Grupos

El objetivo de los grupos es dar o restringir permisos sobre algunos archivos a ciertos usuarios. Por ejemplo un archivo llamado **apunte.txt** que pertenezca al grupo profesores que tenga permiso de lectura para el grupo y no para otros usuarios, podrá ser leído únicamente por el dueño y por usuarios que pertenezcan al grupo profesores. Cada usuario tiene un grupo principal (puede especificarse durante la creación con la opción ***GID***, (-***g nombregrupo de useradd***), puede pertenecer a diversos grupos y si conoce la clave de algún grupo con clave puede volverse miembro durante la sesión.

Los programas relacionados son :

groupadd	*Para agregar un grupo.*
groupdel	*Para borrar un grupo.*
groupmod	*Para modificar un grupo.*
groups	*Un usuario puede ver los grupos a los que pertenece con este programa.*
newgrp	*Para cambiarme a otro grupo (al que debe pertenecer),*

Esto lo que hace es modificar el archivo ***/etc/group***.

Ejemplo:

```
# groupdel pablo
```

Opciones de **groupadd** :

-g , --gid	*Le indicamos que número de GID principal utilizara.*
-h	*Ayuda.*

Ejemplo:

```
# groupadd pablo
```

Opciones de **groupmod** :

-g , --gid	*Le indicamos que número de GID principal utilizara.*
-n, --new-name	*Cambia el nombre del grupo viejo por el nuevo.*
-h	*Ayuda.*

Ejemplo:

```
# groupmod  -n pablonuevo pablo
```

Permisos

Al realizar un listado veremos los permisos de los directorios y archivos:

```
# ls -l /

        drwxr-xr-x   2 root     root      4096 may 24 10:19 bin
        drwxr-xr-x   4 root     root      1024 jun  3 16:04 boot
        drwxr-xr-x  18 root     root      3640 jun 14 09:23 dev
        drwxr-xr-x 170 root     root     12288 jun 14 13:20 etc
        ...
```

Tomamos como ejemplo **/bin** vemos que tiene al principio **drwxr-xr-x**, vemos que luego de la letra **d** (directorio), viene luego tres grupos de con **rwx r-x** y **r-x**. Vemos que el primer grupo corresponde al dueño del archivo, el siguiente al grupo y el ultimo a otros grupos que no sea el principal.

Todos los ficheros y directorios en un sistema **UNIX** tienen asociado un número compuesto de cuatro cifras en octal. Los tres dígitos menos significativos especifican los permisos que tienen los usuarios sobre ese fichero (lectura (**r**), escritura (**w**) y ejecución (**x**) para el usuario, los usuarios pertenecientes al grupo o para otros):

Lectura (r)	***Archivo** : Poder acceder a los contenidos de un archivo.* ***Directorio** : Poder leer un directorio, ver qué ficheros contiene.*
Escritura (w)	***Archivo** : Poder modificar o añadir contenido de un archivo.* ***Directorio** : Poder borrar o mover ficheros en un directorio.*
Ejecución (x)	***Archivo** : Poder ejecutar un programa binario o guión de shell.* ***Directorio** : Poder entrar en un directorio.*

Por ejemplo para sacar permisos de lectura a los usuarios en archivos de configuración de un servidor.

Octal Número	Binario Número	Permisos	Descripción
0	0	---	Ningún permiso garantizado.
1	1	--x	Ejecutable.
2	10	-w-	Modificable.
3	11	-wx	Modificable/Ejecutable.
4	100	r--	Legible.
5	101	r-x	Legible/Ejecutable.
6	110	rw-	Legible/Modificable.
7	111	rwx	Legible/Modificable/Ejecutable.

Los permisos para estos 3 tipos de usuarios puede estar dado por una cadena de 9 caracteres.

Octal Número	Dueño Columna	Grupo Columna	Otros Columna	Completo Código
0777	rwx	rwx	rwx	rwxrwxrwx
0755	rwx	r-x	r-x	rwxr-xr-x
0700	rwx	---	---	rwx------
0666	rw-	rw-	rw-	rw-rw-rw

```
sst rwx rwx rwx
421 421 421 421
S   U   G   O

S=SUID, SGID y Sticky Bit
U=Usuario
G=Grupo
O=Otros
```

También tenemos el ***umask*** (abreviatura de **u**ser **mask**, máscara de usuario) es una orden y una función en entornos POSIX que establece los permisos por defecto para los nuevos archivos y directorios creados por el proceso actual.

