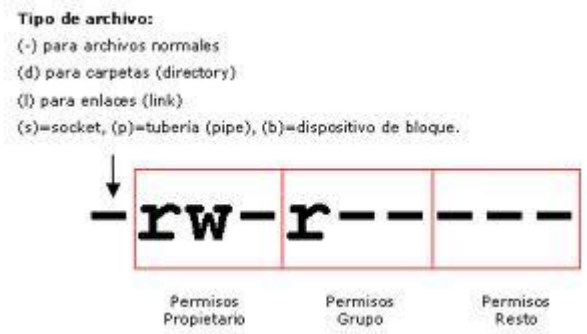
IES Rodrigo Caro

Dpto de Informática

Implantación de Sistemas Operativos.
Material elaborado por Manuel Fco. Domínguez Tienda.

Ud10.- Los permisos

Índice



1.- Introducción.

Los permisos en Linux nos ayuda a proteger los archivos de accesos indebidos.

Un archivo creado por un usuario, en principio, no debería ser visualizado o manipulado por otros usuarios. ¿Por defecto, es posible?.

1.- INTRODUCCIÓN

1.- Abre una sesión como heidi y crea un archivo llamado **agua** que contenga el texto: "Tengo ganas de desayunar"

2.- Abre una consola textual y entra como pedro.

3.- ¿Puede pedro visualizar el fichero agua de heidi?.

4.- ¿Puede pedro borrar el fichero agua de heidi?.

2.- Permisos y propietarios.

2.1.- Tipos de usuarios.

Los usuarios los podemos clasificar en:

- usuario (u): es el propietario.
- Grupo(g): conjunto de usuarios que tienen algo en común.
- Otros (o): resto de usuarios.

Cada usuario tiene un identificador, llamado uid.

Cada grupo tiene un identificador llamado gid.

$id [opciones] [Nombre_usuario] → Nos muestra información del identificador del usuario, así como los grupos a los que pertenece.

Opciones:

- -u → sólo muestra uid
- -g → sólo muestra gid

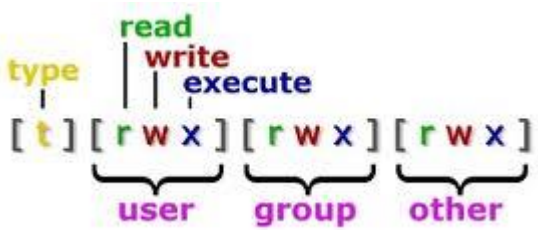
Ejercicio:
1.- visualiza el uid y gid del usuario y del root

2.- Permisos y propietarios.

2.2.- Permisos.

Los permisos fundamentales son:

- lectura → r
- Escritura → w
- Ejecución → x



Estos permisos aplicados a archivos o a directorios tienen diferentes significados.

Existen otros, los llamados especiales, como es el sticky bit, que ya veremos.

2.- Permisos y propietarios.

2.2.- Permisos.

Interpretación de los permisos en archivos:

Permisos	Archivos
r	Podemos visualizar el archivo
w	Podemos modificarlo o borrarlo.
x	Es ejecutable (Equivalente a los .exe de windows)

2.- Permisos y propietarios.

2.2.- Permisos.

Interpretación de los permisos en directorios:

Permisos	Directorios
r	Podemos visualizar su contenido. ls → sí cd → No
w	Podemos escribir y eliminar archivos en su interior. Por sí sólo no hace nada, debe estar combinado con rw o rx.
x	Permite acceder a dicho directorio.

Directorios			
R	W	X	Interpretación
X		X	Podemos visualizar y acceder
X	X		Podemos visualizar. No podemos modificar nada porque no podemos acceder.
	X	X	No podemos visualizar y sí modificarlo.

2.- Permisos y propietarios.

Ejercicio: Interpretación de los permisos:

1.- Crea un archivo llamado nube e interpreta sus permisos.

2.- Crea un directorio llamado fisica e interpreta sus permisos.

3.- Visualiza los permisos del comando date y ping

2.- Permisos y propietarios.

2.2.- Permisos.

