Práctica 1 - Configuración de las aplicaciones

Septiembre 2009

1. Objetivos

El objetivo de esta práctica es aprender a configurar y arrancar un servidor Web, en concreto uno con Apache. También configuraremos un servidor de base de datos MySQL y el módulo de PHP para Apache con soporte para MySQL. Además en la parte final nos familiarizaremos con la configuración avanzada del servidor web Apache. Este programa dispone de un gran número de opciones de configuración. Veremos algunas de ellas y a medida que otras hagan falta se presentarán en las siguientes prácticas.

2. Introducción

En el Laboratorio de Telemática 2 (primera planta del mismo edificio) emplearemos cualquier máquina. Los programas que vamos a configurar y arrancar en esta práctica son:

- Apache 2.2.9
- MySQL 5.0.45

Generalmente existen versiones pre-compiladas de estos programas para diferentes sistemas operativos y microprocesadores y una solución fácil y rápida es descargarse la adecuada e instalarla mediante el sistema que disponga el sistema operativo (.rpm, .pkg, .deb, etc). Sin embargo en las máquinas del laboratorio ya están instalados estos programas de tal forma que cualquier usuario los pueda usar y arrancar con su cuenta y en su espacio. Puedes ver donde están instalados estos programas con el comando ***whereis***:

```
$ whereis httpd
httpd: /usr/sbin/httpd.worker /usr/sbin/httpd /etc/httpd /usr/lib/httpd
 /usr/include/httpd /usr/share/man/man8/httpd.8.gz

$ whereis apachectl
apachectl: /usr/sbin/apachectl /usr/share/man/man8/apachectl.8.gz

$ whereis mysql
mysql: /usr/bin/mysql /usr/lib/mysql /usr/include/mysql /usr/share/mysql
 /usr/share/man/man1/mysql.1.gz

$ whereis mysqladmin
mysqladmin: /usr/bin/mysqladmin /usr/share/man/man1/mysqladmin.1.gz
```

En cualquier caso, lo único que hay que hacer para obtener los paquetes es dirigirse a sus sitios web y descargarlos dado que todos son software abierto (http://www.apache.org, http://www.php.net, http://www.mysql.com).

En esta práctica también te será muy útil el manual del Apache que puedes encontrar en la página web de la asignatura o en la web del Apache. Las referencias a secciones de la documentación se harán con URLs a la página web del Apache pero puedes visitar cualquier otra copia de la documentación.

3. Configuración básica de Apache

3.1. Sobre los módulos

Apache es un servidor modular. Esto quiere decir que acepta añadirle funcionalidades a través de módulos sin necesidad de recompilar el servidor. Solo las funcionalidades básicas suelen estar incluidas en el programa del servidor. Durante la compilación del servidor se puede escoger qué módulos queremos que se compilen para emplear con el mismo. Además, estos módulos podrían incluirse dentro del propio ejecutable del servidor o, si el sistema operativo soporta cargar módulos de forma dinámica (módulos DSO), pueden ser compilados como ficheros separados *.so* para ser cargados cuando se requieran. Podemos especificar qué módulos cargar dinámicamente de los compilados mediante directrices en el fichero de configuración del Apache. Un módulo que siempre estará incluido en Apache será el mod_so que es precisamente el que se encarga de la carga de módulos de forma dinámica (más información en http://httpd.apache.org/docs-2.0/en/dso.html).

Podemos comprobar los módulos que se han incluido en el ejecutable lanzándolo con la opción -l://

```
$ /usr/sbin/httpd -l
Compiled in modules:
  core.c
  prefork.c
  http_core.c
  mod_so.c
```

3.2. Configurar: fichero httpd.conf

Antes de proceder a configurar el Apache y lanzarlo algunas cuestiones sobre las que pensar:

- Si no se especifica el directorio donde se van a tener los documentos web, ¿donde debería suponer Apache que los tiene?
- ¿Puedes manejar (permisos lectura y escritura) ese directorio? ¿Por qué?
- Si lanzámos el Apache ahora mismo, este emplearía la configuración por defecto que incluye emplear el puerto 80 para ejercer sus funciones de servidor Web en el puerto reservado a tal efecto. ¿Podemos lanzar un servidor en ese puerto? ¿Por qué? ¿Quién puede?
- ¿Qué usuario debería lanzar el servidor Apache que vamos a configurar en nuestro HOME?

A lo largo de las diferentes versiones de Apache los ficheros de configuración han cambiado un poco su ubicación y la forma de repartir las directrices de configuración entre un fichero o varios. Se pueden crear varios ficheros de configuración para Apache y luego incluirlos en el principal (como con un **#include** de C). Incluso podemos especificar el fichero de configuración que debe emplear cuando lo lanzamos (con la opción **-f**). Por simplicidad vamos a emplear este último método creando un fichero ***httpd.conf***.

