Curso de administración de sistemas GNU/linux

MySQL: Servidor de Bases de Datos

Por Jorge Fuertes
http://jorgefuertes.com
jorge@jorgefuertes.com

Revisado a 22 de mayo de 2009

Índice

1. Introducción **4**
1.1. El servidor MySQL 4

2. Instalando MySQL **4**

3. Arranque y parada del servidor MySQL **5**

4. El cliente MySQL **6**
4.1. Opciones del cliente 6
4.2. Otras opciones interesantes 7
4.3. La entrada estándar 8

5. Escuchar en otra IP **8**

6. Trabajando con las bases de datos **8**
6.1. Creando bases de datos 8
6.2. Listando las bases de datos 9
6.3. Borrando bases de datos 9

7. Agregando usuarios **9**

8. Trabajando con tablas **11**
8.1. Creando una tabla 11
8.2. Listando las tablas 12
8.3. Borrando una tabla 12

9. Trabajando con los datos **13**
9.1. Insertando datos 13
9.2. Seleccionando datos 13
9.3. Actualizando datos 14
9.4. Borrando datos 14

10. Ficheros y copias de seguridad **15**
10.1. Configuración 15
10.2. Los datos 15
10.3. Copias de seguridad binarias 16
10.4. Copias de seguridad SQL 16

11. Ejercicios **19**
11.1. Notas 19
11.2. Enunciados 19
11.3. Soluciones 23
11.3.1. Programa de Altas 23
11.3.2. Alta de usuarios 24
11.3.3. Comprobar servidores MySQL 25
11.3.4. Programa de gestión (Ej. de 4 a 10) 25
11.3.5. Copia binaria 30

12.Sobre esta unidad didáctica **31**
12.1. Notas y advertencias 31
12.2. Derechos 31
12.3. Agradecimientos 31
12.4. Revisiones 31

13.Anexo: Una biblioteca de funciones de ejemplo **32**
13.1. Introducción 32
13.2. La biblioteca de uso general 32

1. Introducción

1.1. El servidor MySQL

MySQL[1] es un **servidor de Bases de Datos**[2], es decir un programa que es capaz de guardar y servir todo tipo de datos agrupados en tablas, y que además se puede manejar utilizando el lenguaje **SQL**[3].

Es software libre, esto quiere decir que podemos utilizarlo libremente, además de copiarlo o modificarlo, ya que disponemos de acceso a su código fuente. Actualmente el equipo de desarrollo es de la empresa Sun Microsystems[4] aunque dicha empresa ha sido adquirida recientemente por Oracle Corporation[5], el gigante de las bases de datos.

El lenguaje SQL (*Structured Query Language*) es un relativamente sencillo lenguaje para interactuar con sistemas de bases de datos. Permite consultar, insertar, actualizar y borrar datos, además de disponer de muchas otras funciones integradas.

En esta guía veremos como instalar el servidor MySQL y como realizar diversas operaciones con él, todo ello enlazando con lo ya aprendido de bash-scripting.

Esta guía no pretende ser exhaustiva, ni un manual de MySQL. Sólo es una unidad didáctica para introducir al alumno en el uso de este gestor de bases de datos y dotarle de conocimientos de instalación y mantenimiento.

NOTA: Aparecerá en repetidas ocasiones el conjunto de símbolos "`#>`". Son los símbolos que indican que se debe de estar en la línea de comandos de su consola, el *prompt* de su sistema, que puede ser diferente. No debe teclear esos símbolos, son sólo una indicación. Si por el contrario aparece el símbolo "`mysql>`" querrá decir que debemos introducir esos comandos en el cliente de MySQL y no en el shell.

2. Instalando MySQL

Para instalar el servidor y el cliente MySQL en una Debian, necesitamos instalar el paquete *mysql-server - MySQL database server (metapackage depending on the latest version)*, el cual introducirá automáticamente todas las dependencias. Para ello ejecutamos:

```
#> aptitude install mysql-server

Leyendo lista de paquetes... Hecho
Creando árbol de dependencias
Leyendo la información de estado... Hecho
Leyendo la información de estado extendido
Inicializando el estado de los paquetes... Hecho
Leyendo las descripciones de las tareas... Hecho
Se instalarán los siguiente paquetes NUEVOS:
  libdbd-mysql-perl{a} libdbi-perl{a} libhtml-template-perl{a}
```

[1] http://es.wikipedia.org/wiki/MySQL
[2] http://es.wikipedia.org/wiki/Base_de_datos
[3] http://es.wikipedia.org/wiki/SQL
[4] http://es.wikipedia.org/wiki/Sun_Microsystems
[5] http://es.wikipedia.org/wiki/Oracle_Corporation

```
  libmysqlclient15off{a} libnet-daemon-perl{a} libplrpc-perl{a}
  libterm-readkey-perl{a} mysql-client-5.0{a} mysql-common{a}
  mysql-server mysql-server-5.0{a}
0 paquetes actualizados, 11 nuevos instalados, 0 para eliminar y 0 sin actualizar.
Necesito descargar 38,5MB de ficheros. Después de desempaquetar se usarán 115MB.
¿Quiere continuar? [Y/n/?]
```

Contestaremos "Y" a la pregunta y se procederá a la instalación. Como vemos entre las dependencias se encuentra ya el cliente MySQL que necesitaremos más adelante.

Durante la instalación, el sistema de paquetes nos solicitará una contraseña de **root** para *MySQL*. No se trata de la contraseña de root del sistema, sino de una contraseña nueva para el usuario root de MySQL. Podemos poner lo que queramos, algo que vayamos a recordar ya que si nos olvidamos estaremos ante un problema bastante grave, y siempre intentando cumplir unos mínimos requisitos de *fuerza*[6] para nuestras contraseñas, como combinar números y letras y algún signo de puntuación, y no usar palabras del diccionario.

Una vez definida la contraseña maestra, el sistema seguirá con la actualización hasta completarla y ya dispondremos de un servidor de bases de datos operativo.

3. Arranque y parada del servidor MySQL

Nuestra distribución Debian ya habrá hecho lo necesario para arrancar MySQL y para agregar los ficheros de arranque y parada a la correspondiente estructura de `/etc/init.d` para que el servidor arranque al iniciar el sistema (runlevel 3) y se detenga al apagarlo.

Si quisiéramos hacerlo manualmente utilizaríamos `invoke-rc.d` de la siguiente forma:

- Parando:

  ```
  #> invoke-rc.d mysql stop
  Stopping MySQL database server: mysqld.
  ```

- Arrancando:

  ```
  #> invoke-rc.d mysql start
  Starting MySQL database server: mysqld.
  Checking for corrupt, not cleanly closed and upgrade needing
  tables..
  ```

- Reiniciando:

  ```
  #> invoke-rc.d mysql restart
  Stopping MySQL database server: mysqld.
  Starting MySQL database server: mysqld.
  Checking for corrupt, not cleanly closed and upgrade needing
  tables..
  ```

[6] http://es.wikipedia.org/wiki/Contrase%C3%B1a#Factores_en_la_seguridad_de_contrase.C3.B1as_individuales

- Recargando configuración:

```
#> invoke-rc.d mysql reload
Reloading MySQL database server: mysqld.
```

- Obteniendo información de estado:

```
#> invoke-rc.d mysql status
/usr/bin/mysqladmin  Ver 8.41 Distrib 5.0.51a, for
debian-linux-gnu on x86_64
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          5.0.51a-24+lenny1
Protocol version        10
Connection              Localhost via UNIX socket
UNIX socket             /var/run/mysqld/mysqld.sock
Uptime:                 1 min 12 sec

Threads: 1  Questions: 80  Slow queries: 0  Opens: 23
Flush tables: 1  Open tables: 17
Queries per second avg: 1.111.
```

4. El cliente MySQL

El cliente "mysql" es un programa que nos permite interactuar con el servidor de bases de datos desde la línea de comandos. Para invocarlo simplemente utilizaremos el comando `mysql`, si bien al invocarlo sin parámetros obtendremos el siguiente error:

```
#> mysql
ERROR 1045 (28000): Access denied for user 'root'@'localhost'
  (using password: NO)
```

Esto es porque necesitamos la contraseña para conectar con nuestro servidor, mejor lo haremos de la forma explicada en el siguiente punto.

