Permisos del sistema de archivos

Resumen de permisos

Como hemos dicho, Slackware Linux es un sistema operativo multiusuario. Debido a esto, sus sistemas de archivos también son de usuario múltiple. Esto significa que cada archivo o directorio tiene un conjunto de permisos que pueden otorgar o denegar privilegios a diferentes usuarios. Hay tres permisos básicos y tres conjuntos de permisos para cada archivo. Echemos un vistazo a un archivo de ejemplo.

```
darkstar:~$ ls -l /bin/ls
-rwxr-xr-x 1 root root 81820 2007-06-08 21:12 /bin/ls
```

Recuerde del capítulo 4 que ***ls*** *-l* enumera los permisos para un archivo o directorio junto con el usuario y el grupo que *"posee"* el archivo. En este caso, los permisos son rwxr-xr-x, el usuario es root y el grupo también es root. La sección de permisos, aunque agrupada, es en realidad tres piezas separadas. El primer conjunto de tres letras son los permisos otorgados al usuario que posee el archivo. El segundo conjunto de tres son los otorgados al propietario del grupo, y los últimos tres son permisos para todos los demás.

Tabla 10.1. Permisos de /bin/ls

Grupo	Listado	Significado
Propietario	rwx	El propietario "root" puede leer, escribir y ejecutar
Grupo	r-x	El grupo "root" puede leer y ejecutar
Otros	r-x	Todos los demás pueden leer y ejecutar

Los permisos son bastante autoexplicativos, por supuesto, al menos para archivos. Leer, escribir y ejecutar le permite leer un archivo, escribir en él o ejecutarlo. Pero, ¿qué significan estos permisos para los directorios? En pocas palabras, los permisos de lectura otorgan la posibilidad de listar los contenidos del directorio (digamos con ***ls***). El permiso de escritura otorga la posibilidad de crear nuevos archivos en el directorio, así como de eliminar todo el directorio, incluso si de otra forma no podría eliminar algunos de los otros archivos que contiene. El permiso de ejecución otorga la posibilidad de ingresar al directorio (por ejemplo, con cd, comando incorporado de ***bash***).

Echemos un vistazo a los permisos en un directorio ahora.

```
darkstar:~$ ls -ld /home/alan
drwxr-x--- 60 alan users 3040 2008-06-06 17:14 /home/alan/
```

Aquí vemos los permisos en mi directorio personal y su propiedad. El directorio es propiedad del usuario alan y el grupo users. Al usuario se le otorgan todos los derechos (rwx), al grupo solo se le otorgan permisos de lectura y ejecución (r-x), y todos los demás tienen prohibido hacer cualquier cosa.

chmod, chown, y chgrp

Entonces, ahora que sabemos qué son los permisos, ¿cómo los cambiamos? Y para el caso, ¿cómo asignamos la propiedad del usuario y del grupo? La respuesta está aquí en esta sección.

La primera herramienta que analizaremos es el comando útil ***chown*** (1). Utilizando ***chown*** , podemos (lo has adivinado), cambiar la propiedad de un archivo o directorio. ***chown*** se usa históricamente solo para cambiar la propiedad del usuario, pero también puede cambiar la propiedad del grupo.

```
darkstar:~# ls -l /tmp/foo
total 0
-rw-r--r-- 1 alan users 0 2008-06-06 22:29 a
-rw-r--r-- 1 alan users 0 2008-06-06 22:29 b
darkstar:~# chown root /tmp/foo/a
darkstar:~# ls -l /tmp/foo
total 0
-rw-r--r-- 1 root users 0 2008-06-06 22:29 a
-rw-r--r-- 1 alan users 0 2008-06-06 22:29 b
```

Al usar dos puntos después de la cuenta de usuario, también puede especificar una nueva cuenta de grupo.

```
darkstar:~# chown root:root /tmp/foo/b
darkstar:~#  ls -l /tmp/foo
total 0
-rw-r--r-- 1 root users 0 2008-06-06 22:29 a
-rw-r--r-- 1 root root  0 2008-06-06 22:29 b
```

chown también se puede usar recursivamente para cambiar la propiedad de todos los archivos y directorios debajo de un directorio de destino. El siguiente comando cambiaría todos los archivos bajo el directorio `/tmp/foo` para tener su propiedad establecida en root: root.

```
darkstar:~# chown -R root:root /tmp/foo/b
```

Especificar dos puntos y un nombre de grupo sin un nombre de usuario simplemente cambiará el grupo para un archivo y dejará intacta la propiedad del usuario.

```
darkstar:~# chown :wheel /tmp/foo/a
darkstar:~# ls -l /tmp/foo
ls -l /tmp/foo
total 0
-rw-r--r-- 1 root wheel 0 2008-06-06 22:29 a
-rw-r--r-- 1 root root  0 2008-06-06 22:29 b
```

