los archivos .cert y .key se copian en el directorio donde se desee guardarlos y que coincide con el que configuremos en las directivas (el directorio que llamamos *camino* en el ejemplo anterior).

5.10 Apéndice I Del Fichero De Configuración Hhttpd.conf



Se incluye el extracto del Fichero de Configuración como documentación anexa al tema

5.11 Apéndice II Del Fichero De Configuración Openssl.cnf



Se incluye el extracto del Fichero de Configuración como documentación anexa al tema