4.1. Opciones del cliente

```
#> mysql -p
Enter password:
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 34
Server version: 5.0.51a-24+lenny1 (Debian)

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

mysql>
```

Al introducir la opción "-p" estamos diciendo al cliente mysql que queremos introducir una contraseña, cosa que hacemos a continuación cuando se nos pide "*Enter password*". Si introducimos correctamente la contraseña de *root* el cliente nos devolverá el prompt `mysql>`.

Para volver al shell del sistema utilizaremos la palabra **exit** o bien pulsaremos la combinación de teclas **Ctrl+d**.

Para comprobar que estamos correctamente conectados al servidor podemos utilizar la orden "**status**":

```
mysql> status
--------------
mysql  Ver 14.12 Distrib 5.0.51a, for debian-linux-gnu (x86_64)
  using readline 5.2

Connection id:          35
Current database:
Current user:           root@localhost
SSL:                    Not in use
Current pager:          stdout
Using outfile:          ''
Using delimiter:        ;
Server version:         5.0.51a-24+lenny1 (Debian)
Protocol version:       10
Connection:             Localhost via UNIX socket
Server characterset:    latin1
Db     characterset:    latin1
Client characterset:    latin1
Conn.  characterset:    latin1
UNIX socket:            /var/run/mysqld/mysqld.sock
Uptime:                 25 min 16 sec

Threads: 1  Questions: 221  Slow queries: 0  Opens: 148  Flush
  tables: 2  Open tables: 17  Queries per second avg: 0.146
--------------
```

4.2. Otras opciones interesantes

- –**user**: Para especificar el nombre de usuario:
  Ej.: `--user=pepe`
- –**password**: Para especificar la contraseña:
  Ej.: `--password=pe.123-z`
- –**host**: El servidor al que se debe conectar:
  Ej.: `--host=192.168.1.1`
- –**database**: Nombre de la BDD sobre la que operar:
  Ej: `--database=facturas`

Más opciones con `mysql --help`.

4.3. La entrada estándar

El cliente mysql admite la entrada de comandos SQL por la entrada estándar. Por ejemplo para ver una lista de bases de datos podríamos hacer:

```
#> echo "SHOW DATABASES;" | mysql --user=root --password=clave
Database
information_schema
curso_usuario
mysql
```

Se pueden por tanto inyectar comandos con un simple *echo* o bien se pueden volcar todo un fichero de comandos SQL y de datos con un *cat*. De hecho es un mecanismo muy utilizado para transferir BDDs de un sitio a otro o para restaurar copias de seguridad.

5. Escuchar en otra IP

Por defecto el servidor MySQL sólo escuchará en la IP `127.0.0.1`, que es la interna o de *loopback* de la máquina, esto nos sirve para utilizarlo con nuestros propios programas internos, y para el cliente mysql de nuestro ordenador, pero no servirá para admitir ninguna consulta externa.

Para que nuestro servidor MySQL sea *visible* desde fuera, deberemos hacer que escuche en las direcciones externas de la máquina[7]. Para conseguirlo deberemos editar el fichero de configuración de MySQL, que es */etc/mysql/my.cnf*:

```
#> vi /etc/mysql/my.cnf

# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address            = 127.0.0.1
# Se omite gran parte de este fichero para mejorar la
# comprensión.
```

Deberemos cambiar la directiva "*bind-address*" para reflejar la IP en la que deseamos que el servidor escuche o bien eliminarla (comentarla mejor) para que escuche en todas las IPs de la máquina. A los efectos del curso la dejaremos así:

```
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
#bind-address            = 127.0.0.1
```

6. Trabajando con las bases de datos

6.1. Creando bases de datos

Crear una nueva base de datos vacía es tan fácil como esto:

[7] Nótese que el sistema de cortafuegos que utilizamos en nuestra red deberá de contemplar este extremo, permitiendo o denegando el servicio según corresponda. La seguridad basada exclusivamente en los nombres de usuario y contraseñas puede no ser suficiente.

```
mysql> CREATE DATABASE nombrebdd DEFAULT CHARACTER SET utf8
       COLLATE utf8_unicode_ci;
```

Podemos comprobar que se ha creado con el siguiente comando:

```
mysql> SHOW DATABASES;
```

6.2. Listando las bases de datos

Para obtener una lista de las bases de datos ejecutaremos *SHOW DATABASES*:

```
mysql> SHOW DATABASES;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| curso_usuario      |
| mibdd              |
| mysql              |
+--------------------+
4 rows in set (0.06 sec)
```

Las bases de datos *information_schema* y *mysql* son reservadas del gestor y no deben tocarse directamente a no ser que sepamos muy bien lo que hacemos.

6.3. Borrando bases de datos

Podemos borrar una base de datos con la sentencia *DROP DATABASE*, por ejemplo:

```
mysql> DROP DATABASE mibdd;
Query OK, 1 row affected (0.01 sec)
```

Cuidado: La base de datos se borra definitivamente y no será posible recuperarla a menos que se disponga de una copia de seguridad.

7. Agregando usuarios

Para que alguien se pueda conectar a nuestro servidor de bases de datos, y utilizar una o más tablas o bases de datos, necesitamos crearle un usuario y asignarle los *privilegios* correspondientes.

En el siguiente ejemplo crearemos una base de datos[8], un usuario, y otorgaremos a este usuario permisos sobre la BDD creada.

1. En primer lugar **creamos la BDD**, de nombre *curso_usuario*:

   ```
   mysql> CREATE DATABASE curso_usuario DEFAULT CHARACTER SET utf8
          COLLATE utf8_unicode_ci;

   Query OK, 1 row affected (0.07 sec)
   ```

[8] A partir de ahora BDD.

2. Después **creamos el usuario** que la va a utilizar:

```
mysql> CREATE USER 'usuario' IDENTIFIED BY 'clave';
Query OK, 0 rows affected (0.01 sec)
```

Comprobamos que se ha creado:

```
mysql> select user,host from mysql.user;
+------------------+-------------+
| user             | host        |
+------------------+-------------+
| usuario          | %           |
| root             | 127.0.0.1   |
| root             | ProfeDebian |
| debian-sys-maint | localhost   |
| root             | localhost   |
+------------------+-------------+
5 rows in set (0.00 sec)
```

El % de la columna *host* indica que el usuario puede acceder desde cualquier host. Si hubiésemos querido restringir su acceso a una determinada IP o host deberíamos haber utilizado la forma `'usuario'@'192.168.1.23'` o bien `'usuario'@'nombre.del.host'`.