El hermano menor de ***chown*** es un poco menos útil ***chgrp*** (1). Este comando funciona igual que ***chown*** , excepto que solo puede cambiar la propiedad del grupo de un archivo. Como ***chown*** ya puede hacer esto, ¿por qué molestarse con ***chgrp*** ? La respuesta es simple. Muchos otros sistemas operativos utilizan una versión diferente de ***chown*** que no puede cambiar la propiedad del grupo, por lo que si alguna vez te encuentras con uno de ellos, ahora sabes cómo hacerlo.

Hay una razón por la que discutimos el cambio de propiedad antes de cambiar los permisos. El primero es un concepto mucho más fácil de entender. La herramienta para cambiar permisos en un archivo o directorio es ***chmod*** (1). La sintaxis es casi idéntica a la de ***chown*** , pero en lugar de especificar un usuario o grupo, el administrador debe especificar un conjunto de permisos octales o un conjunto de permisos alfabéticos. Ninguno de los dos es especialmente fácil de entender la primera vez. Comenzaremos con los permisos octales menos complicados.

Los permisos octales derivan que su nombre se asigna por uno de ocho dígitos, a saber, los números del 0 al 7. A cada permiso se le asigna un número que es una potencia de 2, y esos números se suman para obtener los permisos finales para uno de los conjuntos de permisos. Si esto suena confuso, tal vez esta tabla ayude.

Tabla 10.2. Permisos octales

Permiso	Significado
Lectura	4
Escritura	2
Ejecución	1

Al sumar estos valores, podemos alcanzar cualquier número entre 0 y 7 y especificar todas las combinaciones de permisos posibles. Por ejemplo, para otorgar privilegios de lectura y escritura mientras se deniega la ejecución, usaríamos el número 6. El número 3 otorgaría permisos de escritura y ejecución, pero negaría la capacidad de leer el archivo. Debemos especificar un número para cada uno de los tres conjuntos cuando se utilizan permisos octales. Por ejemplo, no es posible especificar solo un conjunto de permisos de usuario o grupo.

```
darkstar:~# ls -l /tmp/foo/a
-rw-r--r-- 1 root root  0 2008-06-06 22:29 a
darkstar:~# chmod 750 /tmp/foo/a
darkstar:~# ls -l /tmp/foo/a
-rwxr-x--- 1 root root  0 2008-06-06 22:29 a
```

chmod también puede usar valores de letras junto con + o − para otorgar o denegar permisos. Si bien esto puede ser más fácil de recordar, a menudo es más fácil usar los permisos octales.

Tabla 10.3. Permisos alfabéticos

Permiso	Valor de la letra
Leer	r
Escribir	w
Ejecutar	x

Tabla 10.4. Usuarios y grupos alfabéticos

Cuentas afectadas	Valor letra
Usuario / Propietario	u
Grupo	g
Otros / Mundo	o

Para usar los valores de las letras con ***chmod***, debe especificar con cual conjunto usarlos, ya sea *“u”* para el usuario, *“g”* para el grupo, y *“o”* para todos los demás. También debe especificar si está

agregando o eliminando permisos con los signos “+” y “-”. Se pueden cambiar varios conjuntos a la vez separándolos con una coma.

```
darkstar:/tmp/foo# ls -l
total 0
-rw-r--r-- 1 alan users 0 2008-06-06 23:37 a
-rw-r--r-- 1 alan users 0 2008-06-06 23:37 b
-rw-r--r-- 1 alan users 0 2008-06-06 23:37 c
-rw-r--r-- 1 alan users 0 2008-06-06 23:37 d
darkstar:/tmp/foo# chmod u+x a
darkstar:/tmp/foo# chmod g+w b
darkstar:/tmp/foo# chmod u+x,g+x,o-r c
darkstar:/tmp/foo# chmod u+rx-w,g+r,o-r d
darkstar:/tmp/foo# ls -l
-rwxr--r-- 1 alan users 0 2008-06-06 23:37 a*
-rw-rw-r-- 1 alan users 0 2008-06-06 23:37 b
-rwxr-x--- 1 alan users 0 2008-06-06 23:37 c*
-r-xr----- 1 alan users 0 2008-06-06 23:37 d*
```

Lo que prefieras usar depende completamente de ti. Hay lugares donde uno es mejor que el otro, por lo que un verdadero Slacker lo sabrá de adentro hacia afuera.

