# Práctica 1 - Usuarios

Pablo Blázquez Sánchez, Raúl Jiménez de la Cruz

Laboratorio de Seguridad y Riesgos de Sistemas de Información – Curso 2023/24

**(e01)** Añadir varios usuarios y comprobar los ficheros de configuración donde se encuentra la información de usuarios, grupos, contraseñas …

```
$ useradd -m usuario -p $(openssl passwd 12345)
$ useradd -m otro -p $(openssl passwd 99999) -e 2026-01-01 -c 'Esto es un
nombre completo' -u 1009
```

Considerando este par de usuarios creados vamos a comprobar si se han presentado cambios en los ficheros `/etc/passwd`, `/etc/group`, `/etc/shadow` y `/etc/gshadow`. Para ello, se empleó el siguiente comando, asumiendo que se almacenó la información original en ficheros de texto:

```
$ diff $(echo /etc/passwd) orig_passwd.txt
```

- Cambios en `/etc/passwd`

```
diff $(echo /etc/passwd) orig_passwd.txt
28,29d27
< usuario:x:1001:1001::/home/usuario:/bin/sh
< otro:x:1009:1009:Esto es un nombre completo:/home/otro:/bin/sh
```

Podemos observar que se han añadido al final un par de líneas, correspondiente a la configuración del par de usuarios nuevos. En primer lugar, `usuario` se ha creado en la carpeta predeterminada, usando el UID más próximo al último usado (1000 fue usado para el primero, el siguiente es 1001) y con `sh` como "*shell*" por defecto[1].

No obstante, es interesante observar la configuración que ha adoptado `otro`, está empleando como UID = 1009 (el especificado) y se ha especificado un comentario entre los dos puntos que separan los identificadores de usuario y la carpeta que emplea el usuario para `/home`. Este comentario se suele emplear para especificar un nombre completo. Sin embargo, en este fichero falta la fecha de expiración, más adelante veremos cómo obtenerla.

```
diff $(echo /etc/group) orig_group.txt
57,58d56
< usuario:x:1001:
< otro:x:1009:
```

Aquí queda explícito que, en este caso, el GID es igual a UID (no se especificó lo contrario al crear los usuarios). Sucede exactamente lo mismo para `/etc/shadow` y `/etc/gshadow`, se agregaron al final un par de líneas, correspondientes a las contraseñas de los nuevos usuarios, encriptadas usando `openssl`.

En este punto, resulta interesante comprobar si se aplicó la fecha de expiración para el usuario `otro`. Para ello, podemos usar el siguiente comando:

[1] Aunque sea el mismo número, en este fichero aparecen juntos (en orden) *UserID* y *GroupID*, separados por ':', y no necesariamente son iguales

```
$ chage -l usuario
```

Lo que devolverá será una serie de atributos de expiración y de trazabilidad de la contraseña asociada al usuario en cuestión. Por ejemplo, para **usuario**:

```
Last password change                                    : Feb 26, 2024
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : never
Minimum number of days between password change          : 0
Maximum number of days between password change          : 99999
Number of days of warning before password expires       : 7
```

Para este usuario, podemos ver el usuario cambió la contraseña por última vez el 26 de febrero de 2024 (misma fecha de creación del propio usuario), y no se requiere atención por expiración de usuario ni contraseña, usando los valores por defecto de **/etc/login.defs**:

```
PASS_MAX_DAYS   99999
PASS_MIN_DAYS   0
PASS_WARN_AGE   7
```

A continuación, comprobaremos el usuario otro:

```
Last password change                                    : Mar 04, 2024
Password expires                                        : never
Password inactive                                       : never
Account expires                                         : Jan 01, 2026
Minimum number of days between password change          : 0
Maximum number of days between password change          : 99999
Number of days of warning before password expires       : 7
```

Para este usuario, el campo '**Account expires**' sí contiene un valor: **Jan 01, 2026** (en formato YYYY-MM-DD es 2026-01-01, justo el parámetro -e)

**(e02)** Al crear una cuenta con **useradd**, ¿qué ficheros se tienen en cuenta y en qué orden?

El comando anterior se ejecuta considerando la información de estos ficheros:

- **/etc/default/useradd**. En principio, este es el fichero que contiene la información por defecto para crear un nuevo usuario. Podemos mostrar los atributos más importantes y compararlo con el usuario **usuario** del ejercicio 'e01'. Podemos sacar la información de este fichero con un **cat**:

  ```
  SHELL=/bin/sh
  #GROUP=100
  #HOME=/home
  #INACTIVE=-1
  #EXPIRE=
  #SKEL=/etc/skel
  #CREATE_MAIL_SPOOL=yes
  ```

  Por ejemplo, **usuario** usa la '*shell*' **sh**, su usuario tiene GID = 1001 (dentro de 100), se localiza en **/home/**, usa la jerarquía/esqueleto por defecto y no se han considerado aspectos de expiración de credenciales.

