CICLO FORMATIVO DE GRADO SUPERIOR - TÉCNICO EN ADMINISTRACIÓN DE SISTEMAS INFORMÁTICOS EN RED

IMPLANTACIÓN DE SISTEMAS OPERATIVOS

LINUX

Usuarios y permisos

Índice

1.1.- ¿POR QUÉ EXISTEN GRUPOS, USUARIOS Y PERMISOS?

Ya vimos anteriormente que los ficheros deben estar organizados en directorios (carpetas) con el fin de tenerlos ordenados y poder localizarlos convenientemente.

En el caso de una oficina cada papel está en su sitio, hay carpetas y subcarpetas y todo está organizado. Ahora bien, el contable deberá tener acceso, por ejemplo, a las carpetas donde se encuentras las facturas y los recibos, pero no tienen por qué tener acceso a la información sobre desarrollo de productos o marketing.

En un sistema Linux, las carpetas y los archivos funcionan de esta manera. Por ejemplo, los archivos de configuración que se encuentran en el directorio /etc sólo pueden ser modificados por el administrador del sistema. Esto previene que cualquier usuario pueda cambiar información crítica y estropear algo.

1.2.- ¿QUÉ ES EL SUPERUSUARIO?

El superusuario, administrador del sistema o simplemente el root, es un usuario especial que tiene privilegios para cambiar la configuración, borrar y crear ficheros en cualquier directorio, crear nuevos grupos y usuarios, etc.

IMPORTANTE: ES PELIGROSO TRABAJAR COMO SUPERUSUARIO, SE PUEDE DAÑAR EL SISTEMA DE FORMA IRREVERSIBLE. EL LECTOR DEBE ESTAR SEGURO DE LO QUE HACE CUANDO TRABAJE COMO SUPERUSUARIO.

Una vez hecha esta aclaración, pasemos a hacer algo como root:

```
$ touch /etc/prueba.txt

touch: no se puede efectuar `touch' sobre «/etc/prueba.txt»: Permiso denegado

$ sudo touch /etc/prueba.txt

$ ls /etc/pru*

/etc/prueba.txt
```

Hemos intentado primero crear el fichero prueba.txt en el directorio /etc como usuario normal y acto seguido hemos obtenido un error de “Permiso denegado”, lo que quiere

decir que un usuario sin privilegios no puede hacer eso. A continuación, lo hemos intentado como administrador, para ello hemos usado el comando sudo, tras lo que se nos ha preguntado la clave del administrador.

Esta vez sí lo hemos conseguido. No tendría mucho sentido que el sistema no preguntase por la clave, ya que en ese caso cualquiera podría ejecutar comandos como administrador con el peligro que ello supone.

1.3.- PERMISOS

La información sobre grupos, usuarios y permisos se puede obtener mediante el comando ls junto con la opción -l.
Vamos a ver los permisos que tiene establecidos el fichero whatis que se encuentra en el directorio /usr/bin.

```
$ ls -l /usr/bin/whatis
-rwxr-xr-x 1 root root 87792 2008-03-12 14:24 /usr/bin/whatis
```

En la primera columna aparecen los permisos, en la tercera se indica el usuario (en este caso es el administrador del sistema) y en la cuarta columna aparece el nombre del grupo (que en este caso coincide con el de usuario).

Vamos a ver qué significan exactamente los caracteres de la primera columna:

-	r	w	x	r	-	x	r	-	x
Tipo de fichero.	Permisos para el dueño del fichero.			Permisos para el grupo al que pertenece el fichero.			Permisos para el resto de usuarios		

r	Permiso de lectura.
w	Permiso de escritura.
x	Permiso de ejecución.

El tipo de fichero se indica en la siguiente tabla:

Tipo de fichero	
l	Enlace simbólico.
c	Dispositivo especial de caracteres.
b	Dispositivo especial de bloques.
p	FIFO (estructura de datos).
s	Socket (comunicaciones).
-	Ninguno de los anteriores. Puede ser un fichero de texto, un binario, etc.

En el caso que nos ocupa tenemos un carácter "-" como tipo de fichero, porque se trata de un binario (un programa). El dueño del fichero tiene los permisos rwx, lo que quiere decir que puede leer, escribir y ejecutar el fichero. Que tiene permiso para escribir significa que puede borrarlo, cambiarle el nombre o editarlo. Tanto el grupo como el resto de usuarios tienen los permisos r-x, lo que significa que pueden utilizarlo (pueden leerlo y ejecutarlo) pero no lo pueden modificar.

