Permisos de archivos y directorios

Autor: Sergio González D. **(sergio.gonzalez.duran@gmail.com)**

Introducción

¿Has visto esa combinación de r,w,x y - cuando listas un directorio?, tienes cierta idea que son los permisos, pero ¿como se usan y como funcionan?. En este manual sobre permisos de archivos y directorios de LinuxTotal entenderás totalmente su uso y la manera correcta de utilizarlos.

En Linux, todo archivo y directorio tiene tres niveles de permisos de acceso: los que se aplican al propietario del archivo, los que se aplican al grupo que tiene el archivo y los que se aplican a todos los usuarios del sistema. Podemos ver los permisos cuando listamos un directorio con **`ls -l`**:

```
$> ls -l
-rwxrwxr--  1 sergio ventas    9090 sep  9 14:10 presentacion
-rw-rw-r--  1 sergio sergio 2825990 sep  7 16:36 reporte1
drwxr-xr-x  2 sergio sergio    4096 ago 27 11:41 videos
```

Veamos por partes el listado, tomando como ejemplo la primera línea. La primera columna (-rwxrwxr--) es el tipo de archivo y sus permisos, la siguiente columna (1) es el número de enlaces al archivo, la tercera columna (sergio) representa al propietario del archivo, la cuarta columna (ventas) representa al grupo al que pertence al archivo y las siguientes son el tamaño, la fecha y hora de última modificación y por último el nombre delarchivo o directorio.

El primer caracter al extremo izquierdo, representa el tipo de archivo, los posibles valores para esta posición son los siguientes:

- **-** un guión representa un archivo comun (de texto, html, mp3, jpg, etc.)
- **d** representa un directorio
- **l** link, es decir un enlace o acceso directo
- **b** binario, un archivo generalmente ejecutable

Los siguientes 9 restantes, representan los permisos del archivo y deben verse en grupos de 3.

Los tres primeros representan los permisos para el propietario del archivo. Los tres siguientes son los permisos para el grupo del archivo y los tres últimos son los permisos para el resto del mundo o otros.

```
rwx       rwx      rwx
usuario   grupo    otros
```

En cuanto a las letras, su significado son los siguientes:

- **r** read - lectura
- **w** write - escritura (en archivos: permiso de modificar, en directorios: permiso de crear archivos en el dir.)
- **x** execution - ejecución

Las nueve posiciones de permisos son en realidad un bit que o esta encendido (mostrado con su letra correspondiente) o esta apagado (mostrado con un guión -), asi que, por ejemplo, permisos como rwxrw-r--, indicaría que los permisos del propietario (rwx) puede leer, escribir y ejecutar el archivo, el grupo (o sea los usuarios que esten en mismo grupo del archivo) (rw-) podrá leer y escribir pero no ejecutar el archivo, y cualquier otro usuario del sistema (r--), solo podrá leer el archivo, ya que los otros dos bits de lectura y ejecucción no se encuentran encendidos o activados.

Permisos en formato numérico octal

La cambinación de valores de cada grupo de los usuarios forma un número octal, el bit x es 2^0 es decir 1, el bit w es 2^1 es decir 2, el bit r es 2^2 es decir 4, tenemos entonces:

- r = 4
- w = 2
- x = 1

La combinación de bits encendidos o apagados en cada grupo da ocho posibles combinaciones de valores, es decir la suma de los bits encendidos:

- - -	= 0	no se tiene ningún permiso
- - x	= 1	solo permiso de ejecución
- w -	= 2	solo permiso de escritura
- w x	= 3	permisos de escritura y ejecución
r - -	= 4	solo permiso de lectura
r - x	= 5	permisos de lectura y ejecución
r w -	= 6	permisos de lectura y escritura
r w x	= 7	todos los permisos establecidos, lectura, escritura y ejecución

Cuando se combinan los permisos del usuario, grupo y otros, se obtienen un número de tres cifras que conforman los permisos del archivo o del directorio. Esto es más fácil visualizarlo con algunos ejemplos:

Permisos	Valor	Descripción
rw-------	600	El propietario tiene permisos de lectura y escritura.
rwx--x--x	711	El propietario lectura, escritura y ejecución, el grupo y otros solo ejecución.
rwxr-xr-x	755	El propietario lectura, escritura y ejecución, el grupo y otros pueden leer y ejecutar el archivo.
rwxrwxrwx	777	El archivo puede ser leido, escrito y ejecutado por quien sea.
r--------	400	Solo el propietario puede leer el archivo, pero ni el mismo puede modificarlo o ejecutarlo y por supuesto ni elgrupo ni otros pueden hacer nada en el.

rw-r-----	640	El usuario propietario puede leer y escribir, el grupo puede leer el archivo y otros no pueden hacer nada.

Estableciendo los permisos con el comando chmod

Habiendo entendido lo anterior, es ahora fácil cambiar los permisos de cualquier archivo o directorio, usando el comando **chmod** (change mode), cuya sintaxis es la siguiente:

chmod [opciones] permisos archivo[s], algunos ejemplos:

```
$> chmod 755 reporte1
$> chmod 511 respaldo.sh
$> chmod 700 julio*
$> chmod 644 *
```

Los ejemplos anterior establecen los permisos correspondientes que el usuario propietario desea establecer, el tercer ejemplo (chmod 700 julio*) cambiará los permisos a todos los archivos que empiezen con julio (julio01, julio02, julio_respaldo, etc.) debido al caracter '*' que es parte de las expresiones regulares que el shell acepta, e indica lo que sea. El último ejemplo por lo tanto cambiará los permisos a los archivos dentro del directorio actual.

Una opción común cuando se desea cambiar todo un árbol de directorios, es decir, varios directorios anidados y sus archivos correspondientes, es usar la opción *-R*, de recursividad:

```
$> chmod -R 755 respaldos/*
```

Esto cambiará los permisos a 755 (rwxr-xr-x) del directorio respaldos y de todos los subdirectorios y archivos que estén contenidos dentro de este.

Estableciendo permisos en modo simbólico

Otra manera popular de establecer los permisos de un archivo o directorio es a través de identificadores del bit (r,w, o x) de los permisos, como ya se vió anteriormente, pero ahora identificando además lo siguiente:

- al usuario con la letra **u**
- al grupo con la letra **g**
- a otros usuarios con la letra **o**
- y cuando nos referimos a todos (usuario, grupo, otros) con la letra **a** (all, todos en inglés)
- el signo **+** para establecer el permiso
- el signo **-** para eliminar o quitar el permiso

La sintaxis es muy simple chmod augo[+|-]rwx[,...] archivo[s], asi por ejemplo, si queremos que otros tengan permiso de escritura sería **chmod o+w archivo**, todos los usuarios con permisos de ejecución **chmod a+x archivo**.

En este modo de establecer permisos, solo hay que tomar en cuenta que partiendo de los permisos ya establecidos se agregan o se quitan a los ya existentes.Veámoslo con ejemplos su manera de trabajar:

Actual	chmod	Resultado	Descripción
rw-------	a+x	rwx--x--x	Agregar a todos (all) permisos de escritura.
rwx--x--x	go-x	rwx------	Se eliminan permiso de ejecución para grupo y otros.

rwxr-xr-x	u-x,go-r	rw---x--x	Al usuario se le quita ejecución, al grupo y otros se le quita lectura.
rwxrwxrwx	u-x,go-rwx	rw-------	Al usuario se le elimina ejecución, al grupo y otros se eliminan todos los permisos.
r--------	a+r,u+w	rw-r--r--	A todos se les agrega lectura, al usuario se le agrega escritura.
rw-r-----	u-rw,g+w,o+x	---rw---x	Al usuario se le eliminan lectura y escritura, al grupo se le agrega lectura y otros se le agrega ejecución.

