educación IT
CODE YOUR FUTURE

Introducción a Linux

Módulo 4

Modos de acceso y dueños de archivos

Modos de acceso y Propietarios de archivos

Por seguridad, Linux asocia cada archivo con un grupo de propiedades, conocidas como **permisos. Son atributos especiales que controlan la capacidad de los usuarios para realizar varias operaciones sobre archivos**.

¿Qué son los permisos?

Unix y los sistemas estilo Unix, como Linux, **son inherentemente sistemas operativos multiusuario**, y tienen la necesidad de **proporcionar un control** para que un usuario malicioso no pueda interferir con archivos de otros usuarios. Por ejemplo, un usuario puede tener información sensible a la que otro usuario puede querer acceder. **Sin permisos de archivos, el otro usuario puede acceder a ellos, pero con los permisos debidamente configurados, el usuario no podrá leerlos**.

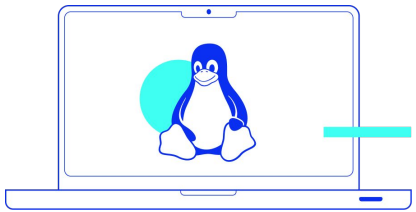
Cada archivo, directorio, o dispositivo tiene un grupo de permisos asociados a él. Estos permisos son visibles con el comando **ls** y la opción **-l**.

Por ejemplo:

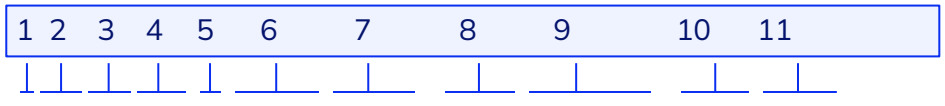
```
# ls -l

-rw-r-r- 1 root root 1059 Nov 18 13:23 /etc/passwd
```

El **bloque de los permisos** es lo primero que aparece en la línea:

```
-rw-r--r--
```



Se parte esta cadena en **secciones:**




```
drwxr-xr-x 2 zeque zeque  4096 May 22   2013 temp

-rw-rw---- 1 zeque zeque   542 Jun 15  09:38 test.txt
```

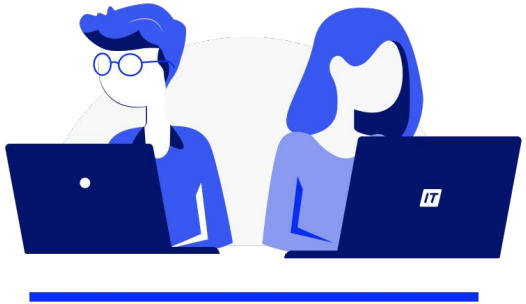
En la tabla de la siguiente diapositiva veremos en detalle estas secciones.

Columna	Descripción	Información adicional
1	Tipo.	**-** (Archivo) **d** (Directorio) **l** (link simbólico) **b** (dispositivo de bloque) **c** (dispositivo de carácter) **s** (socket)
2	Permisos Usuario.	Asociado al usuario que lo creó.
3	Permisos de Grupo.	Asociado al grupo del usuario.
4	Permisos Otros.	Cualquier otro que no sea el usuario o no esté en el grupo.
5	Links asociados o directorios contenidos.	
6	Usuario propietario.	
7	Grupo.	
8	Tamaño.	
9	Fecha.	
10	Año u Hora.	
11	Nombre de archivo/directorio.	

En cada bloque de tres caracteres, **“r”, “w”,** y **“x”** significan que el archivo puede ser **leído *(Read),* escrito/modificado *(Write)***, o **ejecutado *(eXecute).*** Un **guión (-)** significa que **el permiso no está puesto** (por ejemplo, el archivo *test.txt* puede ser leído y modificado por el propietario, pero no se puede ejecutar). Así, por ejemplo **r-x** significa que los permisos de lectura y ejecución están puestos, pero no el de escritura.

Como regla, es más fácil utilizar la sigla **U.G.O (Usuarios, Grupos y Otros)**, a cada grupo de tres caracteres le pertenecen los permisos correspondientes.

Cuando hablamos de *otros* es todo aquel que no sea el usuario propietario, ni se encuentre en el grupo definido del archivo.



