LINUX

3

GESTIÓN DE USUARIOS

Tabla de contenido

Gestión de Usuarios

1. Introducción

Cada usuario del sistema debe tener un nombre de entrada (***login***) único. Con ello se le podrá identificar y se evitará que cualquier otro usuario pueda borrar ficheros de su propiedad. Del mismo modo cada usuario dispone de una clave de entrada o ***password*** que le liga a su cuenta de usuario. Cuando el sistema arranca con éxito nos pide que introduzcamos nuestro nombre de usuario y nuestra contraseña. Si vamos a iniciar una sesión como *superusuario* deberemos dar como login ***root***.

El superusuario es el usuario más potente del sistema, en el sentido de que tiene el control completo de los recursos de la computadora. Puede iniciar y desconectar el sistema, abrir o cerrar el acceso a cualquier archivo o directorio, borrar o cambiar parte del sistema, y en general, modificar la configuración del mismo. El indicativo de petición de orden del ***shell*** es para el administrador el símbolo **#**, mientras que para el resto de usuarios es normalmente el **$** (> en SuSe).

Existen unas normas aconsejables que debe aplicarse el administrador del sistema o *superusuario* y que se pueden resumir en los siguientes puntos:

- Mantener siempre el indicativo de *root* igual al carácter #, esto ayudará a recordar cuando se tiene acceso total al sistema.
- No dar la contraseña de superusuario a nadie que no necesite acceder al control total del sistema.
- Cambiar la contraseña de *root* frecuentemente, mantenerla hace al sistema más vulnerable.
- Limitar la actuación como *root* a labores de administración del sistema exclusivamente.
- Hacer el entorno de *root* diferente al entorno normal no usando los mismos ficheros de configuración como superusuario y como usuario normal y minimizando el uso de alias en la presentación *root*.
- Hacer la variable PATH tan corta como sea posible mientras sea práctica, no incluyendo los directorios propios de usuario en el PATH de *root*.

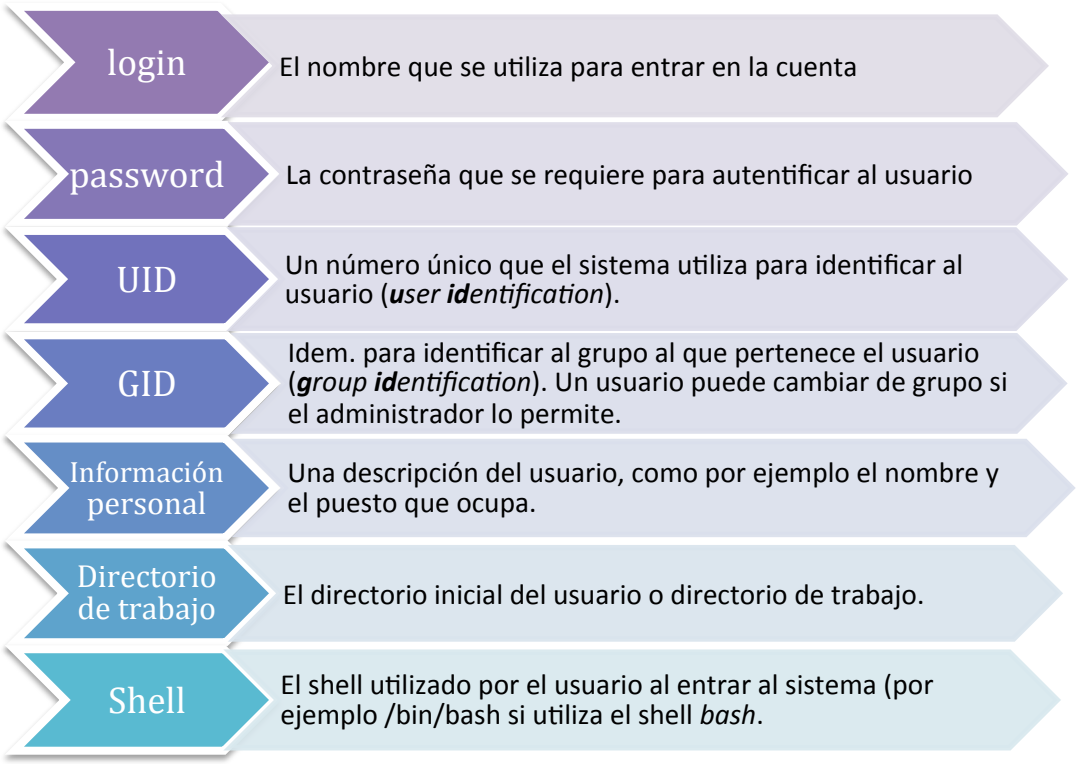
2. Añadir un nuevo usuario al sistema.

1.1. El archivo /etc/passwd.

Cuando se da de alta un usuario, el resultado es una entrada en el archivo de usuarios ***/etc/passwd***. Esta entrada tiene la sintaxis siguiente:

```
<login>:<Password Encriptada>:<UID>:<GID>:<Información Usuario>:<directorio
                        personal>:<orden de acceso>
```

En esta sintaxis los campos están separados por dos puntos. El siguiente gráfico presenta la descripción de estos campos:



Podemos editar el archivo */etc/passwd* si somos *superusuario* y cambiar los valores de algunos de los campos que hemos visto, a excepción de la contraseña encriptada que, en su caso, se ha de modificar con el comando `passwd`.

Para añadir un nuevo usuario sólo tenemos que utilizar la orden `useradd` seguida del nombre de cuenta del usuario a dar de alta en el sistema:

```
useradd [-g grupo][-G grupo[,grupo ...]] [-d dir-de-trabajo[-m]]
        [-s shell] nombre-de-cuenta
```

Además de crear una nueva entrada en el fichero ***/etc/passwd***, esta orden copia los archivos ocultos desde el directorio ***/etc/skel*** al directorio de usuario. En el directorio ***/etc/skel*** deberán estar todos los archivos que el administrador considere que tengan que tener todos los usuarios del sistema, entre ellos los archivos de configuración personales (`profile`, `mailmc`, etc.).

Cuando usamos el comando `useradd` sin modificadores, la información por defecto utilizada para crear la nueva cuenta se extrae del archivo ***/etc/default/useradd*** que veremos más tarde.

Opciones:

- `g grupo` : Se utiliza para asignar el grupo principal[1] al usuario. Se especifica el nombre del grupo y éste debe existir previamente.

- `G grupo[,...]:` Especifica la lista de grupos a los que va a pertenecer el usuario, separada por comas y sin espacios en blanco entre los nombres de grupo.

- `d dir-de-trabajo` : Asigna el directorio de trabajo especificado a la cuenta. Lo normal es que el directorio tenga la ruta ***/home/nombre-usuario***, que es la que asigna el comando por defecto si no usamos `-d`. En todo caso el directorio debe existir previamente. Si no existe y queremos que lo cree el comando automáticamente deberemos usar el modificador `–m.`

- `m:` Crea el directorio de trabajo si no existe y copia en él los archivos de */etc/skel*.

