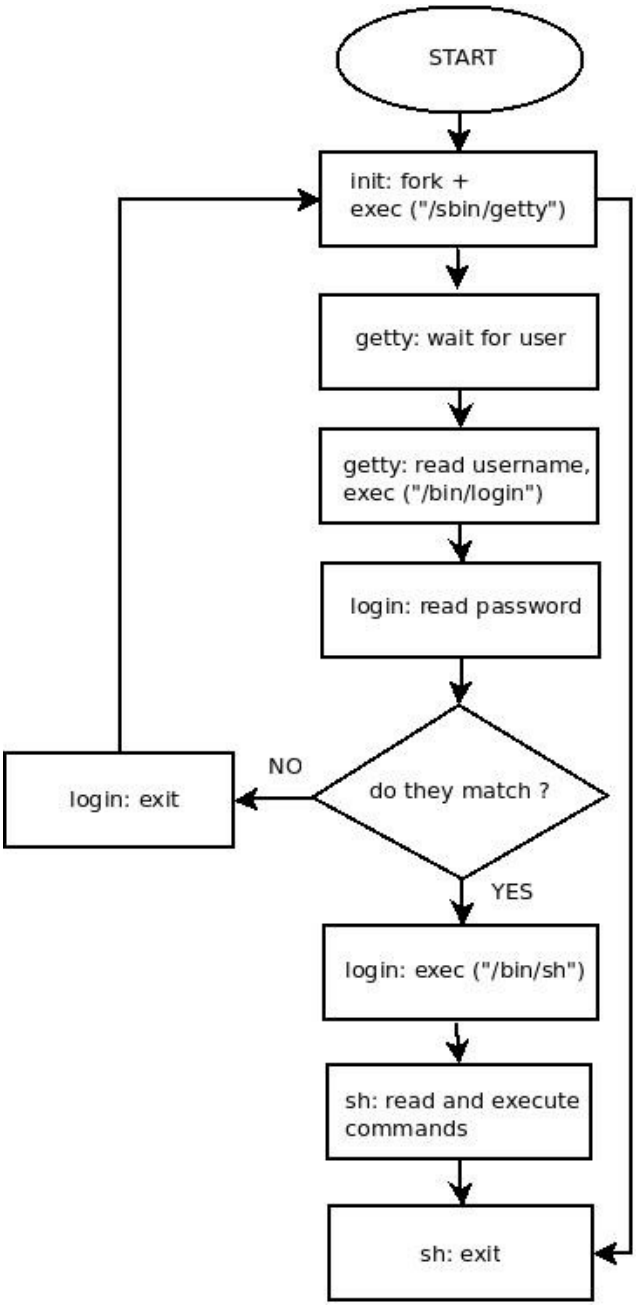
Capítulo 11

Usuario, Grupos y Permisos

Usuarios y Grupos

Init, lee el archivo **/etc/inittab** para ver cuantas terminales hay, y mediante **fork** crea un proceso hijo para cada terminal. Cada proceso hijo ejecuta **getty**, que es quien imprime el **prompt del login** y obtiene el nombre de usuario. Cuando el usuario entra al sistema, se ejecuta login con el nombre como argumento y se ejecuta el s**hell** de ese usuario. El **shell** del usuario se especifica en el **/etc/passwd**. El s**hell** espera por un comando y luego lanza un **FORK** y un **EXEC** por cada comando.



Todos los usuarios se encuentran en ***/home***, salvo **root** que se encuentra en ***/root***. La estructura del directorio ***/home/nombre-usuario*** por ejemplo si doy de alta al usuario **pepe** sería : ***/home/pepe*** donde tendrá sus respectivo documentos, fotos, mp3, etc.

Las contraseñas se guardan en secreto, el sistema encripta la contraseña usando un algoritmo desarrollado **NSA** llamado **DES** y deja el valor encriptado visible. Para romper la clave la forma es mediante **fuerza bruta** mediante la utilización de un diccionario. Pero solo el usuario **root** puede ver el archivo donde se guarda las contraseñas que es ***/etc/shadow***.

Cuando el usuario se logonea, la contraseña que introduce se encripta. El valor encriptado se compara con la entrada de la contraseña del usuario. Si ambos valores coinciden se permite la entrada del usuario.

Una buena contraseña tiene que ser letras mayúscula, minúsculas, números y signos.

Cada vez que creamos un usuario esto se agrega en el archivo ***/etc/passwd***, el grupo que pertenece en ***/etc/group*** y su contraseña encriptada ***/etc/shadow***.