Otra forma de ver los permisos de manera más detallada es la siguiente:

```
$ stat /bin/bash
File: `/bin/bash'
Size: 960544      Blocks: 1880       IO Block: 4096   regular file
Device: fd02h/64770d    Inode: 66159       Links: 1
Access: (0755/-rwxr-xr-x)  Uid: (    0/    root)   Gid: (    0/
root)
Access: 2011-11-21 05:01:01.752935695 -0300
Modify: 2011-06-22 10:49:05.000000000 -0300
Change: 2011-10-04 03:12:19.933687854 -0300
Birth: -
```

De esta forma, estamos viendo los atributos generales que posee un *inodo*, que es el objeto que representa los datos dentro del sistema de archivos.

En síntesis, los componentes informativos de un fichero están compuestos por un campo y tres secciones, que son las siguientes:

Permiso	Descripción	Información adicional
`r`	Permite la lectura *(**R**ead)* de un archivo. Éste es el único permiso necesario para copiar un archivo.	Si se aplica a un directorio, se pueden leer o ver sus archivos.
`w`	Permite escribir *(**W**rite)* en un archivo. Con él se pueden cambiar, modificar o sobreescribir los contenidos del archivo.	Cuando se aplica en un directorio, este permite borrar y mover archivos incluso si no se tiene el permiso de escritura específico sobre el archivo individual.
`x`	Permite ejecutar *(e**X**ecute)* un archivo.	Aplicado a un directorio, este permite acceder a él. Cuando se aplica a un conjunto con permisos de lectura dentro de un directorio, este permite buscar dentro de ese directorio.
`-`	Permiso sin asignar. Por ejemplo, `r-x` indica que ese usuario puede leer y ejecutar, pero no escribir.	

Cambiar los permisos

Comando *chmod*

Muchos permisos se configuran automáticamente cuando se crea un archivo. Estos permisos están basados en el usuario, las preferencias administrativas y los datos.

chmod *(change mode)* permite cambiar los permisos de acceso de un archivo o directorio.

La sintaxis básica de esta utilidad es la siguiente:

```
# chmod [modificadores] [opciones] [archivo/directorio]
```

Los *modificadores* son los permisos antes vistos.

Se pueden utilizar muchas opciones con el comando **`chmod`**. Las usadas más frecuentemente son las siguientes:

- **`-v`** Lista los archivos y directorios a los que se les va aplicando el comando, a medida que ese comando se ejecuta.

- **`-R`** Aplica el comando `chmod` en forma recursiva a todos los archivos y subdirectorios.

Cambiar permisos

A cada archivo se le asigna automáticamente un conjunto de permisos, que se puede definir con el comando **`umask`**. Para cambiar estos permisos predeterminados, se usa el comando **`chmod`**, que **modifica las posibilidades de lectura, escritura, y ejecución** para el usuario, grupo y otros.

La forma de cambiar los permisos es con letras o números. Las explicaremos a continuación.

Con letras, se puede utilizar **`u, g, o, a`** (usuario, grupo, otros, todos los permisos).

Ejemplo:

Agregar ejecución y quitar escritura al usuario, agregar lectura y escritura a otros.

```
chmod u+x,u-w,o+rw hola
```

Se define para usuario, grupo y otros: lectura y escritura en el archivo *test.txt*

```
chmod ugo=rw test.txt
```

Se agrega ejecución para usuario, grupo y otros.

```
chmod a=x test
```

Se definen lectura y escritura para el usuario, para el directorio, etc., de manera recursiva.

```
chmod  -R u=rw etc
```

De esta forma, lo que se puede hacer es **“+” agregar un permiso, “-” quitar un permiso, o “=” igualar un permiso** a todos esos valores.

También se pueden **definir los permisos de manera numérica.**

Permiso	Descripción
4	Lectura.
2	Escritura.
1	Ejecución.
0	Ningún permiso.

En la siguiente diapositiva se mostrarán algunos ejemplos.

Para definir permisos de manera numérica, se deben ingresar tres números: **el primero será para los permisos de Usuario, el segundo para el Grupo y el tercero para Otros**.

Definir lectura para el usuario y el grupo:

```
# chmod 440 test.txt
```

Cuando se quiere definir más de un permiso se combinan los valores.