- `s shell :` Asigna un intérprete de comandos al usuario. Lo normal es que se use el shell **bash** (***/bin/bash***) que además es el que se suele asignar por defecto en el caso de que no utilicemos este modificador.

Ejemplos:

```
[root@lotr:~] # useradd –g profesores –d /home/workarwen –m –s /bin/tcsh  arwen
```

En este caso creamos un usuario ***arwen*** cuyo grupo principal es *profesores*, su directorio de trabajo estará en ***/home/workarwen*** que además es creado por el comando y que va a utilizar como shell el ***tcsh***.

```
[root@lotr:~] # useradd arwen
```

En este otro dejamos al sistema que configure la cuenta con los valores por defecto. Es decir, se crea un usuario ***arwen***, con directorio de trabajo en ***/home/arwen***, grupo principal ***arwen*** (en SuSe) o *users* (en Ubuntu) y shell ***bash***.

No obstante, en distribuciones como SuSe o Ubuntu al usar ***useradd*** dándole como único argumento el nombre de la cuenta, el sistema no crea automáticamente el directorio de trabajo ni, por supuesto, traslada los ficheros de ***/etc/skel*** al directorio de trabajo del usuario. Para conseguir que haga esto basta con usar el modificador ***–m***. Por ejemplo: ***useradd –m arwen***.

La orden `useradd` no asigna una contraseña a la nueva cuenta. El administrador del sistema debe definir una contraseña para cada usuario. Para conseguir esto deberemos utilizar la orden `passwd` que tiene el siguiente formato sintáctico:

[1] Todos los usuarios pertenecen al menos a un grupo que es el grupo principal del usuario, también llamado ***grupo primario*** del usuario, pero pueden pertenecer a más grupos. En caso de que pertenezcan a más grupos, éstos serán ***grupos secundarios***.

`passwd [nombre-de-cuenta]`

`passwd` se utiliza para asignar una contraseña a la cuenta de un usuario por primera vez o para modificar una ya existente. El comando entabla un diálogo con el operador solicitándole que escriba la contraseña dos veces para asegurarse de que ha sido introducida correctamente.

El sistema encripta la contraseña y la almacena en el fichero ***/etc/passwd*** [2]. Es importante elegir la contraseña de acuerdo con determinadas normas:

- Deben tener al menos 6 caracteres (recomendable 8).
- Deben estar compuestas de letras mayúsculas y minúsculas, signos de puntuación y números (al menos deben existir símbolos de dos de estos conjuntos).

Una buena contraseña es un requisito indispensable para la seguridad del sistema. Si un usuario olvida su contraseña el administrador puede suprimir el segundo campo de la entrada de dicho usuario en el archivo ***/etc/passwd*** y asignar una nueva con la orden `passwd`.

Un usuario puede cambiar su propia contraseña si le apetece. Para ello deberá usar el comando `passwd` sin especificar cuenta alguna (sólo el administrador del sistema tiene permiso para especificar un nombre de cuenta en `passwd`). El comando entrará en forma interactiva y solicitará la clave actual de la cuenta para verificarla, antes de proceder a solicitar la nueva clave.

1.2. El archivo /etc/login.defs.

Cuando creamos una cuenta con los valores por defecto, la información necesaria para completar la operación es recogida de dos ficheros que contienen los valores por defecto del proceso de creación de cuentas de usuario. Estos ficheros son */etc/login.defs* y */etc/default/useradd*.

El primero de ellos, ***/etc/login.defs***, contiene parámetros que establecen la localización por defecto del buzón de correo, el plazo para la expiración del *password*, el mínimo de caracteres del *password* o el rango de UID's y GID's disponibles para su uso. Además, determina si serán creados los directorios *home* de los usuarios durante la creación de su cuenta, así como el algoritmo empleado para encriptar las claves en el caso de no usar *shadow passwords*.

Un ejemplo de */etc/login.defs* se muestra a continuación:

```
# *REQUIRED*
# Directory where mailboxes reside, _or_ name of file, relative to the
# home directory.  If you _do_ define both, MAIL_DIR takes precedence.
# QMAIL_DIR is for Qmail
#
#QMAIL_DIR   Maildir
MAIL_DIR     /var/spool/mail
#MAIL_FILE   .mail
# Password aging controls:
```

[2] Si se habilitó durante la instalación la opción ***shadow password*** la almacenará en el fichero ***/etc/shadow*** (hoy en día casi todas las distribuciones la tienen habilitada por defecto).

```
#
# PASS_MAX_DAYS     Maximum days a password may be used.
# PASS_MIN_DAYS     Minimum days allowed between password changes.
# PASS_MIN_LEN      Minimum acceptable password length.
# PASS_WARN_AGE     Number of days warning given before password expiry.
#
PASS_MAX_DAYS      99999
PASS_MIN_DAYS      0
PASS_MIN_LEN       5
PASS_WARN_AGE      7
#
# Min/max values for automatic uid selection in useradd
#
UID_MIN            500
UID_MAX            60000
#
# Min/max values for automatic gid selection in groupadd
#
GID_MIN            500
GID_MAX            60000
#
# If defined, this command is run when removing a user.
# It should remove any at/cron/print jobs etc. owned by
# the user to be removed (passed as the first argument).
#
#USERDEL_CMD       /usr/sbin/userdel_local
#
# If useradd should create home directories for users by default
# On RH systems, we do. This option is overridden with the -m flag on
# useradd command line.
#
CREATE_HOME        yes
# The permission mask is initialized to this value. If not specified,
# the permission mask will be initialized to 022.
UMASK              077
# This enables userdel to remove user groups if no members exist.
#
USERGROUPS_ENAB    yes
# Use SHA512 to encrypt password.
ENCRYPT_METHOD     SHA512
```

1.3. El archivo /etc/default/useradd.

El archivo ***/etc/default/useradd*** contiene información acerca del grupo primario, la localización de los directorios *home*, el número de días por defecto para deshabilitar cuentas con claves expiradas, la Shell utilizada y el directorio *skel* utilizado que necesita el comando `useradd` cuando se usa sin modificadores para establecer los valores por defecto.

Un ejemplo de */etc/default/useradd* puede ser:

```
# useradd defaults file
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
CREATE_MAIL_SPOOL=yes
```

1.4. El archivo /etc/adduser.conf.

En las distribuciones Debian existe otro comando para añadir un usuario al sistema. Este comando es `adduser` (también existe `addgroup` para añadir grupos al sistema). Se utiliza menos que `useradd` porque presenta una sintaxis más compleja y una interfaz menos amigable que éste. El uso más extendido de este comando, no obstante, es añadir usuarios a grupos como veremos más adelante. El archivo */etc/adduser.conf* contiene los valores por defecto que usarán `adduser` y `addgroup` para crear cuentas del sistema y grupos con sus respectivos comandos usados sin modificadores.

