RECOPILATORIO LINUX

ENLACES

Los enlaces le permiten dar a un archivo múltiples nombres.

Los archivos son identificados en el sistema por un inodo, el cual es el único identificador del archivo para el sistema de archivos.

Un directorio es una lista de inodo con sus correspondientes nombres de archivo.

Cada nombre de archivo en un directorio es un enlace a un inodo particular.

HARD LINKS

Si tenemos un archivo llamado archivo con un inodo determinado en el sistema de archivos, podemos crear un enlace a archivo llamado enlace:

#ln archivo enlace

Si haces un ls –i, observarás que los dos ficheros tienen el mismo inodo.

Accediendo a "archivo" o a "enlace" accedemos al mismo archivo.

Si haces cambios en "archivo", estos cambios serán efectuados también en "enlace".

Para todos los efectos "archivo" y "enlace" son el mismo archivo

Los HARD LINK crean el enlace al inodo.

Sólo se pueden crear enlaces entre archivos del mismo sistema de archivos..

Si borras un archivo con rm sólo estás borrando un enlace a un archivo.

Si haces #rm "archivo" la orden ls-i mostrará los números de inodo.

Por tanto observarás que solo el enlace de nombre "archivo" es borrado pero todavía existe "enlace".

¿Por qué rm borra los archivos?

Porque usualmente los archivos tienen un único enlace, pero si el archivo tiene múltiples enlaces, el uso de rm sólo borrará un enlace.

Para borrar el archivo debes borrar todos los enlaces del archivo.

Si haces: # ls –l "archivo" "enlace"

```
-rw-r—r--  2  root  9  feb  5  15:00  enlace
-rw-r—r--  2  root  9  feb  5  15:00  archivo
```

La segunda columna me indica el numero de enlaces:2 al archivo

Un directorio no es más que un archivo que contiene información sobre la traslación enlace a inodo.

Cada directorio tiene al menos dos enlaces duros en él:

".": uno apuntando a sí mismo

"..": otro apuntando al directorio padre

ENLACES SIMBÓLICOS

Son otro tipo de enlace.

Permite dar a un archivo el nombre de otro, pero no enlaza el archivo con un inodo.

Dentro del enlace simbólico se guarda la ruta donde ubicar al archivo destino

```
#ln –s   archivo1    enlace1
```

Haciendo ls –li, podemos ver que los dos archivos tienen inodos distintos:

```
#ls –l   archivo1   enlace1

lrwxrwxrwx 1     root  root  9     feb   15:00      enlace1->archivo1

-rw-r—r--  1     root  root  9     feb   15:00      archivo1
```

Los bits de permiso en un enlace simbólico no se usan. En su lugar, los permisos del enlace simbólico son determinados por los permisos del archivo "apuntado" por el enlace (en nuestro caso "archivo1")

Se puede crear un enlace simbólico a un archivo que no existe.

Los enlaces simbólicos identifican al archivo al que apuntan. En cambio en el enlacace duro no hay forma de saber qué archivo está enlazado al mismo inodo.

La operación "rm" sobre un fichero simbólico no actúa sobre el fichero apuntado sino solo sobre el propio enlace simbólico destruyéndolo.

Cualquier cambio que se haga tanto en el directorio original como en el contenido del enlace simbólico **quedará reflejado en el otro** (si cambias el contenido del enlace simbólico se podrá ver en el original y viceversa)

Borrar el enlace simbólico **no afecta al original** (seguirá existiendo)

Para crear un enlace simbólico lanzaremos el siguiente comando desde la *terminal*

Identificar enlaces simbólicos

Desde el **administrador de archivos** pueden identificarse fácilmente por contar con una **flecha en la parte inferior** del icono de carpeta

Desde la **terminal** (lanzando un *ls -l* para que muestre información adicional) veremos que en el bit de tipo en lugar de aparecer una *d* (de Directorio) **aparece una *l*** (de Link)

```
lrwxrwxrwx 1 jasvazquez jasvazquez
drwx------ 2 jasvazquez jasvazquez
drwx------ 2 jasvazquez jasvazquez
```

Ya que estamos con la terminal y recordando el aviso que hicimos sobre los enlaces duros (al borrar el último se borra el *DIRECTORIO_ORIGINAL*) comentar que para saber cuántos **enlaces DUROS apuntan al** directorio **original** basta con **mirar** el **número** que hay justo **a la derecha de los permisos** (en el ejemplo es un 1 porque sólo hay un enlace duro al *DIRECTORIO_ORIGINAL*)

```
lrwxrwxrwx 1 jasvazquez jasvazquez
drwx------ 2 jasvazquez jasvazquez
drwx------ 2 jasvazquez jasvazquez
```

DETALLES A TENER EN CUENTA:

Puesto que Linux es un sistema multiusuario, permite que varios usuarios usen el sistema simultáneamente. Se hace necesario que cada usuario no pueda acceder a los documentos de los demás, además no todos los usuarios deberían poder instalar programas, modificar ficheros importantes del sistema u otras cosas importantes.

Por ello Linux establece login y password (nombre y contraseña) para cada usuario. Además existe el administrador del sistema o superusuario, cuyo login es root, que tiene acceso a todo y permisos para todo.

Las contraseñas son almacenadas de forma cifrada, por lo que es imposible acceder a ellas (si las guardara simplemente en un fichero de texto, sería muy fácil acceder y romper el sistema de seguridad).

Un ***grupo*** es un conjunto de usuarios que comparten las mismas características. La idea de grupo permite al administrador del sistema dar ciertos permisos (por ejemplo: autorización para usar la grabadora) al grupo, evitando tener que dárselos a cada usuario individualmente.

Todo usuario debe pertenecer al menos a un grupo. Además existen otros usuarios, aparte de los reales, que los usa el sistema para determinadas operaciones. Estos usuarios del sistema (no reales) no se pueden "logear", es decir no se puede entrar al sistema tecleando un login de estos.