1.4.- ¿QUIÉNES SOMOS? (whoami, groups)

Antes de empezar a crear usuarios, crear grupos y cambiar permisos, debemos saber quiénes somos y a qué grupo o grupos pertenecemos. Aunque en principio entremos en el sistema como un determinado usuario, podemos utilizar su para ejecutar comandos como otro usuario distinto, siempre y cuando sepamos la contraseña de ese otro usuario.

```
$ whoami
nira
$ su alumno
Contraseña:
$ whoami
alumno
```

Para volver a ser el usuario original basta con utilizar exit.

```
$ whoami
alumno
$ exit
exit
$ whoami
nira
```

Con el comando groups se puede ver a qué grupo pertenecemos.

```
$ groups
nira adm cdrom sudo dip plugdev lpadmin admin lxd sambashare
```

Se pueden especificar uno o más usuarios detrás de groups. Eso nos dirá a qué grupos pertenece cada uno de ellos.

```
~$ groups alumno
root alumno : alumno
root : root
```

1.5.- GESTIÓN DE GRUPOS (groupadd, groupdel, groupmod)

Los comandos groupadd, groupdel y groupmod permiten crear, borrar y modificar grupos respectivamente.
Vamos a crear los grupos oficina_madrid, oficina_bcn y oficina_lp

```
$ groupadd oficina_madrid
groupadd: incapaz de bloquear el fichero de grupos
$ sudo groupadd oficina_madrid
$ sudo groupadd oficina_bcn
$ sudo groupadd oficina_lp
```

Vemos que si intentamos crear un grupo como usuario sin privilegios obtenemos un error. Para manejar grupos y usuarios es necesario ejecutar los comandos con privilegios de administrador, por tanto deberemos teclear sudo antes del comando en cuestión.

Si hubiéramos escrito mal el nombre del primer grupo, ¡que no cunda el pánico!, este problema se puede solventar con groupmod.

$ sudo groupmod -n oficina_madrid oficina_madrit

La directiva de la empresa ha decidido cerrar la oficina de Barcelona para ahorrar costes y pasar los recursos a la oficina de Madrid, así que no hará falta el grupo oficina_bcn. Lo podemos borrar con groupdel.

$ sudo groupdel oficina_bcn

1.6.- GESTIÓN DE USUARIOS (adduser, userdel, usermod)

La gestión de usuarios, al igual que la de grupos, exige que los comandos se ejecuten con los privilegios del administrador del sistema. Se puede escribir sudo antes de cada comando, o se puede hacer lo siguiente:

$ sudo bash

Si añades esta instrucción el prompt cambia. Ahora se muestra un carácter “#” en lugar de un “$”. A partir de ahora, todos los comandos se ejecutarán con privilegios de administrador del sistema. Hay que acordarse de volver al usuario inicial mediante exit.

Es necesario dar de alta a dos usuarios para el grupo oficina_madird y uno para oficina_lp. Habrá un cuarto usuario que estará yendo y viniendo de una oficina a otra, por tanto, se le dará de alta en las dos.

adduser pedro --ingroup oficina madrid adduser ana --ingroup oficina_madrid adduser berta --ingroup oficina_lp adduser laura --ingroup oficina_madrid adduser laura oficina_lp

Hemos matado dos pájaros de un tiro. Hemos creado los usuarios y al mismo tiempo los hemos incluido dentro de los grupos correspondientes. Estos dos pasos se pueden hacer de forma independiente.

El usuario laura pertenece a dos grupos. En primer lugar, se ha creado el usuario y al mismo tiempo se ha añadido al grupo oficina_madrid con la opción –ingroup. Para añadir un usuario existente a un grupo, se utiliza adduser sin opciones.

groups ana berta laura ana : oficina_madrid berta : oficina_lp

laura : oficina_madrid oficina_lp

Al crear los usuarios, se nos han pedido las claves, no obstante, estas claves se pueden cambiar con el comando passwd.

passwd pedro passwd ana passwd laura

Recuerda salir del modo root con el comando exit cuando no tenga que hacer tareas que requieran privilegios de administrador.

exit

De ahora en adelante, simplemente se indicará con el carácter “$” que se trabaja como usuario sin privilegios y con el carácter “#” que se trabaja como root.

Cabe señalar que, para cada usuario, se crea por defecto un directorio dentro de /home. Cuando un usuario se conecta al sistema, “aterriza” en ese directorio. Es lo que hemos denominado anteriormente como el directorio de trabajo.

