Capítulo 1: Administración de Permisos y Usuarios

La administración de cuentas de usuarios y grupos en Linux es una tarea fundamental de cualquier administrador de sistemas. En este capítulo mostraremos cómo crear, modificar, y eliminar cuentas de usuario y grupos en Linux. Esto también incluye la manera de establecer permisos para que dichos grupos y usuarios tengan acceso a archivos y directorios según lo necesiten.

Introducción a la administración de usuarios

La base de datos interna de los usuarios en GNU/Linux está relacionada con dos archivos:

- **/etc/passwd** contiene los nombres de los usuarios
- **/etc/shadow** contiene la contraseña encriptada con SHA512

Una buena política de seguridad es cambiar las contraseñas periódicamente. Al hacer esto, aunque logren desencriptar nuestras contraseñas, hacerlo les va a demandar mucho más tiempo. Probablemente cuando lo logren ya la hayamos cambiado nuevamente. Las contraseñas deben ser combinaciones de letras y números incluyendo letras mayúsculas y minúsculas además de caracteres especiales como por ejemplo **%**, **_, **o** $**.

Descripción del archivo /etc/passwd

Cuando agregamos un usuario, el sistema operativo agrega una línea en el archivo **/etc/passwd** y además se añade un grupo para cada usuario creado en **/etc/group**. Trabajar con grupos facilita la tarea del administrador ya que le permiten una administración de permisos más eficiente y menos trabajosa.

Al agregar un usuario, no sólo agregamos a este y un grupo sino también su buzón de correo. Se crea el archivo de correos en el directorio **/var/spool/mail/** y por último se crea su directorio personal (por lo general llamado simplemente *home*). Es en esta última ubicación que cada usuario guarda sus archivos personales organizados en subdirectorios. Fuera de allí, no puede escribir (salvo en los directorios **/tmp** y **/var/tmp**).

*Al añadir una nueva cuenta de usuario, también se copian desde **/etc/skel/** a su home los archivos **.bashrc**, **.bash_profile**, y **.bash_logout**. Los dos primeros contienen información del prompt del usuario y de los alias de comandos, mientras que el tercero almacena información sobre qué hacer cuando el usuario ejecuta el logout (salida del sistema).*

Todo esto ocurre en forma simultánea cuando damos de alta nuevos usuarios. Para esto, el sistema operativo usa **PAM** (**Plugable Administration Module**) los módulos PAM están definidos en el directorio **/etc/pam.d**.

El proceso de login de usuarios y servicios está relacionado con **PAM** (**P**lugable **A**uthentication **M**odule). Por lo tanto, antes de empezar a trabajar con los comandos para agregar usuarios veamos como hace el sistema operativo la validación de estos.

PAM es una jerarquía de comprobaciones de autenticación y autorizaciones. Estas comprobaciones están distribuidas en grupos de administración. Dichos grupos están compuestos por uno o más módulos PAM apilados, cada uno de los cuales realiza una función de comprobación determinada. Algunas de estas comprobaciones pueden ser obligatorios (como la existencia del usuario) y otras opcionales.

*La configuración de **PAM** se encuentra en el directorio **/etc/pam.d/**. Cada aplicación o servicio con capacidades PAM tiene un archivo en el directorio **/etc/pam.d/**. Cada archivo en este directorio tiene el mismo nombre del servicio al que controla el acceso. Si una aplicación utiliza **PAM** ella es la responsable por definir su nombre de servicio e instalar su propio archivo de configuración en el directorio **/etc/pam.d/**. Por ejemplo, el programa `login` define su nombre de servicio como **login** e instala el archivo de configuración **PAM** en **/etc/pam.d/login**.*

Observemos las líneas correspondientes a los usuarios root y alumno en el archivo /etc/passwd:

```
root:x:0:0:root:/root:/bin/bash
alumno:x:1000:1000:alumno,,,:/home/alumno:/bin/bash
```