3. Eliminación de un usuario.

```
userdel [-r] nombre-de-cuenta
```

El comando `userdel` borra todas las entradas del usuario especificado en cualquier archivo del sistema, pero por sí solo no borra los ficheros del directorio de trabajo ni el propio directorio de trabajo del usuario. Para ello deberemos usar el modificador `-r`. Si el usuario en cuestión posee archivos en un directorio diferente al suyo de trabajo deberemos buscarlos manualmente.

1.5. Desactivar momentáneamente un usuario.

Puede ser que un usuario necesite estar ausente durante algún tiempo. Por motivos de seguridad es conveniente desactivar su cuenta sin eliminarla. Esto se hace modificando el archivo ***/etc/passwd*** e insertando en el segundo campo, el de la clave encriptada, el símbolo admiración (!) al principio del mismo. Esto equivale a hacer `usermod -L nombre_de_cuenta`, como veremos después.

1.6. Eliminar un usuario manteniendo sus ficheros.

Puede ocurrir que deseemos mantener determinados ficheros de un usuario que hemos de dar de baja. Para conseguir esto debemos seguir los siguientes pasos:

1) Suprimir la entrada del usuario en los ficheros */etc/passwd*, */etc/shadow* y */etc/group* utilizando un editor ASCII o con el comando `userdel` que tiene el formato siguiente:

Ejemplo: `[root@lotr:~] # userdel arwen`

El comando `userdel` sin modificadores borra las entradas de la cuenta especificada de los archivos */etc/passwd* y */etc/shadow* pero no borra el directorio de trabajo del usuario.

2) Cambiar la ubicación de los ficheros con el comando `mv` que cuenta con el formato siguiente:

Ejemplo: `[root@lotr:~] # mv /home/arwen /home/gandalf/newdocs`

3) Cambiar el propietario con la orden `chown` (que veremos detenidamente más adelante):

```
chown  nuevo-propietario[:nuevo-grupo]  lista-de-archivos
```

Ejemplo: `[root@lotr:~] # chown –R gandalf /home/gandalf/newdocs`

1.7. Eliminar totalmente un usuario.

Podemos eliminar totalmente un usuario y todos sus ficheros siguiendo los siguientes pasos:

1) Eliminar el usuario y su directorio de trabajo.

   Ejemplo:

   `[root@lotr:~] # userdel –r arwen`.

2) Eliminar todas las referencias al usuario en otros directorios del sistema distintos al suyo de trabajo. Podemos usar el comando `find` de la forma:

   Ejemplo: `[root@lotr:~] # find /home -uid`[3] `1001 -exec rm -i {} \;`[4]

4. Modificar la cuenta de un usuario.

Podemos modificar las características de una cuenta de usuario **que exista previamente** con el comando `usermod` que posee la siguiente sintaxis:

```
usermod [-c  comentario] [-d  directorio-de-trabajo [-m]]
[-e  fecha-límite ] [-f  num-dias]
[-g  grupo-inicial] [-G  grupo[,...]]
[-l  nuevo-login] [-s shell]
[-u   UID [-o]] [{-L|-U}] nombre-de-cuenta
```

[3] Cuando hacemos *userdel* obviamente desaparece del sistema el nombre de cuenta del usuario que hemos borrado, pero si este usuario posee archivos diseminados por los directorios del sistema éstos aún llevarán asociado el UID del usuario por lo que éste será el criterio de búsqueda apropiado.

[4] La notación {} significa que los archivos que se encuentren deben ocupar este lugar dentro de la orden. La orden que queremos ejecutar (-***exec***) debe terminar en un **;** precedido del carácter \(para "escapar" el carácter ; que de otra forma tendría un significado distinto para la shell). Ambos deben formar un argumento separado para ***find***, por lo tanto debe haber un espacio en blanco delante de él, de lo contrario provocará un error.

Opciones:

`-c comentario:` Especifica un nuevo valor para el campo comentario en la entrada correspondiente del fichero ***/etc/passwd.*** Normalmente este campo se modifica con el comando `chfn`.

`-d dir-trabajo:` Especifica un nuevo directorio de trabajo para el usuario. Si se utiliza el modificador `-m` el contenido del antiguo directorio ***home*** se moverá al nuevo, el cual será creado si no existe previamente.

`-e fecha-límite:` Especifica la fecha en la que la cuenta expirará. Se puede dar en número de días transcurridos desde Enero de 1970 o en el formato **`YYYY/MM/DD`**.

`-f num-dias:` Especifica el número de días que transcurrirán entre la fecha de expiración de la contraseña (especificada por el modificador `-M` del comando `chage` que veremos a continuación) y su eliminación definitiva. El valor **`0`** deshabilita la cuenta tan pronto como expire la validez de la contraseña. El valor **`-1`** (por defecto) deshabilita esta característica.

`-g grupo-inicial:` Especifica el nombre o el ***GID*** del nuevo grupo inicial del usuario. **Este nuevo grupo debe existir previamente**. Todos los archivos del directorio de trabajo del usuario reflejarán el nuevo grupo. Si existen archivos fuera del directorio de trabajo deberemos cambiarles el grupo manualmente.

`-G grupo[,...]`: Especifica la lista de grupos a los que pertenece el usuario, separada por comas y sin espacios en blanco entre los nombres de grupo. El usuario será borrado de cualquier otro grupo al que perteneciese y que no fuese especificado en la lista de grupos de `-G` (salvo que se use el modificador `-a`). Por lo demás sigue las mismas restricciones que `-g`.

`-l nuevo-login`: El nombre del ***login*** será cambiado al valor especificado siempre que el usuario no esté conectado en ese momento.

`-s shell:` Cambia el shell del usuario.

`-u UID [-o]`: Cambia el identificador del usuario. Debe ser único a menos que se utilice `-o` (que habilita la existencia de UID's duplicados). El identificador debe ser un número positivo, teniendo en cuenta que los valores entre **`0`** y **`999`** se reservan a cuentas del sistema. Los ficheros que se hallen en el directorio ***home*** del usuario serán modificados automáticamente. Los demás ficheros pertenecientes al mismo deberán ser modificados manualmente.

`-L` (***lock***) : Bloquea la cuenta del usuario. Actúa insertando en el campo de la clave encriptada del fichero ***/etc/passwd*** un carácter "!" al principio de la misma. No se puede usar con `-u` o con `-g`.

`-U` (***unlock***) : Desbloquea la cuenta del usuario.

La única información de una cuenta de usuario que `usermod` no puede modificar es la relacionada con la expiración de la clave de usuario almacenada en el archivo */etc/shadow*. Para ello podemos usar el comando `chage` con las opciones apropiadas.

```
chage {[-m mindays] [-M maxdays] [-d lastday]
[-I inactive] [-E expiredate] [-W warndays]|
[-l]}nombre_usuario
```

Opciones:

`-m mindays`: Mínimo número de días que el usuario debe esperar para cambiar su clave.