3. Otorgamos privilegios de uso del gestor:

```
mysql> GRANT USAGE ON *.* TO 'usuario' IDENTIFIED BY 'clave';
Query OK, 0 rows affected (0.00 sec)
```

Nota: Esto no da permisos sobre ninguna base de datos, pero permite la conexión con el gestor y posiblemente el acceso a bases de datos con permisos públicos, si las hubiese.

4. Otorgamos permisos específicos para su base de datos:

```
mysql> GRANT ALL PRIVILEGES ON curso_usuario.* TO 'usuario';
Query OK, 0 rows affected (0.00 sec)
```

En este caso una, pero podrían ser más. Incluso se podría hilar más fino y otorgar privilegios para algunas tablas en algunas bases de datos.

5. Recargamos la página de usuarios y códigos:

```
mysql> FLUSH PRIVILEGES;
Query OK, 0 rows affected (0.00 sec)
```

Si no hacemos esto es posible que el usuario no funcione. La página de usuarios es leída en el arranque y normalmente no se vuelve a leer si no se fuerza mediante la orden anterior.

8. Trabajando con tablas

8.1. Creando una tabla

Para crear una tabla podemos escribir el SQL necesario bien dentro del cliente mysql o bien en un fichero de texto que luego cargaremos a través del cliente. En este ejemplo haremos esto último.

Fichero SQL para crear una BDD y un tabla de ejemplo:

```
CREATE DATABASE mibdd CHARACTER SET utf8 COLLATE utf8_unicode_ci;

CREATE TABLE mibdd.amigos (
id INT NOT NULL AUTO_INCREMENT PRIMARY KEY ,
nombre VARCHAR(50) NOT NULL ,
telefono VARCHAR(15) NOT NULL ,
cp VARCHAR(5) NOT NULL
) ENGINE = MYISAM CHARACTER SET utf8 COLLATE utf8_unicode_ci
COMMENT = 'Tabla de pruebas.';
```

Lo guardamos como prueba.sql y para inyectarlo en el gestor de BDD:

```
#> cat prueba.sql | mysql -u root --password=clave
```

Si todo ha ido bien mysql no dirá nada, pero podemos comprobar que realmente se haya creado bien:

```
mysql> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| curso_usuario      |
| mibdd              |
| mysql              |
+--------------------+
4 rows in set (0.00 sec)
```

La base de datos existe. Veamos que contiene:

```
mysql> use mibdd
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Database changed
mysql> show tables;
+-----------------+
| Tables_in_mibdd |
+-----------------+
| amigos          |
+-----------------+
1 row in set (0.00 sec)
```

```
mysql> show columns from amigos;
+----------+-------------+------+-----+---------+----------------+
| Field    | Type        | Null | Key | Default | Extra          |
+----------+-------------+------+-----+---------+----------------+
| id       | int(11)     | NO   | PRI | NULL    | auto_increment |
| nombre   | varchar(50) | NO   |     | NULL    |                |
| telefono | varchar(15) | NO   |     | NULL    |                |
| cp       | varchar(5)  | NO   |     | NULL    |                |
+----------+-------------+------+-----+---------+----------------+
4 rows in set (0.01 sec)
```

Es correcto. Ya podemos utilizar nuestra tabla para guardar datos.

8.2. Listando las tablas

Para obtener un listado de las tablas disponibles en una BDD utilizaremos la sentencia *SHOW TABLES*. Por ejemplo:

```
mysql> USE mibdd;
Database changed
mysql> SHOW TABLES;
+-----------------+
| Tables_in_mibdd |
+-----------------+
| amigos          |
+-----------------+
1 row in set (0.01 sec)
```

También sería válida la forma:

```
mysql> SHOW TABLES FROM mibdd;
+-----------------+
| Tables_in_mibdd |
+-----------------+
| amigos          |
+-----------------+
1 row in set (0.01 sec)
```

8.3. Borrando una tabla

Para borrar una tabla utilizaremos la sentencia sql *DROP TABLE*:

```
mysql> DROP TABLE amigos;
Query OK, 0 rows affected (0.00 sec)
```

¡Cuidado! No es posible recuperar la tabla si no disponemos de una copia de seguridad.

9. Trabajando con los datos

9.1. Insertando datos

Desde el cliente mysql, aunque igual podría hacerse con un *echo* o con un *cat*:

```
mysql> INSERT INTO mibdd.amigos
    -> (nombre, telefono, cp)
    -> VALUES ('José Pelaez Rodriguez', '976-666666', '50001');
Query OK, 1 row affected (0.00 sec)
```

- **INTO**: Con la cláusula *INTO* le decimos en que BDD y en qué tabla queremos insertar.
- **Columnas**: A continuación y entre paréntesis ponemos los nombres de las columnas para las que queremos dar datos.
- **VALUES**: Para la cláusula *VALUES* pondremos, entre paréntesis y en orden, los datos que queremos insertar en las columnas anteriores.
- Siempre cerraremos las sentencias SQL con un punto y coma.

Obsérvese que no hemos definido valor para la columna ID, ya que esta es *autoincremental*, y si no decimos nada ya coge el siguiente valor del contador interno de la tabla.

9.2. Seleccionando datos

Para ver todos los datos de una tabla podemos hacer una sentencia **SELECT** como esta:

```
mysql> SELECT * FROM mibdd.amigos;
+----+------------------------+------------+-------+
| id | nombre                 | telefono   | cp    |
+----+------------------------+------------+-------+
|  1 | José Pelaez Rodriguez  | 976-666666 | 50001 |
+----+------------------------+------------+-------+
1 row in set (0.00 sec)
```

Podría ser que la tabla tuviese muchos datos y sólo quisiéramos ver los de las personas que se llaman "José":

```
mysql> SELECT * FROM mibdd.amigos WHERE nombre LIKE 'José%';
+----+------------------------+------------+-------+
| id | nombre                 | telefono   | cp    |
+----+------------------------+------------+-------+
|  1 | José Pelaez Rodriguez  | 976-666666 | 50001 |
+----+------------------------+------------+-------+
1 row in set (0.00 sec)
```

O sólo queremos ver los números de teléfono de todos los que vivan en Zaragoza:

```
mysql> SELECT telefono FROM mibdd.amigos WHERE cp LIKE '50%';
+------------+
| telefono   |
+------------+
| 976-666666 |
+------------+
1 row in set (0.00 sec)
```

Las sentencias SELECT y las cláusulas WHERE pueden llegar a ser muy complejas. Se recomienda la referencia del manual de MySQL[9] para obtener información más detallada. Hay que prestar especial atención al apartado "Sintaxis de SELECT"[10].

9.3. Actualizando datos

Podemos modificar el contenido de uno de varios registros de la tabla. Para eso se utiliza la sentencia **UPDATE**:

```
mysql> UPDATE mibdd.amigos SET telefono='976-777777' WHERE id=1;
Query OK, 1 row affected (0.01 sec)
Rows matched: 1  Changed: 1  Warnings: 0
```

Vemos que hemos actualizado el teléfono del registro con ID 1. Ahora lo comprobamos:

```
mysql> SELECT * FROM mibdd.amigos WHERE id=1;
+----+-------------------------+------------+-------+
| id | nombre                  | telefono   | cp    |
+----+-------------------------+------------+-------+
|  1 | José Pelaez Rodriguez   | 976-777777 | 50001 |
+----+-------------------------+------------+-------+
1 row in set (0.00 sec)
```

Efectivamente ha funcionando.

La sentencia *UPDATE* podría hacerse para afectar a muchos registros, o para afectar sólo a los que cumpliesen determinadas condiciones. Esto se consigue gracias a la cláusula *WHERE*, que funciona de la misma forma que en una sentencia *SELECT*.