- `/etc/login.defs`. Este archivo contiene la configuración del conjunto de contraseñas para los usuarios del sistema, como la directiva de expiración de contraseñas, los intervalos de ID de usuario utilizados al crear el sistema y los usuarios normales, etc. Sin ir más lejos, en el ejercicio anterior se mostró un fragmento de este fichero para la expiración de una contraseña.
- `/etc/skel/`. Este directorio (directorio esqueleto) contiene una serie de scripts y ficheros necesarios para la generación básica de un usuario, incluyendo configuraciones iniciales y/o de login, que serán copiadas al directorio del nuevo usuario. Contiene los ficheros `.bash_logout`, `.bashrc` y `.profile`.

**(e03)** ¿Qué comando y opción permite ver la configuración utilizada por defecto al crear un nuevo usuario?

El comando **`useradd`** tiene un parámetro u opción que permite obtener qué configuración se aplica al crear por defecto un usuario en el sistema, que es la opción `-D`:

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

Como vimos en el ejercicio 'e02', el usuario **`usuario`** se creó siguiendo los parámetros por defecto que aparecen en la figura superior.

**(e04)** ¿En qué archivo se encuentra especificada la shell por defecto al crear un nuevo usuario? ¿Qué valor tiene en tu sistema Linux?

En general, cuando creamos un nuevo usuario en Linux, los valores por defecto se almacenan en el fichero **`/etc/default/useradd`**, tal y como vimos en el ejercicio (e02). Si nos fijamos en las primeras líneas del contenido de dicho fichero, podemos ver la *shell* por defecto, así como una justificación del uso de esta:

```
# The SHELL variable specifies the default login shell on your system.
# Similar to DSHELL in adduser. However, we use "sh" here because
# useradd is a low level utility and should be as general
# as possible
SHELL=/bin/sh
```

No obstante, el listado completo de *shells* utilizables por el sistema (por ejemplo, en un *login*) se encuentra en **`/etc/shells`**:

```
# /etc/shells: valid login shells
/bin/sh
/bin/bash
/usr/bin/bash
/bin/rbash
/usr/bin/rbash
/usr/bin/sh
/bin/dash
/usr/bin/dash
/usr/bin/tmux
/usr/bin/screen
```

**(e05)** Crear varios usuarios combinando las opciones -b, -d y -m

– ¿Bajo qué circunstancias tiene que existir (o no) el directorio previamente a la creación del usuario?

Para responder a la pregunta, vamos a crear una serie de usuarios de prueba, para comprobar su salida por terminal y su comportamiento al tratar de iniciar sesión. *En todo momento no se habrá creado con anterioridad ningún directorio*, con el fin de entender el funcionamiento de esta utilidad de bajo nivel:

| ***Comando empleado*** | ***¿Se han creado los directorios?*** |
|---|---|
| *useradd -b /paco paco* | No, el *login* se realiza en /. **Debe existir anteriormente** */paco/paco/* |
| *useradd -b /paco -m paco* | Sí, pero fallará el `chown` que aplica `useradd` al no existir antes el directorio. **Debe existir anteriormente** */paco/* |
| *useradd -d /home/antonio antonio* | No, el *login* se realiza en /. **Debe existir anteriormente** */home/antonio/* |
| *useradd -d /home/antonio antonio -m* | Sí |
| *useradd -d /antonio/antonio antonio -m* | Sí, pero fallará el `chown` que aplica `useradd` al no existir antes el directorio. **Debe existir anteriormente** */antonio/* |

Con esto, podemos sacar las siguientes conclusiones:

- La opción `-m` crea automáticamente el directorio del usuario y el directorio de inicio (el superior o padre del usuario) pero con grupos no aplicados adecuadamente, por lo que si se usa esta opción, *es recomendable que exista el directorio inicial*. Si esto se cumple, no importa si existe o no el directorio del usuario, pues se creará automáticamente.
- La opción `-b` permite especificar el directorio inicial o base, entonces *debe de existir* (si no se usa `-m`)
- La opción `-d` permite especificar la ruta para el inicio de sesión. No es necesario que exista, pero sin `-m` no se creará, por lo que *es recomendable*.