El usuario ***root*** o superusuario no debe entrar al sistema con el login de root, sino que debe tener además un login y password para entrar al sistema. Sólo cuando necesite realizar acciones especiales, el sistema le pedirá la contraseña de root y podrá acceder. Aunque es posible acceder como root a un sistema Linux, no es recomendable.

Toda la información sobre usuarios, grupos y contraseñas se guarda en los archivos:

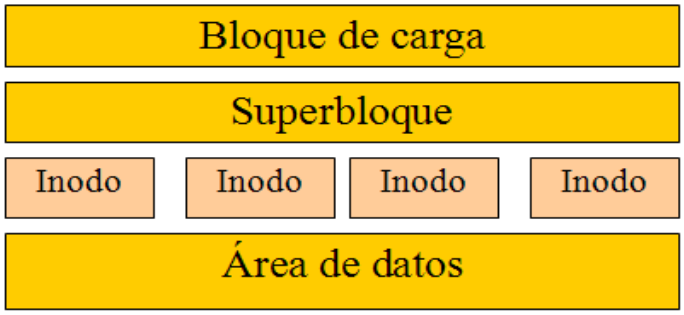
`/etc/passwd` (información sobre usuarios)
`/etc/group` (información sobre grupos)
`/etc/shadow` (contraseñas cifradas)
`/etc/gshadow` (contraseñas cifradas de los grupos) [normalmente no se usa este fichero]

En estos archivos de texto, se almacena la información línea a línea (cada una es un usuario o un grupo).

Dentro de cada línea hay varios campos separados por ":"

El sistema de ficheros
El sistema de ficheros es la organización lógica del disco que nos permite almacenar la información en forma de ficheros de un modo totalmente transparente. Esta palabra tan utilizada significa que no tenemos que preocuparnos de pistas, sectores, cilindros y otras menudencias.
Al igual que el cluster era el elemento mas importante en FAT y NTFS, en **HTFS** lo es el **bloque.**

El esquema del sistema de archivos:



En Linux los elementos del sistema de archivos son el **superbloque**, **i-nodos** y **bloques de datos**.
Bloque de carga o bloque cero de cada sistema, esta reservado para almacenar un programa que utiliza el sistema para gestionar el resto de partes del sistema de archivos.

- **El superbloque** o bloque uno

Contiene la descripción sobre el sistema de ficheros: Tamaño, bloques libres, tamaño de la lista de i-nodos, i-nodos libres, verificaciones, etc.

- Los **i-nodos**. Un i-nodo contiene toda la información sobre cada conjunto de datos en disco, que denominamos fichero:
    1. **Donde se almacenan los datos**, es decir lista de bloques de datos en disco. Esto son una serie de punteros o direcciones de bloques que indican bien donde están los datos en disco, o bien donde están los bloques que tienen más direcciones de bloques de datos (bloques indirectos).

2. Quien es **el propietario** de los datos, un número que lo identifica (UID o User Identifier), y a qué grupo pertenece el fichero GID Group Identifier).
3. Tipo de fichero:

regular, es decir un fichero que contiene información habitual, datos o programas **dispositivo**, un elemento destinado a intercambiar datos con un periférico

enlace, un fichero que apunta a otro fichero; **pipe**, un fichero que se utiliza para intercambiar información entre procesos a nivel de núcleo.

directorio, si el elemento no contiene datos sino referencias a otros ficheros y directorios.

Regulares (datos y comandos) y directorios
Dispositivos (bloque y carácter)
Enlace (duro y simbólico)

Archivos Especiales

Los ficheros especiales no sirven para almacenar información, sino para operaciones especiales. Consideraremos ficheros especiales:

Los **enlaces**. Sirven para que podamos acceder al contenido de un fichero o directorio desde distintos puntos del árbol de directorios. Tienen la apariencia de ficheros normales, pero el contenido de estos ficheros es compartido por otros archivos, de forma que si modificamos el contenido del fichero, queda modificado para todos los enlaces, ya que se refieren al mismo fichero.

Los **dispositivos**. Sirven para manejar los distintos periféricos conectados al computador.
En UNIX todos los periféricos se manejan como si fueran ficheros.

Por ejemplo, la pantalla se maneja como un fichero llamado **/dev/tty**, de forma que si escribimos información en este fichero, la información, en vez de almacenarse en el disco, se muestra en pantalla.

Cada periférico tiene uno o varios ficheros especiales, que llamaremos dispositivos, y todos ellos se colocan en el directorio **/dev**. La tabla siguiente contiene algunos ejemplos de ficheros especiales.

Dispositivo	Significado
/dev/tty	Terminal (consola).
/dev/hda	Primer disco duro
/dev/hda1	Primera partición del primer disco duro
/dev/hdb	Segundo disco duro
/dev/fd0	Primera unidad de disco flexible.
/dev/null	Dispositivo nulo (descarta la información).
/dev/ram	Memoria RAM.

4. Permisos del fichero (quien puede leer(r), escribir(w) o ejecutar(x)). Estos permisos se asignan a se asignan de forma diferenciada a tres elementos: el propietario, el grupo (indicados con anterioridad) y al resto de los usuarios del sistema.

5. Tamaño del fichero

6. Número de enlaces del fichero. Es decir cuantos nombres distintos tiene este fichero.

Hay que observar como el nombre de un fichero no forma parte del i-nodo.