Apache se configura colocando directrices en ficheros de configuración, que están formados por simple texto. Es muy importante tener en cuenta que Apache solo lee el fichero de configuración cuando es lanzado, por lo que cualquier cambio realizado en él requiere pararlo y volverlo a lanzar para que los cambios surtan efecto.

Podemos incluir comentarios en el fichero de configuración comenzando la línea con el carácter **#**. También podemos comprobar la sintaxis del fichero de configuración lanzando el httpd con la opción **-t**:

```
$ /usr/sbin/apachectl -f /opt3/lir/lirXY/www/httpd.conf -t
```

Primero, si no existe crea una carpeta ***www*** en tu HOME. Dentro de ella crea las carpetas ***run***, ***logs*** y ***htdocs***. Además dentro de ***www*** crea un fichero ***httpd.conf*** con el siguiente contenido (donde ***lirXY*** se debe sustituir por su número de grupo de forma que especificamos un directorio dentro del HOME del grupo de prácticas):

```
# Run params
ServerRoot /etc/httpd
PidFile /opt3/lir/lirXY/www/run/httpd.pid
Timeout 120
KeepAlive Off
MaxKeepAliveRequests 100
KeepAliveTimeout 15
UseCanonicalName Off
ErrorLog /opt3/lir/lirXY/www/logs/error_log
LogLevel warn

# Port
Listen 8080

# User & Group
User lirXY
Group lir

# Modules
LoadModule authz_host_module modules/mod_authz_host.so
LoadModule mime_module modules/mod_mime.so
LoadModule mime_magic_module modules/mod_mime_magic.so
LoadModule autoindex_module modules/mod_autoindex.so
LoadModule dir_module modules/mod_dir.so

# Documents directory
DocumentRoot /opt3/lir/lirXY/www/htdocs
<Directory />
    Options FollowSymLinks
    AllowOverride None
</Directory>
<Directory /opt3/lir/lirXY/www/htdocs>
    Options FollowSymLinks
    AllowOverride None
    Order allow,deny
    Allow from all
</Directory>

# Types
DefaultType text/plain
TypesConfig /etc/mime.types
<IfModule mod_mime_magic.c>
    MIMEMagicFile conf/magic
</IfModule>
AddType application/x-compress .Z
AddType application/x-gzip .gz .tgz
```

Mira un poco en la documentación de Apache para que sirve cada directriz:
http://httpd.apache.org/docs/2.0/mod/directives.html

La mayoría son intuitivos sólo por el nombre, como ***Listen, User, Group o DocumentRoot***. Pero algunos, como ***UseCanonicalName o AllowOverride***, no son tan claros.

3.3. Lanzando Apache

Podemos lanzar el servidor Apache simplemente ejecutando el programa **httpd**. Sin embargo, es mucho más cómodo emplear un script preparado para lanzarlo correctamente. Este script es el **apachectl**. El script lanzará el servidor, el cual crea varios procesos para atender a las peticiones de los clientes.

Lanzamos Apache:

```
$ /usr/sbin/apachectl -f /opt3/lir/lirXY/www/httpd.conf -k start
```

Se pueden ver los procesos que quedan corriendo en la máquina como demonios:

```
$ ps aux | grep httpd
8498 ?        Ss     0:00 /usr/sbin/httpd
 -f /opt3/lir/lirXY/www/httpd.conf -k start
8982 ?        S      0:00 /usr/sbin/httpd
 -f /opt3/lir/lirXY/www/httpd.conf -k start
8983 ?        S      0:00 /usr/sbin/httpd
 -f /opt3/lir/lirXY/www/httpd.conf -k start
8984 ?        S      0:00 /usr/sbin/httpd
 -f /opt3/lir/lirXY/www/httpd.conf -k start
8985 ?        S      0:00 /usr/sbin/httpd
 -f /opt3/lir/lirXY/www/httpd.conf -k start
8986 ?        S      0:00 /usr/sbin/httpd
 -f /opt3/lir/lirXY/www/httpd.conf -k start
```

3.4. Probando

Prueba ahora a acceder al contenido web: http://localhost:8080/

¿Qué sucede? ¿Qué te sale? ¿Funciona? Piensa un poco antes de seguir leyendo...

Has configurado un ***DocumentRoot***, pero no has puesto ninguna página que enseñar. Por tanto, ¿qué esperas que te enseñe el Apache? Crea un fichero **vacio.html** en esa carpeta y prueba ahora con:
http://localhost:8080/vacio.html

Pruebe a solicitar la página por defecto desde otra máquina del laboratorio. Recuerde que **localhost** hace referencia a la máquina local en la que se está ejecutando el programa, en este caso el navegador, así que no puede emplear el URI anterior.