Cada línea está dividida en 7 campos separados unos de otros por el caracter **:** (dos puntos). El significado de cada campo es el siguiente:

1. Nombre de usuario
2. Si contiene una **x**, indica que el usuario posee una contraseña encriptada dentro de **/etc/shadow**.
3. UID (User ID)
4. GID (Group ID) 5.- Nombre completo del usuario, y opcionalmente información adicional sobre el mismo, tal como el teléfono del trabajo o el teléfono particular
5. Comentario descriptivo sobre el usuario.
6. Ruta al directorio personal del usuario
7. Intérprete de comando del usuario (normalmente se utiliza **/bin/bash**)

La mayoría de las distribuciones, se reserva los primeros 1000 UID (del 0 al 999). Estos se denominan **UID del sistema** y son utilizados únicamente por el sistema. Los usuarios humanos tendrán siempre un UID a partir del ID 1000.

Crear usuarios y grupos

Para crear el usuario **fulano**, agregando el nombre completo (*Fulano De Tal*) en el campo comentario utilizaremos el comando `useradd` con la opción `-c` (o `adduser`) seguido por el nombre de la cuenta de usuario. Por defecto, también se creará el directorio personal del usuario en **/home/fulano** y se le asignará **/bin/bash** como shell:

```
useradd fulano -c "Fulano De Tal"
```

Si queremos cambiar el shell del usuario fulano a **/bin/sh**, el contenido del campo comentario a *Fulano De Tal actualizado* y el directorio home a **/mnt/homes/fulano** (el cual debe ser creado primero) usaremos `usermod` la opción `-s` (cambiar shell) y `-d` (cambiar home), junto con `-c` (actualizar comentario):

```
mkdir -p /mnt/homes/fulano
usermod -s /bin/sh -c "Fulano De Tal actualizado" -d
/mnt/homes/fulano fulano
```

---

*Para ver la lista de shells disponibles en nuestro sistema podemos inspeccionar el archivo **/etc/shells**. La shell o intérprete de comandos actúa como interfaz entre el*

---

sistema operativo para solicitarle que éste ejecute una tarea (por ejemplo, devolver el listado de un directorio, copiar un archivo y cualquier comando que se nos ocurra ejecutar).

Para asignar una contraseña inicial al usuario **fulano** emplearemos el comando `passwd`.

```
passwd fulano
```

Luego de iniciar sesión por primera vez, el usuario podrá cambiarla utilizando el mismo comando sin necesidad de especificar su nombre de usuario, es decir, simplemente utilizando `passwd` sin ningún argumento.

Una vez que hemos creado la cuenta de usuario, el comando `chage` con su opción `-l` nos permitirá ver información detallada sobre el período de validez de la misma (por defecto, una cuenta de usuario nunca expira) y de la contraseña.

La salida del comando

```
chage -l fulano
```

debería ser similar a la mostrada en la imagen:

```
[root@server ~]# chage -l fulano
Last password change                                    : Aug 02, 2018
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 0
Maximum number of days between password change          : 99999
Number of days of warning before password expires       : 7
[root@server ~]#
```

Ahora veremos cómo editar la cuenta de fulano de tal manera que la cuenta expire (`-E`) el 31 de diciembre de 2018, y mientras tanto se le permita cambiarla solamente (`-m`) una vez por semana (7 días). Cada contraseña será válida como máximo (`-M`) durante 60 días, y 3 días

antes se le debe enviar un recordatorio para que la modifique (`-W`). Finalmente, 5 días después de la fecha de vencimiento de la contraseña se procederá a bloquear (`-I`) la cuenta:

```
chage -E 2018-12-31 -m 7 -M 60 -W 3 -I 5 fulano
```