`-M maxdays`: Número de días de validez de la clave.

`-d lastday:` Establece el día del último cambio de la contraseña al especificado.

`-I inactive:` Desactiva la cuenta después de transcurridos el número de días especificado desde la fecha de caducidad de la clave (si el usuario en esos días no ha vuelto a conectarse y poner la nueva clave suministrada).

`-E expiredate:` Establece la fecha de caducidad de la cuenta.

`-W warndays:` Establece los días de aviso de expiración de la clave.

`-l` : Lista los atributos de caducidad especificados para la cuenta indicada. No se puede usar con el resto de modificadores.

Ejemplo: Cambiar la fecha de expiración de la cuenta del usario `gandalf` al día 28-9-2013:

```
[root@lotr:~] # chage –E 2013-9-28 gandalf
```

Ejemplo: Cambiar el número de días entre cambios de clave a 15 y caducidad por inactividad de la cuenta a 5 días para `gandalf`.

```
[root@lotr:~] # chage –m 15 –M 15 –I 5 gandalf
```

Ejemplo: Obligar a `foo` a cambiar la clave en el próximo intento de acceso:

```
[root@lotr:~] # chage –d 0 foo
```

Ejemplo: Especificar que el usuario `arwen` debe esperar dos días antes de poder cambiar su clave después de recibir una nueva clave y establecer que su clave caduque cada 50 días con 7 días previos de aviso de expiración de este plazo:

```
[root@lotr:~] # chage –m 2 –M 50 –W 7 arwen
```

Ejemplo: Visualizar los atributos de caducidad de la cuenta `galadriel`:

```
[root@lotr:~] # chage –l galadriel
Último cambio de contraseña: sep 07, 2012
La contraseña caduca: nunca
Contraseña inactiva: nunca
La cuenta caduca: sep 28, 2013
Número de días mínimo entre cambios de contraseña: 0
Número de días máximo entre cambios de contraseña: 99999
Número de días de aviso antes de que caduque la contraseña: 7
```

5. Información adicional del usuario.

Para actualizar la información relativa a un usuario se utiliza la orden `chfn` (***change finger information***). La información que podemos actualizar se encuentra almacenada en el fichero ***/etc/passwd*** que ya hemos estudiado.

Cuando ejecutamos esta orden se nos presenta la información disponible hasta ese momento entre corchetes, relativa al nombre, el puesto que ocupa, el teléfono de trabajo y el de casa. Basta con pulsar [ENTER] para mantener la información grabada. Si queremos dejar un campo en blanco deberemos teclear `none`. Si queremos cambiar un campo sólo tenemos que teclear el nuevo valor (recordar que no podemos introducir vocales acentuadas porque serían tomadas como caracteres de control). Esta información sirve de fuente para otros comandos del sistema como `finger`.

Ejemplo:

```
[root@lotr:~] # chfn
      Changing finger information for root
      Enter new value, or press ENTER for the default
      Full Name[]: Mode Martinez Palenzuela
      Room Number[]: aula205
      Work Phone []: 951000999
      Home Phone []: 952010020
      Other []:
      Finger information changed
```

```
finger [{-s|-l}] [-p]  cuenta-de-usuario
```

La orden `finger` nos muestra información relativa al usuario especificado. Si no especificamos ninguno, `finger` nos muestra la información relativa a todos los usuarios conectados en ese momento al sistema (incluso si están en computadoras remotas).

Opciones:

`-s`: (***Short format)***. `Finger` visualiza el ***login***, el nombre real, el terminal asociado, el tiempo de inactividad del terminal (***idle time***) y la hora de conexión, así como el número de oficina y teléfono. Es la opción por defecto si no se especifica cuenta de usuario.

`-l`: (***Long format***). Visualiza todos los datos anteriores además del directorio de trabajo, el shell utilizado y los archivos ***.plan*** y ***.project*** colocados en el directorio de trabajo del usuario en cuestión. Es la opción por defecto si especificamos cuenta de usuario.

`-p`: Omite la visualización de los archivos ***.plan*** y ***.project***.

Ejemplo:

```
[root@lotr:~] # finger root
      Login: root             Name: Gandalf
      Directory: /root        Shell: /bin/bash
      On since Wed Feb 23 22:33 on tty1  11 min 46 sec idle
      …
      No plan
```

6. Comprobar quién se halla conectado al sistema.

Para obtener una lista de aquellos usuarios que están conectados al sistema en un determinado momento podemos utilizar el comando `who`. Esta información la podemos necesitar, por ejemplo, antes de proceder a desconectar el sistema. También nos puede servir para comprobar qué terminales han estado inactivos durante mucho tiempo, ya que una larga inactividad puede significar que los usuarios se han marchado sin desconectar sus terminales.

```
who [{-m|am i}][-u][-H][-q]
```

Opciones:

- `-u` : Visualiza una línea de información por cada usuario conectado. En la primera columna aparece el nombre de cuenta del usuario. Le sigue el terminal donde está presente, la fecha y la hora en la que se conectó, cuánto tiempo ha estado inactivo (aparece un punto "." si está actualmente activo y ***old*** si el usuario estuvo ocioso durante más de 24 horas) y el identificador del proceso correspondiente a cada ***shell*** de usuario.
- `-r` : Podemos utilizar este modificador para comprobar si el sistema está en modo monousuario o en uno de los estados multiusuario. También se visualiza el estado de terminación del proceso y el identificador del mismo.
- `-m|am i` : En ocasiones puede darse el caso de que hayamos modificado varias veces nuestra identidad y queramos saber quiénes somos en cada instante. Para eso se utiliza `who am i`.
- `-H` : Pone cabeceras a la información visualizada (***headers***).
- `-q` : Sólo visualiza los nombres de cuenta y el número total de usuarios conectados.

Ejemplos:

```
[gandalf@lotr:~] $ who am i
      gandalf  pts/0[5]  2010-03-18 18:25 (:0.0)
[gandalf@lotr:~] $ whoami
      gandalf
      …
```

[5] Consola del entorno de ventanas (O: en Debian).

```
[gandalf@lotr:~] $ who -uH
      NOMBRE            LINEA       TIEMPO                  PID  COMENTARIO
      gandalf           tty7        2010-03-18  18:23 antig 1358 (:0)
      gandalf           tty5        2010-03-18  18:25 .     173  (:0.0)
```

7. Grupos: El archivo etc/group.

Cuando se da de alta un usuario también se crea una entrada en este archivo, que está compuesto por una serie de líneas formadas por cinco segmentos separados por ":".Un grupo de usuarios es un conjunto de usuarios unidos por algún propósito específico, por ejemplo, compartir un mismo número de ficheros. El fichero ***/etc/group*** debe pertenecer al superusuario. Los permisos deben ser de lectura/escritura para el propietario y de sólo lectura para el resto de los usuarios.

