

Capítulo VI: Gestión de permisos, usuarios y grupos en Linux:

ÍNDICE DE CONTENIDOS:

- 6.1 Política de permisos en Linux:
 - 6.1.1 Interpretando rwx
 - 6.1.2 Los permisos en decimal
 - 6.1.3 Cambiando permisos con chmod
- 6.2 Gestión de Usuarios:
 - 6.2.1 El archivo /etc/passwd
 - 6.2.2 Añadiendo usuarios
 - 6.2.3 Modificando usuarios
 - 6.2.4 Eliminando usuarios
 - 6.2.5 Añadiendo nuevos grupos
 - 6.2.6 Modificando grupos
 - 6.2.7 Eliminando grupos
- 6.3 Cómo cambiar de propietario y grupo a nuestros ficheros
- 6.4 Otros comandos útiles

Esta obra está protegida por la **Licencia Creative Commons**, bajo las condiciones de: **Reconocimiento - No comercial - Compartir igual**: El material creado por un artista puede ser distribuido, copiado y exhibido por terceros si se muestra en los créditos. No se puede obtener ningún beneficio comercial y las obras derivadas tienen que estar bajo los mismos términos de licencia que el trabajo original.



Reconocimiento (Attribution): El material creado por un artista puede ser distribuido, copiado y exhibido por terceras personas si se muestra en los créditos.



No Comercial (Non commercial): El material original y los trabajos derivados pueden ser distribuidos, copiados y exhibidos mientras su uso no sea comercial.



Compartir Igual (Share alike): El material creado por un artista puede ser modificado y distribuido pero bajo la misma licencia que el material original.

Versión: 1.0

Autor: Alberto Reynolds Moreno.

alberto.reynolds@gmail.com

Revisión: Isabel Rueda Rodríguez

rueda.isabel@gmail.com

Teniendo presente todo lo anteriormente visto, puede ser buen momento para comenzar con la primera tarea de administración, la de los usuarios del sistema.

Recordamos que ya hemos explicado que en Linux existen dos tipos o perfiles de usuarios: el administrador, o 'root', y los usuarios finales.

Para cualquier tarea de administración que tratemos a partir de ahora, será necesario estar **logueado como usuario root**.

La administración de usuarios en Linux es la referida a la creación y gestión de las cuentas de usuarios, grupos de usuarios, establecimiento de permisos y relaciones entre ambos.

Este tipo de administración la llevaremos a cabo cuando queramos establecer políticas de seguridad en un equipo o en una LAN, o para gestionar servidores del tipo NFS, FTP o WEB.

En este apartado vamos a diferenciar dos tareas básicas: política de permisos y gestión de usuarios.



6.1 Política de permisos en Linux:

Linux es un sistema multiusuario, por lo que necesita de una política de permisos segura y planificada para mantener el sistema seguro.

Un administrador de sistemas Linux debe prestar mucha atención y planificar dicha política de permisos para mantener su sistema seguro.

Podemos hacernos una primera idea de cómo se gestionan los permisos en Linux, haciendo un 'ls -l' desde nuestro intérprete de comandos:

```
[albertux@debian]$ ls -l
total 284
drwxr-xr-x  5 alberto alberto 4096 2007-11-26 17:38 2006r3
drwxr-xr-x  5 root    root    4096 2007-09-17 15:48 AlberTUX_LIVE
drwxr-xr-x  3 alberto alberto 4096 2007-04-02 11:38 Beryl
drwxr-xr-x  2 alberto alberto 4096 2007-12-14 15:05 bin
-rw-r--r--  1 alberto alberto 16548 2008-04-03 15:07 CELIA-DSL.odt
drwxr-xr-x  1 alberto alberto 4096 2008-03-27 14:03 COMOS
drwx----- 3 alberto alberto 4096 2008-04-07 15:02 curso-ASL
-rw-r--r--  1 alberto alberto 9490 2008-04-07 13:44 Curso-ASOL.odt
```

Como vemos, cada línea tiene un formato del estilo:

```
{T} {rwx} {rwx} {rwx} {N} {usuario} {grupo} {tamaño} {fecha de creación}{nombre}
1    2    3    4    5    6    7    8    9    10
```

1- Nos indica que tipo de archivo es:

- Si es un fichero normal
- d Si es un directorio
- c Especial de modo carácter (Dispositivo tty, impresora...)
- p Pipe
- l Enlace simbólico

2- Nos indica los permisos que tiene el propietario del archivo.