Recuerde que los nombre de las máquinas del laboratorio siguen la forma ***tlmAB. net.tlm.unavarra.es***. Si tiene configurado un proxy en el navegador Web le recomienzo que le indique al navegador que no use el proxy para las conexiones con máquinas dentro del dominio ***net.tlm.unavarra.es***, para ello, en la sección de preferencias avanzadas de un Netscape o Mozilla, en la subsección de Proxies añada a “No Proxy for” algo como ***.net.tlm.unavarra.es*** para que cuando intente acceder a una máquina en una máquina cuyo nombre de dominio termine así no le haga la solicitud al proxy.

CHECKPOINT 1

Configura en el Apache:

- que liste el contenido del directorio
- email de administrador
- nombre del servidor
- charset por defecto UTF-8

4. Configuración de Apache con PHP

A continuación vamos a configurar Apache para que ejecute páginas con PHP.

4.1. Configuración necesaria para Apache

Hay dos directrices necesarias en el fichero de configuración de Apache **httpd.conf** para que emplee el módulo PHP y reconozca las páginas PHP.

En primer lugar debemos decirle que cargue el módulo de PHP. Cargar módulos se realiza con la directriz ***LoadModule***. Esta directriz es procesada por el módulo ***mod_so***. Su sintaxis es la siguiente:

```
LoadModule nombre_del_modulo fichero_del_modulo
```

En nuestro caso, debemos tener junto al resto de comandos ***LoadModule*** la siguiente línea en el fichero de configuración:

```
LoadModule php5_module modules/libphp5.so
```

Como podemos ver el último argumento incluye el path para llegar al fichero desde el directorio donde se instaló el Apache.

Lo segundo que debemos modificar en la configuración del Apache es indicarle cómo reconocer los ficheros que contienen código PHP. Lo normal para esto es emplear la extensión del fichero de forma que cuando el servidor vaya a servir un fichero con esa extensión lo reconozca como un fichero con código PHP y lo procese a través del módulo de PHP. La extensión típica es ***.php*** aunque ha habido otras como ***.php3***.

La forma de configurar esto es indicándole al servidor un nuevo tipo MIME asociado a la extensión que hayamos escogido. El tipo MIME para los ficheros PHP debe ser: **application/x-httpd-php** y la forma que tenemos de declararlos en el fichero de configuración es mediante la directriz ***AddType***. La sitanxis de AddType es:

```
AddType MIME_type extension
```

Donde podemos poner varias extensiones que asociar al mismo tipo MIME La línea que debemos incluir será simplemente:

```
AddType application/x-httpd-php .php
```

Reiniciamos el servidor Apache para que cargue la nueva configuración.

```
$ /usr/sbin/apachectl -f /opt3/lir/lirXY/www/httpd.conf -k restart
```

4.2. Página PHP de prueba

Creamos un fichero llamado **prueba.php** en el directorio ***htdocs*** con el siguiente código:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
        <head>
                <title>Ejemplo</title>
        </head>
        <body>
                Si despues de esto no ves nada
                es que PHP no funciona<br>
                <?php
                     echo "Vaya, pues funciona!";
                ?>
        </body>
</html>
```

Y solicitamos esa página desde un navegador poniendo el siguiente URI: http://localhost:8080/prueba.php

El resultado debe ser el siguiente, si no tenemos algún error:

```
Si despues de esto no ves nada es que PHP no funciona

Vaya, pues funciona!
```

4.3. Configuración necesaria para PHP

Hay una directiva de PHP que hay que establecer en el fichero de configuración de PHP, ***php.ini***, para que funcione de una manera óptima. Hay que decirle que no use ***magic quotes***, ni que use ***register globals***. Para esto primero hay que indicar dónde está el fichero de configuración de PHP. Es suficiente con introducir en el **httpd.conf** la siguiente directiva:

```
PHPIniDir "/opt3/lir/lirXY/www"
```

Ahora para realizar la configuración es suficiente con crear el fichero **php.ini** en **/opt3/lir/lirXY/www** con:

```
magic_quotes_gpc = Off
register_globals = Off
```

Y por supuesto, reiniciar de nuevo el servidor Apache. Podemos comprobar que todo ha ido bien haciendo un nuevo PHP **phpinfo.php** con:

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

Al solicitar la página (http://localhost:8080/phpinfo.php), nos debería de salir un listado con toda la configuración de Apache y PHP, incluido estas variables.

5. Configuración de MySQL

Ahora vamos a configurar correctamente MySQL y a arrancar el servidor de base de datos para poder usarlo con Apache-PHP.

5.1. Pasos previos

Antes de lanzar por primera vez el servidor de base de datos debemos ejecutar un script que termina la configuración necesaria. Este script creará la base de datos en la que se guardan los permisos de acceso que demos a las diferentes bases de datos que se creen, así como información de los usuarios. También se crea una base de datos para pruebas y un usuario inicial (root). Con la opción ***–datadir*** le podemos decir dónde queremos tener las bases de datos (las ponemos en nuestro directorio) y con la opción ***–socket*** indicamos dónde se creará el socket UNIX para la comunicación local con la base de datos (para cuando no se hace por la red sino en la misma máquina), que también colocamos en el directorio del usuario.