/etc/passwd

Veremos que los campos están separados con (**:**), como bien dijimos anteriormente aca se guardan la información del usuario dado de alta.

```
# more /etc/passwd

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

Descripción de los distintos campos :

Campo 1	*Es el nombre del usuario, identificador de inicio de sesión (**login**). Tiene que ser único.*
Campo 2	*La '**x**' indica la contraseña encriptada del usuario, además también indica que se está haciendo uso del archivo , si no se hace uso de este archivo, este campo se vería algo así como: ''.*
Campo 3	*Número de identificación del usuario (**UID**). Tiene que ser único. **0** para **root**, generalmente las cuentas o usuarios especiales se numeran del 1 al 100 y las de usuario normal del 101 en delante, en las distribuciones mas recientes esta numeración comienza a partir del 500.*
Campo 4	*Numeración de identificación del grupo (**GID**). El que aparece es el número de grupo principal del usuario, pero puede pertenecer a otros, esto se configura en .*
Campo 5	*Comentarios o el nombre completo del usuario.*
Campo 6	*Directorio de trabajo (**Home**) donde se sitúa al usuario después del inicio de sesión.*
Campo 7	***Shell** que va a utilizar el usuario de forma predeterminada.*

/etc/shadow

Anteriormente (en sistemas Unix) las contraseñas cifradas se almacenaban en el mismo archivo de **/etc/passwd**, el problema es que '**passwd**' es un archivo que puede ser leído por cualquier usuario del sistema, aunque solo puede ser modificado por **root**. Con cualquier computadora potente de hoy en día, un buen programa de descifrado de contraseñas y paciencia es posible "crackear" contraseñas débiles (por eso la conveniencia de cambiar periódicamente la contraseña de **root** y de otras cuentas importantes). El archivo '**shadow**', resuelve el problema ya que solo puede ser leído por **root**. Considérese a '**shadow**' como una extensión de '**passwd**' ya que no solo almacena la contraseña encriptada, sino que tiene otros campos de control de contraseñas.

```
# more /etc/shadow

        root:ghy675gjuXCc12r5gt78uuu6R:10568:0:99999:7:7:-1::
```

Descripción de los distintos campos :

Campo 1	*Nombre de la cuenta del usuario.*
Campo 2	*Contraseña cifrada o encriptada, un '*' indica cuenta de 'nologin'.*
Campo 3	*Días transcurridos desde el 1/ene/1970 hasta la fecha en que la contraseña fue cambiada por última vez.*
Campo 4	*Número de días que deben transcurrir hasta que la contraseña se pueda volver a cambiar.*
Campo 5	*Número de días tras los cuales hay que cambiar la contraseña. (-1 significa nunca). A partir de este dato se obtiene la fecha de expiración de la contraseña.*
Campo 6	*Número de días antes de la expiración de la contraseña en que se le avisará al usuario al inicio de la sesión.*
Campo 7	*Días después de la expiración en que la contraseña se inhabilitara, si es que no se cambio.*
Campo 8	*Fecha de caducidad de la cuenta. Se expresa en días transcurridos desde el 1/Enero/1970 (epoch).*
Campo 9	*Reservado.*

/etc/group

Este archivo guarda la relación de los grupos a los que pertenecen los usuarios del sistema, contiene una línea para cada usuario con tres o cuatro campos por usuario:

```
# more /etc/group

        root:x:0:root
```

Descripción de los distintos campos :

Campo 1	*Indica el usuario.*
Campo 2	*'***x**' indica la contraseña del grupo, que no existe, si hubiera se mostraría un 'hash' encriptado.
Campo 3	*Es el Group ID (**GID**) o identificación del grupo.*

Campo 4 *Es opcional e indica la lista de grupos a los que pertenece el usuario*

Actualmente al crear al usuario con se crea también automáticamente su grupo principal de trabajo **GID**, con el mismo nombre del usuario. Es decir, si se añade el usuario 'sergio' también se crea el **/etc/group** el grupo 'sergio'. Aun así, existen comandos de administración de grupos que se explicarán más adelante.

Shell

Por defecto la **shell** que se utiliza por defecto es **bash**. GNU/Linux viene con varios **shell** que podemos elegir entre una lista.