3- Nos indica los permisos que tiene el grupo al que pertenece el archivo.

4- Nos indica los permisos del resto de usuarios. (Otros)

5- Es el número de archivos/directorios que contiene. Si es un fichero aparecerá 1, si se trata de un directorio aparecerá como mínimo 2 (los directorios '.' y '..' más los que contenga).

6- Indica el nombre del usuario al que pertenece el archivo o directorio.

7- Indica el nombre del grupo al que pertenece el archivo o directorio.

8- Indica el tamaño.

9- Indica la fecha de creación.

10- Indica su nombre.

6.1.1 Interpretando rwx:

Los grupos de permisos se agrupan de 3 en 3: rwx, donde (r) significa permiso de lectura, (w) permiso de escritura y (x) permisos de ejecución.

Por lo tanto un fichero con esta apariencia:

```
rwX r-X ---
```

significa que tendrá permisos de lectura, escritura y ejecución para el propietario, permisos de lectura y ejecución para el grupo, y ningún permiso para el resto.

PERMISOS	SIGNIFICADO
rwX rwX rwX	Permiso para todos (Muy poco seguro para archivos sin importancia)
rwX r-- ---	Todos los permisos para el propietario y solo de lectura para el grupo.
r-X --- ---	Sólo lectura y ejecución para el propietario.

Si nos fijamos en el directorio donde comentamos que estaban los binarios (ejecutables) comunes para todos los usuarios del sistema, directorio **/bin**, y pedimos un listado con 'ls -l' observaremos que todos los binarios tienen una sintaxis como esta:

```
-rwxr-xr-x 1 root root 3518 2006-09-19 14:38 znew
```

Todos los binarios son propiedad del root, y el resto de usuarios (nosotros) únicamente podemos leer su contenido y ejecutarlo, algo bastante lógico y seguro, ¿no? Sería poco serio si cualquier usuario pudiese modificar la funcionalidad de este comando.

Que pasaría si a un archivo ejecutable del directorio /bin le diésemos los siguientes permisos:

```
-rwxr-x--x 1 root root 3518 2006-09-19 14:38 znew
```

Un usuario cualquiera (excepto root) teóricamente debería poder ejecutarlo, sin embargo, necesita leerlo para interpretarlo (cosa que tiene prohibida), por lo tanto no se ejecutaría.

¿Qué pasaría en este otro caso?

```
-rwxr-xr-- 1 root root 3518 2006-09-19 14:38 znew
```

Como usuario normal podríamos leer su contenido, pero tampoco podríamos ejecutarlo, ya que carece de permisos de ejecución.

6.1.2 Los permisos en decimal:

Como ya hemos comentado, la operación de administrar una política de seguridad en Linux es una tarea fundamental para un administrador de sistemas.

La única persona que puede cambiar los permisos de un archivo o directorio es su propietario. No hace falta decir, que el root podrá cambiar los permisos de cualquier otro usuario del sistema. Es triste, pero ... siempre ha habido clases y clases... en este Sistema Operativo si no eres root, no eres nadie. ;)

Para entender bien como gestionar los permisos, nos interesa recordar cómo es sistema binario. En un sistema BINARIO sólo pueden haber dos valores para cada dígito: ya sea un 0=DESACTIVADO ó un 1=ACTIVADO.

Para representar el número 22 en notación BINARIA lo haríamos como **00010110**, notación que se explica según la siguiente tabla:

Posición del BIT:	7	6	5	4	3	2	1	0
Valor Binario:	0	0	0	1	0	1	1	0
Valor Decimal:	128	64	32	16	8	4	2	1
Valores a Sumar:	0	0	0	16	0	4	2	0
Valor Resultante:	16 + 4 + 2=22							

Para calcular el valor de un permiso nos basaremos en la sumas de sus valores decimales según esta correspondencia:

r	w	x	Permiso
4	2	1	Valor Decimal

Por lo tanto los posibles valores para unpermiso serán:

Permisos	Valor
rwx	7

rw-	6
r-x	5
r--	4
-wx	3
-w-	2
--r	1
---	0

Lo anterior, aplicado a un fichero que agrupa los permisos en 3 grupos (propietario, grupo y otros), resultaría así.

```

    rw-  r-x  ---
     7   5   0
    
```

Por lo tanto,

PERMISO	VALOR
rw- rw- rw-	777
rw- r-x r--	754
r-x r-- --	540

Teniendo claro esto, ya podemos administrar permisos de forma sencilla utilizando el comando **chmod**.