Los permisos del directorio predominan frente a los de un archivo.

```
usuario@debian:~$ ls -ld apuntes
drwxrwxrwx 2 usuario usuario 4096 feb 14 18:59 apuntes
usuario@debian:~$ ls -l apuntes/
total 0
-rw-r--r-- 1 usuario usuario 0 feb 14 18:59 prueba
usuario@debian:~$
```

```
heidi@debian:/home/usuario/apuntes$ rm -r prueba
rm: ¿borrar el fichero regular vacío 'prueba'  protegido contra escritura? (s/n)
 s
```

3.- Asignar permisos.

Para asignar permisos utilizaremos el comando:

$chmod [opciones] permisos

-R : aplica permisos recursivamente.

3.1.- Asignar permisos de forma simbólica.

Tipos de usuario	Operaciones	Permisos
Usuario: u	+ añadir permisos - Quitar permisos = Establecer o asignar permisos	R lectura W escritura X ejecución
Grupo: g		
Otros: o		
Todos: a= ugo		
Ejemplo: $ chmod g+w agua		

3.- Asignar permisos.

3.1.- Asignar permisos de forma simbólica.

Ejercicios: Interpreta los siguiente permisos:

1.- $chmod a+r agua →

2.- $chmod og-x agua →

3.- $chmod u+rwx agua →

4.- $chmod o-rwx agua →

5.- $chmod g=x agua →

6.- $chmod g+x agua →

3.- Asignar permisos.

3.1.- Asignar permisos de forma simbólica.

Ejercicios:

3.1 .- DE FORMA SIMBÓLICA

1.- Vamos a trabajar con el archivo agua de heidi.

2.- Quita al usuario heidi el permiso de escritura. Visualiza los nuevos permisos. ¿Puede heidi modificar el archivo agua?.

3.- Abre una consola textual como pedro. ¿A qué grupo de usuarios pertenecería pedro?. Intenta visualizar el fichero agua de heidi. ¿Puede modificarlo?.

4.- Añade los permisos que creas oportuno para que el usuario pedro pueda modificar dicho archivo.

3.- Asignar permisos.

3.2.- Asignar permisos de forma numérica.

$chmod abc archivo

a=representa los permisos del usuario

b=representa los permisos del grupo

c=representa los permisos de los otros.

Esta forma de dar los permisos no tiene en cuenta los permisos Anteriores.

Ejemplo: 700, 400, 777

Ejemplo:

Usuario	Grupo	Otros	Comando
rwx 421	r- - 421	rw- 421	$chmod 746 agua
7	4	6	

4.- La máscara.

Cómo has podido comprobar cuando se crea un archivo se crea con los permisos:

```
usuario@servidor200:~$ touch ejemplo
usuario@servidor200:~$ ls -l ejemplo
-rw-r--r-- 1 usuario usuario 0 mar  3 19:37 ejemplo
usuario@servidor200:~$
```

¿Qué puedo hacer si quiero que mis archivos se creen por defecto con el permiso:

rw- --- --- ?

```
usuario@servidor200:~$ touch resultado
usuario@servidor200:~$ ls -l resultado
-rw------- 1 usuario usuario 0 mar  3 19:39 resultado
usuario@servidor200:~$
```

4.- La máscara.

¿Cómo podemos cambiar los permisos que asigna el sistema por defecto?.

A través de la máscara.

La máscara nos permite variar los permisos por defecto.

El comando para llevarlo a cabo es:

$umask → nos devuelve el valor de la máscara.

Ejemplo: 0022

octal ugo

$umask xxx → cambia el valor de la máscara.

4.- La máscara.

Permisos reales:

Para averiguar los permisos reales que se establecen, debemos restar los permisos por defecto y la máscara.

	Archivo	Directorio
Permisos por defecto	666	777
	-	-
Máscara	022	022
	____	____
	644	755
	rw- r– r--	rwx r-x r-x

4.- La máscara.

Ejercicios:

4.- LA MÁSCARA

1.- Escribir la máscara actual.

2.- Crear un directorio arboles y un archivo abeto. Visualiza los permisos.

3.- Cambiar temporalmente la mácara a 000. ¿Con qué permisos se crearán los directorios y archivos?. Crea un directorio oceano y un archivo atlantico. Visualiza los nuevos permisos.

4.- La máscara.