```
$ mkdir -p /opt3/lir/lirXY/www/mysql/var
$ /usr/bin/mysql_install_db --datadir=/opt3/lir/lirXY/www/mysql/var
 --socket=/opt3/lir/lirXY/www/socket.sock
```

El programa servidor de la base de datos se llama ***mysqld*** (la 'd' viene de 'daemon' o 'demonio', igual que en ***httpd***). Hay scripts para lanzarlo pero en estas prácticas vamos a comenzar lanzándolo directamente con el ejecutable:

```
$ /usr/libexec/mysqld --datadir=/opt3/lir/lirXY/www/mysql/var
 --socket=/opt3/lir/lirXY/www/socket.sock &
```

Ahora podemos comprobar que el servidor está corriendo correctamente empleando el programa ***mysqladmin*** que permite realizar algunas tareas de gestión.

```
$ /usr/bin/mysqladmin --socket=/opt3/lir/lirXY/www/socket.sock
 -u root version
```

Deberíamos obtener información sobre el servidor. Como por ejemplo:

```
$ /usr/bin/mysqladmin  Ver 8.41 Distrib 5.0.45, for redhat-linux-gnu on i386
Copyright (C) 2000-2006 MySQL AB
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL license

Server version          5.0.45
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /opt3/lir/lirXY/www/socket.sock
Uptime:                 53 sec

Threads: 1  Questions: 1  Slow queries: 0  Opens: 12  Flush tables: 1
  Open tables: 6  Queries per second avg: 0.019
```

Cuestión: Averigüe cómo detener el servidor de base de datos con la opción ***shutdown*** del comando ***mysqladmin***.

El servidor de la base de datos controla los accesos a las mismas mediante un sistema de usuarios y privilegios. Inicialmente existe un usuario llamado root que tiene todos los privilegios posibles y carece de password. No se deben confundir los usuarios de la base de datos con los usuarios de una máquina Unix. Ambos conjuntos de usuarios son independientes. Se emplea el nombre ***root*** para ese superusuario por la tradición existente en máquinas Unix pero el usuario root de nuestro servidor de base de datos es independiente del usuario root de la máquina.

Podemos cambiar la password de este usuario empleando el siguiente comando: (*NOTA: sustituir* ***nuevapassword*** *por el password nuevo que querais usar*)

```
$ /usr/bin/mysqladmin --socket=/opt3/lir/lirXY/www/socket.sock
 -u root password nuevapassword
```

La instalación de MySQL ofrece un cliente para realizar comandos sobre la base de datos. Este cliente es un programa para la shell llamado ***mysql***:

```
$ /usr/bin/mysql --socket=/opt3/lir/lirXY/www/socket.sock -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 3
Server version: 5.0.45 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql>
```

Como se puede ver empleamos la opción ***-u*** para especificar el usuario con el que nos queremos conectar a la base de datos y con la opción ***-p*** hacemos que nos solicite la password del usuario.

Para salir del programa basta con emplear el comando ***quit***:

```
mysql> quit
Bye
```

Vamos a comprobar la correcta instalación realizando algunas consultas básicas al servidor. Primero solicitamos una lista de las bases de datos existentes. Si estamos empleando el usuario root tendremos acceso a ver todas las que hay que al instalar son:

```
$ /usr/bin/mysql --socket=/opt3/lir/lirXY/www/socket.sock -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 4
Server version: 5.0.45 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| mysql              |
| test               |
+--------------------+
3 rows in set (0.01 sec)
```

```
mysql> quit
Bye
```

Las bases de datos ***information_schema***, ***mysql*** y ***test*** fueron creadas por el script ***mysql_install_db***. La primera contiene información sobre el esquema del resto de base de datos y la segunda contiene la información referente a los usuarios y sus privilegios de acceso al servidor y por eso solo es visible para el superusuario. La última (***test***) es una base de datos creada para realizar pruebas.

Ahora vamos a conectarnos al servidor y a crear una nueva tabla dentro de la base de datos de pruebas:

```
$ /usr/bin/mysql --socket=/opt3/lir/lirXY/www/socket.sock -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
Server version: 5.0.45 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> use test;
Database changed
mysql> create table precios (nombre varchar(50), valor double(6,2));
Query OK, 0 rows affected (0.15 sec)

mysql> describe precios;
+--------+--------------+------+-----+---------+-------+
| Field  | Type         | Null | Key | Default | Extra |
+--------+--------------+------+-----+---------+-------+
| nombre | varchar(50)  | YES  |     | NULL    |       |
| valor  | double(6,2)  | YES  |     | NULL    |       |
+--------+--------------+------+-----+---------+-------+
2 rows in set (0.15 sec)

mysql> insert into precios values ("El taxi", 10000);
Query OK, 1 row affected, 1 warning (0.00 sec)

mysql> insert into precios values ("Un duro", .03);
Query OK, 1 row affected (0.00 sec)

mysql> select * from precios;
+----------+----------+
| nombre   | valor    |
+----------+----------+
| El taxi  | 9999.99  |
| Un duro  |    0.03  |
+----------+----------+
2 rows in set (0.02 sec)

mysql> quit
Bye
```

Con el comando ***use*** hemos indicado la base de datos con la que queremos trabajar. El comando ***create*** sirve para crear una nueva tabla en la base de datos, en este caso el nombre de la tabla es ***precios*** y tiene dos columnas: una cadena de 50 caracteres de nombre ***nombre*** y un número flotante con presentación de 6 cifras enteras y 2 decimales con nombre ***valor***. Con el comando ***describe*** hemos podido ver las columnas de la base de datos que coinciden con lo deseado. A continuación hemos insertado (comando ***insert*** por si alguien se lo preguntaba) dos nuevas filas en la tabla: el precio del taxi y el precio de un duro. Finalmente hemos realizado una petición al servidor con ***SELECT*** para que nos mostrase todas las entradas de la tabla ***precios*** recién creada.

Finalmente, vamos a crear un usuario de la base de datos MySQL (no tiene nada que ver con un usuario UNIX) al que se le permitirá acceder a las tablas de esa base de datos desde cualquier máquina en la red (en este punto vamos a ser bastante permisivos, en prácticas futuras pueden estudiar los privilegios de usuarios y restringir un poco el acceso si lo desean). También vamos a darle permisos al usuario root del servidor de base datos para conectarse mediante TCP siempre que sea desde la máquina local.

*NOTA: sustituir las palabras **laclave** y **laclavepararoot** por los passwords que querais y recordarlos.*

```
$ /usr/bin/mysql --socket=/opt3/lir/lirXY/www/socket.sock -u root -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 6
Server version: 5.0.45 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> grant all on test.* to lir@'%' identified by 'laclave';
Query OK, 0 rows affected (0.01 sec)

mysql> grant all on *.* to root@'localhost.localdomain' identified
 by 'laclavepararoot';
Query OK, 0 rows affected (0.00 sec)

mysql> quit
Bye

$ /usr/bin/mysql -h 127.0.0.1 -u lir -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 7
Server version: 5.0.45 Source distribution

Type 'help;' or '\h' for help. Type '\c' to clear the buffer.

mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| test               |
+--------------------+
2 rows in set (0.00 sec)

mysql> quit
Bye
```

5.2. Probando el soporte de MySQL desde Apache-PHP

Vamos a comprobar que somos capaces de acceder a la base de datos desde scripts PHP. Para ello crearemos un simple script que se conecte a la base de datos y solicite el contenido de la tabla precios de forma similar a como hicimos anteriormente con el cliente mysql solo que ahora mostrando nosotros el contenido con una tabla HTML.

Creamos un fichero llamado **pruebasql.php** en el directorio ***/opt3/lir/lirXY/www/htdocs/*** con el siguiente código:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
 "http://www.w3.org/TR/html4/strict.dtd">
<html>
<head>
        <title>Probando acceso a MySQL</title>
</head>
<body>
        Si despues de esta linea no ves nada es que PHP no funciona<br>
        <?php
        /* Conectarse empleando un usuario en concreto */
        /* Si no exites usuario, es como emplear usuario anonimo. */
        $link = mysql_connect("127.0.0.1", "lir", "laclave")
 or die("Fallo al conectar con la base de datos");
        print "Conectado<br>";
        /* Seleccionar la base de datos a emplear */
        mysql_select_db("test") or die("Error al seleccionar test");
        print "Base de datos <tt>test</tt> seleccionada<br>";

        /* Realizar una busqueda */
        $query = "SELECT * FROM precios";
        $result = mysql_query($query) or die("Query failed");

        /* Sacar el contenido como una tabla HTML */
        print "<table border=1>\n";
        while ($line = mysql_fetch_array($result, MYSQL_ASSOC)){

                print "\t<tr>\n";
                foreach ($line as $col_value){
                        print "\t\t<td>$col_value</td>\n";
                }
                print "\t</tr>\n";
        }
        print "</table>\n";
        ?>
</body>
</html>
```