6.1.3 Cambiando permisos con chmod:

Lo más habitual es utilizar el comando chmod para administrar permisos, del siguiente modo:

[chmod] [modo] [permisos] [fichero/s]

Como modo sólo veremos el modo '-R', para cambiar los permisos de un modo recursivo dentro de los directorios.

```

[albertux@debian]$ chmod -R 755 mi_directorio
[albertux@debian]$ ls -l
[albertux@debian]$
drwxr-xr-x 2 albertux albertux 4096 2007-07-13 13:57 mi_directorio
    
```

Otra manera de añadir o quitar permisos, será utilizando estos modos:

```

a Indica que se aplicará a todos.(all)
u Indica que se aplicará al usuario.(user)
g Indica que se aplicará al grupo.(group)
o Indica que se aplicará a otros.(other)
+ Indica que se añade el permiso.
- Indica que se quita el permiso.
r Indica permiso de lectura.
w Indica permiso de escritura.
x Indica permiso de ejecución.
    
```

La manera de aplicar este nuevo método será:

A quién se aplica +/- - Qué permisos aplica

Aplicando lo anterior, resultarían estas posibles combinaciones:

```
a+r Permisos de lectura para todos.
+r Igual que antes, si no se indica nada se supone 'a'.
og-x Quita permiso de ejecución a todos menos al usuario.
u+rwx Da todos los permisos al usuario.
o-rwx Quita los permisos a los otros.
```

Aplicaremos '-R' (modo recursivo) cuando se trate de directorios:

```
[albertux@debian]$ chmod -R o-rx mi_directorio
[albertux@debian]$ ls -l
[albertux@debian]$
drwxr-x--- 2 albertux albertux 4096 2007-07-13 13:57 mi_directorio
```

Con estos conceptos básicos ya podremos administrar los permisos de los usuarios de nuestro sistemas. El comando 'chmod' tiene multiples opciones, recordad la existencia del comando 'man' para conocer más:

```
[albertux@debian]$ man chmod
```

6.2 Gestión de Usuarios:

Una vez comprendida cómo cambiar los permisos a los usuarios del sistema, es necesario complementarlo con la gestión de dichos usuarios.

Un administrador de sistemas necesitará crear, modificar, eliminar y gestionar usuarios y grupos de usuarios a nivel global.

En Linux existe un archivo editable que contiene TODA la información acerca de los usuarios creados en nuestro sistema. Lógicamente, su configuración se encontrará en el directorio de configuraciones: /etc

El archivo en cuestión se llama passwd y se localiza en **/etc/passwd**.

6.2.1 El archivo /etc/passwd:

Gran parte de la seguridad de un sistema Linux, viene por el buen uso del archivo /etc/passwd, ya que es objetivo primario para muchos ataques de crackers. (*crackers != hackers*).

En este archivo, existe una línea que contiene toda la información propia por cada usuario existente en la máquina, con una sintaxis parecida a esta:

nombre:clave_encriptada:UID:GID:GECOS:directorio_inicial:intérprete

Ej.: alberto:x:1000:1000:Alberto Reynolds,,,:/home/alberto:/bin/bash

Cada campo está separado por ':'. El campo '**nombre**' indica el alias con el que se logea el usuario en el sistema. Seguidamente está la **clave encriptada** aunque lo que todos encontraréis seguramente es una 'x'.

Después nos encontramos con el **UID** y **GID**, el primero es el identificador de usuario para el sistema y el segundo el identificador de grupo primario al que el usuario pertenece, puesto que un usuario puede pertenecer a muchos grupos.