```
# cat /etc/shells

 # /etc/shells: valid login shells
 /bin/csh
 /bin/sh
 /usr/bin/es
 /usr/bin/ksh
 /bin/ksh
 /usr/bin/rc
 /usr/bin/tcsh
 /bin/tcsh
 /usr/bin/esh
 /bin/dash
 /bin/bash
 /bin/rbash
```

Nosotros podemos bajar otros tipo de **shell** como por ejemplo **scponly** que en este caso restringe el uso de **scp** y **sftp**.

Estos tipos de **shell** al crear el usuario o mediante un usuario ya existente se lo indicamos.

pwconv y pwunconv

El comportamiento por defecto de todas las distros modernas de GNU/Linux es activar la protección extendida del archivo , que (se insiste) oculta efectivamente el 'hash' cifrado de la contraseña de .

Pero si por alguna bizarra y extraña situación de compatibilidad se requiriera tener las contraseñas cifradas en el mismo archivo de se usaría el comando :

```
# more /etc/passwd
 root:x:0:0:root:/root:/bin/bash
 sergio:x:501:500:Sergio González:/home/sergio:/bin/bash
 ...
 (La 'x' en el campo 2 indica que se hace uso de /etc/shadow)

# more /etc/shadow

 root:ghy675gjuXCc12r5gt78uuu6R:10568:0:99999:7:7:-1::
 sergio:rfgf886DG778sDFFDRRu78asd:10568:0:-1:9:-1:-1::
```

```
# pwunconv
# more /etc/passwd

 root:ghy675gjuXCc12r5gt78uuu6R:0:0:root:/root:/bin/bash
 sergio:rfgf886DG778sDFFDRRu78asd:501:500:Sergio González:/home/sergio:/bin/bash
 …

# more /etc/shadow
 /etc/shadow: No such file or directory

(Al ejecutar pwunconv, el archivo shadow se elimina y las contraseñas cifradas 'pasaron' a passwd)
```

En cualquier momento es posible reactivar la protección de **shadow**:

```
# pwconv

# ls -l /etc/passwd /etc/shadow

 -rw-r--r-- 1 root root 1106 2007-07-08 01:07 /etc/passwd
 -r--------  1 root root  699 2009-07-08 01:07 /etc/shadow
```

Se vuelve a crear el archivo, además nótese los permisos tan restrictivos (**400**) que tiene este archivo, haciendo sumamente difícil que cualquier usuario que no sea **root** lo lea.

/etc/login.defs

En el archivo de configuración están definidas las variables que controlan los aspectos de la creación de usuarios y de los campos de usadas por defecto. Algunos de los aspectos que controlan estas variables son:

- Número máximo de días que una contraseña es válida **PASS_MAX_DAYS**.
- El número mínimo de caracteres en la contraseña **PASS_MIN_LEN**.
- Valor mínimo para usuarios normales cuando se usa **UID_MIN**.
- El valor por defecto **UMASK**.
- Si el comando debe crear el directorio **home** por defecto **CREATE_HOME**.

Basta con leer este archivo para conocer el resto de las variables que son auto descriptivas y ajustarlas al gusto. Recuérdese que se usaran principalmente al momento de crear o modificar usuarios con los comandos y que en breve se explicaran.

/etc/skel

Es un directorio, contiene todos los archivos que se copiaron al crear los usuarios (***.bashrc, .bash_profile, .profile***).

Cualquier cosa que pongamos acá al crear un nuevo usuario se copiara dentro de su **home**.

.bash_profile	*Se ejecuta cuando el usuario inicia la sesión.*
.bash_logout	*Se ejecuta cuando el usuario abandona la sesión.*
.bashrc	*Se ejecuta cuando el usuario inicia la sesión.*

Comando: *adduser*

Con este comando veremos que nos pedirá los datos siendo un asistente :

```
# adduser nombre-usuario
```

Tenemos un archivo de configuración llamado ***/etc/adduser.conf*** donde ya tenemos varias variables configuradas por defecto que tomara el comando **adduser**.

```
# more /etc/adduser.conf

        # /etc/adduser.conf: `adduser' configuration.
        # See adduser(8) and adduser.conf(5) for full documentation.

        # The DSHELL variable specifies the default login shell on your
        # system.
        DSHELL=/bin/bash

        # The DHOME variable specifies the directory containing users' home
        # directories.
        DHOME=/home

        # If GROUPHOMES is "yes", then the home directories will be created as
        # /home/groupname/user.
        GROUPHOMES=no

        # If LETTERHOMES is "yes", then the created home directories will have
        # an extra directory - the first letter of the user name. For example:
        # /home/u/user.
        LETTERHOMES=no

        # The SKEL variable specifies the directory containing "skeletal" user
        # files; in other words, files such as a sample .profile that will be
        # copied to the new user's home directory when it is created.
        SKEL=/etc/skel