**(e06)** Cambiar el contenido del directorio esqueleto y crear varios usuarios para comprobar su funcionamiento

Para probar que funciona adecuadamente el cambio del directorio esqueleto, hemos añadido un fichero de texto de prueba llamado **mi_archivo.txt**, y también se ha modificado **.bashrc**, para que cada vez que se abra una terminal muestre el calendario por defecto y el **sudo** ha sido personalizado, reemplazando dicho término por otra palabra, manteniendo su funcionalidad:

```
 ... (resto de .bashrc)
alias porfi='sudo'
cal
```

En este punto, vamos a modificar el directorio esqueleto y crear un nuevo usuario que tome por defecto la ruta original del directorio esqueleto, para forzar a que se tomen los nuevos cambios:

```
$ rm /etc/skel/.bashrc
$ cp /home/raul/.bashrc /etc/skel/ #en la primera ruta está la modificación
$ cd /etc/skel/
$ touch mi_archivo.txt
$ useradd -m ruben -p $(openssl passwd ruben)
```

Cuando se crea el usuario, podemos iniciar sesión y comprobar si se aplicaron correctamente las modificaciones:

```
root@equipo:~# login ruben
Password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.133.1-microsoft-standard-WSL2
x86_64)
 ... (aquí sigue el texto de bienvenida)
$ bash
     March 2024
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
ruben@equipo:~$ porfi --help
sudo - execute a command as another user
 ... (el resto del manual de sudo)
ruben@equipo:~$ ls
mi_archivo.txt
```

En este caso, se han aplicado correctamente porque se muestra el calendario al abrir una sesión **bash** nueva desde el usuario **ruben**, podemos usar el alias customizado y el fichero de bienvenida está presente.

Por ejemplo, a partir de una copia de seguridad del directorio esqueleto, podemos crear un usuario de forma que se tome este directorio de resguardo:

```
$ useradd -k /home/raul/skel/ -m rodolfo -p $(openssl passwd rodolfo)
root@HPRAUL:~# login rodolfo
Password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.133.1-microsoft-standard-WSL2
x86_64)
 ...
$ bash
rodolfo@equipo:~$ porfi
porfi: command not found
rodolfo@equipo:~$
```

En este caso, el usuario no muestra el calendario ni alias personalizado, está completamente por defecto, lo cual reafirma que el directorio esqueleto original se modificó satisfactoriamente.

**(e07)** Comprueba las diferentes formas para añadir o sustituir grupos a un usuario

Para este ejercicio, vamos a partir de una serie de usuarios con los que jugaremos para la gestión de grupos en Linux. Por ejemplo, sacando por pantalla la información de `/etc/subuid` podemos ver los usuarios y sus id de usuario subordinados:

```
raulillo:100000:65536
usuario:165536:65536
otro:231072:65536
ruben:296608:65536
rodolfo:362144:65536
```

Lo primero, vamos a quedarnos en modo super-usuario para comenzar a crear una serie de grupos. Esto es posible a través de la primitiva `groupadd`, utilidad de bajo nivel que permite crear grupos:

```
$ groupadd nuevogrupo
$ groupadd -g 3401 grupoalt
```

De esta forma, el archivo de grupos `/etc/group` tiene estas nuevas entradas:

```
nuevogrupo:x:1012:
grupoalt:x:3401:
```

Es correcto ya que el último usuario es Rodolfo con enumeración 1011, por lo que el siguiente, por defecto, es 1012, y para `grupoalt` se seleccionó manualmente con la opción `-g`, que permite establecer un GID concreto, siempre y cuando esté entre `GID_MIN` y `GID_MAX` (y no esté en uso). Los valores límite se pueden encontrar en `/etc/login.defs` (en nuestro caso es `[1000,60000]`).

A partir de aquí podemos emplear otra utilidad para añadir los usuarios a los grupos. Por ejemplo, vamos a agregar a `raul` y a `usuario` al grupo `nuevogrupo`. Esto lo haremos con la primitiva `usermod`. Esta utilidad sirve para modificar la información de una cuenta o usuario de un sistema Linux, lo cual nos servirá para la gestión de grupos.

En este caso, vamos a combinar dos opciones en una: **-a** y **-G**: la primera es para encolar o añadir al final información de grupos[2], y la segunda es para incluir un conjunto de grupos a los que pertenece la cuenta, separados por comas.

```
usermod -aG nuevogrupo raulillo
usermod -a -G nuevogrupo usuario
```

Ahora comparamos el nuevo contenido en **/etc/group**:

```
nuevogrupo:x:1012:raulillo,usuario
grupoalt:x:3401:
```

Entonces, al final de cada id de grupo de este fichero, se listarán tras ':' los usuarios pertenecientes. No obstante, también es posible añadir un usuario a varios grupos distintos. Por ejemplo, lo vamos a aplicar sobre **otro**:

```
$ usermod -aG nuevogrupo,grupoalt otro
$ cat /etc/group
root:x:0:
  … (el fichero sigue)
nuevogrupo:x:1012:raulillo,usuario,otro
grupoalt:x:3401:otro
```