Y solicitamos esa página desde un navegador poniendo el siguiente URI: http://localhost:8080/pruebasql.php

El resultado debe ser el siguiente, si no tenemos algún error:

```
Si despues de esta linea no ves nada es que PHP no funciona
Conectado
Base de datos test seleccionada
El taxi          9999.99
Un duro          0.03
```

CHECKPOINT 2

Muestre al responsable de prácticas que todo le funciona correctamente

5.3. Arranque de los servicios

En un entorno real con un ordenador como servidor web, al arrancarse el ordenador todos los servicios relacionados con el servidor web se deberían de iniciar. En nuestro caso eso significa en que el Apache y el MySQL

arranquen. En el entorno del laboratorio de Telemática realizar esta tarea es difícil, pero si que se puede crear un pequeño script que al ejecutarlo inicie estas dos aplicaciones. Crea el fichero bash **arraque.sh**:

```
#!/bin/bash
/usr/libexec/mysqld --datadir=/opt3/lir/lirXY/www/mysql/var
 --socket=/opt3/lir/lirXY/www/socket.sock &
/usr/sbin/apachectl -f /opt3/lir/lirXY/www/httpd.conf -k start
```

Dale permisos de ejecución. Si quieres puedes probar a ejecutarlo, pero como en teoría ya tienes MySQL corriendo te dará un error y sólo reiniciará Apache.

Ahora lo único que tienes que hacer es recordar ejecutar este script cada vez que empieces las prácticas de LIR.

También podemos preparar un script para parar los servidores cada vez que te acabes las prácticas de LIR, puesto que sino los servidores se quedarán en ejecución y si algún compañero tuyo quiere usar esa máquina tendrá que reiniciar.

Crea el fichero bash **parada.sh** y dale permisos de ejecución:

```
#!/bin/bash
/usr/sbin/apachectl -f /opt3/lir/lirXY/www/httpd.conf -k stop
/usr/bin/mysqladmin --socket=/opt3/lir/lirXY/www/socket.sock
 -u root -p shutdown
```

6. Configuración avanzada de Apache

6.1. Otras opciones en Apache

Las directrices colocadas directamente en el fichero de configuración se aplican a todo el servidor. Sin embargo, se pueden colocar directrices dentro del ámbito de otra, de forma que solo apliquen dentro del alcance de la segunda. Por ejemplo, las directrices colocadas entre un ***<Directory>*** y su correspondiente ***</Directory>*** se aplican solo para el directorio indicado. Ejemplo:

```
<Directory /home/miweb>
        AllowOverride All
</Directory>
```

Pero hay algunas directrices que solo tienen sentido de forma global y por lo tanto no se pueden colocar dentro de estas directrices de bloque.

Estudie el significado de las siguientes directrices de bloque:

- Directory
- Files
- Location

Encontrará una buena explicación de las diferencias entre ellas en:
http://httpd.apache.org/docs/2.0/sections.html

Cuestión: El sistema de ficheros ext3 empleado en los linux del laboratorio distingue mayúsculas de minúsculas en los nombres de ficheros. Sin embargo, por ejemplo el sistema de ficheros HFS+ no las distingue, para él

el fichero MiFichero y MIfichero son el mismo. ¿Qué consecuencias puede tener esto a la hora de limitar el acceso a ciertos directorios del servidor web?

Modifique la configuración actual para que:

- El servidor espere conexiones al puerto ***2020*** y al ***8080***
- Sirva los ficheros que se coloquen en ***/opt3/lir/lirXY/www/htdocs2***
- Cuando se solicite un directorio intente primero servir un fichero ***index.php*** que esté en el directorio y si no lo encuentra busque un ***index.html*** y si no un ***index.htm***

Podemos controlar a qué clientes serviremos ciertas páginas y a cuáles no. Para ello disponemos de las directrices Allow y Deny. Puede encontrar una explicación de su funcionamiento en:
http://httpd.apache.org/docs/2.0/mod/mod_access.html#allow

Continuando con la configuración modifique la configuración para que ***sólo se sirvan las páginas a las máquinas de la tlm11 a la tlm31***.

Hasta ahora hemos visto directrices que hemos colocado en el fichero de configuración global httpd.conf. Sin embargo, podemos colocar directrices también en otros ficheros. Especialmente interesante son los ficheros que normalmente se llaman **.htaccess** que permiten cambiar la configuración que se aplica a los directorios en los que se encuentran. Todo lo que se puede hacer con ellos se puede hacer desde el fichero de configuración global pero hay situaciones en las que son útiles. Podemos controlar el tipo de directrices que se pueden colocar en estos ficheros mediante la directriz ***AllowOverride***. Puede encontrar un tutorial respecto al empleo de estos ficheros en:
http://httpd.apache.org/docs/2.0/howto/htaccess.html.