Veamos los datos de la cuenta nuevamente para verificar los cambios:

```
[root@server ~]# chage -E 2018-12-31 -m 7 -M 60 -W 3 -I 5 fulano
[root@server ~]# chage -l fulano
Last password change                                    : Aug 02, 2018
Password expires                                        : Oct 01, 2018
Password inactive                                       : Oct 06, 2018
Account expires                                         : Dec 31, 2018
Minimum number of days between password change          : 7
Maximum number of days between password change          : 60
Number of days of warning before password expires       : 3
[root@server ~]#
```

Por seguridad, podemos forzar a que **fulano** cambie su contraseña la próxima vez que inicie sesión:

```
chage -d 0 fulano
```

Otra forma de visualizar información de una cuenta dada consiste en utilizar el comando `finger` seguido por el nombre de usuario en cuestión.

Recordemos que como medida de seguridad es una buena idea requerir un cambio de clave cada cierto período de tiempo.

Administración de grupos

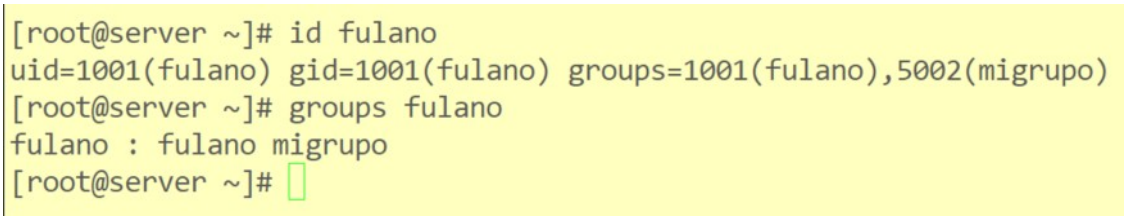
En Linux, los grupos constituyen el primer método de control de acceso a los recursos del sistema. Cuando se crea una cuenta de usuario, también se crea un grupo con el mismo nombre (conocido como *grupo primario* para dicha cuenta) y ambos son vinculados.

Para crear un nuevo grupo adicional llamado **migrupo** utilizaremos el comando groupadd. A continuación, para agregar a **fulano** al mismo volveremos a usar el comando `usermod` con sus opciones `-a` (de *append*) y `-G` (*group*) seguidos por el nombre del grupo y el usuario que deseamos agregar al mismo:

```
groupadd migrupo
usermod -a -G migrupo fulano
```

Para ver la lista de grupos a los que pertenece un determinado usuario podemos utilizar los comandos `groups` o `id`, seguido por el nombre de usuario:

```
id fulano
groups fulano
```



Para eliminar una cuenta de usuario de un grupo suplementario tenemos dos alternativas:

- Utilizar `usermod -G` seguido de los grupos en los que se necesita que el usuario permanezca activo, y finalmente del nombre del usuario en cuestión. Esta opción puede tornarse un tanto engorrosa si el usuario pertenece a muchos grupos suplementarios.
- Emplear `gpasswd -d` seguido del nombre de usuario y del grupo del que se desea eliminarlo.

En otras palabras, teniendo en cuenta que a este punto el usuario **fulano** pertenece a los grupos que se muestran en la imagen anterior, cualquiera de los dos comandos siguientes lo quitarán del grupo **migrupo**:

```
usermod -G fulano fulano
gpasswd -d fulano migrupo
```

Si lo que queremos es borrar la cuenta del usuario **fulano**, junto con su directorio home y la cola de correos haremos lo siguiente (en el caso de que nos interese eliminar la cuenta, pero conservando dichos recursos deberemos omitir la opción `-r`):

```
userdel -r fulano
```

Para borrar un grupo dado (**migrupo** en el siguiente ejemplo), utilizaremos

```
groupdel migrupo
```

Permisos en Linux

¿Para qué nos sirve que algo tenga dueño, pertenezca a algo o a alguien? El sistema operativo permite que los usuarios accedan a los recursos o ejecuten comandos teniendo en cuenta una política de permisos. Estos permisos son administrados por el usuario root (administrador del sistema).

