Práctica 3 Permisos

Sistemas Operativos

Grado en Ingeniería Informática
Universidad de Cádiz

Curso 12-13

Contenido

Objetivos I

Al finalizar esta práctica, el estudiante deberá ser capaz de:

1. Explicar los conceptos de UID, GID, grupo principal y grupo secundario.
2. Identificar los diferentes permisos que pueden asignarse a los ficheros y directorios en un sistema GNU/Linux.
3. Describir qué permisos debe tener activados un fichero o un directorio para poder realizar determinadas operaciones sobre ellos.
4. Dado un fichero, activar o desactivar un determinado conjunto de permisos utilizando la notación octal y la simbólica.
5. Establecer la máscara de permisos que deben tener los ficheros y directorios por omisión cuando se creen.

Objetivos II

Al finalizar esta práctica, el estudiante deberá ser capaz de:

6. Determinar qué permisos debería tener un determinado fichero y el directorio en el que se encuentra catalogado para que se garantice el grado de seguridad apropiado.
7. Explicar la utilidad de los permisos SUID, SGID y el bit *sticky*
8. Reconocer cuando están activados los permisos SUID, SGID y el bit *sticky* para un fichero
9. Dado un conjunto de ficheros y directorios y los permisos que estos poseen, deberá poder indicar si es posible realizar una serie de operaciones sobre estos, indicando las razones

Conceptos

- Permisos Todo fichero lleva asociado un conjunto de permisos que definen quién puede acceder a él y qué operaciones puede realizar.
- Superusuario (root) Usuario especial que puede hacer cualquier operación sobre un fichero, independientemente de sus permisos. También puede cambiar los permisos de cualquier fichero.
- UID Número que identifica a cada usuario dentro del sistema.
- GID Número que identifica a cada grupo dentro del sistema
- Grupo principal Todo usuario pertenece a un grupo principal (su GID aparece en el fichero /etc/passwd).
- Grupos secundarios Los usuarios pueden pertenecer a uno o varios grupos secundarios (/etc/group).

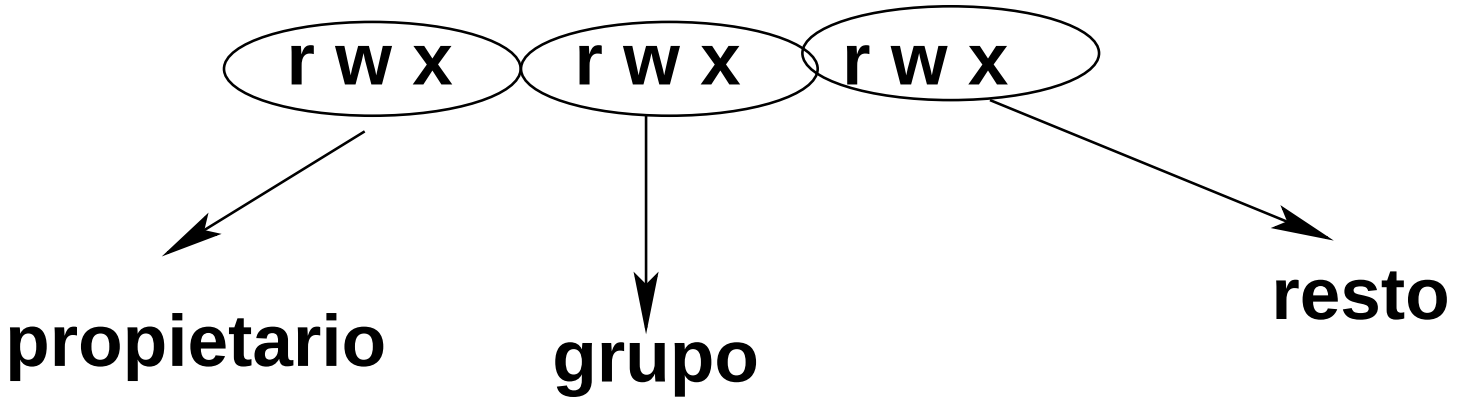
Los ficheros /etc/passwd y /etc/group

- /etc/passwd Fichero de usuarios. Para cada usuario del sistema mantiene: login_id:contraseña_cod:UID:GID:varios:dir_entrada:shell
- /etc/group Ficheros de grupos. Para cada grupo mantiene: nombre_grupo:x:GID:lista_usuarios_grupo_secundario

Permisos de ficheros y directorios

- Los ficheros regulares y los directorios pueden llevar asociados 3 tipos de permisos: lectura (r), escritura (w) y ejecución (x).

```
ls -l /etc/passwd
-rw-r--r-- 1 root root .... /etc/passwd
```



Significado de los permisos

Ficheros regulares

- r Nos permite examinar su contenido
- w Nos permite modificar su contenido
- x Nos permite ejecutarlo, si se trata de un programa

Directorios

- r Nos permite ver el contenido del directorio (ls)
- w Nos permite crear nuevos ficheros, borrar ficheros existentes, modificar el identificador de un fichero.
- x Permiso de búsqueda Nos permite buscar en el directorio el número de nodo-i que le corresponde a un fichero a partir de su nombre