Para más información dirigirse al apartado "Sintaxis de UPDATE" del manual de Mysql[11].

9.4. Borrando datos

Para borrar datos utilizaremos las sentencias **DELETE**. Por ejemplo:

```
mysql> DELETE FROM mibdd.amigos WHERE id=1;
Query OK, 1 row affected (0.00 sec)
```

[9] Manual de MySQL: http://dev.mysql.com/doc/refman/5.0/es/index.html
[10] Sintaxis de SELECT: http://dev.mysql.com/doc/refman/5.0/es/select.html
[11] Sintaxis de UPDATE: http://dev.mysql.com/doc/refman/5.0/es/update.html

Con esa sentencia habremos borrado el registro con ID 1 de nuestra tabla. Por supuesto con otra cláusula WHERE podríamos haber borrado otros registros mediante otros criterios, por ejemplo si quisiéramos haber borrado a todos los que se llamen José:

```
mysql> DELETE FROM mibdd.amigos WHERE nombre LIKE 'José%';
Query OK, 1 row affected (0.00 sec)
```

También en este caso será la cláusula *WHERE* la que tenga el poder sobre nuestros registros. Hay que tener mucho cuidado al construir estas cláusulas, y antes de poner un programa en producción hay que probarlas sobre BDD de pruebas.

10. Ficheros y copias de seguridad

10.1. Configuración

Toda la configuración se encuentra en `/etc/mysql`:

```
#> la /etc/mysql/
total 24K
49172 drwxr-xr-x  3 root root 4,0K may 11 20:16 ./
48481 drwxr-xr-x 89 root root 4,0K may 15  2009 ../
48856 drwxr-xr-x  2 root root 4,0K may 11 16:00 conf.d/
50275 -rw-------  1 root root  312 may 11 16:00 debian.cnf
50266 -rwxr-xr-x  1 root root 1,2K abr 19 03:12 debian-start*
50312 -rw-r--r--  1 root root 3,9K may 11 20:16 my.cnf
```

El fichero más importante tal vez sea `my.cnf` que es donde están la mayor parte las configuraciones. En él tendremos que cambiar, por ejemplo, las IPs en las que escucha el servidor, tal y como se dice en el punto 5.

En este fichero podemos además escribir un usuario y un password para que se pase siempre al cliente mysql y no nos haga falta ponerlo cada vez (cuidado luego con los permisos del fichero y con quién puede ejecutar el cliente mysql), y podemos realizar muchos ajustes finos del funcionamiento del gestor que se escapan al objeto de esta unidad didáctica. Para más información se recomienda la lectura del manual de MySQL [12].

10.2. Los datos

Todos los datos de MySQL se guardan en `/var/lib/mysql`:

```
#> la /var/lib/mysql
total 21M
2097287 drwxr-xr-x  4 mysql mysql  135 may 15 13:10 ./
2097305 drwxr-xr-x 33 root  root  4,0K may 11 18:28 ../
4315079 drwx------  2 mysql mysql   19 may 12 12:34 curso_usuario/
2097355 -rw-r--r--  1 root  root     0 may 11 16:00 debian-5.0.flag
2097360 -rw-rw----  1 mysql mysql  10M may 15 13:10 ibdata1
```

[12] Manual de MySQL: http://dev.mysql.com/doc/refman/5.0/es

```
2097373 -rw-rw----  1 mysql mysql 5,0M may 15 13:10 ib_logfile0
2097380 -rw-rw----  1 mysql mysql 5,0M may 11 16:01 ib_logfile1
 733377 drwxr-xr-x  2 mysql root  4,0K may 11 16:01 mysql/
2097283 -rw-------  1 root  root     7 may 11 16:01 mysql_upgrade_info
```

Como puede verse, las bases de datos y tablas son ficheros de disco, aunque esto no quiere decir que podamos hacer una copia de seguridad simplemente copiando estos ficheros en caliente. Explicaremos en el siguiente punto (10.3) el procedimiento correcto.

El gestor puede ser reconfigurado para almacenar los datos en otra parte, si esto fuese necesario, y también podríamos montar otro medio en este directorio, por ejemplo una cabina *iscsi* de discos, o una unidad *LVM*, para poder tomar instantáneas, etc...

10.3. Copias de seguridad binarias

Para realizar una copia de seguridad binaria, necesitamos parar el gestor de Bases de Datos. De otra forma no podríamos garantizar la integridad de los datos, ya que muchas operaciones podrían estar realizándose mientras nosotros copiamos los datos y podrían no estar escritas, estarlo a medias, y además perderíamos todos los cambios pendientes de caché o RAM. Un fichero de datos, copiado con el gestor en marcha, podría ser totalmente inservible.

La forma correcta es detener el gestor y después realizar una de estas operaciones:

- Un cp, por ejemplo:

  ```
  #> cp -a /var/lib/mysql /root/backup/mysql-bk
  ```

- Empaquetar con tar, por ejemplo:

  ```
  #> tar czvf /root/backup/mysql-bk.tgz /var/lib/mysql
  ```

- Parar el gestor, generar un snapshot (si es un dispositivo *LVM* o similar) y arrancar de nuevo el gestor. Posteriormente haríamos una copia o un empaquetado de la instantánea resultante, y sólo habríamos parado el gestor durante unos segundos. Consulte la unidad didáctica de *LVM* de este mismo curso para saber como tirar una instantánea de un volumen.

10.4. Copias de seguridad SQL

MySQL nos ofrece una utilidad para realizar volcados SQL de nuestros datos. Se trata de `mysqldump`. Si le pedimos ayuda nos va a dar un montón de opciones, pero el funcionamiento básico está explicado al principio:

```
#> mysqldump --help
mysqldump  Ver 10.11 Distrib 5.0.51a, for debian-linux-gnu (x86_64)
By Igor Romanenko, Monty, Jani & Sinisa
This software comes with ABSOLUTELY NO WARRANTY. This is free software,
and you are welcome to modify and redistribute it under the GPL license
```

```
Dumping definition and data mysql database or table
Usage: mysqldump [OPTIONS] database [tables]
OR     mysqldump [OPTIONS] --databases [OPTIONS] DB1 [DB2 DB3...]
OR     mysqldump [OPTIONS] --all-databases [OPTIONS]
```