Los sistemas Unix modernos permiten que las máscaras se especifiquen de dos modos:

- Un permiso por defecto, también llamado máscara simbólica. Por ejemplo, u=rwx,g=rwx,o=
- Un número en octal que controla qué permisos se enmascararán (no se establecerán) para cualquier nuevo archivo, por ejemplo, 007.

En ambos casos debe tenerse en cuenta que la mayoría de los sistemas Unix no permiten que nuevos archivos sean creados con permisos de ejecución activados, independientemente de la máscara.

Ejemplo:

```
# umask

  0022
```

Por defecto la creación de archivos es : 666 – umask (0022) = 644.
Por defecto la creación de directorios es : 777 – umask (0022) = 755.

Comando: *chmod*

Para cambiar la máscara de permisos de un archivo o directorio, usamos el comando ***chmod***. La sintaxis de este comando es la siguiente:

```
chmod [opciones] archivo|directorio
```

Opciones :

-v	*Muestra lo que realiza.*
-R	*Recursivo.*

Se puede usar números y letras para indicar los permisos.

- Con letras [ugo][+-=][rwx] (u=usuario,g=grupo,o=otros).
- Con números [0-7][0-7][0-7]

Ejemplo:

```
# chmod u+x,g-w,o-w archivo1
# chmod u=rx archivo1
```

El signo (+) significa que agrega permisos, y el signo (-) que saca permisos, si no ponemos ninguno de estos dos signos reemplaza lo que ponemos por lo que hay.

Sticky bit

El *sticky bit* tiene significado cuando se aplica a directorios. Si el *sticky bit* está activo en un directorio, un usuario sólo puede borrar archivos que son de su propiedad o para los que tiene permiso de escritura, incluso cuando tiene acceso de escritura al directorio. En un directorio evita que un usuario que no sea el dueño del directorio pero que tenga permiso de escritura, pueda borrar o renombrar archivos que no le pertenecen. Esto está pensado para directorios como ***/tmp***, que tienen permiso de escritura global, pero no es deseable permitir a cualquier usuario borrar los archivos que quiera. El *sticky bit* aparece como '***t***' en los listados largos de directorios. Ejemplo:

```
# ls -ld /tmp

        drwxrwxrw**t** 3 root root 4096 jun 14 14:17 /tmp
```

Esto se agrega con el comando **chmod** con el valor **1** y agrega una **t** minúscula que tapa el valor de la **x**, si es mayúscula significa que en su lugar no habia una **x**.

```
# chmod **1**777 /tmp
```

Para sacarlo :

```
# chmod o-t /tmp
```

Atributo SGID: *(Para Archivos)*

Este ***bit*** describe permisos al grupo del archivo. Cuando el atributo ***SGID*** (poner id del grupo) está activo en los permisos del grupo, y ese archivo es ejecutable, los procesos que lo ejecutan obtienen acceso a los recursos del sistema basados en el grupo que crea el proceso (no el grupo que lo lanza). Resumiendo esta explicación, podemos decir : el ***bit SGID*** empleado con archivos ejecutables cambia la identificación del grupo por la del dueño del archivo para otros.

Esto se agrega con el comando **chmod** con el valor **2** y agrega una **s** minúscula que tapa el valor de la **x**, si es mayúscula significa que en su lugar no habia una **x**.

```
# chmod 2755 mi-programa
```

Para sacarlo :

```
# chmod g-s mi-programa
```

Atributo SUID: *(Para Archivos)*

Este ***bit*** describe permisos al usuario del archivo. Cuando el atributo ***SUID*** (poner id de usuario) está activo en los permisos del propietario, y ese archivo es ejecutable, los procesos que lo ejecutan obtienen acceso a los recursos del sistema basados en el usuario que crea el proceso (no el usuario que lo lanza). Resumiendo esta explicación, podemos decir : el ***bit SUID*** empleado con archivos ejecutables cambia la identificación del usuario por la del dueño del archivo para otros.

Esto se agrega con el comando **chmod** con el valor **4** y agrega una **s** minúscula que tapa el valor de la **x**, si es mayúscula significa que en su lugar no habia una **x**.

```
# chmod 4755 mi-programa
```

Para sacarlo :

```
# chmod u-s mi-programa
```

Comando: *chrgp*

Con este comando me permite cambiar solo el grupo del archivo o directorio.

```
chgrp grupo archivos
```

Opciones :

-v	*Muestra lo que realiza.*
-R	*Recursivo.*

Ejemplo:

```
# chgrp users archivo.txt
# chgrp -R users directorio-pablo
```

Comando: *chown*

Con este comando me permite cambiar el dueño y/o grupo del archivo o directorio.

```
chown dueño[:grupo] archivos
```

Opciones :

-v	*Muestra lo que realiza.*
-R	*Recursivo.*

Ejemplo:

```
# chown nobody archivo.txt
# chown nobody:nogroup archivo.txt
# chown nobody:nogroup -R directorio-pablo
```

***Comando:* chattr**

Cambia los atributos de los ficheros en un sistema de ficheros ***ext2/ext3/ext4***. Esta incluido en el paquete ***e2fsprogs***.

```
chattr [opciones] [modo] fichero
```

Opciones :

-v	*Muestra lo que realiza.*
-R	*Recursivo.*

Atributos :

- + Se usa para añadir atributos.
- - Se usa para sacar atributos.
- = Se usa para especificar los atributos.