Cada línea de este archivo corresponde a un grupo de usuarios y tiene un formato como el siguiente:

```
<nombre de grupo>:<contraseña>:<identificador de grupo>:
                  <lista de usuarios>
```

nombre de grupo	• Corresponde al nombre de grupo asociado al identificador de grupo
contraseña	• Actualmente no se usa
GID	• Identificador de grupo. Es el número asignado al grupo que aparece en el fichero /etc/passwd en todas las entradas de los usuarios pertenecientes al mismo.
lista de usuarios	• Es una lista separada por comas de los nombres de usuarios

La creación de grupos puede ser útil en casos en donde se desea que varios usuarios adquieran permisos para un conjunto particular de archivos o directorios. Por ejemplo, puede desearse asignar los usuarios que están escribiendo páginas de un manual a un grupo llamado ***manual*** y darles permiso para un directorio que contenga las páginas de ese manual. Se puede crear un nuevo grupo editando este archivo y escribiendo la información correspondiente a ese nuevo grupo. Eso sí, deberemos tener cuidado de no asignar al nuevo grupo un GID que ya utilice otro, pues Linux utiliza el identificador de grupo y no su nombre para buscarlo. También podemos utilizar el comando `groupadd` que crea un nuevo grupo con los valores especificados y posee la sintaxis siguiente:

```
groupadd [-g GID[-o]][-r][-f]  nombre-de-grupo
```

Opciones:

`-g GID :` Asigna el identificador de grupo especificado al nuevo grupo. Este número debe ser único (a menos que se especifique `-o`) y positivo. Por defecto se asigna un valor mayor que **`500`** y que cualquier ***GID*** ya existente. Los valores comprendidos entre **`0-999`** se reservan normalmente para cuentas del sistema. Podemos especificar el rango de los GID's en el archivo ***/etc/login.defs***.

`-r` : Añade un grupo del sistema. Sólo en algunas distribuciones (Red Hat y SuSe, por ejemplo). El GID estará comprendido entre los valores de **SYSTEM_GID_MIN** y **SYSTEM_GID_MAX** que se inicializan en el archivo ***/etc/login.defs*** y por defecto valen 0 y 500 respectivamente. En estas mismas distribuciones los valores para GID están comprendidos entre **GID_MIN** y **GID_MAX**, definidas en el mismo archivo y cuyos valores predeterminados son respectivamente 1000 y 60000.

`-f` (***force***) : (**No existe en SuSe**) Provoca que el comando termine con un error si el grupo a dar de alta existe ya (esta acción está por defecto activada en SuSe). El grupo existente no será alterado. Modifica también el funcionamiento del parámetro `-g:` si especificamos un *GID* ya existente en vez de dar error actúa como si no hubiésemos especificado `-g`, es decir, fuerza la creación del grupo con un GID asignado por el sistema.

Del mismo modo podemos eliminar un grupo editando el fichero y suprimiendo la entrada relativa a este grupo. También podemos utilizar el comando `groupdel` de la siguiente forma:

```
groupdel  grupo
```

`groupdel` modifica el sistema de cuentas borrando todas las entradas referidas al grupo especificado. El grupo debe existir. No se puede eliminar el grupo primario de un usuario cuya cuenta existe aún (primero habremos de eliminar la cuenta del sistema). Después de eliminar un grupo deberemos comprobar manualmente que no existen ficheros en el sistema que lo referencien. En el caso de querer mantener estos archivos será necesario asignarles un nuevo grupo. Por ejemplo para cambiar de grupo a todos los archivos ordinarios de la carpeta */home/arwen* de *arwen* a *lotr* podemos utilizar la orden `find` de la siguiente forma:

```
[root@lotr:~] # find /home/arwen -gid 1001 -exec chgrp  lotr  {} \;
```

También podemos modificar determinados parámetros de la entrada del fichero ***/etc/group*** con el comando:

```
groupmod [-g GID[-o]][-n nuevo-nombre-de-grupo]  nombre-de-grupo
```

Opciones:

`-g GID:` Especifica un nuevo identificador de grupo. Este número debe ser positivo y único, a menos que utilicemos `-o`. Los números comprendidos entre 0 y 999 están reservados para grupos del sistema. Todos los ficheros con el antiguo GID deben ser asignados manualmente al nuevo identificador de grupo.

`-n nuevo-nombre-de-grupo:` Cambia el nombre del grupo por el especificado.

Para conocer a qué grupos pertenecemos podemos usar el siguiente comando:

```
groups
```

... que visualiza primero el grupo principal del usuario y después los grupos secundarios a los que pertenece.

Ejemplo:

```
[root@lotr:~] # groups
      root appserveradm appserverusr admin
```

```
chgrp [-cRv] nuevo-grupo lista-de-archivos
```

Cambia la propiedad de grupo a un archivo o lista de archivos. Si usamos el modificador `-R` (*recursive*) afectará a directorios y subdirectorios. Tanto `-c` como `-v` hacen que la salida del comando sea detallada. La diferencia es que `-c` sólo informará si se realiza algún cambio.

El modificador `-h` hace que en el caso de enlaces simbólicos, cambie el propietario del destino en lugar del propio enlace.

Ejemplo:

```
[root@lotr:~] # chgrp -hR staff /u
```

1.8. Añadir un usuario a un grupo (no existe en SuSe).

Para añadir un nuevo usuario a un grupo podemos utilizar el comando `adduser` seguido del nombre del usuario y del nombre del grupo al que queremos incorporarlo.

```
adduser   nombre-de-usuario   grupo
```

Ejemplo: Añadir el usuario `tombombadil` al grupo `lotr`.

```
[root@lotr:~] # adduser tombombadil lotr
```

1.9. Quitar un usuario de un grupo (no existe en SuSe).

Se utiliza el comando `deluser` seguido del nombre del usuario y del nombre del grupo del que queremos eliminarlo.

```
deluser   nombre-de-usuario   grupo
```

Ejemplo: Quitar al usuario `gandalf` del grupo `lotr`.

```
[root@lotr:~] # deluser gandalf lotr
```

8. Permisos.

El número identificador de un usuario, es decir su ***UID***, es un entero que se almacena en el archivo ***etc/passwd*** y que está asociado con el nombre del ***login*** del usuario. Cuando un usuario inicia una sesión, la orden ***/bin/login*** asocia este número al primer proceso lanzado: el intérprete de órdenes. Los procesos lanzados a partir de ese momento por el usuario llevarán adherido también este identificador. Los procesos están organizados a su vez en grupos que también poseen un número de identificación, su ***GID*** (***group identification***), almacenado en el fichero ***/etc/group***. El fichero ***/etc/group*** lleva una relación de los grupos existentes. Estos dos identificadores se denominan ***reales*** porque son representativos del usuario que ejecutó el proceso ***login*** y para distinguirlos de otros dos identificadores (uno de usuario y otro de grupo) denominados ***efectivos*** que pueden ser distintos a los reales y que se usan para determinar los permisos, mientras que los reales se usan para saber la verdadera identidad del usuario.