SUID, SGID, y el "Sticky" Bit

Todavía no hemos terminado con los permisos. Hay otros tres *permisos “especiales”* además de los mencionados anteriormente. Son SUID, SGID, y el sticky bit. Cuando un archivo tiene uno o más de estos permisos establecidos, se comporta de manera especial. Los permisos SUID y SGID cambian la forma en que se ejecuta una aplicación, mientras que el sticky bit restringe la eliminación de archivos. Estos permisos se aplican con ***chmod*** como lectura, escritura y ejecución, pero con un giro.

SUID y SGID representan *“Establecer ID de usuario”* (*“Set User ID”* y *“Establecer ID de grupo”* respectivamente. Cuando se establece una aplicación con uno de estos bits, la aplicación se ejecuta con los permisos de propiedad del usuario o grupo de esa aplicación, independientemente de qué usuario la ejecutó realmente. Echemos un vistazo a una aplicación SUID común, humilde ***passwd*** y los archivos que modifica.

```
darkstar:~# ls -l /usr/bin/passwd \
  /etc/passwd \
  /etc/shadow
-rw-r--r-- 1 root root    1106 2008-06-03 22:23 /etc/passwd
-rw-r----- 1 root shadow   627 2008-06-03 22:22 /etc/shadow
-rws--x--x 1 root root   34844 2008-03-24 16:11 /usr/bin/passwd*
```

Observe los permisos en ***passwd***. En lugar de una `x` en la ranura de ejecución del usuario, tenemos una `s`. Esto nos dice que ***passwd*** es un programa SUID, y cuando lo ejecutamos, el proceso se ejecutará como usuario *“root”* en lugar del usuario que realmente lo ejecutó. La razón de esto es evidente tan pronto como usted mira los dos archivos que modifica. Ni `/etc/passwd` ni `/etc/shadow` son grabables por otra persona que no sea root. Dado que los usuarios necesitan cambiar su información personal, ***passwd*** debe ejecutarse como root para modificar esos archivos.

Entonces, ¿qué pasa con el sticky bit? El sticky bit restringe la capacidad de mover o eliminar archivos y solo se establece en directorios. Los usuarios que no son root no pueden mover o eliminar ningún archivo de un directorio con el conjunto de sticky bit a menos que sean los propietarios de ese archivo. Normalmente, cualquier persona que tenga permiso de escritura en el archivo puede hacer esto, pero el sticky bit lo impide para cualquier persona que no sea el propietario (y, por supuesto, root). Echemos un vistazo a un directorio común *"sticky bit"*.

```
darkstar:~# ls -ld /tmp
drwxrwxrwt 1 root root   34844 2008-03-24 16:11 /tmp
```

Naturalmente, al ser un directorio para el almacenamiento de archivos temporales en todo el sistema, `/tmp` debe ser legible, grabable y ejecutable por todos y cada uno. Dado que es probable que cualquier usuario tenga un archivo o dos almacenados aquí en cualquier momento, solo tiene sentido evitar que otros usuarios eliminen esos archivos, por lo que se ha establecido el sticky bit. Puede verlo por la presencia de la `t` en lugar de la `x` en la sección de permisos globales.

Tabla 10.5. Permisos SUID, SGID y "sticky bit"

Tipo de permiso	Valor octal	Valor de letra
SUID	4	s
SGID	2	s
Sticky	1	t

Al utilizar permisos octales, debe especificar un valor octal inicial adicional. Por ejemplo, para recrear el permiso en `/tmp`, usaríamos 1777. Para recrear esos permisos en `/usr/bin/passwd`, usaríamos 4711. Básicamente, toda vez que éste cuarto octeto principal no está especificado, ***chmod*** asume que su valor es 0.

```
darkstar:~# chmod 1777 /tmp
darkstar:~# chmod 4711 /usr/bin/passwd
```

El uso de los valores de permiso alfabético es ligeramente diferente. Asumiendo que los dos archivos anteriores tienen permisos de 0000 (sin permisos), aquí es cómo los estableceríamos.

```
darkstar:~# chmod ug+rwx,o+rwt /tmp
darkstar:~# chmod u+rws,go+x /usr/bin/passwd
```

Navegación de capítulos

Capítulo anterior: Usuarios y Grupos

Capítulo siguiente: Trabajando con sistemas de archivos

Fuentes

- Fuente original: http://www.slackbook.org/beta

- Escrito originalmente por Alan Hicks, Chris Lumens, David Cantrell, Logan Johnson
- Traducción por: Victor 2019/02/03 03:34 (UTC)

slackbook, filesystem, permissions, suid, sgid, sticky bit, chmod, chown, chgrp