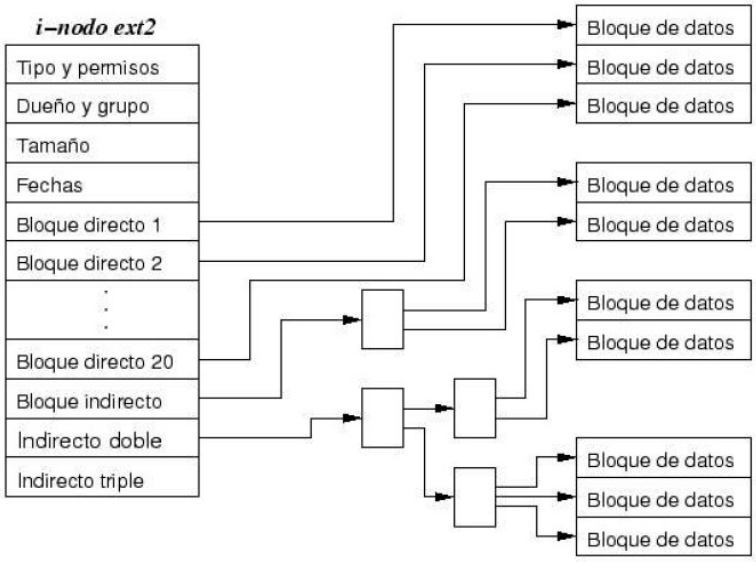
El nombre de fichero se asocia a un i-nodo dentro de un fichero especial denominado directorio. Esto le proporciona al sistema de ficheros la posibilidad de que un mismo i-nodo pueda tener varios nombres si aparece en varios directorios o con distintos nombres.

- **Bloques o zonas de datos**

  Ocupa el resto del disco**, y es** equivalente a la zona de datos de DOS.

  En esta zona están almacenados los ficheros y directorios de nuestro sistema.

En la figura se representa el esquema de un i-nodo.



El sistema de ficheros de Linux de forma lógica posee una estructura de árbol cuya raíz es única y se representa por / que además constituye el carácter separador en el camino (Path) de cualquier recurso del sistema.

Esquema general de los principales directorios presentes en el sistema de ficheros de Linux.
Esto quiere decir que los distintos discos no aparecen como tales discos independientes en el sistema, sino que aparecen como un directorio más y esto lo podemos organizar de acuerdo con nuestros intereses.

Cada sistema de ficheros se integra como un árbol con ficheros y directorios en el árbol de directorio como una rama más. Los dispositivos extraíbles se integran como ramas que se pueden poner o quitar

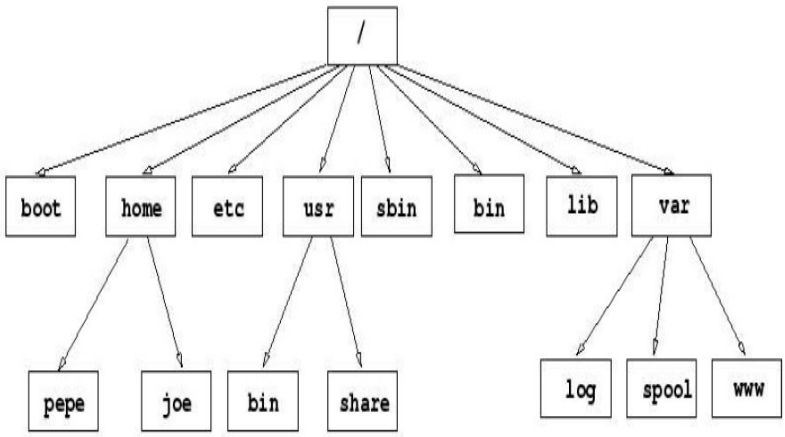


Figura: Estructura lógica en el sistema de ficheros de Linux

/ : es el directorio raíz donde empiezan los directorios .
/bin : contiene los comandos básicos que utilizan los usuarios para trabajar con el ordenador: ver el contenido de archivos, copiar archivos, crear directorios etc.
/boot : contiene la información de arranque del sistema, los ficheros de inicio y otros ficheros de carga.
/dev : aquí se almacenan los ficheros de los dispositivos. Se utilizan para acceder a los dispositivos físicos que hay instalados en el ordenador (ratones, discos, CDrom etc.)
/etc : aquí se guardan los ficheros de configuración de la máquina. Archivos de configuración de impresoras o la información de las cuentas de usuario.
/home: contiene las carpetas personales de los usuarios.
/lib : contiene las librerías de los programas que están instalados en el sistema.
/mnt: es el directorio sobre el que se montan los sistemas de ficheros de los discos, CDrom, disquetera.
/opt : aquí se coloca el software que se agrega después de la instalación.
/sbin : contiene los comandos para la administración del sistema. Los usa el superusuario o administrador.
/tmp : contiene los ficheros temporales que se crean durante la instalación, el arranque, la ejecución de programas, etc.
/usr : aquí se instala algún software del sistema, la documentación del sistema y los complementos multimedia(iconos, imágenes, sonidos…)
/var : aquí está la información variable, como los buzones de correo de los usuarios, archivos de registro del programa, información de las colas de impresión, etc.

Propiedad
En la descripción del sistema de ficheros, cada **i-nodo** guarda un espacio para indicar los números de usuario y grupo. Entonces cada i-nodo pertenece a un usuario y a un grupo. También el usuario que tenga el UID (número de identificación) descrito en la base de datos de usuario **/etc/passwd** que coincida con el del fichero será su propietario. Esto también es válido para directorios.
Los propietarios, se utilizan como mecanismo de seguridad para poder asignar ciertas propiedades en función del usuario, en particular los permisos.

Permisos
Todos los archivos en un sistema Linux tienen permisos que posibilitan o impiden a otros verlos, modificarlos o ejecutarlos. El superusuario «root» tiene la capacidad de poder acceder a cualquier archivo en el sistema.
El nombre del fichero se asocia al i-nodo en un fichero de directorio
Cada archivo tiene restricciones de acceso y restricciones de usuario, y está asociado con un propietario y un grupo.
Cada **i-nodo** guarda un espacio para almacenar los permisos bajo los cuales se puede acceder a un fichero.