Cambiando propietario y grupo

Volviendo a mostrar el listado al inicio de este artículo:

```
$> ls -l
-rwxrwxr--  1 sergio ventas    9090 sep  9 14:10 presentacion
-rw-rw-r--  1 sergio sergio 2825990 sep  7 16:36 reporte1
drwxr-xr-x  2 sergio sergio    4096 ago 27 11:41 videos
```

Vemos en la tercera y cuarta columna al usuario propietario del archivo y al grupo al que pertence, es posible cambiar estos valores a través de los comandos **chown** (change owner, cambiar propietario) y **chgrp** (change group, cambiar grupo). La sintaxis es muy sencilla: **chown usuario archivo[s]** y **chgrp grupo archivo[s]**. Además al igual que con **chmod**, también es posible utilizar la opción *-R* para recursividad.

```
#> ls -l presentacion
-rwxrwxr--  1 sergio ventas    9090 sep  9 14:10 presentacion
#> chown juan presentacion
#> ls -l presentacion
-rwxrwxr--  1  juan ventas    9090 sep  9 14:10 presentacion
#> chgrp gerentes presentacion
#> ls -l presentacion
-rwxrwxr--  1  juan gerentes  9090 sep  9 14:10 presentacion
```

Solo el usuario root puede cambiar usuarios y grupos a su voluntad sobre cualquier usuario, queda claro que habiendo ingresado al sistema como usuario normal, solo podrá hacer cambios de grupos, y eso solo a los que pertenezca.

Una manera rápida para el usuario root de cambiar usuario y grupo al mismo tiempo, es con el mismo comando **chown** de la siguiente manera:

```
#> chown juan.gerentes presentacion  (o en vez de punto, con : puntos)
#> chown juan:gerentes presentacion
```

Así, cambiará el usuario.grupo en una sola instrucción.

Bits SUID, SGID y de persitencia (sticky bit)

Aún hay otro tipo de permisos que hay que considerar. Se trata del bit de permisos SUID (Set User ID), el bit de permisos SGID (Set Group ID) y el bit de permisos de persistencia (sticky bit). Para entender los dos primeros el SUID y el SGID veamos los permisos para un comando de uso común a todos los usuarios, que es el comando **passwd**, que como se sabe sirve para cambiar la contraseña del usuario, y puede ser invocado por cualquier

usuario para cambiar su propia contraseña, si vemos sus permisos observaremos un nuevo tipo de permiso:

```
#> ls -l /usr/bin/passwd
-r-s--x--x 1 root root 21944 feb 12  2006 /usr/bin/passwd
```

SUID

En vez de la 'x' en el grupo del usuario encontramos ahora una 's' (suid). **passwd** es un comando propiedad de root, pero sin embargo debe de poder ser ejecutado por otros usuarios, no solo por root. Es aqui donde interviene el bit SUID, donde al activarlo obliga al archivo ejecutable binario a ejecutarse como si lo hubiera lanzado el usuario propietario y no realmente quien lo lanzó o ejecutó. Es decir, es poder invocar un comando propiedad de otro usuario (generalmente de root) como si uno fuera el propietario.

SGID

El bit SGID funciona exactamente igual que el anterior solo que aplica al grupo del archivo. Es decir si el usuario pertenece al grupo 'ventas' y existe un binario llamado 'reporte' que su grupo es 'ventas' y tiene el bit SGID activado, entonces el usuario que pertenezca al grupo 'ventas' podrá ejecutarlo. También se muestra como una 's' en vez del bit 'x' en los permisos del grupo.

STICKY BIT (Bit de persistencia)

Este bit se aplica para directorios como en el caso de /tmp y se indica con una 't':

```
#> ls -ld /tmp
drwxrwxrwt 24 root root 4096 sep 25 18:14 /tmp
```

Puede apreciarse la 't' en vez de la 'x' en los permisos de otros. Lo que hace el bit de persistencia en directorios compartidos por varios usuarios, es que el sólo el propietario del archivo pueda eliminarlo del directorio. Es decir cualquier otro usuario va a poder leer el contenido de un archivo o ejecutarlo si fuera un binario, pero sólo el propietario original podrá eliminarlo o modificarlo. Si no se tuviera el sticky bit activado, entonces en estas carpetas públicas, cualquiera podría eliminar o modificar los archivos de cualquier otro usuario.