Podemos dividir la política de permisos en tres grandes categorías. La clasificación de estas es la siguiente:

1. Permisos aplicados usuario dueño (*Owner*)

2. Permisos aplicados al grupo (*Group*)

3. Permisos aplicados a los otros (*Other*) usuarios del sistema no contemplados en **1** o en **2**.

De acuerdo con estas categorías podemos aplicar a cada una de estas los tres permisos: **read** (lectura), **write** (escritura) y **execute** (ejecución). En particular:

- **r**: indica que el usuario tiene permiso de lectura. Podrá ejecutar comandos de lectura como `ls`, `cat`, etc.
- **w**: indica que el usuario tiene permiso de escritura. Esto le permitirá editar archivos de texto, por ejemplo.
- **x**: indica que tiene permisos de ejecución. Podrá ejecutar un script.

Veamos cómo crear un directorio llamado **midirectorio** en **/tmp** y mostrar el usuario y el grupo dueños del mismo:

```
mkdir /tmp/midirectorio
ls -ld /tmp/midirectorio
```

Si queremos cambiar el usuario dueño o el grupo propietario de **/tmp/midirectorio**, podemos utilizar el comando `chown` de la siguiente manera:

```
chown fulano: /tmp/midirectorio # Cambiar usuario dueño
chown :fulano /tmp/midirectorio # Cambiar grupo propietario
(similar a chgrp)
chown fulano:fulano /tmp/midirectorio # Cambiar ambos
```

Otra opción para cambiar el grupo propietario es `chgrp`:

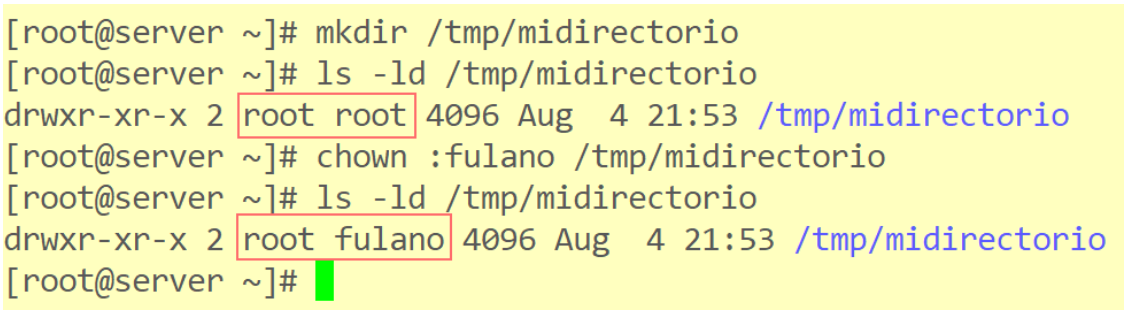
```
chgrp fulano /tmp/midirectorio
```

Ahora cambiemos el grupo propietario a **fulano**:

```
chgrp fulano /tmp/midirectorio
```

Tanto en el caso de `chown` como para `chgrp` una de las opciones más utilizadas en `-r`, la cual hace que la operación se aplique de forma recursiva a todo el contenido de un directorio.

En la imagen siguiente podemos apreciar el resultado de haber cambiado el grupo propietario del directorio a **fulano**:



En la primera columna de la salida de `ls -ld` podemos ver los permisos asignados para el usuario dueño y el grupo propietario de **/tmp/midirectorio**, y para el resto de los usuarios del sistema:

- La primera letra (**d**) indica que **/tmp/midirectorio** es un directorio. Si bien nosotros sabíamos esto por haberlo creado previamente, en ocasiones tendremos que identificar si el recurso indicado es un directorio, un archivo común y corriente (indicado por un **-** en vez de la **d**), o un enlace simbólico (por la presencia de una **l**), entre otros.