Cada archivo contiene en su inodo el ***UID*** y el ***GID***, un conjunto de permisos de lectura, escritura y ejecución para el propietario, un grupo de usuarios (un departamento por ejemplo) y otros usuarios, además de otra información concerniente al archivo como hemos visto. Este conjunto de permisos determina cuando un proceso o un usuario puede ejecutar una acción sobre un archivo determinado. Este sistema de permisos funciona de la siguiente forma:

- Si el número de identificación de usuario efectivo es 0, entonces se dan los permisos como propietario (0 es el UID efectivo del administrador del sistema).
- Si el número de identificación de usuario efectivo coincide con el UID real marcado por su i-nodo, entonces se dan los permisos de propietario establecidos.
- Si el identificador de grupo efectivo coincide con el número de identificación de grupo propietario del archivo marcado en su i-nodo, entonces se dan los permisos de grupo.
- Si no se da ninguna de las tres anteriores suposiciones se darán los permisos reservados a otros usuarios.

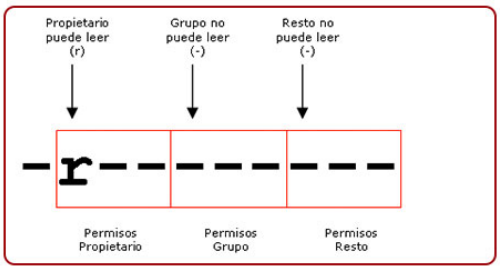
Los derechos de un archivo vienen representados por una cadena de nueve bits (un uno indica que el derecho está activo), los tres primeros para el propietario, los tres siguientes para el grupo de usuarios y los restantes para otros usuarios. Estos nueve bits se visualizan en la máscara de permisos de los archivos como caracteres **`r`** (solo lectura), **`w`** (lectura/escritura) y **`x`** (ejecución). Lo podemos comprobar utilizando la orden `ls` con el modificador `-l`.

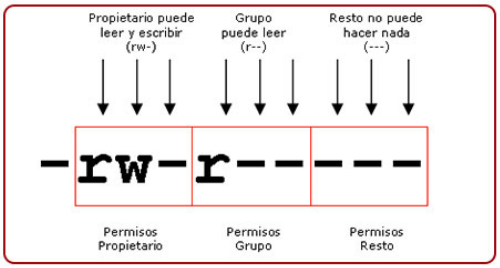
Ejemplo:

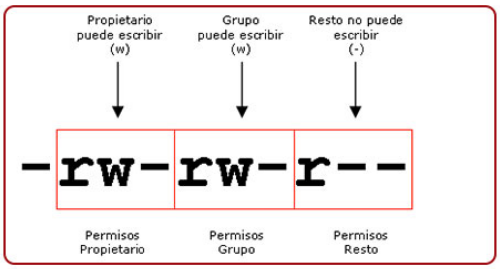
```
[root@lotr:~] # ls -l Mifichero
      -rwxr-x---   1   mode    grmode    890   Feb 5 20:30 Mifichero
```

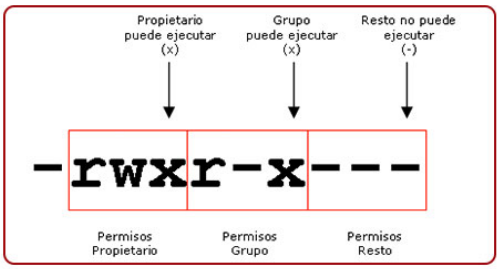
El primer carácter representa el tipo de archivo (el guión nos muestra que es un archivo ordinario en este caso). Los tres siguientes caracteres son los **permisos del propietario** (en este caso posee los tres), los siguientes los del **grupo** (sólo lectura y ejecución) y los tres últimos los de **otros** usuarios (en este caso no tienen permiso de acceso alguno).

Ejemplos:









Cómo afectan los permisos a las carpetas:

Cuando un usuario tiene permiso de lectura de una carpeta, significa que puede visualizar el contenido de la carpeta, es decir, puede ver los archivos y carpetas que contiene, bien sea con el comando 'ls' o con un explorador de archivos. Si el usuario no tiene permiso de lectura sobre la carpeta, no podrá ver lo que contiene.

Cuando un usuario tiene permiso de escritura sobre una carpeta, significa que puede modificar el contenido de la carpeta, es decir, puede crear y eliminar archivos y otras carpetas dentro de ella. permite crear archivos en el directorio, bien sean archivos ordinarios o nuevos directorios. También se pueden copiar archivos en el directorio, mover, cambiar el nombre, etc. Si el usuario no tiene permiso de escritura sobre la carpeta, no podrá crear ni eliminar archivos ni carpetas dentro de ella.

Cuando un usuario tiene permiso de ejecución sobre una carpeta, significa que puede entrar en ella, bien sea con el comando **cd** o con un explorador de archivos. Es decir, permite situarse sobre el directorio para poder examinar su contenido, copiar archivos de o hacia él. Si no dispone del permiso de ejecución significa que no puede ir a dicha carpeta.

Además existen otros tres bits conocidos como ***sticky bit*** (bit pegajoso), ***set-gid*** y ***set-uid*** en la máscara de permisos. Son los llamados bits raros, que hacen que la máscara tenga 3 permisos más de los que hemos visto hasta ahora.

El **set-uid** es una idea relativamente simple que nos permite ahondar más en el tema de la seguridad. Si un programa posee este bit activo cuando sea ejecutado tomará como identificador de usuario el identificador del propietario. Si el propietario fuese el administrador entonces el programa se ejecutará como si lo hubiese lanzado el propio administrador. De esta manera podemos explicarnos como un usuario normal puede modificar su palabra clave cuando el archivo **/etc/passwd** sólo tiene permiso de escritura por parte del administrador del sistema. Esto se debe a que el programa **passwd** tiene este bit activo y cuando lo ejecutamos, y sólo mientras se ejecuta, actuamos como superusuario. Este bit está activo cuando en la máscara de permisos del programa aparece una **s** en lugar de una x en el campo de ejecución para el propietario(**passwd** tiene los permisos **r-s—x--x**). Para activarlo deberemos sumar 4000 a la máscara de permisos en octal.

El **set-gid** posee una utilidad similar pero referida al grupo de propietarios. Los directorios con el bit **set-gid** activado fuerzan a todos los archivos y subdirectorios creados en ellos a heredar el grupo del directorio padre. Para activarlo deberemos sumar 2000 a la máscara de permisos en octal. Aparecerá en la máscara de permisos como una **s** en el permiso de ejecución para usuarios del grupo.