Es decir, que podemos ordenar el volcado de una tabla de esta forma:

```
#> mysqldump mibdd amigos
-- MySQL dump 10.11
--
-- Host: localhost    Database: mibdd
-- ------------------------------------------------------
-- Server version       5.0.51a-24+lenny1

/*!40101 SET @OLD_CHARACTER_SET_CLIENT=@@CHARACTER_SET_CLIENT */;
/*!40101 SET @OLD_CHARACTER_SET_RESULTS=@@CHARACTER_SET_RESULTS */;
/*!40101 SET @OLD_COLLATION_CONNECTION=@@COLLATION_CONNECTION */;
/*!40101 SET NAMES utf8 */;
/*!40103 SET @OLD_TIME_ZONE=@@TIME_ZONE */;
/*!40103 SET TIME_ZONE='+00:00' */;
/*!40014 SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0 */;
/*!40014 SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS,
FOREIGN_KEY_CHECKS=0 */;
/*!40101 SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='NO_AUTO_VALUE_ON_ZERO' */;
/*!40111 SET @OLD_SQL_NOTES=@@SQL_NOTES, SQL_NOTES=0 */;

--
-- Table structure for table `amigos`
--

DROP TABLE IF EXISTS `amigos`;
SET @saved_cs_client     = @@character_set_client;
SET character_set_client = utf8;
CREATE TABLE `amigos` (
  `id` int(11) NOT NULL auto_increment,
  `nombre` varchar(50) collate utf8_unicode_ci NOT NULL,
  `telefono` varchar(15) collate utf8_unicode_ci NOT NULL,
  `cp` varchar(5) collate utf8_unicode_ci NOT NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM AUTO_INCREMENT=4 DEFAULT CHARSET=utf8 COLLATE=utf8_unicode_ci
COMMENT='Tabla de pruebas.';
SET character_set_client = @saved_cs_client;

--
-- Dumping data for table `amigos`
--

LOCK TABLES `amigos` WRITE;
/*!40000 ALTER TABLE `amigos` DISABLE KEYS */;
INSERT INTO `amigos` VALUES (1,'José Pelaez Rodriguez',
'976-666666','50001'),(2,'Pepe Perez González',
'333-666666','50002'),(3,'Juan Martin Soria',
'333-664566','50003');
/*!40000 ALTER TABLE `amigos` ENABLE KEYS */;
```

```
UNLOCK TABLES;
/*!40103 SET TIME_ZONE=@OLD_TIME_ZONE */;

/*!40101 SET SQL_MODE=@OLD_SQL_MODE */;
/*!40014 SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS */;
/*!40014 SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS */;
/*!40101 SET CHARACTER_SET_CLIENT=@OLD_CHARACTER_SET_CLIENT */;
/*!40101 SET CHARACTER_SET_RESULTS=@OLD_CHARACTER_SET_RESULTS */;
/*!40101 SET COLLATION_CONNECTION=@OLD_COLLATION_CONNECTION */;
/*!40111 SET SQL_NOTES=@OLD_SQL_NOTES */;

-- Dump completed on 2009-05-15 12:36:40
```

El volcado es lenguaje SQL, simplemente. Si no decimos nada, como en el caso anterior, es volcado por la salida estándar y, ya sabemos como enviarlo a un fichero:

```
#> mysqldump mibdd amigos > mibdd-amigos-bk.sql

#> la mbdd-amigos-bk.sql
88927 -rw-r--r-- 1 root root 2,1K may 15 14:42 mbdd-amigos-bk.sql
```

El lenguaje SQL es texto, así que un compresor producirá resultados sorprendentes en cuanto a ahorro de espacio:

```
#> mysqldump mibdd amigos | gzip > mibdd-amigos-bk.sql.gz

# la mibdd-amigos-bk.sql.gz
88938 -rw-r--r-- 1 root root 849 may 15 14:44 mibdd-amigos-bk.sql.gz
```

En lugar de una sola tabla podemos volcar una BDD entera:

```
#> mysqldump mibdd
```

O todas las BDDs:

```
#> mysqldump --all-databases
```

Si queremos asegurarnos de que los datos no cambian mientras los estamos volcando, podemos utilizar la opción *-l* para bloquear las tablas en lectura. Esto no suele ser necesario y un volcado en *caliente* suele ser más que suficiente.

11. Ejercicios

11.1. Notas

- **Scripts o guiones**: Cree un script o guión de *Bash* para cada uno de los ejercicios, y llámelo `ej-sql-num.sh`, siendo *num* el número de ejercicio[13]. Por ejemplo el script del ejercicio 1 deberá llamarse `ej-sql-1.sh`.
- **Funciones**: También debe crear un fichero llamado `funciones.inc.sh` que contendrá todas las funciones que necesite para sus programas.
- **Modificaciones**: Si se le pide modificar un programa anterior, lo que debe hacer es copiar el programa anterior pero con el nombre correspondiente al ejercicio que esté haciendo, es decir, al acabar los ejercicios debe tener un programa para cada uno de ellos.
- **Ficheros de configuración**: Si necesita crear un fichero de configuración este deberá llamarse `ej-func-num-conf.inc.sh`, siendo *num* el número de ejercicio. Por ejemplo la configuración del ejercicio 1, de necesitarse, deberá llamarse `ej-func-1-conf.inc.sh`.

11.2. Enunciados

1. **Programa de altas**:
   - Programe una función, llamada *creabdd* que sea capaz de crear bases de datos en su servidor mysql. La función deberá admitir el nombre de la BDD como argumento.
   - Programe otra función que sea capaz de dar de alta usuarios en MySQL, otorgando todos los privilegios sobre una determinada BDD. Esta función admitirá dos argumentos, el nombre del nuevo usuario y la BDD sobre la que darle privilegios.
   - La clave de cada usuario debe ser "*sqlusu.nombredeusuario*".
   - Utilizando las dos funciones anteriores, cree un programa que pregunte un nombre usuario, si es que no le le ha pasado como argumento, y que cree una base de datos de nombre "*curso_nombredeusuario*", siendo *nombredeusuario* el introducido como argumento o pregunta, y que cree un usuario (simplemente *nombredeusuario*), con privilegios totales sobre la base de datos anterior.

2. **Alta de usuarios**: Cree un fichero con la lista de nombres de usuario de sus compañeros de clase, llámele *usuarios.txt*. Después escriba un programa que lea dicha lista con un bucle *for* y que llame al programa del ejercicio 1 para crear la BDD y el usuario en MySQL.

[13] En Español, la construcción "siendo num tal cosa" quiere decir que hay que sustituir *num* por lo que se dice a continuación, no que haya que poner literalmente *num*.

3. **Comprobar MySQLs**: Escriba un programa que siguiendo la lista *usuarios.txt* intente conectar a cada base de datos, en los hosts *nombredeusuario-debian*, enviando el comando "*status*" mediante el cliente `mysql`, y con su nombre de usuario y su clave "*sqlusu.nombredeusuario*". La única información que deberá aparecer por pantalla es el nombre del host y al lado un "OK" o un "FALLO" dependiendo de si se logra conectar o no.

4. **Programa de altas y bajas**: Escriba un programa con las siguientes características:

   - El programa creará, si no existe, una base de datos llamada *clientes*, y una tabla llamada *datos* con los campos:

     ```
     id.......: numérico autoincremental
     nombre...: varchar 150
     telefono.: varchar 15
     email....: varchar 100
     cp.......: varchar 5
     ```

   - El programa permitirá mediante un menú de opciones, dar altas y bajas por **id** y por **código postal**.

5. Modifique el programa anterior para que también permita obtener listados ordenados por nombre, tanto por pantalla como por un fichero a elección del usuario.

6. Modifique el programa anterior para que permita obtener listados "*agenda*" en los que sólo aparezca el nombre completo, el teléfono, y el email en el siguiente formato:

   ```
   NOMBRE:                        TELÉFONO:   EMAIL:
   ========================================================
   José Pérez Pelaez              555-555555  pepe@pepe.es
   ```

   El listado deberá tener una línea de cabecera, igual que en el ejemplo, y en la versión de pantalla deberá salir paginado (introduzca datos hasta hacer que sea más de una pantalla). Por supuesto también deberá ofrecerse la opción de obtener el listado "*agenda*" en un fichero.

7. Construya un programa para cron que consulte periódicamente la BDD de clientes y le envíe un mensa con todos los correo electrónicos de dicha base de datos, en el siguiente formato:

```
GRUPO DE EMAIL DE CLIENTES:
---------------------------

email1, email2, email3, email4, email5...
```

8. Consiga que su programa (el del ejercicio 4) funcione contra cualquier servidor de BDD. Para ello cree un fichero de configuración en el que se indique el host, el usuario y la contraseña del gestor de BDD. Su programa deberá funcionar de igual modo contra el servidor local que contra uno remoto.