- A continuación, la primera terna de caracteres (**rwx**) indica que el usuario dueño posee permisos de lectura (**r**), escritura (**w**), y ejecución (**x**) sobre el directorio. La segunda terna (**r-x**) indica que los miembros del grupo propietario poseen permisos de lectura y ejecución (pero no de escritura). Finalmente, la última terna muestra que todos los demás usuarios del sistema poseen los mismos permisos que en el caso anterior.

Veamos la forma en que los permisos **rwx** nos permiten restringir el acceso a los recursos del sistema:

- El permiso de lectura (**r**) es requerido para poder listar el contenido del directorio.
- El permiso de escritura (**w**) permite que se agreguen archivos al directorio, que se renombren otros ya existentes, que se borren dichos archivos, o que se modifiquen los atributos del directorio. De todas formas, hay que tener en cuenta que para poder realizar estas operaciones es necesario el permiso de ejecución sobre el mismo directorio.
- El permiso de ejecución (**x**) es necesario para ingresar al directorio y acceder a archivos y a subdirectorios que se encuentren dentro del mismo.

Para dar permisos de escritura sobre **/tmp/midirectorio** a miembros del grupo **fulano** deberemos usar el comando `chmod` seguido de la opción `g` (grupo), del signo `+` y del permiso que deseemos asignar (en este caso `w`):

```
chmod g+w /tmp/midirectorio
```

Si quisiéramos asignar (o quitar, en cuyo caso utilizaremos el signo **-**) más de un permiso a la vez podemos hacerlo fácilmente. Por ejemplo, veamos cómo quitar los permisos de lectura y ejecución para el resto de los usuarios del sistema (simbolizados por la opción `o`, de *others* en inglés):

```
chmod o-rx /tmp/midirectorio
```

En el caso de que queramos modificar los permisos de un directorio y de sus contenidos simultáneamente, podemos usar la opción `-R` (de *recursiva*) de `chmod`. Tomando el caso anterior como ejemplo:

```
chmod -R o-rx /tmp/midirectorio
```

Por último, para asignar o quitar permisos al usuario dueño, utilizamos la opción `u` (del inglés *user*), mientras que si queremos otorgar o remover un permiso en particular para todos usaremos la opción `a` (de `all`). De esta manera, el siguiente comando activa los permisos de ejecución sobre **/tmp/midirectorio** para todos los usuarios:

```
chmod a+x /tmp/midirectorio
```

El usuario dueño de un recurso puede administrar los permisos de grupo y otros. Esto significa que puede otorgar o quitar derechos sobre sus propios recursos.

Adicionalmente, podemos indicar un determinado permiso en forma octal en vez de **r**, **w**, o **x**:

	Read	Write	Execute
Owner	4	2	1
Group	4	2	1
Other	4	2	1

Para encontrar ayuda en la conversión podemos consultar cualquier calculador de permisos en línea.

Por ejemplo, para asignar permisos **rwx** para el dueño, **rw** para el grupo, y solamente **r** para el resto sobre el archivo test.txt podemos hacer:

```
chmod u+rwx,g+rw,o+r test.txt
```

o su equivalente

```
chmod 764 test.txt
```

En la próxima sección explicaremos de dónde provienen los permisos que se establecen sobre nuevos archivos y directorios automáticamente cuando son creados.

Máscaras de archivos y directorios

Por lo general, cada vez que utilizamos una cuenta de usuario limitado para crear un archivo o un directorio, se establecen por defecto los permisos **rw-rw-r–** (octal 664) y **rwxrwxr-x** (octal 775), respectivamente. Si lo hacemos como root, los permisos se setean como **rw-r–r–** (octal 644) y **rwxr-xr-x** (octal 755). Esto es resultado de aplicar lo que se conoce como máscaras de archivos y directorios, cuyos valores podemos visualizar y cambiar mediante el comando `umask`.