Así pues, esta cuenta está vinculada a dos grupos. Lo cual podemos utilizarlo para probar si podemos aplicar una sustitución completa de los grupos asociados a **otro** a un único grupo.

```
$ groupadd tercergrupo
$ usermod -G tercergrupo otro
$ cat /etc/group
root:x:0
  … (el fichero sigue)
nuevogrupo:x:1012:raulillo,usuario
grupoalt:x:3401:
tercergrupo:x:3402:otro
```

Aquí se puede ver como el uso único de **-G**[3] permite un reemplazo completo. Por último, vamos a ver cómo se quita un grupo de un usuario. En este caso se requiere de otra primitiva: **gpasswd**. Es una herramienta para gestionar **/etc/group** y **/etc/gshadow**. Tiene una opción que nos permitirá eliminar un usuario de un grupo muy cómodamente, con **-d**:

```
root@HPRAUL:~# gpasswd -d usuario nuevogrupo
Removing user usuario from group nuevogrupo
$ cat /etc/group
root:x:0
  … (el fichero sigue)
nuevogrupo:x:1012:raulillo
grupoalt:x:3401:
tercergrupo:x:3402:otro
```

En la figura anterior podemos ver cómo hemos quitado del grupo **nuevogrupo** la cuenta **usuario**.

---

[2] Esta primera opción no funciona si no está incluida también **-G**

[3] Importante la **-G** en mayúscula, la opción **-g** tiene otro uso relacionado a grupos, más concretamente asigna una cuenta a un grupo primario o principal, el cual se puede ver en **/etc/passwd**

**(e08)** ¿Qué opciones son iguales para los comandos `useradd` y `usermod`?

La manera más sencilla de comparar las opciones entre ambas primitivas es consultando ambos manuales. Esto lo podemos hacer con `man [comando]`. Por comodidad, resumiremos en una tabla las opciones de ambos comandos, *resaltando en fondo verde los coincidentes*:

| useradd | usermod |
|---|---|
| | -a, --append |
| --badname | -b, --badname |
| -b, --base-dir | |
| -c, --comment | -c, --comment |
| -d, --home-dir | -d, --home |
| -D, --defaults | |
| -e, --expiredate | -e, --expiredate |
| -f, --inactive | -f, --inactive |
| -F, --add-subids-for-system | |
| -g, --gid | -g, --gid |
| -G, --groups | -G, --groups |
| -h, --help | -l, --login |
| -k, --skel | -L, --lock |
| -K, --key | |
| -l, --no-log-init | |
| -m, --create-home | -m, --move-home |
| -M, --no-create-home | |
| -N, --no-user-group | |
| -o, --non-unique | -o, --non-unique |
| -p, --password | -p, --password |
| -r, --system | -r, --remove |
| -R, --root | -R, --root |
| -P, --prefix | -P, --prefix |

| | |
|---|---|
| -s, --shell | -s, --shell |
| -u, --uid | -u, --uid |
| -U, --user-group | -U, --unlock |
| | -v, --add-subuids |
| | -V, --del-subuids |
| | -w, --add-subgids |
| | -W, --del-subgids |
| -Z, --selinux-user | -Z, --selinux-user |
| --selinux-range | --selinux-range |

**(e09)** Buscar el *shell* por defecto al crear una cuenta en el archivo de configuración correspondiente

- Establecer, con el comando correspondiente, el *shell* por defecto a `/bin/bash`
- Comprobar el cambio en el fichero pertinente
- Comprobar que el cambio se hace efectivo al crear nuevas cuentas de usuario

Tal y como vimos en el ejercicio e02, existe información almacenada en el fichero `/etc/default/useradd` relacionada con los parámetros usados por defecto a la hora de crear un usuario. En este caso, el *shell* por defecto es `SHELL=/bin/sh`, ya que tiene que ser el de más bajo nivel posible (recogido en dicho fichero de configuración).