Como hemos visto la máscara por defecto tiene un valor de 022 para todos los usuarios, pero ¿Qué ocurre si un usuario necesita establecer una máscara diferente permanentemente?.

~/.bashrc y ~/.profile.

El primero se encarga de las sesiones en líneas de comando, mientras que el segundo se encarga de todos los demás casos, es decir, del entorno gráfico, y no tiene absolutamente ningún sentido que el umask sea diferente según el método de acceso del usuario.

Debemos cerrar la sesión para que los cambios surtan efectos.

umask [VALOR DESEADO]

Ejemplo: umask 000

4.- La máscara.

Ejercicio:

4.1.- FIJAR LA MÁSCARA DE UN USUARIO.

1.- ¿Qué máscara deberíamos establecer para que los archivos se crearan con los siguientes permisos: rw- --- ---?.

2.- Fija la anterior máscara al usuario heidi. Cierra la sesión y vuelve a entrar. Comprueba el nuevo valor de la máscara.

3.- Vuelve a dejar la máscara con su valor por defecto (022).

5.- Cambio de grupo y de propietario.

Cuando se crea un archivo se le asigna un usuario y un grupo.

Por ejemplo:

touch glaciar → - rw- r– r-- heidi heidi

Nos está indicando el usuario y el grupo, que en este caso es heidi y heidi.

El comando que utilizaremos para cambiar un archivo de grupo es:

$chgrp [opciones] grupo fichero

Para poder realizar esta operación, se necesita cumplir dos operaciones:

.- Ser el propietario.

.- Pertenecer al nuevo grupo.

5.- Cambio de grupo y de propietario.

Otro comando de mucho interés es:

```
$chown [opciones] usuario_nuevo[:grupo_nuevo]  fichero
```

Nos permite cambiar a un fichero de usuario y grupo.

¿Para qué puede ser interesante?.

Pues, a veces el root realiza unas operaciones y a continuación debe transferirle el archivo a un usuario.

Ya veremos algún ejemplo cuando creemos usuarios manualmente.

Semejanza con chgrp:

$chown :grupo_nuevo fichero → $chgrp grupo_nuevo fichero

6.- Permisos especiales

A parte de los permisos de lectura, escritura y ejecución, existe unos permisos especiales, que son:

Sticky bit o bit pegajoso (t)

Setuid (s)

Éstos dos, nos permitirán ejecutar archivos como si fuésemos los propietarios

Setgid (s)

Ejemplo setuid:

passwd sirve para cambiar la contraseña.

/etc/bin/passwd → cuando se ejecuta llama al archivo → /etc/passwd

-rws r-x r-x root root

Lo puede ejecutar todo el mundo

-rw- r– r— root root

No puede ser modificado por todos

Sin embargo heidi puede ejecutarlo y los cambios afectan al fichero /etc/passwd. ¿Por Qué?.

6.- Permisos especiales

6.1.- Sticky bit o bit pegajoso (t)

Se aplica a directorios.

Este permiso aplicado a un directorio, hará que los archivos creados en él sólo pueden ser borrados por el propietario.

Nadie más podrá borrarlo, aunque tenga permiso de escritura sobre el directorio.

Procedimiento:

1.- Damos todos los permisos al directorio en cuestión.
Ejemplo: $chmod 777 prueba

2.- Asignamos el sticky bit a dicho directorio.
Ejemplo: $chmod o+t prueba.

Esos dos comandos se pueden sustituir por $chmod 1777 prueba

6.- Permisos especiales

6.1.- Sticky bit o bit pegajoso (t)

6.1.- Sticky bit o bit pegajoso

1.- Visualizar los permisos que tiene el directorio /tmp.

2.- El root va a crear en /home un directorio llamado sugerencias para que los usuarios del sistema manifieste sus sugerencias.

3.- Necesitamos que el directorio sugerencias esté configurado de forma que:

a) Todos los usuarios puedan acceder y escribir en él .
b) Los archivos sólo pueden ser borrados por los propietarios.

4.- Como heidi escribe un archivo llamado horario que ponga: "Me gustaria que el horario de biblioteca fuese de 24h, en la semana de exámenes".

5.- Ahora como pedro, visualiza el archivo horario y a continuación intenta borrarlo.