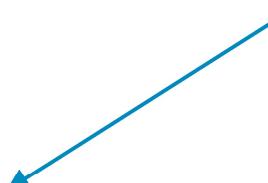


## Acceso a recursos y permisos locales

Los archivos son recursos del sistema operativo y, por tanto, este debe disponer de herramientas para decidir **qué usuarios y grupos pueden acceder a ellos y qué operaciones realizar sobre ellos**.

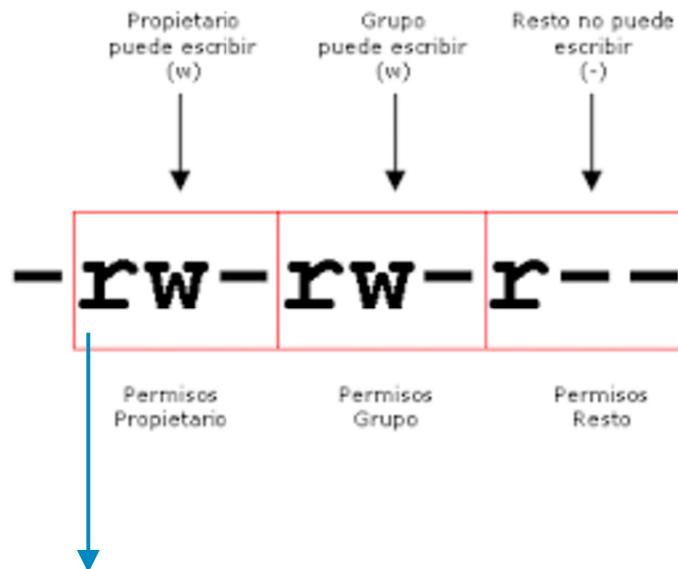
El i-nodo de los archivos almacena el usuario propietario del archivo (UID), su grupo propietario(GID) y la máscara de permisos con este tipo de estructura:

**-rwxrwxrwx**



El primer carácter indica (-) archivo normal, (d) directorio, (l) enlace

# Acceso a recursos y permisos locales



Cada grupo de tres bits representa un **(r) lectura (w) escritura (x) ejecución (-) Ningún permiso**

- **Los permisos de usuario (propietario)** son los privilegios que tiene éste sobre un archivo. (primer grupo de tres bits)
- *Los permisos de grupo* son los que tienen sobre él los usuarios que pertenecen al grupo del archivo. (segundo grupo de tres bits)
- **Otros (resto)** ni son propietarios ni pertenecen al grupo (tercer grupo de tres bits)

# Acceso a recursos y permisos locales

Los permisos se comportan de distinta manera sobre archivos y carpetas.

Permisos	Archivos	Carpetas
<b>Permiso de lectura (r)</b>	Puede ser leído o visualizado	Se puede visualizar su contenido, mostrando los archivos y carpetas que contenga
<b>Permiso de escritura (w)</b>	Puede modificar su contenido, sus permisos, el propietario y el grupo	Permite modificar el contenido, creando o eliminando archivos o carpetas en ella
<b>Permiso de ejecución (x)</b>	Permite ejecutarlo	Permite acceder a ella

En principio, solo los privilegios se asocian al propietario, a los usuarios del grupo del fichero y al resto de usuarios del sistema. Si se desea aplicar permisos especiales sobre un archivo o carpeta a otro conjunto de usuarios, no se puede establecer, a no ser que se empleen listas de control de acceso (ACL).

# Acceso a recursos y permisos locales

Existen unos bits “raros” que se asocian a unos nodos especiales:

**Set-uid.** Este bit hace referencia a la propiedad que un archivo o carpeta puede adoptar cuando un usuario (que no sea el propietario) lo ejecuta, ya que tomaría el UID del propietario. Si el propietario es el superusuario, entonces otro usuario podría ejecutar tal archivo como si fuera el root. Esta propiedad solo es posible durante la ejecución, evitando problemas de seguridad. El bit set-uid no tiene efecto si se aplica sobre directorios o si el propietario no tiene permiso de ejecución. Cuando está activo el set-uid, se simboliza con “**s**” en lugar de “**x**” en los bits de la máscara de servicios del propietario. Si no es ejecutable el archivo y dispone de set-uid, se representará por “**S**”.

# Acceso a recursos y permisos locales

**Set.gid** Presenta características diferentes en archivos o directorios.

- Cuando está activo en archivos su uso es similar a set-uid. Permite que, al ejecutarse un fichero, se realice con el GID de su grupo y, por tanto, con los privilegios del GID del grupo durante la ejecución. Esto hace que los usuarios de un grupo puedan trabajar con los ficheros de ese mismo grupo.
- Cuando está activo en directorios, los archivos y subdirectorios creados en su interior serán obligados a pertenecer al grupo del directorio y no al grupo del usuario que lo ha creado.