GECOS es un campo opcional para guardar información. Normalmente contiene el nombre completo del usuario. GECOS significa General Electric Comprehensive Operating System. En este campo podemos poner lo que queramos ya que el sistema no lo usa para ninguna gestión. Se puede omitir sin poner entre los ':' nada, quedando '::'.

Posteriormente aparece el **directorio inicial** es donde empezará el usuario una vez logueado, y seguidamente el **interprete de comandos** usará (bash, sh, tcsh...).

Si en el campo intérprete encontramos '/sbin/nologin' es un usuario que no puede loguearse, esto tiene sentido ya que el propio sistema tiene usuarios con privilegios para dar servicios al sistema como pueden ser 'bin', 'daemon', 'adm'...

El usuario root siempre tiene como UID y como GID un '0', y cualquier usuario que lo tenga tendrá los mismos privilegios que él. Hay que recordar que un usuario sólo puede tener un UID, pero varios usuarios pueden tener el mismo UID.

PELIGRO: Algo totalmente desaconsejable, debido a que puede provocar agujeros de seguridad importantes, es tener algún usuario con UID=0 distinto al root. Si esto ocurre, lo más seguro es que hayas sido atacado por algún, ¿hacker?, bastante cutre.

Normalmente el sistema guarda los UID bajos (por ejemplo por debajo de 500 o por debajo de 1000) para los usuarios del propio sistema, los que necesita para los servicios que ofrece (por ejemplo, si tienes un servidor de ftp lo normal es tener un usuario en el sistema que se llame 'ftp'), y los UID altos se utilizan para los usuarios normales.

PELIGRO: Si donde debiera aparecer la clave aparece ':::', esto indica que no hay clave, y por supuesto, no es nada bueno. Si aparece una 'x' es que las claves o la clave de ese usuario están gestionadas en el archivo **/etc/shadow** ya que la mayoría de los sistemas usan "claves en sombra".

Esto es debido a que **/etc/passwd** debe ser visible a todos, porque si no ciertos comandos como puede ser 'ls' dejarían de funcionar, y aunque la clave está encriptada no es bueno que se encuentre a la vista por la existencia de programas de fuerza bruta que se dedican a descifrarlas (el famoso '*john deriper*' por ejemplo), y cualquier usuario con acceso a nuestro sistema tendría las claves usando dichos programas.

En vez de esto se usa el fichero **/etc/shadow**, que sólo es visible por root.

No es aconsejable 'jugar' con esta información editando y retocando directamente dichos archivos. Para ello existen distintas herramientas que nos ayudarán a gestionar los usuarios y grupos.

6.2.2 Añadiendo usuarios:

Utilizaremos el comando **adduser** o **useradd**, desde nuestro interprete de comandos y siendo administrador, root, para añadir usuarios.

Normalmente bastará con la orden **adduser nombre_usuario** para crear nuevos usuarios, aunque su sintaxis completa será:

addusr [-c comentario] [-d home] [-e fecha] [-f dias] [-g grupo] [-G lista de grupos] [-m [-k template] | -M] [-n] [-o] [-p passwd] [-r][-s shell] [-u uid] usuario

Podremos 'ahorrarnos' muchas de las opciones, ya que, si no indicamos nada el 'home' del usuario se creará por defecto en '/home/nombre_del_usuario', con la estructura del '/etc/skel'.

Este, será un usuario indefinido y no se bloqueará por no usarse, tendrá la shell por defecto (normalmente bash), se le asignará un uid automáticamente y se creará un grupo con el mismo nombre.

Normalmente usaremos la forma sencilla:

```
[debian:/home/alberto]# adduser pepito
Adding user `pepito' ...
Adding new group `pepito' (1001) ...
Adding new user `pepito' (1001) with group `pepito' ...
Creating home directory `/home/pepito' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
```

Si preferimos personalizar o forzar algún campo, utilizaremos la sintaxis completa:

addusr [-c comentario] [-d home] [-e fecha] [-f dias] [-g grupo] [-G lista de grupos] [-m [-k template] | -M] [-n] [-o] [-p passwd] [-r][-s shell] [-u uid] usuario

Donde:

- [-c comentario]:** Pondremos el comentario que queremos en el campo GECOS.
- [-d home]:** Directorio home (el de inicio) de la cuenta.
- [-e fecha]:** fecha en formato año-mes-día en que la cuenta caduca y se bloquea.
- [-f dias]:** Número de días en los que la cuenta se bloqueará si no se usa.
- [-g grupo]:** Nombre del grupo primario. Ojo el grupo debe ya existir.
- [-G lista de grupos]:** Listado de grupos al que el usuario pertenecerá.
- [-m [-k template] | -M] -m:** Crea el directorio home de la cuenta si es que no existe ya. Con -k usa el directorio template para crear el directorio home o sea que copia lo de template al home, en caso contrario se copia / etc/skel. Si en vez de -m ponemos -M el directorio no es creado.
- [-n]:** Añade un grupo con el mismo nombre que el usuario
- [-o]:** Permite crear usuarios con uid duplicada.
- [-p passwd]:** Añade el password al usuario.
- [-r]:** Se usa para crear un usuario del sistema.
- [-s shell]:** Indica que shell
- [-u uid]:** Indicamos que uid queremos.

Para saber más...

```
[albertux@debian]$ man adduser
```

NOTA: Podemos modificar la estructura del /home del usuario, modificando el /etc/skel del sistema.

6.2.3 Modificando usuarios:

Como comentamos anteriormente, existen herramientas para evitar modificar el archivo

/etc/passwd 'a mano'. El comando **usermod** nos permitirá hacer estos cambios de estampanera:

usermod [-c comentario] [-d home] [-e fecha] [-f dias] [-g grupo] [-G lista de grupos] [-m] [-n] [-p passwd] [-s shell] [-u uid [-o]] usuario [-L | -U] usuario

El significado de los parámetros es el mismo que en el anterior comando, así que no necesitan explicación, salvo dos nuevos:

```
-L Bloquea la cuenta de usuario.
-U La desbloquea.
```

Cambiaremos por ejemplo una contraseña así:

```
[debian:/home/alberto]# usermod -p nueva_clave pepito
```

Y bloquearemos la cuenta de 'pepito' de estemodo:

```
[debian:/home/alberto]# usermod -L pepito
```

Para saber más...

```
[alberTUX@debian]$ oman usermod
```

NOTA: Un usuario puede cambiarse su propia contraseña utilizando la orden **passwd**. No hace falta molestar al administrador del sistema para esto.

```
[alberTUX@debian]$ passwd
Changing password for alberto
(current) UNIX password:
```

6.2.4 Eliminando usuarios:

Llegó el momento de eliminar usuarios que ya no pintan nada en nuestro sistema. Ya es sabido por todos que es mucho más fácil destruir que crear. En este caso bastará con aplicar el comando **deluser** o **userdel**.

Aquí tenéis un ejemplo:

```
[debian:/home/alberto]# deluser -R pepito
```

Con la opción '-R' eliminará el directorio home del usuario; sin esta opción se limitará a eliminar únicamente la cuenta de usuario, dejando el home intacto.

Para saber más...

```
[alberTUX@debian]$ man deluser
```

6.2.5 Añadiendo nuevos grupos:

Ahora toca administrar los grupos de usuarios. Cada usuario pertenece como mínimo a un

grupo, (su grupo principal de usuario), aunque podrá pertenecer a varios grupos más.

Existen ciertos grupos del sistema, tales como grupo root, grupo bin... y por lo general, salvo raras excepciones, los usuarios **nunca** deben pertenecer a ninguno de esos grupos.

Para los grupos existe igualmente un archivo de configuración **/etc/group**, al que por supuesto debemos proteger y cuidar como hacemos con **/etc/passwd**.

El fichero de shadow para los grupos es **'/etc/gshadow'**, e igual que en los usuarios sólo es accesible por el root.

El formato de **'/etc/group'** es el siguiente:

nombre_grupo:clave:GID:lista de usuarios miembros

Aunque en el campo clave aparezca una 'x' al igual que pasaba con **/etc/passwd**, no quiere decir que cada grupo tenga una clave. Si miramos el archivo **/etc/gshadow** veremos que casi todos los campos están vacíos, es decir, no existe ninguna clave.