9. Añada al programa (el del ejercicio 4) la funcionalidad de realizar copias de seguridad. Cuando el usuario seleccione la opción de *backup* el programa generará un volcado sql comprimido con gzip. El usuario podrá elegir el nombre del fichero y la ruta, si bien se le ofrecerá por defecto[14] la siguiente:

   `~/backup/clientes-fechahora.sql.gz`

   Cree el directorio /backup si no existe. En el nombre anterior *fechahora* deberá sustituirse por la fecha y la hora actuales en formato *ddmmaa_hhmmss*.

10. Dote al programa (el del ejercicio 4) de la posibilidad de recuperar una copia de seguridad que se descomprimirá y se inyectará en la base de datos automáticamente. El usuario podrá elegir la ruta y el fichero de respaldo, pero por defecto se le mostrará la más moderna del directorio `~/backup`.

[14] Por defecto: Opción que se escogerá automáticamente en caso de que la respuesta esté en blanco.

11. Programe una tarea *cron* para que diariamente se haga una copia de seguridad binaria y comprimida de todas las bases de datos de su servidor de MySQL.

11.3. Soluciones

Las soluciones a estos ejercicios son complejas y además múltiples, así que los programas pueden adolecer de cierta subjetividad que no tiene porqué concordar con la del lector. Estas soluciones no deben aceptarse como dogma, pero si como guía, y desde luego son más que correctas.

Nótese que las soluciones dependen de la configuración de la BDD y que esta parte habrá de ajustarse correctamente.

Por lo demás, el ejercicio final constituye un magnífico ejemplo de programa de manejo de BDD desde Bash, y podrá fácilmente ser adaptado para cualquier necesidad en ese sentido.

11.3.1. Programa de Altas

Fichero de funciones necesario para el programa del ejercicio 1:

```
# Este fichero se llama:
# ej-sql-1-func.inc.sh

# Configuración:
MYSQLCMD="mysql -h localhost -u root --password=my6667"

function creabdd {
        echo "CREATE DATABASE $1 DEFAULT CHARACTER SET utf8 COLLATE utf8\_unicode\_ci;" \
        | $MYSQLCMD
}

function creauser {
        echo "CREATE USER '$1' IDENTIFIED BY 'sqlusu.$1';" | $MYSQLCMD
        echo "GRANT USAGE ON *.* TO '$1' IDENTIFIED BY 'sqlusu.$1';" | $MYSQLCMD
        echo "GRANT ALL PRIVILEGES ON $2.* TO '$1';" | $MYSQLCMD
        echo "FLUSH PRIVILEGES;" | $MYSQLCMD
}

function tabla {
        echo "CREATE TABLE $1.$2 (id INT NOT NULL AUTO\_INCREMENT PRIMARY KEY, \
              nombre VARCHAR(150) NOT NULL, \
              telefono VARCHAR(15) NOT NULL, \
              email VARCHAR(100) NOT NULL, \
              cp VARCHAR(5) NOT NULL \
              ) ENGINE = MYISAM CHARACTER SET utf8 COLLATE utf8\_unicode\_ci \
              COMMENT = 'Tabla de datos.';" | $MYSQLCMD
}

function database {
        echo "CREATE DATABASE IF NOT EXISTS $1 DEFAULT CHARACTER SET utf8 COLLATE \
        utf8\_unicode\_ci;" | $MYSQLCMD
}

function altas {
        echo "INSERT INTO $1.$2 \
              (nombre, telefono, email, cp) \
              VALUES ('$3', '$4', '$5', '$6');" | $MYSQLCMD
}
```

```
function bajas {
        echo "DELETE FROM $1.$2 WHERE ('$3');" | $MYSQLCMD
}

function lista {
        echo "SELECT nombre,email FROM clientes.datos ORDER BY nombre;" | $MYSQLCMD
}
```

El programa completo del ejercicio 1:

```
#!/bin/bash

# ej-sql-1.sh

# Cargar funciones:
source ej-sql-1-func.inc.sh

if [ -z "$1" ]

then
        echo "Introduzca nombre por favor..."
        read $1

else
        echo -n "Creando BDD curso_$1..."
        creabdd "curso_$1"
        echo "OK"
        echo "Creando usuario $1..."
        creauser "$1" "curso_$1"
        echo "ok"
fi
```

11.3.2. Alta de usuarios

Lista de usuarios para el alta (fichero *usuarios.txt*):

```
slipe
aluna
jlvargas
gdiaz
ebychina
ioanpc
airis
esther
jfuertes
```

Programa para realizar las altas en la BDD de los usuarios de la lista (*ej-sql-2.sh*):

```
#!/bin/bash
```

```
# ej-sql-2.sh

for usuario in $(cat usuarios.txt)

do
        sh "ej-sql-1.sh" $usuario
done
```

11.3.3. Comprobar servidores MySQL

El siguiente programa realiza una comprobación de estado de todos los servidores MySQL de la clase. Se comprueba la conexión y también que el nombre de usuario y contraseña son correctos. El programa hace uso de la biblioteca de funciones del anexo, sección 13.

```
#!/bin/bash

# ej-sql-3.sh

source funciones.inc.sh

clear
titulo "Comprobando servidores de BDD"

for i in $(cat usuarios.txt)
do
        haciendo "Comprobando servidor $i-debian"
        echo "STATUS"|mysql -h $i-debian -u usuario \
        --password=sqlusu.usuario &>/dev/null
        ok $?
done
```

11.3.4. Programa de gestión (Ej. de 4 a 10)

El siguiente programa da respuesta a los ejercicios desde el número 4 hasta el número 10.

Debe tenerse en cuenta que:

- Se hace uso de la biblioteca de funciones del anexo, sección 13.
- Tiene un fichero de configuración: *ej-sql-10-conf.inc.sh*.
- Tiene un fichero de funciones propias: *ej-sql-10-func.inc.sh*.
- Todo debe estar en el mismo directorio para funcionar.
- El usuario deberá configurar el programa para que pueda acceder a su BDD.

Este es el fichero de configuración:

```
# ej-sql-10-conf.inc.sh
# Fichero de configuracion de la aplicación:

# Base de datos:
HOST="mihost-debian"
USER="miusuario"
PASS="mipasswod"
BDD="clientes"
TABLA="datos"
MYSQLOPTS="--host=${HOST} --user=${USER} --password=${PASS}"
MYSQLCMD="mysql ${MYSQLOPTS}"
DUMPCMD="mysqldump ${MYSQLOPTS}"
```