Estableciendo los permisos especiales

Para cambiar este tipo de bit se utiliza el mismo comando **chmod** pero agregando un número octal (1 al 7) extra al principio de los permisos, ejemplo:

```
#> ls -l /usr/prog
-r-x--x--x 24 root root 4096 sep 25 18:14 prog
#>chmod 4511 /usr/prog
#> ls -l /usr/prog
-r-s--x--x 24 root root 4096 sep 25 18:14 prog
```

Nótese que el valor extra es el '4' y los demás permisos se dejan como se quieran los permisos para el archivo. Es decir, los permisos originales en este ejemplo eran 511 (r-x--x--x), y al cambiarlos a 4511, se cambió el bit SUID reemplazando el bit 'x' del usuario por 's'.

Los posibles valores serían los siguientes:

- - - - - - - - -	= 0	Predeterminado, sin permisos especiales. No se requiere indicar.
- - - - - - - - t	= 1	Bit de persistencia, sticky bit
- - - - - s - - -	= 2	Bit sgid de grupo
- - - - - s - - t	= 3	Bit sgid y sticky

- - s - - - - - -	= 4	Bit suid
- - s - - - - - t	= 5	Bit suid y sticky
- - s - - s - - -	= 6	Bit suid y sgid
- - s - - s - - t	= 7	Bit suid, sgid y sticky

MUY IMPORTANTE: Algo sumamente delicado y que se tiene que tomar muy en cuenta es lo que decidas establecer con permisos de bit SUID y SGID, ya que recuerda que al establecerlos de esta manera, cualquier usuario podrá ejecutarlos como si fueran el propietario original de ese programa. Y esto puede tener consecuencias de seguridad severas en tu sistema. Siempre considera y reconsidera si conviene que un usuario normal ejecute aplicaciones propias de root a través del cambio de bits SUID o SGID. Mejores alternativas pueden ser los comandos **sudo** y **su**, en este tip (index.php?cont=info__tips_016) de LinuxTotal.com.mx encuentras una manera de identificar archivos y ejecutables con estos bits establecidos.

Permisos preestablecidos con `umask`

El comando **umask** establece la máscara de permisos de directorio y de archivos. Es decir los nuevos directorios y archivos que se crean obtienen el valor de los permisos a partir de los valores de **umask**.

```
$> umask
0002
(o en formato simbólico con la opción -S)
$> umask -S
u=rwx,g=rwx,o=rx
```

Lo anterior indica que un directorio y archivos ejecutables se crearán con los permisos 775 y los archivos comunes con los permisos 664. Esto se logra restando de 777 el valor de umask (777-002) y (666-002) respectivamente. El primer valor de umask corresponde para valores de Sticky bit, GUID o SUID, que por default es 0.

```
$> umask
0002
(Creamos un archivo y según la máscara debemos de tener 666-002=664 o rw-rw-r--)
$> touch archivo
$> ll archivo
-rw-rw-r-- 1 sergio sergio 0 sep 25 20:14 archivo
(Ahora creamos un directorio y según la máscara debemos de tener 777-002=775 o rwxrwxr-x)
$> mkdir dir
$> ls -ld dir
drwxrwxr-x 2 sergio sergio 4096 sep 25 20:20 dir
```

Para establecer el valor de la máscara, simplemente se usa el mismo comando **umask** seguido del valor de máscara que se desee:

```
$> umask 0022
```

Para dejarlo fijo en la sesión, entonces conviene agregarlo a .bash_profile o .bash_rc de nuestro directorio de inicio.

Si te gustó este artículo quizás también encuentres interesantes este otro artículo relacionado:

- Administración de usuarios (index.php?cont=info_admon_008)

¿Requieres de una instalación o configuración de Linux o sus servicios?
¿Un desarrollo WEB empresarial a la medida?
¿Un curso o capacitación a la medida?
Revisa el sitio de SERVICIOS (index.php?cont=servicios) de LinuxTotal

---

info@linuxtotal.com.mx · linuxtotal.com.mx@gmail.com