Búsqueda de ficheros

Directorio raíz

1	.
1	..
4	bin
7	dev
14	lib
9	etc
6	usr
8	tmp

La búsqueda de /usr produce el nodo−i 6

Nodo−i 6 de /usr

tamaño tiempos modos
132

El nodo−i 6 indica que /usr está en el bloque 132

El bloque 132 es el directorio de /usr

6	.
1	..
19	dick
30	erik
51	jim
26	ast
45	bal

/usr/ast es el nodo−i 26

El nodo−i 26 es de /usr/ast

tamaño tiempos modos
406

El nodo−i 26 indica que /usr/ast es el bloque 406

El bloque 406 es el directorio /usr/ast

26	.
6	..
64	gran
92	libros
60	buzon
81	linux
17	mtp−ii

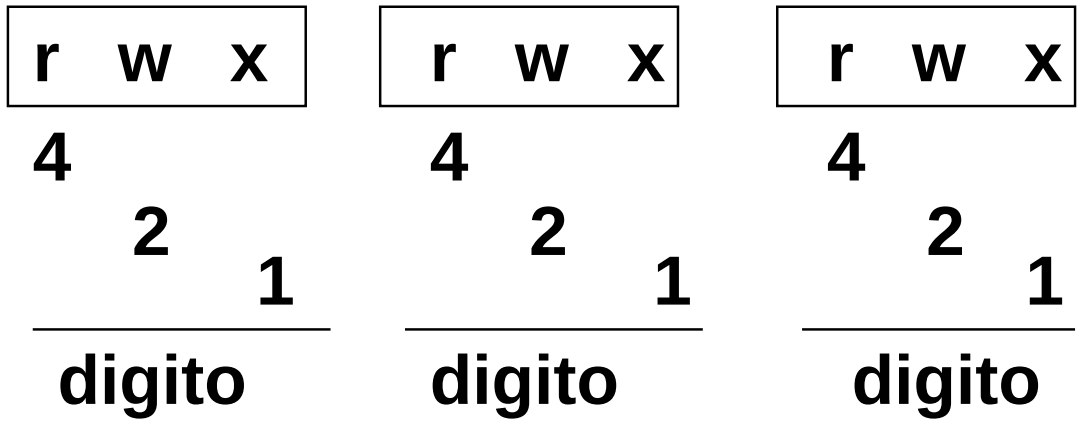
/usr/ast/buzon es el nodo−i 60

Cambio de permisos

- La orden chmod permite cambiar los permisos de un fichero

  chmod *modo fichero* ...

- Sólo puede cambiar los permisos de un fichero el superusuario (`root`) o el propietario del fichero
- El modo se puede especificar de 2 formas:
  - Números octales `chmod 750 miprograma`



  - Descripción simbólica

Cambio de permisos (cont.)

- **Descripción simbólica**

```
chmod ug-x, o+r miprograma
```

[*quién*] *op permiso*

quién	*op*	*permiso*
u (prop.)	+ (añade permiso)	r
g(grupo)	- (quita permiso)	w
o (otros)	= (asigna abs.)	x
a (todos)		s, t

Ejercicios de Ejemplo

1. Suponga que da la orden `ls -l mifichero` y obtiene:

   `-rw--w-r-- user1 grupo1 ... mifichero`

   ¿Qué línea de órdenes debería dar para que el fichero quedara con la máscara de permisos `rw-r---`? El modo debe especificarlo mediante números octales y en forma simbólica.
2. Suponga que da la siguiente secuencia de órdenes. Indique de forma razonada qué permisos tendría el fichero tras la ejecución de cada orden.
   1. chmod 351 listado
   2. chmod u+w listado
   3. chmod g-r listado
   4. chmod o=x listado

La orden umask

- Permite especificar los permisos que tendrán los ficheros y directorios en el momento de su creación

  umask [*modo*]
- El administrador del sistema establece una máscara por omisión para todos los ficheros del sistema.
- Los usuarios pueden establecer su propia máscara.
- El modo se puede dar de forma octal (indica los permisos que no se van a establecer) y en forma simbólica (se indican los permisos que se van a establecer)
- Hay que tener en cuenta que en el caso de los ficheros regulares el permiso de ejecución nunca se activa al crearlo

Ejercicios de ejemplo

1. Supongamos que da la orden `umask 046` ¿Qué máscara de permisos tendrán activados los ficheros ordinarios y directorios que se creen a partir de ese momento?
2. Dados los ficheros:

```
drwxr-xr-x  root  alum  /home/alum
drwxr-xr-x  juan  alum  /home/alum/juan
drwxr--r--  juan  alum  /home/alum/juan/docs
-rw-rw-rw-  juan  alum  /home/alum/juan/docs/apuntes
```

¿Podría el usuario `pepe` perteneciente al grupo `alum` dar las órdenes siguientes? ¿Y el usuario manuel perteneciente al grupo `profs`?