Definir todo para el usuario, lectura y ejecución para el grupo, y lectura para otros:

```
# chmod 754 test.txt
```

En este ejemplo el **7** es el resultado de la suma de **4** (lectura), **2** (escritura) y **1** (ejecución). Define la totalidad de los permisos para el usuario propietario. Luego para el grupo, tiene el valor **5** de la suma de los valores **4** (lectura) y **1** (ejecución). Por último, para otros solo el permiso de lectura **4.**

Cambiar el propietario

Comando *chown*

El comando **chown cambia el propietario** de cada archivo, usuario o grupo, pero solo si se le especifica un nombre de usuario (o UID numérico); entonces, ese usuario se convierte en el propietario de cada archivo dado, y el grupo al que pertenece, no cambia.

Si al nombre de usuario lo sigue un signo de dos puntos y un nombre de grupo o GID numérico, sin espacios entre ellos, entonces, también se **cambia el grupo al que pertenece** cada archivo.

La sintaxis básica de esta utilidad es la siguiente:

```
# chown [opciones] usuario:grupo fichero
```

Se pueden utilizar opciones con el comando **chown**. Las utilizadas con más frecuencia son las que se muestran en la tabla de la siguiente slide.

Opciones más utilizadas para el comando **chown**.

Opciones	Descripción
`-c`	Muestra un mensaje donde menciona solamente aquellos ficheros cuyo propietario cambia realmente.
`-f`	No muestra mensajes de error sobre los ficheros a cambiarse.
`-R`	Cambia recursivamente el propietario de directorios y sus contenidos.
`-v`	Describe la acción efectuada (o no) para cada fichero de forma interactiva)

Ejemplo:

Asignar el propietario `educacionit` al directorio **`/directorio1`** de manera recursiva (a todo su contenido):

```
# chown -R educacionit  /directorio1
```

Cambiar el grupo propietario

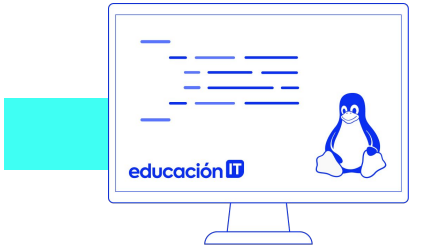
Comando *chgrp*

El comando **chgrp** se usa para **cambiar solamente el grupo propietario de un archivo o directorio.** El comando busca en el fichero **/etc/group** para confirmar la existencia del grupo especificado antes de cambiar los permisos.

La sintaxis básica de esta utilidad es la siguiente:

```
# chgrp [opciones] grupo fichero
```

Se pueden utilizar varias opciones con el comando **chgrp**. Las usadas más frecuentemente son las que se muestran en la tabla de la pantalla a continuación.



Opciones más utilizadas para el comando **`chgrp`**.

Opciones	Descripción
`-c`	Muestra un mensaje donde menciona solamente aquellos ficheros cuyo propietario cambia realmente.
`-h`	Actúa sobre enlaces simbólicos propiamente en vez de sobre lo que apuntan. Disponible solamente si el sistema proporciona la primitiva **`lchown`.**
`-f`	No muestra mensajes de error sobre ficheros cuyo grupo no pueda cambiarse.
`-R`	Cambia recursivamente el grupo al que pertenecen directorios y sus contenidos. Continúa incluso si se encuentran errores.
`-v`	Describe la acción efectuada (o no) para cada fichero de forma interactiva.

El **usuario root** puede efectuar los cambios al grupo que desee; para que un usuario pueda hacerlo, debe ser el propietario del archivo y pertenecer al grupo donde está efectuando el cambio.

El comando *umask*

Este comando usa para **visualizar/configurar la máscara de permisos** para la creación de archivos y directorios. Algunos ejemplos:

```
$ umask -p
umask 0002
```

```
$ umask -S
u=rwx,g=rwx,o=rx
```

El **umask** es la inversa del **chmod.** En el caso de umask el valor 022 es el equivalente a 755 en chmod. **El umask lo podemos calcular como la cantidad de números que faltan para llegar a 7.**

En el caso de definir un umask 000 al crear un archivo el permiso máximo chmod será 666, en cambio para un directorio será 777.

En el resultado del comando se ven cuatro cifras. La primera es la que va a definir el SUID, SGID o *Stickybit.* Lo veremos en detalle a continuación.

educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!