En total tendremos nueve bits que indican los distintos permiso en el siguiente orden: usuario, grupo, otros

Cada archivo esta asegurado por los siguientes grupos de permisos, por orden de importancia:
- ***usuario*** se aplica al usuario propietario del archivo
- ***grupo*** se aplica al grupo asociado al archivo
- ***otro*** se aplica a todos los demás usuarios

Dentro de cada uno de los tres grupos de permisos están los permisos en sí. A continuación se describen los permisos, junto con la forma en la que se aplican a **archivos** y a directorios por separado:
- ***lectura*** **los archivos se pueden ver**/abrir se puede ver el contenido del directorio
- ***escritura*** **los archivos se pueden cambiar o borrar** se puede modificar el contenido del directorio
- ***ejecución*** **los archivos ejecutables se pueden arrancar** como un programa se puede entrar en los directorios

permiso de escritura aplicado a ficheros y a directorios:

Ficheros: permite modificarlo o borrarlo

Directorios: permite crear o borrar ficheros en el directorio

ANALISIS de ficheros importantes

Cat /etc/passwd

Cada una de las líneas representa un usuario y responde al esquema:

```
usuario : x : UID : GID : comentarios :
directorio_home : shell
```

usuario: es el login o nombre de usuario (nombre único para cada usuario)
x: contraseña: aparece una x; la contraseña se encuentra cifrada en /etc/shadow
UID: User IDentifier (nº de Identidad de Usuario). Es único. Debe ser un entero≥0 (el cero se reserva para root y del 1 al 99 se reservan para tareas del sistema). Usar de 100 en adelante.
GID: nº de Identidad de Grupo (el cero se reserva para el grupo root)
comentarios: nombre real u otros comentarios sobre el usuario (puede contener espacios)
directorio_home: directorio de inicio del usuario
shell: intérprete de comandos (normalmente bash). Si ponemos /bin/false el usuario no podrá ejecutar ningún comando del sistema

Ejemplo:

```
dani@guadalinex:~$ cat /etc/passwd

root:x:0:0:root:/root:/bin/bash

daemon:x:1:1:daemon:/usr/sbin:/bin/sh

bin:x:2:2:bin:/bin:/bin/sh

  ...

identd:x:100:65534::/var/run/identd:/bin/false

sshd:x:101:65534::/var/run/sshd:/bin/false

gdm:x:102:103:Gnome Display
Manager:/var/lib/gdm:/bin/false

dani:x:1000:100::/home/dani:/bin/bash

yaiza:x:1001:1001:,,,:/home/yaiza:/bin/bash
```

Tecleando en un terminal `cat /etc/passwd` nos muestra el contenido del fichero `/etc/passwd`

Observe como el primer usuario que aparece (primera línea) es root

A continuación una serie de usuarios (no reales) del sistema

Por último (dos últimas líneas) los usuarios "dani" y "yaiza"

Podemos comprobar como en esta distribución de Linux (GuadaLinex 2004) los números de usuarios empiezan desde 1000 en adelante (en otras distribuciones lo hacen desde 100 o desde 500)

Nomenclatura:

A:B:C:D:E:F:G

A: nombre de usuario
B: password (si es una **x** es que esta encriptado en /etc/shadow)
C: ID del usuario
D: grupo principal del usuario
E: Información adicional del usuario (nombre real, telefono, departamento, etc.), separado con comas
F: Carpeta personal del usuario (no siempre la carpeta personal tiene el mismo nombre del usuario)
G: Shell del usuario. Existen varias shell, la clásica es bash, pero hay otras como zsh, por ejemplo.

Cuando en el editor aparece /bin/false, significa que el usuario no puede iniciar sesión

cat /etc/group

Cada una de las líneas representa un grupo y responde al esquema:

```
grupo : x : GID : lista_usuarios
```

grupo: es el nombre del grupo
x: contraseña: aparece una x; la contraseña se encuentra cifrada en /etc/shadow. Si este campo aparece vacío, significa que el grupo no necesita contraseña.
GID: nº de Identidad de Grupo (el cero se reserva para el grupo root)
lista_usuarios: lista de usuarios que pertenecen al grupo separados por comas. También se pueden añadir usuarios de otros grupos, con lo que tendrían todos los privilegios de este grupo

cat /etc/shadow

Cada una de las líneas representa un usuario y responde al esquema:

```
usuario : contraseña_cifrada : d1 : d2 : d3 :
d4 : d5 : d6 : reservado
```

usuario: es el login o nombre de usuario (el mismo que en /etc/passwd)
x: contraseña: aparece una x; la contraseña se encuentra cifrada en /etc/shadow.

d1: nº de días desde el 01/01/1970 hasta último cambio de la contraseña.
d2: nº de días que deben pasar hasta que se pueda cambiar la contraseña.
d3: nº de días que deben pasar para que caduque la contraseña y deba ser cambiada.
d4: nº de días de antelación con los que avisará el sistema de la caducidad de la contraseña.
d5: nº de días con contraseña caducada antes de deshabilitar la cuenta.
d6: nº de días desde el 01/01/1970 y el día en que se deshabilitó la cuenta.
reservado: campo reservado

Ejemplos de creación de usuarios en la shell:

BASE introductoria con detalles a cambiar:

1.-Crea tres usuarios: operario1, operario2, operario3 con contraseñas:

Adduser operario1 passwd operario1

Adduser operario2 passwd operario2

Adduser operario3 passwd operario3

2.- Crea el grupo trabajadores e introduce los dos primeros en el grupo trabajadores:

Groupadd trabajadores

Adduser operario1 trabajadores

Adduser operario2 trabajadores

userdel borrar usuarios

groupdel borrar grupos

PERMISOS DE ARCHIVOS Y DIRECTORIOS

¿Has visto esa combinación de r,w,x y - cuando listas un directorio?, tienes cierta idea que son los permisos, pero ¿como se usan y como funcionan?.