Por último, el **sticky-bit** indica al núcleo de Linux que el archivo es un programa con capacidad para que varios procesos compartan su código y que este código debe mantenerse en memoria aunque alguno de los procesos que lo utilicen dejen de ejecutarse. Esto permite ahorrar memoria cuando se trabaja con programas muy utilizados, como editores de texto o compiladores. Cuando está activo aparece una **t** en vez de una x en el permiso de ejecución del resto de usuarios. Se activa sumando 1000 a la máscara de permisos en octal.

Lo que hace el *sticky-bit* de persistencia en directorios compartidos por varios usuarios es que sólo el propietario del archivo pueda eliminarlo del directorio. Es decir cualquier otro usuario va a poder leer el contenido de un archivo o ejecutarlo si fuera un binario, pero sólo el propietario original podrá eliminarlo o modificarlo. Si no se tuviera el *sticky-bit* activado, entonces en estas carpetas públicas, cualquiera podría eliminar o modificar los archivos de cualquier otro usuario.

Hay que tener en cuenta que todos los bits raros aparecen en minúscula en la máscara de permisos si además del bit raro correspondiente está activado el permiso de ejecución. En caso contrario aparecerán en mayúscula.

Ejemplo: el archivo *Mifichero* tiene activado el ***set-uid*** y permiso de ejecución para el propietario:

```
[root@lotr:~] # ls –l Mifichero
      -rwsr-x---   1   mode    grmode    890   Feb 5 20:30 Mifichero
```

Ejemplo: el archivo *Mifichero* tiene activado el ***sticky-bit*** pero **no** tiene permiso de **ejecución** para otros usuarios:

```
[root@lotr:~] # ls –l Mifichero
      -rwxr-x--T   1   mode    grmode    890   Feb 5 20:30 Mifichero
```

Ejemplo: el archivo *Mifichero* tiene activado el ***sticky-bit*** y también tiene activado el permiso de **ejecución** para otros usuarios:

```
[root@lotr:~] # ls –l Mifichero
      -rwxr-x--t   1   mode    grmode    890   Feb 5 20:30 Mifichero
```

```
chmod  permisos  lista-de-archivos
```

La orden chmod (***change mode***) nos permite modificar los permisos de un archivo. Para poder modificar estos derechos debemos ser propietarios del mismo. También el administrador del sistema tiene la posibilidad de cambiarlos. Para cambiar los permisos podemos seguir los siguientes pasos:

1. Cambiar la máscara de permisos que queremos asignar a binario (recordar que la máscara es una cadena de 9+3 bits) poniendo a 1 los permisos que queremos activar y a 0 los que queremos desactivar. Por ejemplo, una máscara con estos permisos **rwxr-wr-w** generaría la siguiente cadena binaria: **111101101**.
2. Pasar esta cadena binaria a código octal (**755**).
3. Reunir los tres dígitos octales en uno sólo. Acabamos de obtener la máscara de permisos en octal que nos servirá como argumento a la orden chmod.

Ejemplo:

```
[root@lotr:~] # chmod 755 Mifichero
[root@lotr:~] # ls –l Mifichero
      -rwxr-xr-x   1   mode    grmode    890   Feb 5 20:30 Mifichero
```

```
chmod  [u][g][o][a]{+|=|-}[r][w][x][s][t]  lista-de-archivos
```

Para cambiar los permisos en relación con su valor actual disponemos de otro formato de la sentencia chmod. Con este formato primero se especifican los permisos que se van a cambiar (**u** para usuario propietario, **g** para grupo, **o** para otros usuarios y **a** que hace referencia a cualquier tipo de usuario). En segundo lugar se especifican cómo se cambian (**+** añade permisos, **–** quita permisos, **=** establece los permisos especificados y anula el resto de permisos) para leer (**r**), escribir (**w**), ejecutar (**x**), establecer el ***set-uid*** (**s**) si se asigna al propietario o ***set-gid*** (**s**) si se asigna al grupo del propietario o establecer el ***sticky-bit*** (**t**) si se asigna a otros usuarios. Por último se especifica el fichero o ficheros a los que se quiere modificar los permisos.

Ejemplos:

```
[root@lotr:~] # ls –l Mifichero
      -rwxr-xr-x        3     root  root  78    Apr 1 21:30 Mifichero
[root@lotr:~] # chmod go-rx Mifichero
[root@lotr:~] # ls –l Mifichero
      -rwx------        3     root  root  78    Apr 1 21:30 Mi fichero

[root@lotr:~] # chmod ugo+rwx Mifichero
[root@lotr:~] # ls –l Mifichero
      -rwxrwxrwx        3     root  root  78    Apr 1 21:30 Mi fichero
```

Este último ejemplo equivale a poner:

```
[root@lotr:~] # chmod a+rwx Mifichero
```

Ejemplo: Establecer el ***sticky-bit***:

```
[root@lotr:~] # chmod o+t Mifichero
[root@lotr:~] # ls –l Mifichero
      -rwxrwxrwt        3     root  root  78    Apr 1 21:30 Mi fichero
```

Ejemplo: Establecer el ***set-uid***:

```
[root@lotr:~] # chmod u+s Mifichero
[root@lotr:~] # ls –l Mifichero
      -rwsrwxrwt        3     root  root  78    Apr 1 21:30 Mi fichero
```

```
umask [valor-en-octal]
```

Los permisos asignados a un archivo o a un directorio cuando son creados dependen de una variable denominada ***user mask***. Podemos visualizar el contenido de esa variable con la orden `umask` sin argumento (normalmente vale 0022 para el usuario root y 0002 para el resto). Cuando se crea un archivo, los permisos originales por defecto[6] son **666** y cuando se crea una carpeta, los permisos por defecto son **777**.

Si analizamos el valor de `umask` en binario, cada bit a '1' desactiva un permiso y cada bit a '0' lo activa, es decir, si tiene un valor 022 (000 010 010) cuando creemos una carpeta, tendrá permisos rwxr-xr-x y cuando creemos un archivo tendrá permisos rw-r--r-- ya que el permiso de ejecución para archivos hay que fijarlo con `chmod` al tener los archivos el permiso original 666.

Cada usuario tiene su máscara. Se puede fijar la máscara por defecto para todos los usuarios en el archivo ***/etc/profile*** o para cada usuario en el archivo ***/home/usuario/.bashrc.***

Una forma sencilla de hallar el valor de `umask` es poner la máscara de permisos en binario, hallarle el complemento a uno y pasarla a octal. Ese será el valor de `umask` anteponiéndole un 0 (para los bits raros –***set-uid, set-guid*** y ***sticky bit***). Hay que tener en cuenta que el núcleo del sistema ignora, como hemos dicho, por motivos de seguridad los permisos de ejecución que se derivan del valor de `umask` al asignarlos a ficheros nuevos.

---

[6] Parece ser que por motivos de seguridad los máximos permisos que permiten establecer los sistemas UNIX son sólo de lectura y escritura para propietario, grupo y otros usuarios pero nunca de ejecución. Es decir que cómo mucho al crear un fichero dándole todos los permisos posibles estos serían **-rw-rw-rw-.** Esto explica que al establecer umask como **022** obtengas los permisos 644 o **(-rw-r--r--**) al crear un fichero.