Modifique la configuración de su servidor para que dentro de ***/opt3/lir/lirXY/www/htdocs2*** emplee ficheros **.htaccess** y configure mediante un fichero de este tipo que no liste el contenido del directorio.

CHECKPOINT 3

Muestre al responsable de prácticas que todo le funciona correctamente

6.2. Virtual Hosts

Existen más directrices de bloque. Una de ellas es **<*VirtualHost*>** la cual nos permite servir más de un sitio Web con el mismo servidor Apache.

Suponga que queremos instalar el servidor web para dos empresas independientes. Una posibilidad es tener dos máquinas distintas, cada una con su interfaz de red, su dirección IP, su nombre DNS y ejecutando Apache, una contiene las páginas de la empresa 1 y se llama enterprise1.midominio.net y la otra contiene las páginas de la empresa 2 y se llama enterprise2.sudominio.org.

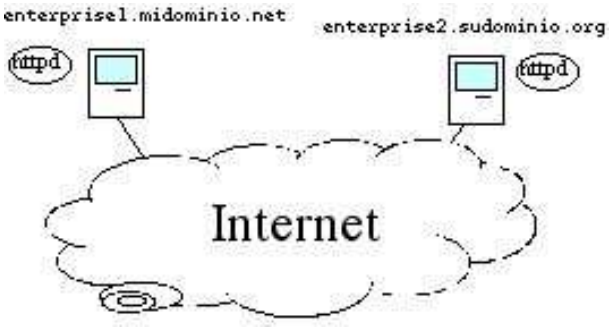

Figura 1: Dos servidores independientes

Perfecto, pero hay más formas de hacerlo. Por ejemplo, podríamos tener una sola máquina pero con dos tarjetas de red, cada una con una dirección IP. Para cada IP tenemos un nombre de dominio (DNS) diferente. Entonces podemos ejecutar dos copias del programa Apache simultáneamente, una de ellas atendiendo a peticiones que vengan a una de las direcciones IP y el otro atendiendo a las peticiones que vayan a la otra. Cada programa Apache corriendo posee un fichero de configuración diferente y un directorio con páginas web a servir diferente. Estudie cómo debería configurar y lanzar cada copia de Apache.

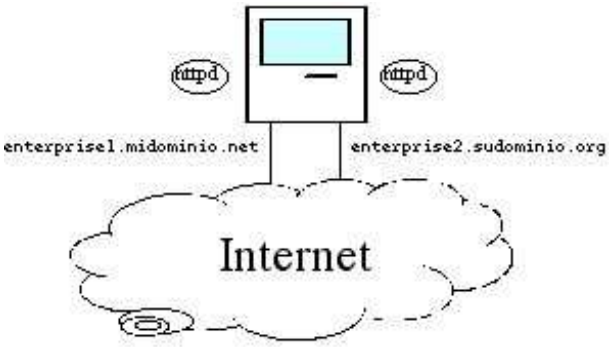

Figura 2: Un servidor con 2 interfaces y 2 programa Apache corriendo

Una tercera posibilidad es tener dos direcciones IP pero solo ejecutar un servidor Apache, no dos. Ese servidor tendrá que servir las páginas de una u otra empresa según a qué interfaz se dirijan las peticiones.

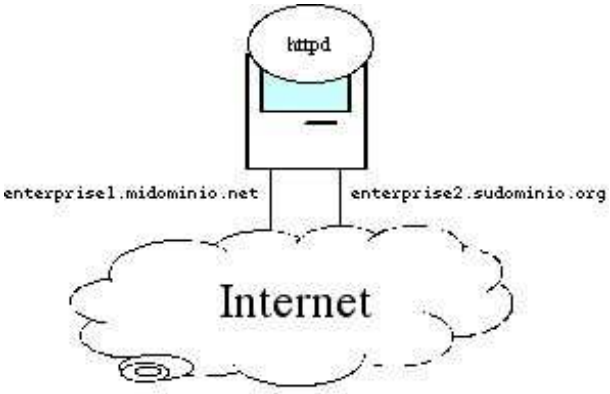

Figura 3: Un servidor con 2 interfaces y 1 programa Apache corriendo

Esta tercera posibilidad es la que vamos a intentar configurar a continuación. Las máquinas del laboratorio no tienen dos tarjetas de red pero sí tienen dos interfaces de red y por tanto dos direcciones IP (lo cual puede comprobar ejecutando el programa ***/sbin/ifconfig***). Esto se debe a que normalmente todas las máquinas que emplean TCP/IP tienen un interfaz que se llama de Loopback que no corresponde a una tarjeta de red sino a software dentro del sistema operativo (podríamos tener este interfaz sin tener tarjeta de red). Este interfaz tiene configurada la dirección IP 127.0.0.1 y todo el tráfico que se dirija a esa IP se queda dentro de la máquina.