A continuación el fichero de funciones específicas:

```
# ej-sql-10-func.inc.sh
# Funciones específicas del programa 10.

# Alta de cliente.
# alta <nombre> <telefono> <email> <cp>
function alta
{
    sql "Alta de cliente" \
        "INSERT INTO $BDD.$TABLA (nombre, telefono, email, cp) \
VALUES ('$1', '$2', '$3', '$4');" "$MYSQLOPTS"
}

# Borra un cliente por el criterio campo/valor indicado.
# borrauser <campo> <valor>
function borrauser
{
        query "Buscando_usuario" "SELECT * FROM ${BDD}.${TABLA} \
          WHERE ${1}=\"${2}\";" "$MYSQLOPTS"
        sino "¿Desea dar de baja a este cliente?"
        if [ $? -eq 0 ]
        then
            query "Borrando $2" "DELETE FROM ${BDD}.${TABLA} \
  WHERE ${1}=\"${2}\";" "$MYSQLOPTS"
        else
            aviso "Operación de baja abortada."
        fi
}

# Produce un listado.
# listar <a/c> <fichero>
function listar
{
    SEPARADOR="================================================="

    if [ "$1" == "a" ]
    then
        # Tipo agenda:
        COLUMNAS="nombre,telefono,email"
        CABECERA="NOMBRE:\tTELEFONO:\tEMAIL:"
```

```
    else
        # Tipo completo:
        COLUMNAS="id,nombre,telefono,email,cp"
        CABECERA="ID:\tNOMBRE:\tTELEFONO:\tEMAIL:\tCP:"
    fi

    if [ -z "$2" ]
    then
        # Por pantalla:
        echo
        echo -e $CABECERA
        echo $SEPARADOR
        echo "SELECT $COLUMNAS FROM $BDD.$TABLA ORDER BY nombre;" \
      | $MYSQLCMD
        echo
    else
        # A fichero:
        echo -e $CABECERA > $2
        echo $SEPARADOR >> $2
        echo "SELECT $COLUMNAS FROM $BDD.$TABLA ORDER BY nombre;" \
          | $MYSQLCMD >> $2
    fi

    informa "Fin de listado."

    pregunta "Pulse para continuar" NADA
}
```

El programa:

```
#!/bin/bash

# Funciones genéricas:
source funciones.inc.sh
# Funciones de este programa:
source ej-sql-10-func.inc.sh
# Fichero de configuración:
source ej-sql-10-conf.inc.sh

# Comprobar si existe la BDD:
echo "USE $BDD;" | $MYSQLCMD &> /dev/null
if [ $? -ne 0 ]
then
    informa "La BDD $BDD no existe."
    sql "Creando BDD $BDD" \
        "CREATE DATABASE IF NOT EXISTS $BDD CHARACTER SET \
        utf8 COLLATE utf8_unicode_ci;" "$MYSQLOPTS"
    sleep 2
fi

# Comprobar si existe la tabla:
echo "SHOW TABLES FROM $BDD" | $MYSQLCMD | grep $TABLA &> /dev/null
if [ $? -ne 0 ]
then
    informa "La tabla $BDD.$TABLA no existe."
```

```
    sql "Creando tabla $BDD.$TABLA" \
        "CREATE TABLE IF NOT EXISTS $BDD.$TABLA ( \
            id INT NOT NULL AUTO_INCREMENT PRIMARY KEY, \
            nombre VARCHAR(150) NOT NULL, \
            telefono VARCHAR(15) NOT NULL, \
            email VARCHAR (100) NOT NULL, \
            cp VARCHAR(5) NOT NULL \
        ) ENGINE = MYISAM CHARACTER SET utf8 COLLATE utf8_unicode_ci \
        COMMENT = 'tabla de $BDD.';" "$MYSQLOPTS"
    sleep 3
fi

# Menú de opciones:
while [ 1 ]
do
    clear
    titulo "MENÚ DE OPCIONES"
    echo
    echo "  1 - Altas"
    echo "  2 - Bajas"
    echo "  3 - Listados"
    echo "  4 - Backup"
    echo "  5 - Restaurar backup"
    echo
    echo "  0 - Salir"
    echo
    preguntaobg "Opción" opcion "0"
    echo -e "\n\n"

    case $opcion in
    0)
        informa "Que tenga un buen día."
        finalizado 0
    ;;
    1)
        clear
        titulo "Alta de cliente"
        preguntaobg "Nombre" nom
        preguntaobg "Teléfono" tel
        preguntaobg "eMail" mail
        preguntaobg "CP" cp

        alta $nom $tel $mail $cp
    ;;
    2)
        while [ 1 ]
        do
            clear
            titulo "BAJA de cliente"
            echo
            echo "  1 - Baja por ID"
            echo "  2 - Baja por CP"
            echo
            echo "  0 - Salir a menú ppal"
```

```
        echo
        preguntaobg "Opción" OPC

        case $OPC in
        1)
            preguntaobg "ID que quiere dar de baja" ID
            borrauser "id" $ID
            sleep 2
        ;;
        2)
            preguntaobg "CP de los clientes a dar de baja" CP
            borrauser "cp" $CP
            sleep 2
        ;;
        *)
            break
        ;;
        esac
    done
;;
3)
    preguntaobg "¿Listado por pantalla o a un fichero (p/f)?: " option "p"
    fichero=""
    if [ "$option" == "f" ]
    then
        preguntaobg "Nombre del fichero" fichero "listado.txt"
    fi
    preguntaobg "¿Listado completo o agenda (c/a)?" tipo "c"
    listar $tipo $fichero
;;
4)
    clear
    titulo "Copia de seguridad"
    echo

    HORA=$(fechahora)
    preguntaobg "¿Destino?" DESTINO "${HOME}/backup/clientes-${HORA}.sql.gz"
    RUTA=$(dirname $DESTINO)
    haciendo "Creando directorio ${RUTA}"
    mkdir -p $RUTA
    ok $?
    haciendo "Salvando la base de datos ${BDD} en ${DESTINO}"
    $DUMPCMD $BDD | gzip > $DESTINO
    ok $?
    echo
    pregunta "Pulse para continuar" NADA
;;
5)
    clear
    titulo "Restaurar copia de seguridad"
    echo
    ARCH=$HOME/backup/$(ls $HOME/backup -t|head -1)
    preguntaobg "¿Archivo a recuperar?" ARCH $ARCH
    haciendo "Recuperando copia de seguridad"
```

```
        zcat $ARCH | $MYSQLCMD $BDD
        ok $?
        echo
        pregunta "Pulse para continuar" NADA
    ;;
    *)
        echo "La opción es incorrecta, inténtelo de nuevo."
    ;;
    esac

    informa "Regresando al menú ppal."
    sleep 1
done
```

11.3.5. Copia binaria

Solución al ejercicio 11 para realizar una copia binaria desde cron:

```
#!/bin/bash

# Configuración:
DATOS="/var/lib/mysql"
DESTINO="/root/backup"

# Copia binaria de todas las BDD.

# Primero detenemos el gestor:
/etc/init.d/mysql stop

# Realizamos la copia de seguridad:
tar cjf $DESTINO/mysql-datos-bin.tbz $DATOS

# Arrancamos de nuevo el gestor:
/etc/init.d/mysql start
```

12. Sobre esta unidad didáctica

12.1. Notas y advertencias

Debian: Esta guía está basada en el sistema *Debian GNU/Linux*, podría haber pequeños cambios si se aplica a otras distribuciones de *GNU*, pero en su mayor parte funcionará bien con la excepción de lo referido al sistema de paquetería de programas, los comandos que empiezan por *apt*, ya que otras *distros* no basadas en *Debian* podrían incorporar sistemas diferentes para el manejo de sus paquetes.

12.2. Derechos

Esta guía se cede bajo contrato Coloriuris. Sólo puede ser utilizada previa aceptación del contrato de cesión sito en:

- http://www.coloriuris.net/contratos/ef5af6aaa441ab9c213273fade56dca1

Dicho contrato garantiza que estoy cediendo los derechos de uso y modificación sin ánimo de lucro.