# Acceso a recursos y permisos locales

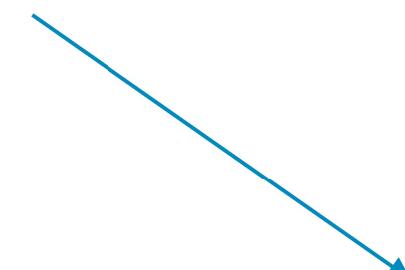
**Sticky-bit** o bit de permanencia. Este bit permite establecer características especiales sobre directorios, ya que **sobre archivos no tiene efecto**. Permite en los directorios que solo el propietario del archivo creado dentro del directorio con sticky-bit activo o el propietario del directorio con sticky-bit activo pueda eliminar o modificar su contenido. Si suponemos que un usuario (que no sea el propietario del directorio con sticky-bit activo) dispone de permisos de lectura, escritura y ejecución, no podría eliminar o modificar su contenido. Cuando está activo el sticky-bit , se simboliza con “t” en lugar de “x”, en los bits de la máscara de permisos del resto de usuarios. Si no es ejecutable el archivo o el directorio y sí dispone de sticky-bit, se representará por “T”. No obstante, no tiene efecto si el resto de usuarios no tiene permiso de ejecución.

# Modificación de permisos

Para modificar los permisos de un archivo utilizamos el comando **chmod** (change mode) siendo propietarios o superusuario.

**chmod [-R] permisos archivos**

**-R es para que sea recursivo.**



Permisos lo podemos especificar en modo  
**octal o simbólico.**

# Modificación de permisos

## OCTAL

0	000	---
1	001	--x
2	010	-w-
3	011	-wx
4	100	r--
5	101	r-x
6	110	r w-
7	111	rwx

activar o desactivar cada bit sobre los permisos en binario mediante un 1 o un 0 respectivamente

Para **rw-rw-r--** deberíamos aplicar **111110100** sería **764**

los modos especiales:

- **sticky-bit**: sumar 1000 en octal al resultado anterior
- **set-gid**: sumar 2000 en octal al resultado anterior
- **set-uid**: sumar 4000 en octal al resultado anterior.

**chmod 5776 prueba.txt**

quedando una máscara de permisos: **rwsrwxtw T**

# Modificación de permisos

## SIMBÓLICO

**chmod modo\_simbolico[,modo\_simbolico] archivos**

El modo simbólico se puede repetir más de una vez, separándose por comas

[u] [g] [o] {+ | = | -} [r] [w] [x] [s] [t]

- |   |   |
|---|---|
| <ul style="list-style-type: none"><li>✓ Destinatarios de los permisos para modificar:<ul style="list-style-type: none"><li>✓ "u": propietarios</li><li>✓ "g": grupo</li><li>✓ "o": otros (el resto de usuarios)</li><li>✓ "a": propietario, grupo y otros</li></ul></li><br/><li>✓ Tipo de modificación<ul style="list-style-type: none"><li>✓ "+": se añaden al valor actualización</li><li>✓ "=": establece los permisos especificados y anula al resto</li><li>✓ "-": se quitan al valor actual.</li></ul></li></ul> | <ul style="list-style-type: none"><li>✓ Permisos:<ul style="list-style-type: none"><li>✓ "r": lectura</li><li>✓ "w": escritura</li><li>✓ "x": ejecución</li><li>✓ "s" set uid o set guid dependiendo de su aplicación al propietario o grupo.</li><li>✓ "t": sticky-bit</li></ul></li></ul> |
|---|---|

deshabilitar para el grupo y otros el permiso de lectura sería:

**chmod go - r prueba.txt**

# Permisos por defecto

Los permisos originales de un **archivo** y un **directorio** son **0666** y **0777** en octal, respectivamente. Sobre estos permisos se aplica una máscara de permisos, dando como resultado los permisos que posee un archivo o directorio cuando son creados. Su finalidad es aportar seguridad a los archivos y directorios al crearlos, siendo el valor, por defecto, de la máscara de permisos **0022** o **0002** (**depende de la distribución de Linux y el usuario**)

Para ver o modificar la máscara de permisos, empleamos el comando **umask (user mask)**

```
umask [mascara]
```

mascara debe ser el valor de la máscara de permisos en octal (aunque se puede usar la opción -S para usar notación simbólica)

# Permisos por defecto

El procedimiento de aplicación de la máscara de permisos es el siguiente:

- Convertimos la máscara de permisos de octal a binario
- Aplicamos el operador **NOT** sobre la máscara en binario
- Realizamos la operación **AND** entre los permisos originales de archivos o directorios de la máscara.

*el resultado no activa los bits de ejecución para los archivos, puesto que los permisos originales de estos (**0666**) no tienen habilitado el bit de ejecución.*

La modificación de la máscara de permisos mediante umask es temporal y solo afecta al proceso asociado al shell o a los procesos generados por este.

podemos establecer el valor de la máscara de permisos mediante los archivos de configuración globales, como “/etc/profile” (*afectan a todos los usuarios y solo se podría hacer con permisos de superusuario*), o locales como ~.bashrc (solo afecta a la cuenta de usuario)