También mencionamos que con el comando -D podíamos ver las opciones por defecto utilizadas, relacionadas al fichero de configuración anterior. Además de ver dichas opciones predeterminadas, si agregamos una segunda opción, podremos cambiar el valor por defecto. Vamos a aprovecharlo cambiando la *shell* por defecto a `bash` para nuevos usuarios creados:

```
$ useradd -D -s /bin/bash
$ useradd -D
GROUP=100
HOME=/home
INACTIVE=-1
EXPIRE=
SHELL=/bin/bash
SKEL=/etc/skel
CREATE_MAIL_SPOOL=no
```

Aquí podemos ver cómo `SHELL` ya no es `=/bin/sh`, sino `=/bin/bash`. Vamos a comprobarlo creando un usuario y *logeándonos*:

```
root@HPRAUL:~# useradd -m pruebabash -p $(openssl passwd 123)
root@equipo:~# login pruebabash
Password:
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.133.1-microsoft-standard-WSL2
x86_64)
 ...
     March 2024
Su Mo Tu We Th Fr Sa
                1  2
 3  4  5  6  7  8  9
10 11 12 13 14 15 16
17 18 19 20 21 22 23
24 25 26 27 28 29 30
31
pruebabash@equipo:~$
```

En el listado anterior podemos ver la creación de un usuario de prueba, y cuando iniciamos sesión podemos ver cómo se muestra por pantalla automáticamente el calendario del ejercicio 'e06', lo cual indica explícitamente que la sesión es en una *shell* de tipo **bash**, ya que está tomando el fichero de configuración creado en el ejercicio anteriormente mencionado (y que no ha sido cambiado para el ejercicio actual).

También podemos ver si en el propio fichero de configuración se aplicó el cambio:

```
$ cat /etc/default/useradd
# Default values for useradd(8)
#
# The SHELL variable specifies the default login shell on your
# system.
# Similar to DSHELL in adduser. However, we use "sh" here because
# useradd is a low level utility and should be as general
# as possible
SHELL=/bin/bash
 ...
```

En consecuencia, la *shell* por defecto vuelve a ser **bash**, lo cual es un cambio si lo comparamos con la información mostrada en el ejercicio 'e02'.

**(e10)** ¿Cuándo se deshabilita una cuenta, qué cambios se producen en los ficheros de configuración? Investiga el fichero de contraseñas de la cuenta al ser desactivada

Para deshabilitar una cuenta (sin borrarla), podemos emplear **usermod** con la opción **-L**. Por ejemplo, vamos a deshabilitar la cuenta del ejercicio 'e09' **pruebabash**. Para comparar el fichero de contraseñas, se realizó una copia anteriormente en un **.txt**:

```
cat /etc/shadow > /home/raul/copia_shadow_11.txt
# usermod -L pruebabash
# diff $(echo /etc/shadow) /home/raul/copia_shadow_11.txt
32c32
< pruebabash:!$1$tcmLtwE1$pZykCU4YvYdyWPlv8ll4y0:19792:0:99999:7:::
---
> pruebabash:$1$tcmLtwE1$pZykCU4YvYdyWPlv8ll4y0:19792:0:99999:7:::
```

En este caso, lo que ha sucedido en **/etc/shadow** es que se ha insertado una exclamación (**'!'**) al principio de la contraseña (encriptada).

Esto significa que la cuenta está deshabilitada, reflejado en el propio manual de la primitiva, donde se especifica que el cambio más sustancial se produce en dicho fichero; no hay cambios presentes en **/etc/passwd**, **/etc/group** o **/etc/gshadow**.

**(e11)** Crear una cuenta de usuario que tenga un directorio **home**. A continuación, borrar el usuario.

- Utilizar el comando **find** con la opción correspondiente para listar sus ficheros y también borrarlos
- ¿Qué opción tenemos que utilizar con el comando **ls** para ver el UID?

En primer lugar, vamos a crear el usuario el cual se llamará **UserEx11** y le asignaremos un directorio home cuyo nombre será **NewHome**:

```
$ useradd –m –d /NewHome UserEx11
```

Antes de borrar este usuario comprobaremos el UID que tiene para operar con él más adelante mediante **sudo cat /etc/passwd**, con el que vemos que el usuario creado tiene como UID el valor **1001**.

Para eliminar este usuario utilizaremos el comando **userdel**:

```
$ userdel UserEx11
```

Para listar los archivos del usuario que acabamos de eliminar los buscaremos a partir de su UID obtenido anteriormente. Haciendo uso del comando **find** con la opción requerida, obtenemos:

```
$ sudo find / –user 1001
/NewHome
/NewHome/.bashrc
/NewHome/.gtkrc-2.0
/NewHome/.bash_logout
/NewHome/.profile
/NewHome/.config
/NewHome/.config/qt5ct
/NewHome/.config/qt5ct/qt5ct.conf
/NewHome/.config/hexchat
/NewHome/.config/hexchat/hexchat.conf
/NewHome/.config/hexchat/servlist.conf
/NewHome/.config/caja
/NewHome/.config/caja/desktop-metadata
/NewHome/.gtkrc-xfce
```