        # FIRST_SYSTEM_[GU]ID to LAST_SYSTEM_[GU]ID inclusive is the range for UIDs
        # for dynamically allocated administrative and system accounts/groups.
        # Please note that system software, such as the users allocated by the base-passwd
        # package, may assume that UIDs less than 100 are unallocated.
        FIRST_SYSTEM_UID=100
        LAST_SYSTEM_UID=999

        FIRST_SYSTEM_GID=100
        LAST_SYSTEM_GID=999

        # FIRST_[GU]ID to LAST_[GU]ID inclusive is the range of UIDs of dynamically
        # allocated user accounts/groups.
        FIRST_UID=1000
        LAST_UID=29999

        FIRST_GID=1000
```

```
LAST_GID=29999

# The USERGROUPS variable can be either "yes" or "no".  If "yes" each
# created user will be given their own group to use as a default.  If
# "no", each created user will be placed in the group whose gid is
# USERS_GID (see below).
USERGROUPS=yes

# If USERGROUPS is "no", then USERS_GID should be the GID of the group
# `users' (or the equivalent group) on your system.
USERS_GID=100

# If DIR_MODE is set, directories will be created with the specified
# mode. Otherwise the default mode 0755 will be used.
DIR_MODE=0751

# If SETGID_HOME is "yes" home directories for users with their own
# group the setgid bit will be set. This was the default for
# versions << 3.13 of adduser. Because it has some bad side effects we
# no longer do this per default. If you want it nevertheless you can
# still set it here.
SETGID_HOME=no

# If QUOTAUSER is set, a default quota will be set from that user with
# `edquota -p QUOTAUSER newuser'
QUOTAUSER=""

# If SKEL_IGNORE_REGEX is set, adduser will ignore files matching this
# regular expression when creating a new home directory
SKEL_IGNORE_REGEX="dpkg-(old|new|dist|save)"

# Set this if you want the --add_extra_groups option to adduser to add
# new users to other groups.
# This is the list of groups that new non-system users will be added to
# Default:
#EXTRA_GROUPS="dialout cdrom floppy audio video plugdev users"

# If ADD_EXTRA_GROUPS is set to something non-zero, the EXTRA_GROUPS
# option above will be default behavior for adding new, non-system users
#ADD_EXTRA_GROUPS=1


# check user and group names also against this regular expression.
#NAME_REGEX="^[a-z][-a-z0-9_]*\$"
```

Comando: *useradd*

Con este comando nosotros le tenemos que pasar los parámetros.

```
useradd [opciones] Nombre-usuario
```

Opciones :

-c comentario	*Para especificar un comentario para la nueva cuenta.*
-d directorio-del-usuario	*Para establecer el directorio de trabajo del usuario. Es conveniente, a fin de tener un sistema bien organizado, que este se localice dentro del directorio /**home**.*
-f fechaexpiracion	*Para establecer la fecha de expiración de la cuenta de usuario. Esta debe ingresarse en el siguiente formato: **AAAA-MM-DD**.*

-g grupoinicial	*Para establecer el grupo inicial al que pertenecerá el usuario. De forma predeterminada se establece como único grupo.* ***Nota:*** *El grupo asignado ya debe existir.*
-G grupo1,grupo2	*Para establecer grupos extra a los que pertenecerá el usurio. Se debe separar utilizando una coma y sin espacios. Esto es muy conveniente cuando se desea que el usuario tenga acceso a determinados recursos del sistema, como acceso a la unidad de disquetes, administración de cuentas PPP y POP.* ***Nota:*** *los grupos asignados deben existir.*
-m	*Para especificar que el directorio de trabajo del usuario debe ser creado, y dentro se copiarán los archivos especificados en* ***/etc/skel****. Para cambiar esto, hay que completar la opción -****m*** *con -****k <nuevodirskel>*** *donde* ***nuevodirskel*** *es la carpeta de la cual se sacarán los perfiles.*
-s shell	*Se utiliza para establecer la* ***shell*** *que podrá utilizar el usuario. De forma predeterminada, se establece* ***bash*** *como* ***shell*** *predeterminado.*
-u uid	*Se utiliza para establecer el* ***UID****, es decir, la* ***ID*** *del usuario, el número de identificación de usuario. Este debe ser único. Cuando se crea una cuenta de usuario por primera vez, generalmente se asignará 500 como* ***UID*** *del usuario. Los* ***UID*** *entre 0 y 99 son reservados para las cuentas de los servicios del sistema.*

Ejemplo:

```
# useradd -g 100 -d /home/pablo -s /bin/bash -G floppy,pppuser  -m pablo
# passwd pablo
```

Toma la configuración del archivo ***/etc/default/useradd***, este comando no nos pide que le indiquemos la **password** por eso mismo nosotros tendremos que ejecutar el comando **passowd** seguido del nombre del usuario..

Ejemplo :

```
# useradd -D

 GROUP=100
 HOME=/home
 INACTIVE=-1
 EXPIRE=
 SHELL=/bin/sh
 SKEL=/etc/skel
 CREATE_MAIL_SPOOL=no
```

Comando: *usermod*

Con este comando nosotros podremos modificar el usuario.