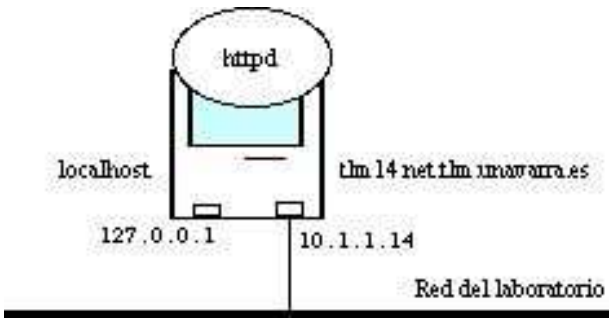

Figura 4: Un servidor con 2 interfaces (uno es localhost) y 1 programa Apache corriendo

Estudie la sintaxis de la directriz <***VirtualHost***>. Por ejemplo lea:
http://httpd.apache.org/docs/2.0/mod/core.html#virtualhost

Configure su servidor Apache para que:

- Atienda a conexiones dirigidas al puerto 8080
- Sirva el fichero ***/opt3/lir/lirXY/www/htdocs2/presentacion.html*** cuando se le solicite la página http://localhost:8080/presentacion.html
- y cuando se le solicite la página http://tlmAB.net.tlm.unavarra.es:8080/presentacion.html sirva el fichero ***/opt3/lir/lirXY/www/htdocs/presentacion.html***, donde AB debe substituirse por lo que corresponda a la máquina donde tenga corriendo el Apache
- Cree /opt3/lir/lirXY/www/htdocs2/presentacion.html y /opt3/lir/lirXY/www/htdocs/presentacion.html para probarlo

CHECKPOINT 4

Muestre al profesor de prácticas que le funciona esta última configuración y explique cómo lo ha hecho.

6.3. Configuración de 2 direcciones con sólo una interfaz de red

Supongamos que nuestra máquina tiene un solo interfaz de red (ignoremos la IP 127.0.0.1 dado que nadie de fuera de nuestra máquina puede en realidad comunicarse con ella!). Las máquinas del laboratorio tienen como nombre tlmAB.net.tlm.unavarra.es el cual se convierte en la dirección IP 10.1.1.AB.

Hemos configurado el servidor de DNS del laboratorio para que ADEMÁS, también se resuelva como la dirección 10.1.1.AB el nombre wwwAB.net.tlm.unavarra.es. Es decir, podemos dirigirnos a nuestras máquinas por cualquiera de los dos nombres y ambos se convertirán en la misma dirección IP antes de mandar los paquetes IP (recuerde que en los paquetes IP se ponen direcciones IP, no nombres, si tenemos un nombre, antes de poder enviar un paquete IP a esa máquina debemos averiguar a qué dirección IP corresponde ese nombre).

Tarea:

- Modifique la configuración de su servidor web para que ofrezca los ficheros en ***/opt3/lir/lirXY/www/htdocs*** cuando se soliciten con el formato http://tlmAB.net.tlm.unavarra.es:8080
- Además debe servir (con la misma configuración) del directorio ***/opt3/lir/lirXY/www/htdocs2*** cuando se le soliciten con el nombre http://wwwAB.net.tlm.unavarra.es:8080
- No debe aceptar peticiones a ningún otro puerto.

Cuestión: ¿Cómo puede saber el servidor Web que se le está solicitando una página dirigiéndose a él con un nombre o con otro si en la cabecera de los paquetes IP solo aparecen direcciones IP y no nombres?

CHECKPOINT 5

Muestre al responsable de prácticas que todo le funciona correctamente

7. Resumen y conclusiones

En esta práctica hemos configurado un servidor Apache con soporte de PHP, un servidor de bases de datos MySQL y hemos comprobado el correcto funcionamiento de todo.

La configuración ha sido a partir de los programas ya instalados en las máquinas compartidas y se ha realizado en directorio HOME del usuario.

Se han utilizado ejemplos simples de HTML, PHP y SQL a sabiendas de que se profundizará en el uso de todos ellos en las siguientes prácticas.

También hemos aprendido a configurar Apache para que ofrezca varias webs diferentes. Esto nos servirá en las siguientes prácticas dado que el servidor Web va a ser la herramienta que dé el soporte principal a los scripts que creemos.