Algunos atributos son :

- **A** evita que se modifique el campo **atime** al acceder a un fichero.
- **a** sólo permite abrir el fichero para añadir datos.
- **c** el fichero se guarda automáticamente comprimido por el kernel.
- **D** cuando un directorio es modificado,los cambios son escritos síncronamente.
- **d** excluye al fichero para ser respaldado por **dump**.
- **i** impide modificar,eliminar,renombrar el fichero y también enlazarlo.
- **s** al borrar un fichero con este atributo,sus bloques son rellenados con ceros.
- **S** cuando un fichero es modificado,los cambios son escritos síncronamente.
- **u** cuando un fichero es eliminado,su contenido es guardado.

Nota: **D** es equivalente a la opción de montaje «dirsync» **S** es equivalente a la opción de montaje «sync».

Ejemplo:

```
# chattr +i mi-archivo
# chattr -i mi-archivo
```

Comando: lsattr

Muestra los atributos de los ficheros en un sistema de ficheros **ext2/ext3/ext4**.

```
lsattr [opciones] fichero
```

Opciones :

-v	*Muestra lo que realiza.*
-R	*Recursivo.*
-a	*Muestra todos los ficheros de un directorio.*

Ejemplo:

```
# lsattr mi-archivo

 ----i------------e- a
```

Listas de acceso con ACL

Los **ACLs** permiten otorgar privilegios de acceso adicionales.
El propietario de un archivo puede, gracias a los **ACLs**, otorgar privilegios a uno o más usuarios y/o grupos que se sustituirán a los privilegios de acceso de base.

Con los **ACLs** es posible otorgar privilegios a un usuario que no es parte del grupo sin modificar los privilegios de los otros.
Igualmente se pueden autorizar privilegios de acceso a un grupo de usuarios que no pertenecen al grupo del archivo.
No hay límites en lo que respecta al número de **usuarios** o **grupos** a adicionar con los **ACLs**.

El respaldo hecho con **tar** no memoriza los **ACLs** definidos.

Las **ACL** cumplen con el estándar **POSIX** (**P**ortable **O**perating **S**ystem for Un**IX**).

Un **ACL** está compuesto de varias entradas de tipo **ACL**. Una entrada especifica los permisos de acceso a un objeto asociado a un usuario o grupo de usuarios utilizando una combinación de privilegios tradicionales **r**, **w** y **x**.

Es decir **una entrada ACL** esta compuesta de un:

- **tag** que especifica una identidad de usuario
- **tag opcional** de usuarios o grupos
- la **lista** de permisos otorgados

Hay que bajar el paquete **acl**.

```
# apt-get install acl
```

El kernel de la familia 2.6 incluye el soporte de acl para ext2/ext3/ext4, jfs y xfs. Si el kernel no incluye el soporte hay que compilar el kernel.

Para verificarlo :

```
# uname -a && grep -i 'acl' /boot/config-$(uname -r)

        CONFIG_EXT2_FS_POSIX_ACL=y
        CONFIG_EXT3_FS_POSIX_ACL=y
        CONFIG_EXT4_FS_POSIX_ACL=y
        CONFIG_REISERFS_FS_POSIX_ACL=y
        CONFIG_JFS_POSIX_ACL=y
        CONFIG_FS_POSIX_ACL=y
        CONFIG_XFS_POSIX_ACL=y
        CONFIG_OCFS2_FS_POSIX_ACL=y
        CONFIG_BTRFS_FS_POSIX_ACL=y
        CONFIG_GENERIC_ACL=y
        CONFIG_TMPFS_POSIX_ACL=y
        CONFIG_JFFS2_FS_POSIX_ACL=y
        CONFIG_NFS_V3_ACL=y
        CONFIG_NFSD_V2_ACL=y
        CONFIG_NFSD_V3_ACL=y
        CONFIG_NFS_ACL_SUPPORT=m
```

Nota: Nosotros al implementar acl tenemos que tener un filesystem aparte.