En Linux, todo archivo y directorio tiene tres niveles de permisos de acceso:

-los que se aplican al propietario del archivo,

-los que se aplican al grupo que tiene el archivo

-los que se aplican a todos los usuarios del sistema. Podemos ver los permisos cuando listamos un directorio con ls -l:

```
$> ls -l
-rwxrwxr--  1 sergio ventas    9090 sep  9 14:10 presentacion
-rw-rw-r--  1 sergio sergio 2825990 sep  7 16:36 reporte1
drwxr-xr-x  2 sergio sergio    4096 ago 27 11:41 videos
```

Veamos por partes el listado, tomando como ejemplo la primera línea.

a) La primera columna (-rwxrwxr--) es el **tipo de archivo** y sus permisos
b) la siguiente columna (1) es el número de enlaces al archivo
c) la tercera columna (sergio) representa al **propietario** del archivo, la cuarta columna (ventas) representa al **grupo** al que pertence al archivo y las siguientes son el tamaño, la fecha y hora de última modificación y por último el nombre del archivo o directorio.

El primer caracter al extremo izquierdo, representa el **tipo de archivo**, los posibles valores para esta posición son los siguientes:

- un guión representa un archivo comun (de texto, html, mp3, jpg, etc.)
d representa un directorio
l link, es decir un enlace o acceso directo
b binario, un archivo generalmente ejecutable

Los siguientes 9 restantes, representan los permisos del archivo y deben verse en grupos de 3.

Los tres primeros representan los permisos para el propietario del archivo. Los tres siguientes son los permisos para el grupo del archivo y los tres últimos son los permisos para el resto del mundo u otros.

rwx rwx rwx
usuario grupo otros

En cuanto a las letras, su significado son los siguientes:

r read - lectura
w write - escritura (en archivos: permiso de modificar, en directorios: permiso de crear archivos en el dir.)
x execution - ejecución

Las nueve posiciones de permisos son en realidad un bit que o esta encendido (mostrado con su letra correspondiente) o esta apagado (mostrado con un guión -), asi que, **por ejemplo**, permisos como **rwxrw-r--**, indicaría que los permisos del propietario (rwx) puede leer, escribir y ejecutar el archivo, el grupo (o sea los usuarios que esten en mismo grupo del archivo) (rw-) podrá leer y escribir pero no ejecutar el archivo, y cualquier otro usuario del sistema (r--), solo podrá leer el archivo, ya que los otros dos bits de lectura y ejecución no se encuentran encendidos o activados.

Permisos en formato numérico octal

La combinación de valores de cada grupo de los usuarios forma un número octal, el bit x es 2^0 es decir 1, el bit w es 2^1 es decir 2, el bit r es 2^2 es decir 4, tenemos entonces:

r = 4
w = 2
x = 1

La combinación de bits encendidos o apagados en cada grupo da ocho posibles combinaciones de valores, es decir la suma de los bits encendidos:

- - - = 0 no se tiene ningún permiso
- - x = 1 solo permiso de ejecución
- w - = 2 solo permiso de escritura
- w x = 3 permisos de escritura y ejecución
r - - = 4 solo permiso de lectura
r - x = 5 permisos de lectura y ejecución
r w - = 6 permisos de lectura y escritura
r w x = 7 todos los permisos establecidos, lectura, escritura y ejecución

Cuando se combinan los permisos del usuario, grupo y otros, se obtienen un número de tres cifras que conforman los permisos del archivo o del directorio. Esto es más fácil visualizarlo con algunos ejemplos:

Permisos	Valor	Descripción
rw-------	600	El propietario tiene permisos de lectura y escritura.
rwx--x--x	711	El propietario lectura, escritura y ejecución, el grupo y otros solo ejecución.
rwxr-xr-x	755	El propietario lectura, escritura y ejecución, el grupo y otros pueden leer y ejecutar el archivo.
rwxrwxrwx	777	El archivo puede ser leido, escrito y ejecutado por quien sea.
r--------	400	Solo el propietario puede leer el archivo, pero ni el mismo puede modificarlo o ejecutarlo y por supuesto ni elgrupo ni otros pueden hacer nada en el.
rw-r-----	640	El usuario propietario puede leer y escribir, el grupo puede leer el archivo y otros no pueden hacer nada.

Estableciendo los permisos con el comando chmod

Habiendo entendido lo anterior, es ahora fácil cambiar los permisos de cualquier archivo o directorio, usando el comando chmod (change mode), cuya sintaxis es la siguiente:

chmod [opciones] permisos archivo[s], algunos ejemplos:

```
$> chmod 755 reporte1
$> chmod 511 respaldo.sh
```

```
$> chmod 700 julio*
$> chmod 644 *
```

Los ejemplos anteriores establecen los permisos correspondientes que el usuario propietario desea establecer.

El tercer ejemplo (chmod 700 julio*) cambiará los permisos a todos los archivos que empiezen con julio (julio01, julio02, julio_respaldo, etc.) debido al caracter '*' que es parte de las expresiones regulares que el shell acepta, e indica lo que sea.

El último ejemplo por lo tanto cambiará los permisos a los archivos dentro del directorio actual.

Una opción común cuando se desea cambiar todo un árbol de directorios, es decir, varios directorios anidados y sus archivos correspondientes, es usar la opción -R, de recursividad:

```
$>chmod-R755respaldos/*
```

Esto cambiará los permisos a 755 (rwxr-xr-x) del directorio respaldos y de todos los subdirectorios y archivos que estén contenidos dentro de este.