```
usermod [opciones] Nombre-usuario
```

Opciones :

-c comentario	*Para especificar un comentario para la nueva cuenta.*
-d directorio-del-usuario	*Para establecer el directorio de trabajo del usuario. Es conveniente, a fin de tener un sistema bien organizado, que este se localice dentro del directorio* ***/home****.*
-f fechaexpiracion	*Para establecer la fecha de expiración de la cuenta de usuario. Esta debe ingresarse en el siguiente formato:* ***AAAA-MM-DD****.*
-g grupoinicial	*Para establecer el grupo inicial al que pertenecerá el usuario. De forma predeterminada se establece como único grupo.* ***Nota:*** *El grupo asignado ya debe existir.*
-G grupo1,grupo2	*Para establecer grupos extra a los que pertenecerá el usurio. Se debe separar utilizando una coma y sin espacios. Esto es muy conveniente cuando se desea que el usuario tenga acceso a determinados recursos del sistema, como acceso a la unidad de disquetes, administración de cuentas PPP y POP.* ***Nota:*** *los grupos asignados deben existir.*
-m	*Para especificar que el directorio de trabajo del usuario debe ser creado, y dentro se copiarán los archivos especificados en* ***/etc/skel****. Para cambiar esto, hay que completar la opción* ***-m*** *con* ***-k <nuevodirskel>*** *donde* ***nuevodirskel*** *es la carpeta de la cual se sacarán los perfiles.*
-s shell	*Se utiliza para establecer la* ***shell*** *que podrá utilizar el usuario. De forma predeterminada, se establece* ***bash*** *como* ***shell*** *predeterminado.*
-u uid	*Se utiliza para establecer el* ***UID****, es decir, la* ***ID*** *del usuario, el número de identificación de usuario. Este debe ser único. Cuando se crea una cuenta de usuario por primera vez, generalmente se asignará 500 como* ***UID*** *del usuario. Los* ***UID*** *entre 0 y 99 son reservados para las cuentas de los servicios del sistema.*
-a, --append	*Agrega al usuario los grupos adicionales* ***-G*** *sino agrego esto reemplazo los que pongo por los que tiene.*
-L, --lock	*Pone frente de la contraseña* ***/etc/shadow*** *un* ***!*** *esto desactiva la contraseña del usuario aparte de bloquear de esta forma también tenemos que utilizar* ***EXPIRE_DATE****.*

Ejemplo:

```
# usermod -G www-data,users -a pablo
```

Comando: *userdel*

Con este comando podemos borrar un usuario determinado también podemos borrar si queremos su directorio **home**. Esto lo que hace es borrar dentro de los archivos **/etc/passwd**, **/etc/shadow** y **/etc/group**.

```
userdel [opciones] Nombre-usuario
```

Opciones :

-r	*Remueve el usuario (****/etc/passwd****,* ***/etc/shadow*** *y* ***/etc/group)*** *y su*

	directorio.
-f	Remueve el usuario y su directorio en forma forzada, es decir si el usuario esta logueado. Esto puede traer problemas al sistema.
	*Para establecer la fecha de expiración de la cuenta de usuario. Esta debe ingresarse en el siguiente formato: **AAAA-MM-DD**.*
-h	*Ayuda.*

Ejemplo:

En este ejemplo solo borra el usuario pero no borra su directorio.

```
# userdel pablo
```

```
# userdel -r pablo
```

Comando: *id*

Muestra el nombre y grupo del usuario en números (el ***UID*** y ***GID***) y a que grupos adicionales pertenecen.

```
id [opciones] [usuario]
```

Opciones :

-g, --group	*Imprime solamente el grupo.*
-G, --groups	Imprime los grupos adicionales.
-u, --user	*Imprime solamente el **UID**.*

Ejemplo:

```
# id pablo
```

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.*

Ejecución (x)	***Directorio** : Poder borrar o mover ficheros en un directorio.* ***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

        drwxrwxrwt 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 1777 /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)
```