$ ls /home/

alumno ana berta ftp laura Luis jose pedro

1.7.- CAMBIO DE GRUO Y DE DUEÑO (chown, chgrp)

Imaginemos que el fichero informe.txt ha sido creado por el usuario pedro. Por defecto, el dueño de un archivo es el usuario que lo crea, en este caso pedro. El grupo del usuario pedro, como hemos visto antes es oficina_madrid.

```
$ su pedro

$ cd

$ pwd

/home/pedro

$ touch informe.txt

$ ls -l

-rw-r--r-- 1 pedro oficina_madrid       0 2022-03-19 12:46 informe.txt
```

Todo esto se puede cambiar. Moveremos el fichero al directorio de trabajo del usuario laura y le cambiaremos el dueño.

```
mv informe.txt /home/laura/ cd /home/laura/

chown laura informe.txt ls -l

-rw-r--r-- 1 laura oficina_malaga       0 2022-03-19 12:46 informe.txt
```

Ahora el fichero tiene al usuario laura como propietario.

Tanto chown como chgrp se pueden usar con la opción -R para cambiar el dueño o el grupo en un directorio completo, de forma recursiva.

1.8.- CAMBIO DE PRIVILEGIOS (chmod)

El comando chmod sirve para cambiar los permisos de uno o varios ficheros. Esos mismos permisos que se pueden ver con ls -l.

```
$ ls -l

-rw-r--r-- 1 pedro oficina_madrid       0 2022-03-19 15:38 hola_mundo.rb

$ chmod +x hola_mundo.rb

$ ls -l

-rwxr-xr-x 1 pedro oficina_madrid       0 2022-03-19 15:38 hola_mundo.rb
```

Hemos añadido el permiso de ejecución al fichero hola_mundo.rb. Vemos que ahora hay tres x, la que corresponde al dueño del fichero, la de todos los usuarios que pertenecen al grupo y la del resto de usuarios.

Cuando no se especifica ninguna de estas tres letras correspondientes a los usuarios (u, g, o) como en el ejemplo anterior, se sobreentiende que nos referimos a todos ellos. Se puede indicar de forma explícita con el carácter a (all).

Para entenderlo mejor, en la siguiente tabla, se muestran de forma esquemática, los parámetros del comando chmod:

u	g	o	+ -	r	w	x
(user) dueño del fichero	(group) usuarios que pertenecen almismo grupo	(others) el resto de usuarios	dar permiso quitar permiso	(read) lectura	(write) escritura	(execution) ejecución

Quitaremos ahora el permiso de ejecución para el resto de usuarios (others) y daremos permiso de escritura (write) a los usuarios del mismo grupo (group).

```
$ ls -l

-rwxr-xr-x 1 pedro oficina_madrid      0 2022-03-19 15:38 hola_mundo.rb

$ chmod o-x hola_mundo.rb

$ chmod g+w hola_mundo.rb

$ ls -l

-rwxrwxr-- 1 pedro oficina_madrid      0 2022-03-19 15:38 hola_mundo.rb
```

A este método, que utiliza los caracteres rwx se le denomina método simbólico. Podemos utilizar de forma análoga el método numérico.

4 2 1	Total
r w x	4 + 2 + 1 = **7**
r w -	4 + 2 + 0 = **6**
r - x	4 + 0 + 1 = **5**
r - -	4 + 0 + 0 = **4**
- w x	0 + 2 + 1 = **3**
- w -	0 + 2 + 0 = **2**
- - x	0 + 0 + 1 = **1**

De esta forma, esta línea

```
$ chmod 755 hola_mundo.rb
```

sería equivalente a estas tres

```
$

$

$
```

Y entonces

```
$ ls -l

-rwxr-xr-x 1 pedro oficina_madrid      0 2022-03-19 15:38 hola_mundo.rb
```

Los permisos de los directorios se pueden cambiar de la misma forma que los ficheros, aunque el significado es algo diferente. Si un directorio tiene el permiso de lectura quiere decir que se puede ver su contenido. Si tiene permiso de escritura, quiere decir que se pueden crear ficheros dentro y si tiene permiso de ejecución quiere decir que se puede entrar dentro.

RESUMEN

Los comandos vistos son los siguientes:

COMANDO	*ACCIÓN*
ls -l	Muestra, entre otras cosas, información sobre los permisos, el usuario y elgrupo al que pertenece el fichero.
sudo	Permite ejecutar comandos como root.
su	Cambia de usuario.
whoami	Muestra el nombre del usuario actual.
groups	Muestra el/los grupos/s a los que pertenece el usuario actual.
groupadd	Añade un nuevo grupo.
groupdel	Borra un grupo.
groupmod	Modifica las características de un grupo.
adduser	Añade un nuevo usuario.
userdel	Borra un usuario.
usermod	Modifica las características de un usuario.
passwd	Asigna o cambia la clave de un usuario.
chown	Cambia el dueño de un archivo.
chgrp	Cambia el grupo al que pertenece un archivo.
chmod	Cambia los permisos.