Por ejemplo si en linea de comandos por el momento queremos montar una partición con **acl** realizamos lo siguiente :

```
# mount /dev/particion -o defaults,acl /punto-de-montaje
```

Si queremos que sea permanente tenemos que editar el archivo **/etc/fstab**.

```
# vi /etc/fstab

  /dev/particion    /punto-de-montaje   ext4     defaults,acl   0    2
```

Una vez que tenemos montada nuestra partición podremos utilizar los siguiente comandos:

setfacl

El comando **setfacl** nos permite **posicionar** los **ACLs**. Mostraremos solamente las opciones **-m**, **-x**, **-L** y **-R**.

Opciones :

-m, --modify	*Modifica los **ACL** de un archivo o directorio*
-x, --remove	*Elimina las entradas **ACLs** .*
-b, --remove-all	*Elimina todas las entradas **ACLs**.*
-L, --logical	*Enrutamiento de los enlaces simbólicos.*
-R, --recursive	*Aplicación de los **ACLs** de forma recursiva.*
-d	*Asigna los permisos por defecto.*
-k	*Borra los permisos por defecto.*

Ejemplo:

Asigna/modifica (**-m**) recursivamente (**-R**) permisos por defecto (**-d**) al grupo: "**grupo1**", como lectura, escritura, exploración/ejecucion (**rwx**) a partir de: "**/dir**"

```
# setfacl -R -d -m group:grupo1:rwx /dir
```

Al listarlo veremos que agrega un '+' esto significa que tiene **ACL**.

```
# ls -ld /dir

  drwxrwxr-x+ 2 root root 4096 jun 21 10:09 dir
```

Asigna/modifica (**-m**) permisos de: lectura, exploración (**rx**) al grupo: "**grupo2**", en el directorio: "**/dir/sub-dir**"

```
# setfacl -m group:grupo2:rw /dir/sub-dir
```

Asigna/modifica (**-m**) permisos de: lectura, escritura y ejecución (**rwx**) al usuario: "**usuario1**", para el archivo: "**/dir/sub-dir/script.sh**"

```
# setfacl -m user:usuario1:rwx /dir/sub-dir/script.sh
```

Para eliminar un usuario :

```
# setfacl -x user:usuario1 /dir/sub-dir/script.sh
```

getfacl

El comando getfacl nos permite mostrar los **ACLs**.

Opciones :

-L	*El enrutamiento de los enlaces simbólicos*
-R	*Permite ver los **ACLs** de forma recursiva*

Ejemplo:

```
# getfacl /mi-directorio/archivo
```

Guardar los **acl** del directorio y su contenido en forma recursiva, para luego restaurarlo.

```
# getfacl  -R /mi-directorio > /mi-directorio.acl
# setfacl -m group:grupo2:rw /mi-directorio
# setfacl --restore=/mi-directorio.acl
```

Quota de disco

En sistemas con muchos usuarios se presenta un problema, el espacio en disco duro, los usuarios guardan y guardan cosas en el disco duro y si no existe un limite para esto el espacio de disco duro se terminará, esto posiblemente se solucione agregando mas discos duros, pero al final resultará lo mismo, el disco se llenará de nueva cuenta.

Las cuotas de disco no son más que un limite para los usuarios que les indica que cantidad de espacio en disco pueden almacenar y en caso de alcanzar este limite no podrán almacenar más cosas.

Para poder aplicar cuotas de disco a los usuarios nuestro kernel debe soportar cuotas, es decir debió haber sido compilado con soporte para cuotas, por defecto los sistemas debian y ubuntu traen el kernel compilado con ésta opción, lo siguiente es ejecutar el comando:

Bajamos el paquete quota :

```
# apt-get install quota
```

Las cuotas puede ser utilizadas dentro de un filesystem por ejemplo el **home**.

```
# grep home /etc/fstab

 /dev/mapper/VolGroup00-lvhome /home          ext4    defaults        0       2
```

Editamos el archivo **/etc/fstab** y agregamos lo siguiente :

```
# vi /etc/fstab

 /dev/mapper/VolGroup00-lvhome /home          ext4    defaults,usrquota       0       2
```

- **usrquota** = Cuota para usuario determinados.
- **grpquota** = Cuota para grupos de usuarios (todos los que pertenecen a un grupo determinado se le da un espacio de disco).

Se puede agregar las dos opciones si queremos.

Luego volvemos a remontar el filesystem :

```
# mount -o remount /home
```

Verificamos :

```
# mount | grep home

  /dev/mapper/VolGroup00-lvhome on /home type ext4 (rw,usrquota)
```