Ejemplo:

```
[gandalf@lotr:~] $ umask
   0002

[gandalf@lotr:~] $ mkdir nueva-carpeta

[gandalf@lotr:~] ls -l

   drwxrwxr-x    2 gandalf     profes       1024 Feb 12 19:46 nueva-carpeta

[gandalf@lotr:~] $ umask 022

[gandalf@lotr:~] $ mkdir otra-carpeta
[gandalf@lotr:~] $ ls -l
   drwxrwxr-x   2 gandalf     profes       1024 Feb 12 19:46 nueva-carpeta
   drwxr-xr-x   2 gandalf     profes       1024 Feb 12 19:46 otra-carpeta
```

La modificación con `umask` de la mascara por defecto no afecta a los archivos y carpetas existentes sino sólo a los nuevos que cree ese usuario a partir de ese momento.

```
umask [u][g][o][a]{+|=|-}[r][w][x]
```

También se puede establecer `umask` de forma simbólica. Por ejemplo, una máscara establecida a u=rwx,g=rwx,o= implica que los nuevos archivos tendrán los permisos rw-rw----, y los nuevos directorios tendrán los permisos rwxrwx---. Esto es debido a que los permisos asignados a los ficheros en su creación son el resultado de la operación AND entre la `umask` y los permisos originales (666). Y para un directorio los permisos al crearlo serán el resultado de la operación AND entre la `umask` y los permisos originales para directorios (777). Por ejemplo:

```
[gandalf@lotr:~] $ umask u=rwx,g=rwx,o=

[gandalf@lotr:~] $ mkdir foo
[gandalf@lotr:~] $ touch bar

[gandalf@lotr:~] $ ls -l
   drwxrwx---   2   dave   dave    512   Sep  1 20:59   foo
   -rw-rw----   1   dave   dave      0   Sep  1 20:59   bar
```

9. Cambio del propietario de un archivo.

Cada archivo tiene un propietario que es, normalmente, la persona que lo creó. El propietario tiene, por lo general, permisos más amplios que los otros usuarios para manipular el archivo que ha creado. A veces es necesario cambiar el propietario de un archivo, bien porque se adquiere la responsabilidad de un archivo que no hemos creado o bien porque otra persona nos lo ha cedido. Una de las formas de hacerse propietario de un archivo es mediante la realización de una copia

(cp), pero presenta las desventajas de que se genera un archivo extra y además requiere que el usuario que va a hacer la copia tenga permiso de lectura para el archivo en cuestión. Una forma más sencilla y directa de transferir la propiedad consiste en utilizar la orden chown (***change owner***).

```
chown [-R][-h] nuevo-propietario[.nuevo-grupo] archivo
```

La orden chown tiene dos argumentos: el nombre del usuario que va a ser el nuevo propietario y el nombre del archivo. Por ejemplo, la sentencia siguiente hace que gandalf sea el nuevo propietario de Midocumento.

```
[root@lotr:~] # chown gandalf Midocumento
```

Sólo el propietario de un archivo o el superusuario pueden utilizar chown para cambiar su propiedad.

Con versiones avanzadas de Unix/Linux podemos utilizar esta orden en forma recursiva (-R) para cambiar el propietario de todos los archivos de un directorio de la forma:

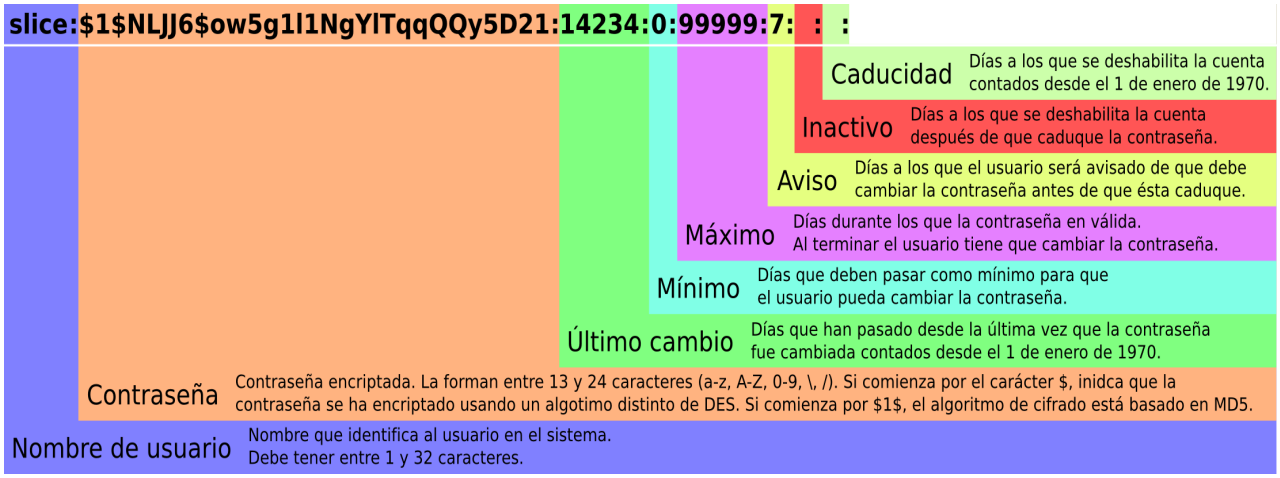
```
[root@lotr:~] # chown –R gandalf MisDocumentos
```

Siendo MisDocumentos un directorio.

10. El archivo /etc/shadow.

El archivo */etc/shadow* almacena las claves de los usuarios encriptadas y algunas propiedades más relacionadas con la seguridad. Todos los campos están separados por el carácter ":" y existe una línea por cada cuenta de usuario existente en el archivo */etc/passwd*, del que se puede considerar una extensión y una mejora en cuanto a la seguridad del sistema, ya que */etc/passwd* aunque puede ser modificado sólo por el administrador, puede ser leído por cualquier usuario lo que significa un agujero para la seguridad.

La estructura del fichero es la siguiente:



11. Comunicación con los usuarios.

1.10. Mensaje del día.

El ***login*** muestra el contenido del fichero ***/etc/motd*** (***message of the day***) después de un inicio de sesión, justo antes de que ejecute el intérprete de comandos. Podemos cambiar el contenido de este fichero con cualquier procesador de textos o con el comando `cat`.

1.11. Aviso a todos los terminales.

Para el envío simultáneo e inmediato de un mensaje a todos y cada uno de los usuarios conectados en ese momento al sistema podemos utilizar la orden `wall` (***write all***). El mensaje aparecerá en mitad del trabajo del usuario pero no producirá ningún daño.

Ejemplo:

```
[root@lotr:~] # wall
  Necesito apagar el sistema en 5 minutos.
  El sistema se reiniciará en dos horas.
  [CTRL] +[D]
```