Estos archivos existen debido a que, al crear un usuario, este se crea con los archivos de configuración del usuario desde donde se está creando, los cuales podemos encontrar en el directorio **/etc/skel**.

```
$ ls –al /etc/skel
```

```
total 40
drwxr-xr-x   3 root root  4096 ene  9 13:59 .
drwxr-xr-x 150 root root 12288 mar  8 17:58 ..
-rw-r--r--   1 root root   220 ene  6  2022 .bash_logout
-rw-r--r--   1 root root  3771 ene  6  2022 .bashrc
drwxr-xr-x   5 root root  4096 ene  9 13:59 .config
-rw-r--r--   1 root root    22 sep  8  2011 .gtkrc-2.0
-rw-r--r--   1 root root   516 dic 17  2013 .gtkrc-xfce
-rw-r--r--   1 root root   807 ene  6  2022 .profile
```

Para borrar los ficheros dentro del comando **find** haremos uso de la opción **-exec** seguido del comando **rm -r** para borrar también los directorios asociados al usuario. El comando resultante quedará así:

```
sudo find / -user 1001 -exec rm -r {} \;
```

Para ver el UID mediante el comando **ls** utilizaremos la opción **-n**, la cual funciona de la misma manera que la opción **-l**, solo que muestra el usuario y el grupo de forma numérica:

```
$ ls / -n
total 2097232
lrwxrwxrwx   1    0    0          7 feb 26 08:53 bin -> usr/bin
drwxr-xr-x   4    0    0       4096 feb 26 09:51 boot
drwxr-xr-x   2    0    0       4096 feb 26 08:57 cdrom
drwxr-xr-x  20    0    0       4860 mar  4 09:18 dev
drwxr-xr-x 149    0    0      12288 mar  4 09:16 etc
drwxr-xr-x   3    0    0       4096 feb 26 08:57 home
lrwxrwxrwx   1    0    0          7 feb 26 08:53 lib -> usr/lib
lrwxrwxrwx   1    0    0          9 feb 26 08:53 lib64 -> usr/lib64
drwx------   2    0    0      16384 feb 26 08:53 lost+found
drwxr-xr-x   3    0    0       4096 feb 26 09:47 media
drwxr-xr-x   2    0    0       4096 ene  9 13:59 mnt
drwxr-x---   3 1001 1001       4096 mar  4 09:06 NewHome
drwxr-xr-x   3    0    0       4096 feb 26 09:11 opt
dr-xr-xr-x 456    0    0          0 mar  4 09:33 proc
drwx------   3    0    0       4096 mar  4 09:27 root
drwxr-xr-x  37    0    0       1140 mar  4 08:34 run
lrwxrwxrwx   1    0    0          8 feb 26 08:53 sbin -> usr/sbin
drwxr-xr-x   2    0    0       4096 ene  9 13:59 srv
-rw-------   1    0    0 2147483648 feb 26 08:53 swapfile
dr-xr-xr-x  13    0    0          0 mar  4 09:33 sys
drwxrwxrwt  18    0    0       4096 mar  4 09:45 tmp
drwxr-xr-x  12    0    0       4096 ene  9 13:59 usr
drwxr-xr-x  11    0    0       4096 ene  9 13:59 var
```

Como podemos observar en la salida del comando en el directorio raíz, las columnas correspondientes al usuario y al grupo se muestran numéricamente. También apreciamos como el directorio **NewHome** tiene un valor numérico distinto al de los demás directorios, correspondiendo al UID del usuario creado anteriormente y que ha sido eliminado.

**(e12)** Crear una cuenta de usuario que tenga un directorio ***home***, a continuación, realiza los siguientes pasos:

- Creamos el usuario con un directorio **home** mediante el comando **useradd**:

```
$ sudo useradd -m -d /Ex12 UserEx12
```

- Mostrar el ID del usuario en el fichero correspondiente

Podemos ver el UID del usuario creado en el fichero `/etc/passwd` haciendo uso del comando `sudo cat /etc/passwd`, donde el usuario se encuentra al final del fichero:

```
$ sudo cat /etc/passwd
.
.
.
UserEx12:x:1001:1001::/Ex12:/bin/sh
```

- Listar los archivos de su directorio ***home***

Listamos los archivos de su directorio home mediante el comando `ls`:

```
$ ls -al /Ex12
total 32
drwxr-x---  3 UserEx12 UserEx12 4096 mar  9 18:46 .
drwxr-xr-x 20 root     root     4096 mar  9 18:46 ..
-rw-r--r--  1 UserEx12 UserEx12  220 ene  6  2022 .bash_logout
-rw-r--r--  1 UserEx12 UserEx12 3771 ene  6  2022 .bashrc
drwxr-xr-x  5 UserEx12 UserEx12 4096 ene  9 13:59 .config
-rw-r--r--  1 UserEx12 UserEx12   22 sep  8  2011 .gtkrc-2.0
-rw-r--r--  1 UserEx12 UserEx12  516 dic 17  2013 .gtkrc-xfce
-rw-r--r--  1 UserEx12 UserEx12  807 ene  6  2022 .profile
```

