Los permisos de Linux

En Linux, los permisos controlan quién puede leer, escribir o ejecutar un archivo o directorio. Cada archivo y carpeta tiene un conjunto de permisos asociados que determinan qué acciones pueden realizar diferentes usuarios.

El usuario administrador (root) al tener el control completo del sistema, puede realizar estas operaciones sobre cualquier fichero o directorio de cualquier usuario (esta es una de las maneras de evitar que un usuario pueda entrar en su directorio personal).

Los permisos en Linux se clasifican en dos categorias (permisos normales y especiales). Los permisos normales se subdividen en tres grupos:

1. Permisos del propietario
2. Permisos para el grupo
3. Los permisos para el resto de usuarios del sistema.

Permisos Normales

En Linux, cada usuario tiene un nombre de conexión único y pertenece a uno o varios grupos de usuarios. El propietario de un archivo o directorio puede definir qué permisos se activan o deshabilitan, configurando así el acceso según sus necesidades.

Para entender mejor el concepto de permisos, podemos usar el comando `ls -l` en la terminal, que muestra los detalles de los archivos y directorios. Por ejemplo:

```
# ls –l
total 0
-rw-r--r-- 1 root root  0 Oct  1 23:08 a.txt
drwxr-xr-x 2 root root  6 Oct  1 23:09 fotos
lrwxrwxrwx 1 root root 17 Oct  1 23:09 messages -> /var/log/messages
```

En el ejemplo anterior, los 10 caracteres que aparecen al inicio de cada línea, como `-rw-r–r–` en el caso del archivo `a.txt`, representan los permisos. Estos caracteres se interpretan de la siguiente manera:

- El primer carácter muestra el tipo: fichero normal(`-`), directorio(`d`), enlace(`l`), tubería(`p`).
- El segundo campo (`rw-`) indica los permisos que tiene el propietario del archivo.
- El tercer campo(`r–`) indica los permisos que tiene el grupo sobre el archivo.
- El 4to campo (`r–`) indica los permisos del resto de usuarios.
- El 5to campo (`n`) indica la cantidad de archivos/directorios que contiene.
- El 6to campo indica (`usuario`) indica el nombre del usuario al que pertenece el archivo o directorio.
- El 7mo campo indica el nombre del grupo al que pertenece el archivo.
- El 8vo campo indica la fecha de creación
- El 9no indica el nombre del archivo.

Para modificar los permisos de acceso, se utiliza el comando `chmod`, que permite gestionarlos mediante dos notaciones principales: numérica en formato octal y simbólica.

Notación Octal

Consiste en el uso de números enteros que van del 0 al 7, teniendo en cuenta la siguiente regla:

- 0 – Ningún permiso
- 1 – Dar Permiso de Ejecución.
- 2 – Dar permiso de escritura.
- 4 – Dar permiso de Lectura

Para formar un permiso se suma los anteriores números, agrupando por: usuario, grupo y otros usuarios. Ejemplo:

```
# touch archivo1
# chmod 750 archivo1
```

En el ejemplo anterior, el valor 750 representa los permisos de acceso en formato octal, distribuidos de la siguiente manera:

- 7 (4+2+1): El propietario tiene permisos de lectura ®, escritura (w) y ejecución (x).
- 5 (4+1): El grupo tiene permisos de lectura ® y ejecución (x), pero no de escritura.
- 0: Otros usuarios no tienen ningún permiso.

Esta configuración garantiza que solo el propietario tenga control total sobre el archivo, mientras que el grupo tiene acceso limitado y otros usuarios no pueden interactuar con él.

Notación Simbólica

En esta notación se utilizan los símbolos u, g, o para representar a los usuarios, grupos y otros, seguido de los modificadores:

- = Asigna un permiso de forma de absoluta.
- + Añadir un permiso.
- - Quitar un permiso.
- r Permiso de lectura.
- w permiso de escritura.
- x permiso de ejecución.

La sintaxis de un permiso usando la notación simbólica sería de la siguiente forma:

```
# chmod u=rw,g=rw,o=r archivo1.html
```

para el ejemplo anterior el propietario puede leer y escribir en el archivo. Los miembros del grupo pueden leer y escribir en el archivo. Otros usuarios solo pueden leer el archivo

```
# chmod u+x archivo1
```

El permiso indica la ejecución al propietario (usuario) del archivo archivo1, mientras mantiene intactos los permisos existentes para el grupo y otros usuarios

```
# chmod u-x archivo1
```

Después de ejecutar el anterior comando, el propietario ya no podrá ejecutar el `archivo1`