1. `ls ~juan`
2. `ls ~juan/docs`
3. `rm ~juan/docs/apuntes`

Control de acceso de un proceso a un fichero

- Cuando se crea un proceso se le asignan UID_real, UID_efectivo, GID_real y GID_efectivo
- Normalmente:
  - UID_real = UID_efectivo = UID_usuario_ejecuta
  - GID_real = GID_efectivo = GID_usuario_ejecuta
- El sistema determina si un proceso tiene o no acceso a un fichero:

```
si UID_efectivo == UID_del_fichero
entonces
   Acceder al fichero como su propietario
si no si GID_efectivo == GID_del_fichero
entonces Acceder al fichero como miembro de
si no
Acceder al fichero como el resto de usuario
fin si
```

El bit SUID

- Cuando un programa tiene establecido el bit SUID, todos los procesos creados por ese programa tendrán el UID efectivo del propietario del programa y no el del usuario que lo ejecuta
- Esto nos puede interesar cuando el programa necesita acceder a un fichero que no es de nuestra propiedad
- Ejemplo:

```
-rw-r--r--    1 root   root   /etc/passwd
-rwsr-xr-x    1 root   root   /usr/bin/passwd
```

El bit SUID (cont.)

- Es independiente del permiso de ejecución.
- Para establecerlo:

  `chmod u+s` *fichero*

  `chmod 4`*xxx fichero*
- Para eliminarlo: `chmod u-s` *fichero*

El bit SGID

- Se comporta de la misma forma que el SUID, pero afecta al GID_efectivo
- Para establecerlo:

  `chmod g+s` *fichero*

  `chmod 2`*xxx fichero*
- Para eliminarlo: `chmod g-s` *fichero*
- También se puede activar en directorios, pero con un significado diferente que se verá más adelante

Ejercicios de ejemplo

1. Considere la siguiente información:

```
drwxr-x---  ppp alum /home/alum/ppp
drwxr-x---  ppp alum /home/alum/ppp/juegos
-rwsr-x--x  ppp alum /home/alum/ppp/juegos/tetris
-rw-r-----  ppp alum /home/alum/ppp/juegos/puntos
```

El usuario ppp ha creado el juego tetris para que todos los usuarios del grupo alum lo puedan utilizar. Hay que tener en cuenta que el programa tetris intenta escribir la puntuación obtenida en el fichero puntos. ¿Son adecuados los permisos que tienen los ficheros tetris y puntos para que cualquier usuario del grupo alum pueda ejecutarlo sin problemas? Razone su respuesta.

Solución

Los permisos del programa tetris son:

```
-rwsr-x--x  ppp alum /home/alum/ppp/juegos/tetr
```

Un usuario del grupo alum tiene permiso de ejecución en el fichero tetris por lo que en principio podría ejecutarlo. Ahora bien, ¿puede escribir en el fichero puntos? El fichero puntos no tiene permiso de escritura para el grupo, pero ...

El bit *sticky*

- Se representa mediante una t que aparece en el campo de ejecución de otros
- Se aplica a directorios de uso público (tienen todos los permisos activados: rwxrwxrwt)
- Impide que un usuario pueda borrar ficheros que no le pertenecen
- Para activarlo:

  `chmod 1`*xxx directorio*

  `chmod o+t` *directorio*

- Para desactivarlo:

  `chmod o-t` *directorio*

Cambio de propietario y grupo de ficheros

La orden chgrp

- Permite cambiar el grupo al que pertenece un fichero
- `chgrp` *grupo fichero ...*
- Para poder darla hay que ser el superusuario o el propietario del fichero y pertenecer al nuevo grupo
- También podemos cambiar el grupo al que pertence un directorio (si damos la orden con la opción -R también cambiarán de grupo todos los ficheros que están ya en el directorio)
- ¿A qué grupo pertenecerán los nuevos ficheros que creemos después?
  - Si el directorio tiene activado el bit SGID, los ficheros pertencecerán al nuevo grupo
  - Si no está activado, pertenecerán al grupo antiguo

Cambio de propietario y grupo de ficheros (cont.)

La orden chown

- Permite cambiar el propietario o el grupo de un fichero
- Sólo el superusuario puede cambiar el propietario de un fichero
- `chown` *usuario*[:*grupo*] *fichero* ...

Ejercicio de ejemplo

1. Si damos la orden chmod 6621 ejercicios ¿qué permisos tendrá asignados el fichero ejercicios? Escriba su máscara de permisos.
2. Supongamos que tenemos la siguiente situación:

```
drwxrwxrwt   root   system   /usr/publico
-rw-------   pepe   system   /usr/publico/examen
-rw-rw-rw-   juan   system   /usr/publico/apuntes
-rw-r--r--   luis   system   /usr/publico/memoria
```

Conteste a las siguientes preguntas, razonando la respuesta:

1. ¿Podría borrar `juan` el fichero `/usr/publico/examen`?
2. ¿Podría borrar `luis` `/usr/publico/apuntes`?

`juan` y `luis` pertenecen al grupo `alumnos`.