- Borrar el usuario recién creado

```
$ sudo userdel UserEx12
```

- Crear un usuario (con nombre distinto al anteriormente borrado) que tenga un directorio ***home***

```
$ sudo useradd -m -d /Ex121 UserEx121
```

- Mostrar el ID del usuario en el fichero correspondiente
    - ¿Existe algún problema con el ID de este usuario y el del anterior?

Mediante el mismo comando utilizado anteriormente obtenemos que el UID del nuevo usuario es **1001**.

```
UserEx121:x:1001:1001::/Ex121:/bin/sh
```

Un problema que podemos apreciar es que el UID del usuario creado coincide con el UID del usuario eliminado anteriormente.

- ¿Qué sucede con el usuario recién creado con respecto al directorio home del usuario borrado anteriormente?
    - ¿Existe algún problema de seguridad en el sistema en estos casos?

En estos casos, el problema de que el directorio del usuario eliminado anteriormente se vincula al nuevo usuario puesto que los UID coinciden. Esto puede causar problemas de seguridad, ya que el nuevo usuario accede a los directorios y ficheros del usuario borrado

antes, puede acceder a documentos y directorios con información sensible sobre el anterior usuario, como credenciales, datos bancarios, proyectos, archivos de configuración, etc.

**(e13)** ¿Dónde se almacenan contraseñas de los grupos?

Antes de nada, vamos a ver dónde se almacena la información relativa a los grupos.

Esta información la podemos encontrar en el fichero `/etc/group`, que es donde se definen los grupos en Linux. Cada línea de este fichero contiene:

```
nombre_grupo:x:ID_grupo:miembro1,miembro2,...
```

Donde:

- `nombre_grupo`: El nombre del grupo.
- `x`: Indica que la contraseña del grupo se almacena en el archivo `/etc/gshadow`.
- `ID_grupo`: Identificador numérico del grupo.
- `miembro1, miembro2,...`: Usuarios miembros de este grupo.

Una vez visto cómo se ve la información de los grupos, podemos ver las contraseñas de los grupos en el fichero `/etc/gshadow`, mencionado más arriba mediante el comando `sudo cat etc/gshadow`.

Al igual que los usuarios, se emplean algoritmos hash de encriptación y "técnicas de salado" para proteger las contraseñas de los grupos. Estas técnicas de salado consisten en aplicar una cadena aleatoria a la contraseña antes de utilizar la función hash en ella.

**(e14)** Crear varios usuarios y grupos y asignar de forma temporal un grupo primario a un usuario.

- Experimentar creando/borrando archivos

Creamos un grupo donde colocaremos dos usuarios también creados:

```
$ sudo groupadd GroupEx14
$ sudo useradd -m User1 -p$(openssl passwd 1234)
$ sudo useradd -m User2 -p$(openssl passwd 1234)
```

Creamos un directorio en el que añadiremos dos ficheros de prueba más adelante:

```
$ sudo mkdir /home/Ejercicio14
```

Cambiamos el grupo propietario del directorio:

```
$ sudo chown :GroupEx14 /home/Ejercicio14/
```

Añadimos los usuarios al grupo y a User2 le asignamos el grupo como grupo primario

```
$ sudo usermod -aG GroupEx14 User1
$ sudo usermod -aG GroupEx14 User2
$ sudo usermod -g GroupEx14 User2
```

Damos permisos al directorio para lectura, escritura y ejecución únicamente al usuario dueño y al grupo

```
$ sudo chmod 770 /home/Ejercicio14
```

A continuación, vamos a iniciar sesión como **User2**, crearemos dos ficheros y eliminaremos uno de ellos y dejaremos el otro para que **User1** intente eliminarlo

```
$ sudo login User2
Contraseña:
$ bash
User2@pablozar12-GP66-Leopard-10UE:~$ touch /home/Ejercicio14/Prueba1.txt
User2@pablozar12-GP66-Leopard-10UE:~$ touch /home/Ejercicio14/Prueba2.txt
$ ls /home/Ejercicio14/
Prueba1.txt Prueba2.txt
```

Ahora vamos a intentar eliminar el fichero **Prueba2.txt** logeados como **User2**

```
$ rm /home/Ejercicio14/Prueba2.txt
$ ls /home/Ejercicio14/
Prueba1.txt
```