12.3. Agradecimientos

El autor quiere reflejar su agradecimiento a todas las páginas de Internet que ponen a disposición de todo el mundo sus contenidos, así como a todo aquél que publica artículos, manuales y experiencias en Internet, ya que eso favorece a la difusión del conocimiento y al desarrollo humano. *La información quiere ser libre.*

Un agradecimiento muy especial a toda la comunidad del Software Libre. Sin ellos el autor viviría en la oscuridad: Programadores, traductores, asociaciones, hacktivistas, webmasters, etc...

También quiero agradecer muy especialmente su ayuda a mis alumnos y lectores, por tomarse la molestia de comunicarme las erratas y por darme ideas para mejorar los ejercicios.

12.4. Revisiones

El autor irá eventualmente publicando revisiones de esta unidad en su página personal, y estará encantado de recibir sugerencias y dudas en la misma o en su email:

- http://jorgefuertes.com.
- cursos@jorgefuertes.com.

Por supuesto se puede contactar con el autor para contratarle para hacer nuevas unidades, adaptaciones, modificaciones, cursos, etc...

13. Anexo: Una biblioteca de funciones de ejemplo

13.1. Introducción

La siguiente biblioteca es una colección de funciones que el autor utiliza habitualmente para sus propios programa de Bash. Su utilidad para el lector puede ser evidente o no, ya que dependiendo de la naturaleza de nuestro trabajo necesitaremos unos u otros apoyos en nuestro desarrollo, sin embargo, las funciones aquí presentadas son bastante generales y contemplan algunos *trucos* para mejorar y facilitar la interacción con el usuario y la entrada/salida.

El alumno debería construir un programa que utilice todas las funciones al menos una vez, con la salvedad de las relacionadas con base de datos, que sólo funcionarán de estar instalado **MySQL**.

En cuanto a su uso fuera del propio de estos apuntes que ya tienen su propia licencia, podrá considerarse que la biblioteca es GPL2 a todos los efectos. El autor agradecerá correcciones y mejoras a la misma.

13.2. La biblioteca de uso general

```
# Biblioteca de funciones de uso general.
#
# Copyright (C) 2007 Jorge Fuertes (queru@queru.org)
#
# Este programa es software libre: usted puede redistribuirlo y/o modificarlo
# bajo los términos de la Licencia Pública General GNU publicada
# por la Fundación para el Software Libre, ya sea la versión 3
# de la Licencia, o (a su elección) cualquier versión posterior.
#
# Este programa se distribuye con la esperanza de que sea útil, pero
# SIN GARANTÍA ALGUNA; ni siquiera la garantía implícita
# MERCANTIL o de APTITUD PARA UN PROPÓSITO DETERMINADO.
# Consulte los detalles de la Licencia Pública General GNU para obtener
# una información más detallada.
#
# Debería haber recibido una copia de la Licencia Pública General GNU
# junto a este programa.
# En caso contrario, consulte <http://www.gnu.org/licenses/>.

# Configuración de colores:
NORMAL="\e[0m"
BOLD="\e[1m"
INVERSO="$BOLD\e[30;47m"
ROJO="$BOLD\e[31m"
VERDE="$BOLD\e[32m"
MARRON="$BOLD\e[33m"
AZUL="$BOLD\e[34m"
MAGENTA="$BOLD\e[35m"
CYAN="$BOLD\e[36m"
BLANCO="$BOLD\e[37m"
```

```
FORTUNECOLOR="$MARRON"
AMARILLO="$MARRON"

# titulo <texto>
# Escribe un título en pantalla, para que todos los programas
# tengan un aspecto común.
function titulo
{
    echo -e "\n${BLANCO}---=[${CYAN}${1}${BLANCO}]=---${NORMAL}"
}

# ok <errorlevel>
# Para usar después de un 'haciendo', cierra la línea con OK o FALLO
# dependiendo del errorlevel pasado. Normalmente 'ok $?'.
function ok
{
    if [ $1 -eq 0 ]
    then
        echo -e "${VERDE}OK${NORMAL}"
    else
        echo -e "${ROJO}FALLO${NORMAL} (Cod.${1})"
    fi
}

# pregunta <texto> <var_sin_dolar> [por defecto]
# Hace una pregunta al usuario, poniéndo el resultado en la variable
# del segundo argumento y poniéndo el tercer argumento como respuesta
# si el usuario responde en blanco.
function pregunta
{
    RESPUESTA=""
    echo -e "${VERDE}>${BLANCO}${1}${NORMAL} (${3}): \c"
    read RESPUESTA
    if [ -z "$RESPUESTA" ]
    then
        RESPUESTA=$3
    fi
    eval "$2=\"$RESPUESTA\""
}

# preguntaobg <texto> <var_sin_dolar> [por defecto]
# Igual que la anterior, pero una respuesta es obligatoria
# si no se pasa valor por defecto.
function preguntaobg
{
    RESPUESTA=""
    while [ -z "$RESPUESTA" ]
    do
        echo -e "${VERDE}>${BLANCO}${1}${NORMAL} (${3})(*): \c"
        read RESPUESTA
```

```
        if [ -z "$RESPUESTA" ]
        then
            RESPUESTA=$3
        fi
    done
    eval "$2=\"$RESPUESTA\""
}

# haciendo <texto>
# Para iniciar una acción informando al usuario.
# Al terminar dicha acción se deberá usar 'ok $?'.
function haciendo
{
    echo -e "  ${AMARILLO}- ${BLANCO}${1}${NORMAL}...\c"
}

# informa <texto>
# Da una información al usuario.
function informa
{
    echo -e "${AMARILLO}+${NORMAL} ${1}${NORMAL}"
}

# finalizado <errorlevel>
# Finaliza el programa saliendo con el errorlevel que se le diga.
function finalizado
{
    echo -e "\n*** ${BLANCO}Finalizado${NORMAL} ***"
    exit $1
}

# query "<texto>" "<sql>" "[opciones mysql]"
# Lanza una consulta a MySQL y muestra el resultado.
function query
{
    #informa "Query: ${2}"
    haciendo $1
    RES=$(echo "${2}" | mysql $3 |tr "\n" "|")
    ok $?
    if [ -z "$RES" ]
    then
        informa "Sin resultado."
        return 1
    else
        informa "Resultado:"
        echo $RES|tr "|" "\n"
    fi
}

# sql <texto> <sql> [opciones mysql]
```

```
# Envía SQL sin esperar respuesta:
function sql
{
    #echo "echo \"${2}\" | mysql $3 |tr \"\n\" \"|\""
    haciendo $1
    echo $2 | mysql $3
    ok $?
}

# sino <texto>
# Hace una pregunta al usuario pero sólo le permite
# responder 's' o 'n'. Devuelve el estado 0 o 1.
function sino
{
    echo -e "${VERDE}>${BLANCO}${1}${NORMAL} (s/N): \c"
    read -n1 RESPUESTA
    echo
    if [[ "$RESPUESTA" == "s" || "$RESPUESTA" == "S" ]]
    then
        return 0
    else
        return 1
    fi
}

# errorgrave <texto>
# Muestra un error grave y sale del programa.
function errorgrave
{
    echo -e "\n${ROJO}> ERROR${NORMAL}: ${1}\n"
    exit 1
}

# aviso <texto>
# Muestra un aviso por pantalla.
function aviso
{
    echo -e "\n${AMARILLO}> ${ROJO}AVISO${NORMAL}: ${1}\n"
}

# Muestra la fecha en formato ddmmaa_hhmmss
# Ej.: HORA=$(fechahora)
function fechahora
{
     echo "$(date +'%d-%m-%y_%H:%M:%S')"
}
```