Estableciendo permisos en modo simbólico

Otra manera popular de establecer los permisos de un archivo o directorio es a través de identificadores del bit (r,w, o x) de los permisos, como ya se vió anteriormente, pero ahora identificando además lo siguiente:

al usuario con la letra **u**
al grupo con la letra **g**
a otros usuarios con la letra **o**
y cuando nos referimos a todos (usuario, grupo, otros) con la letra **a** (all, todos en inglés)
el signo **+** para establecer el permiso
el signo **-** para eliminar o quitar el permiso

La sintaxis es muy simple chmod augo[+|-]rwx[,...] archivo[s], asi por ejemplo, si queremos que otros tengan permiso de escritura sería **chmod o+w archivo**, si queremos que todos los usuarios con permisos de ejecución **chmod a+x archivo**.

En este modo de establecer permisos, solo hay que tomar en cuenta que partiendo de los permisos ya establecidos se agregan o se quitan a los ya existentes. Veámoslo con ejemplos su manera de trabajar:

Actual	chmod	Resultado	Descripción
rw-------	a+x	rwx--x--x	Agregar a todos (all) permisos de escritura.
rwx--x--x	go-x	rwx------	Se eliminan permiso de ejecución para grupo y otros.
rwxr-xr-x	u-x,go-r	rw---x--x	Al usuario se le quita ejecución, al grupo y otros se le quita lectura.
rwxrwxrwx	u-x,go-rwx	rw-------	Al usuario se le elimina ejecución, al grupo y otros se eliminan todos los

			permisos.
r--------	a+r,u+w	rw-r--r--	A todos se les agrega lectura, al usuario se le agrega escritura.
rw-r-----	u-rw,g+w,o+x	---rw---x	Al usuario se le eliminan lectura y escritura, al grupo se le agrega lectura y otros se le agrega ejecución.

Cambiando propietario y grupo

Volviendo a mostrar el listado al inicio de este artículo:

```
$> ls -l
-rwxrwxr--  1 sergio ventas    9090 sep  9 14:10 presentacion
-rw-rw-r--  1 sergio sergio 2825990 sep  7 16:36 reporte1
drwxr-xr-x  2 sergio sergio    4096 ago 27 11:41 videos
```

Vemos en la tercera y cuarta columna al **usuario propietario del archivo** y **al grupo al que pertenece**, es posible cambiar estos valores a través de los comandos **chown (change owner, cambiar propietario)** y **chgrp (change group, cambiar grupo).** La sintaxis es muy sencilla: **chown usuario archivo[s]** y **chgrp grupo archivo[s]**. Además al igual que con **chmod**, también es posible utilizar la opción -R para recursividad.

```
#> ls -l presentacion
-rwxrwxr--  1 sergio ventas    9090 sep  9 14:10 presentacion
#> chown juan presentacion
#> ls -l presentacion
-rwxrwxr--  1  juan ventas    9090 sep  9 14:10 presentacion
#> chgrp gerentes presentacion
#> ls -l presentacion
-rwxrwxr--  1  juan gerentes  9090 sep  9 14:10 presentacion
```

Solo el usuario root puede cambiar usuarios y grupos a su voluntad sobre cualquier usuario, queda claro que habiendo ingresado al sistema como usuario normal, solo podrá hacer cambios de grupos, y eso solo a los que pertenezca.

Una manera rápida para el usuario root de cambiar usuario y grupo al mismo tiempo, es con el mismo comando **chown** de la siguiente manera:

```
#> chown juan.gerentes presentacion  (o en vez de punto, con : puntos)
#> chown juan:gerentes presentacion
```

Así, cambiará el usuario.grupo en una sola instrucción.

Permisos preestablecidos con umask

El comando **umask** establece la máscara de permisos de directorio y de archivos. Es decir los nuevos directorios y archivos que se crean obtienen el valor de los permisos a partir de los valores de **umask**.

```
$> umask
0002
(o en formato simbólico con la opción -S)
$> umask -S
u=rwx,g=rwx,o=rx
```

Lo anterior indica que un directorio y archivos ejecutables se crearán con los permisos 775 y los archivos comunes con los permisos 664. Esto se logra restando de 777 el valor de umask (777-002) y (666-002) respectivamente. El primer valor de umask corresponde para valores de Sticky bit, GUID o SUID, que por default es 0.

```
$> umask
0002
(Creamos un archivo y según la máscara debemos de tener 666-002=664 o rw-rw-r--)
$> touch archivo
$> ll archivo
-rw-rw-r-- 1 sergio sergio 0 sep 25 20:14 archivo
(Ahora creamos un directorio y según la máscara debemos de tener 777-002=775 o rwxrwxr-x)
$> mkdir dir
$> ls -ld dir
drwxrwxr-x 2 sergio sergio 4096 sep 25 20:20 dir
```

Para establecer el valor de la máscara, simplemente se usa el mismo comando **umask** seguido del valor de máscara que se desee:

```
$>umask0022
```

Para dejarlo fijo en la sesión, entonces conviene agregarlo a .bash_profile o .bash_rc de nuestro directorio de inicio.