Para añadir un grupo utilizaremos el comando **addgroup** o **groupadd**. Al igual que anteriormente, existirá el modo 'rápido' y el modo 'personalizado'.

Modo rápido:

```
[debian:/home/alberto]# addgroup migrupe
Adding group `migrupe' (GID 1002) ...
Hecho.
```

Modo personalizado: **groupadd [-g gid [-o]] [-r] [-f] [nombre del grupo]**

Donde:

```
-g indica explícitamente el GID del grupo.
-o no obliga a que el identificador de grupo sea único, cosa totalmente desaconsejable.
-r para crear un grupo del sistema.
-f hace que groupadd no de error si el grupo ya existe.
```

Para saber más...

```
[alberTUX@debian]$man groupadd
```

6.2.6 Modificando grupos:

Llegará el momento de que surja la necesidad de modificar algún grupo existente. Utilizaremos para ello el comando **modgroup**:

groupmod [-g gid [-o]] [-n group_name] [nombre-del-grupo]

Vamos a cambiar el nombre del grupo 'tarde' a grupo 'noche':

```
[debian:/home/alberto]# groupmod -n tarde noche
groupadd noche
```

Para saber más...

```
[alberTUX@debian]$ man groupmod
```

6.2.7 Eliminando grupos:

Por último, vamos a ver como se eliminan los grupos de usuarios que ya no necesitamos. Utilizaremos para ello el comando **groupdel** o **delgroup**:

```
[debian:/home/alberto]# groupdel noche
```

Para saber más...

```
[alberTUX@debian]$ man groupdel
```

6.3 Cómo cambiar de propietario y grupo a nuestros ficheros:

Queda ahora conocer como hacer propietario de un archivo o directorio a un usuario, o hacer que este fichero sea propiedad de un grupo.

Para ello existen los comandos **chown** y **chgrp**.

El primero cambia de propietario a un fichero existente de esta forma:

[chown] [usuario] [archivo]

Si por ejemplo tenemos un fichero llamado 'prueba.sh' propiedad de 'alberto':

```
[debian:/home/alberto]# ls -l |grep prueba.sh
-rw-r--r-- 1 alberto alberto    2 2008-04-10 15:41 prueba.sh
```

y queremos que su propietario pase a ser 'root', haremos:

```
[debian:/home/alberto]# chown root prueba.sh
```

Vemos el resultado...

```
[debian:/home/alberto]# ls -l |grep prueba.sh
-rw-r--r-- 1 root alberto    2 2008-04-10 15:41 prueba.sh
```

Si queremos aplicar esto a directorios, simplemente añadiremos la opción **-R** para que lo haga de modo recursivo:

[chown] -R [usuario] [directorio]

Para saber más...

```
[alberTUX@debian]$ man chown
```

Para cambiar de grupo a un fichero utilizaremos el comando **chgrp** de este modo:

[chgrp] [grupo] [archivo]

Si por ejemplo tenemos un fichero llamado 'prueba.sh' perteneciente al grupo 'alberto':

```
[debian:/home/alberto]# ls -l |grep prueba.sh
-rw-r--r-- 1 root alberto  2 2008-04-10 15:41 prueba.sh
```

y queremos que su grupo principal sea 'root', haremos:

```
[debian:/home/alberto]# chgrp root prueba.sh
```

Vemos el resultado...

```
[debian:/home/alberto]# ls -l |grep prueba.sh
-rw-r--r-- 1 root root  2 2008-04-10 15:41 prueba.sh
```

Al igual que antes, añadiremos la opción '-R' para que lo haga de modo recursivo:

[chgrp] -R [grupo] [directorio]

Para saber más...

```
[alberTUX@debian]$ man chgrp
```

6.4 Otros comandos útiles:

Otros comando útiles para unadministrador relativos a la gestión de usuarios pueden ser los comandos **users** y **groups**.

Users nos indica qué usuarios se encuentran validados en el sistema. Muy útil cuando trabajamos en una máquina compartida o un servidor,

Groups nos indica los grupos a los que pertenece un usuario:

```
[debian:/home/alberto]# groups root
root : root
```

Para saber más...

```
[alberTUX@debian]$ man users
[alberTUX@debian]$ man groups
```