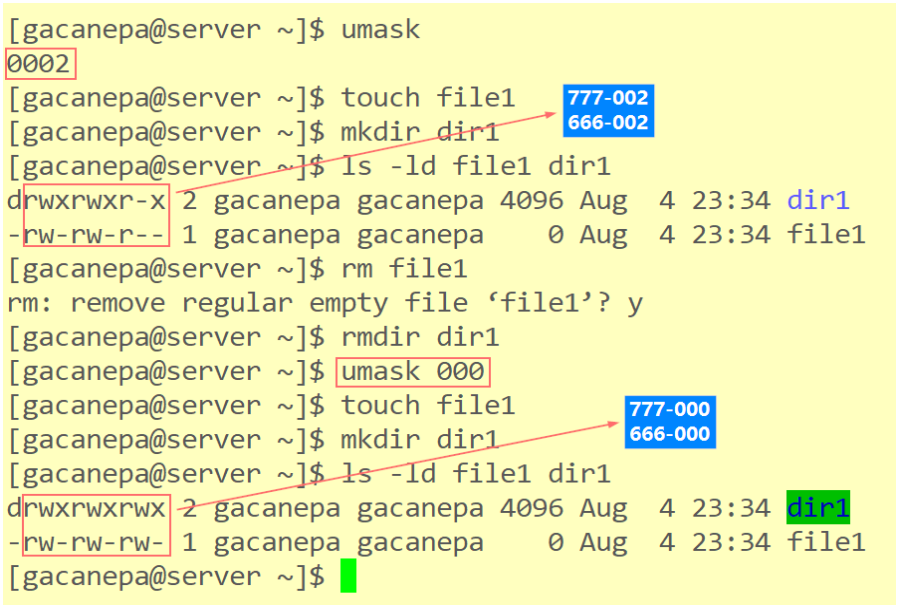
En esencia, el permiso resultante surge de restar las máscaras del usuario a los valores 666 para archivos y 777 para directorios.

El siguiente ejemplo nos permitirá ilustrar:

1. Ejecutemos `umask` para un usuario cualquiera.
2. Al crear un archivo llamado **file1** los permisos resultarán de la resta entre 666 y la máscara de creación de archivos (002 para un usuario común y 022 para root).
3. Cuando añadamos un directorio **dir1**, los permisos resultarán de la resta entre 777 y la máscara de creación de directorios (mismos valores que en el caso anterior).
4. Eliminemos **file1** y **dir1**.
5. Establezcamos un nuevo valor para las máscaras (000 en el siguiente ejemplo) y volvamos a ejecutar los comandos de los pasos 1 a 3.

```
umask
touch file1
mkdir dir1
ls -ld file1 dir1
rm file1
rmdir dir1
umask 000
touch file1
mkdir dir1
ls -ld file1 dir1
```

En la imagen siguiente podemos ver el resultado de los comandos anteriores:



Los cambios que hicimos en la máscara permanecerán hasta que reiniciemos el equipo o que efectuemos otra modificación. Si deseamos aplicarlos de manera permanente deberemos colocar el comando `umask` junto con el valor deseado en el archivo de inicialización de nuestra shell (**~/bash_profile** o **~/.profile**) y volver a iniciar sesión o aplicar los cambios a nuestro entorno mediante

```
source ~/.bash_profile
```

o

```
source ~/.profile
```

según el archivo en donde hayamos hecho la modificación.

Ejercicio integrador

1. Crear cuatro usuarios llamados **juan**, **jose**, **pedro**, y **felipe**.
2. Crear tres grupos llamados **compras**, **ventas**, y **administracion**.
3. Crear tres directorios llamados **dir-compras**, **dir-ventas**, y **dir-administracion**.
4. Agregar los usuarios a los grupos que correspondan y establecer los propietarios y accesos según la tabla que se indica a continuación:

GRUPOS	MIEMBROS	DIRECTORIOS	DUEÑOS	DENEGAR
compras	juan, jose, pedro	dir-compras	juan:compras	felipe
ventas	jose, felipe	dir-ventas	jose:ventas	juan, pedro
administracion	jose, pedro, felipe	dir-administracion	felipe:administracion	juan