NO TE OLVIDES DE LO QUE TIENES QUE SABER:
Lo que hemos hecho en clase más todo lo siguiente bien claro:

EJERCICIOS COMANDOS:

1) Diferencias entre :
- **a) EJECUCION CONSECUTIVA: comando 1; comando2**
- **b) EJECUCION CONDICIONAL:**
  Diferencias entre &&, ||

2) ls opciones –i,-l,-a . Saber las diferencias
3) Saber crear enlaces duros y simbólicos y saber las diferencias
4) Crear directorios con mkdir y saber la diferencia si pones la opción mkdir-p
5)Crear ficheros con touch, con gedit y saber la diferencia si pongo cat > nombre fichero
6) Qué es redireccionar. Diferencias entre > y >>
7) Qué es pipe
8) comando tee
9) Grep con opciones –c –i –v
Entender el significado de la búsqueda con ^ $ ^[^a] [a-z]
10) visualizar contenido de fichero con cat
11) filtrar filas: cut –d: -f6
12) filtrar columnas: cut –d: -c1-5
13) tail
14)head
15) sort con opciones –r y opcion –m para fusionar
16) copiar cp de donde a dónde
17) Diferencia entre rmdir y rm –r
18) Saber borrar directorios
19)saber borrar ficheros
20) Saber borrar pidiendo confirmación: opción –i

21)wc con opciones –l, -c, -w
22)echo….
23) who
24)find con opciones –name y –type
25) mover ficheros mv
26) renombrar ficheros: mv
27)pwd
28)cal
29)date
30)umask
31)sed con opciones delete, sustituir en la primera o en todas

En las distribuciones Ubuntu la cuenta root viene desactivada por defecto.

En el caso de Ubuntu para trabajar como root se usa "sudo"

- sudo passwd root
- y pide contraseña .....

sudo su → root
sudo passwd

Significa que estas trabajando como root

Para salir del modo superusuario: # exit

ls

-l muestra la salida en formato largo.
-R muestra recursivamente un directorio.
-a lista ademas las filas ocultas (sus nombres empiezan por punto).
-h muestra el tamaño del fichero.
-i muestra el identificador del i-nodo asociado a cada elemento.

mkdir

Crea directorios.

- **mkdir** -p /dir1/dir2/dir3 crea varios directorios de golpe, en vez de uno en uno.

cd

Para cambiar de directorio.

pwd

indica el camino absoluto del directorio donde estamos.

- $ pwd Te dice donde estas, por ejemplo: /home/pepe.

Para crear por ejemplo dentro de pepe, un directorio uno, seria:

- $ **mkdir** /uno

Si hago un ls, me lista los directorios y lo que tengo dentro del lugar que estoy.

- $ **ls** -l

Si quiero ver lo que hay dentro del fichero etc que sabemos que esta en el raíz, haría:

- $ ls -lia /etc l-formato, i-inodo, a-ocultos

Si estoy en /home/pepe y quiero pasar a uno:

- $ **cd** uno y me saldria _____/uno $

Para crear un fichero: **gedit** mifichero y entro en el editor y guardo.

Para ver el contenido de un fichero: **type** mifichero

Para crear un fichero vació: $ **cat >** misegundofichero me ha creado midsegundofichero VACIO.

touch 'mifichero' me crea 1 fichero vacio.

Para ver el contenido de un fichero y es largo....

- $ **less** /etc/passwd
- $ **more** /etc/passwd
- $ **cat** nombrefichero o le dices que te muestre el contenido con cat

cp

copia ficheros de un sitio a otro.

- **cp** mifichero /home copia mifichero de donde estoy al directorio home.

grep

-c devuelve solo la cantidad de lineas que contienen el patrón.
-i ignora la diferencia entre mayúsculas y minúsculas.
-v devuelve las lineas que no contienen el patrón.

Busca todas las palabras que empiezan por a en el fichero fich

- **grep** 'a*' fich
- **cat** fich | **grep** 'a*' haría lo mismo, ya que muestra el contenido de fich y saca las palabras que empiezan por a solo.

$ **cat** archivo_demo

ESTA ES LA PRIMERA LINEA EN MAYSCULAS DE ESTE ARCHIVO
esta es la primera linea en minúsculas de este archivo
Esta Es La Primera Linea Con El Primer Carácter De Cada

Hay dos lineas vaciás sobre esta
Y esta es la ultima linea

$ **grep** "esta" archivo_demo — Me saca todas las lineas en las que exista la palabra 'esta'
$ **grep** -i "esta" archivo_demo — Me saca todas las lineas en las que exista la palabra 'esta' este en mayúsculas o minúsculas indistintamente
$ **grep** -c "esta" archivo_demo — Saca cuantas ocurrencias de la palabra 'esta' hay, en nuestro caso **3**
$ **grep** -ic "esta" archivo_demo — Me saca todas las lineas en las que existe la palabra 'esta' sea mayúscula o minúscula, en este caso **5**

head Te visualiza la 1ª parte del contenido de un archivo.

tail 'nombre archivo' Te visualiza el final del contenido de un archivo.

mv mueve archivos y también los re-nombra

- **mv** origen destino

En el directorio actual existe un archivo llamado arch1.txt y lo quiero llevar al dir. /usr/doc

- $**mv** arch1.txt /usr/doc

En el directorio actual existe un archivo llamado arch1.txt y lo quiero llevar al directorio /user/doc y ademas le quiero poner de nombre archivoNuevo.txt en lugar de arch1.txt

- $ mv arch1.txt /usr/doc/archivoNuevo.txt

ps visualiza los procesos que se estan ejecutando, seria:$ **ps** aux.

- ps -a muestra todos los procesos
- ps -x muestra todos los procesos del usuario
- ps -u muestra el usuario que ha lanzado el proceso

wc

-c archivo cuenta el nº byte.
-l archivo cuenta el nº lineas.
-w archivo cuenta el nº palabras

wc -l /etc/passwd cuenta el numero de lineas que tiene el fichero passwd.

date

%a	dia semana abreviado
%A	dia semana completo
%b	nombre mes abreviado
%B	nombre mes completo
%d	dia mes
%m	nº mes
%H	hora, formato 24h
%M	minutos
%S	segundos

$ **date** +"%A %d %B" sale: Sunday 08 April
$ **echo** "la fecha de hoy es 'date +%D' " sale: la fecha de hoy es 08/21/02

find Busca ficheros

$ **find** /home/carlos -name "*.png" donde /home/carlos es donde busca y -name "*.png" indica los ficheros con extensión png.

passwd para cambiar la contraseña de usuario

rm para borrar directorios o archivos

- **rm** arch1.txt
- **rm** -r borra directorios también
- **rm** -i pregunta antes de borrar los archivos
- **rm** -f borra los archivos SIN preguntar
- **rm** -r dir1/dir2/dir3 borro el dir3, luego dir2 y luego dir1

Borra los ficheros cancion1 y cancion2
rm cancion1 cancion2

rmdir dir1 dir2 para borrar directorios con este comando, los directorios tienen que estar vacíos

cut filtra

- **cut** -c 3-9 /etc/passwd
- **cut** -c d ":" -f4 /etc/passwd

sed

- **sed** "3d" canciones En la salida borra la 3ª linea
- **sed** "a\Hola" canciones Después de cada linea se escribe Hola en una linea nueva
- **sed** -n 2p canciones Solo muestra la 2ª linea de canciones
- **sed** -n 3,9p canciones Solo muestra las lineas 3 a la 9 de canciones
- **sed** -n '3,$p' canciones Solo muestra de la linea 3 al final
- **sed** -n /^ho/ canciones Solo muestra las lineas que comienzan por ho
- **sed** -n '/^t, $p' canciones Muestra desde la 1ª linea que comience por t hasta la ultima linea
- **sed** 's/antes/despues/g' canciones Sustituye todas las palabras que ponga 'antes' por 'despues' en todo el fichero

- **sed** 's/antes/despues/' canciones Sustituye el 1er 'antes' por 'despues' en cada linea
- **sed** '1, 6 s/antes/despues/g' canciones Sustituye todos los 'antes' por 'despues' pero solo en las lineas de la 1 a la 6

halt permite suspender el sistema

poweroff permite apagar el sistema

shutdown permite echar abajo el sistema

- **shutdown** now Lo apaga inmediatamente
- **shutdown** 20:00 Lo apaga a las 8
- **shutdown** +10 Lo apaga en 10 minutos

chmod permite cambiar permisos

Si haces ls -l te sale por ejemplo:

- -rw-r--r-- 1 alberto staff 467010 13 sep 2010 prueba.txt
  Se trata de un fichero (el primer '–' lo indica), si fuera un directorio seria una 'd'
  rw- permisos para el propietario (alberto)
  r-- permisos para el grupo al que pertenece el fichero, que en este caso es 'staff'
  r-- permisos para aquellos que no son el propietario ni el grupo al que pertenece el fichero

Si quiero cambiar los permisos de ese fichero usamos **chmod** , se puede usar de forma octal, en este caso

- permiso lectura (r) vale 4 r-- = 100
- permiso escritura (w) vale 2 -w- = 010
- permiso ejecución (x) vale 1 --x = 001

Si al fichero le queremos dar todos los permisos al propietario, de lectura al grupo y al resto nada haría:

- **chmod** 740 prueba.txt (111 100 000)

Si al fichero le queremos dar todos los permisos al propietario, grupo y al resto haría:

- **chmod** 777 prueba.txt (111 111 111)

La otra forma de dar permisos se indica de la siguiente manera a quien se los queremos dar:

- u propietario del fichero
- g grupo al que pertenece el fichero
- o aquellos que no son el propietario
- a a todos
- + se añaden permisos
- - se quitan permisos
- = se igualan permisos

Quiero quitar al grupo el permiso de ejecución

- **chmod** g -x prueba.txt

Quiero dar todos los permisos al propietario

- **chmod** u +rwx prueba.txt

Quiero al grupo dar el permiso de escritura y al resto quitar el permiso de ejecucion

- **chmod** g +w, o-x prueba.txt

MÁS EJERCICIOS

A) Obtener un listado de los ficheros del directorio /home/iraide en un fichero llamado LISTADO
ls -la /home/iraide > LISTADO
Explicación: Redirecciona la salida de lo que me visualiza ls -la /home/iraide, al fichero LISTADO

B) Obtén el numero de procesos que se esta ejecutando
ps | **wc** -l
Explicación: **ps** visualiza los procesos que se están ejecutando y **wc** -l cuenta el nº de lineas

C) Comprobar si el usuario pepe esta conectado
who | **grep** '^pepe'
Explicación: con **who** salen los que están conectados y **grep** '^pepe' busca si entre los conectados esta pepe

D) Crear un fichero vació llamado NADA
touch NADA

E) Obtén la lista de ficheros del directorio actual obviando los posibles errores
ls -la 2>/dev/null
Explicación: con 2>/dev/null le decimos que obvie errores

F) Convierte el fichero NADA en un fichero ejecutable solamente por el propietario
chmod u+x NADA

G) Añade adiós al final del fichero NADA
echo 'adios' >> NADA
Añadir lo indicamos con >>

H) Muestra el directorio de trabajo del usuario 'pepe'
cat /etc/passwd | **grep** 'pepe' | **cut** -d ':' -f6
Explicación: El fichero donde salen todos los usuarios, es /etc/passwd. Entonces debo hacerme con el contenido de etc/passwd, después buscar la linea donde sale el usuario pepe y de esa linea hacer un filtro para quedarme con el 6º campo que es el que indica el directorio de trabajo

I) Muestra las 5 primeras lineas del fichero /etc/inicial
head -5 /etc/inicial

J) Muestra el nº de palabras de /etc/inicial
wc -w /etc/inicial

K) Borra el directorio /practica/mio y todo su contenido pidiendo confirmación
rm -ri /practica/mio

L) Muestra el nº directorios que cuelgan del actual
ls -la | **grep** '^d' | **wc** -l
Explicación: ls -la lista los directorios que cuelgan del actual y **grep** '^d' filtra las lineas que empiezan por d (que indica que son directorios) y **wc** -l saca cual es el nº de lineas

M) Muestra el contenido del fichero oculto HOLA
cat .HOLA