Como podemos observar el fichero se ha borrado correctamente

Ahora vamos a intentar eliminar el fichero **Prueba1.txt** logeados como **User1**

```
$ sudo login User1
Contraseña:
$ bash
User1@pablozar12-GP66-Leopard-10UE:~$ rm /home/Ejercicio14/Prueba1.txt
rm: no se puede borrar '/home/Ejercicio14/Prueba1.txt': Permiso denegado
```

Así, vemos que al no ser miembros del grupo primario actual de **User2** nos deniega el permiso de borrar el fichero

**(e15)** ¿Los grupos son creados exclusivamente por el usuario **root** a través del comando **groupadd**?

Pese a ser recomendable, no es estrictamente necesario que sea el usuario **root** el responsable de la creación de grupos, a través de la primitiva **groupadd**. También está capacitado cualquier usuario que tenga permisos o privilegios de super-usuario, a través de la utilidad **sudo**, seguido del comando a usar, que en este caso es **groupadd**.

**(e16)** Crear un grupo y ver que ID se le ha asignado. Luego cambiar el ID por otro y también su nombre.

Para crear un grupo hacemos uso del comando **groupadd**:

```
$ sudo groupadd GroupEx16
```

Una vez creado, vamos a comprobar su id visualizando el fichero **/etc/group** mediante el comando **sudo cat /etc/group**, en donde vemos que al final de dicho documento encontramos el grupo creado:

```
GroupEx16:x:1001:
```

Así pues, vemos que el identificador asociado al grupo es el **1001**.

A continuación, vamos primero a cambiar el identificador del grupo y mostrar el fichero `/etc/group` modificado y también haremos lo mismo, pero modificando el nombre.

Para modificar un grupo hacemos uso del comando `groupmod` con las opciones `-g` y `-n`, las cuales sirven para modificar el identificador y el nombre del grupo respectivamente.

Primero vamos a modificar el ID del grupo a 8888, donde para ello usaremos la opción `-g`:

```
$ sudo groupmod -g 8888 GroupEx16
```

El resultado de modificar el identificador es `GroupEx16:x:8888:`

Una vez modificado el identificador, vamos a modificar el nombre del grupo a `ModifiedGEx16`, empleando la opción `-n`:

```
$ sudo groupmod -n ModifiedGEx16 GroupEx16
```

El resultado queda así: `ModifiedGEx16:x:8888:`

**(e17)** Crear un archivo y modificar permisos ACL añadiendo varios permisos para otros usuarios/grupos

- ¿Cambia la forma en que se muestran los permisos del archivo con el comando `ls`?

Para la realización de este ejercicio vamos a crear un usuario y un grupo, aparte de hacer uso de nuestro usuario personal.

```
$ sudo useradd UserEx17
$ sudo groupadd GroupEx17
```

En primer lugar, crearemos un fichero llamado Ejercicio17.txt mediante el comando `touch`:

```
$ touch Ejercicio17.txt
```

Una vez creado el fichero, gestionaremos los permisos ACL empleando el comando `setfacl`: donde asignaremos permisos de lectura y escritura a nuestro usuario personal, permisos de solo lectura al usuario `UserEx17` y permisos de lectura, escritura y ejecución al grupo creado anteriormente:

```
$ setfacl -m u:pablozar12:rw Ejercicio17.txt
$ setfacl -m u:UserEx17:r Ejercicio17.txt
$ setfacl -m g:GroupEx17:rwx Ejercicio17.txt
```

Una vez gestionados los permisos ACL, sólo podemos ver como se han modificado estos permisos mediante el comando `getfacl`, ya que los permisos que se muestran haciendo uso de `ls` no se ven afectados:

```
$ getfacl Ejercicio17.txt
# file: Ejercicio17.txt
```

```
# owner: pablozar12
# group: pablozar12
user::rw-
user:pablozar12:rw-
user:UserEx17:r--
group::rw-
group:GroupEx17:rwx
mask::rwx
other::r--
```

**(e18)** Establecer los permisos ACL por defecto en un directorio y crear archivos en este directorio.

- ¿Se heredan los permisos ACL?

Para la realización de este ejercicio vamos a utilizar el usuario creado en el ejercicio anterior, `UserEx17`.

Para empezar, crearemos un directorio llamado Ejercicio18:

```
$ mkdir Ejercicio18
```

A continuación, le asignaremos permisos ACL de solo lectura al usuario `UserEx17` y comprobamos que los permisos se han modificado correctamente:

```
$ setfacl -m u:UserEx17:r Ejercicio18
$ getfacl Ejercicio18
# file: Ejercicio18
# owner: pablozar12
# group: pablozar12
user::rwx
user:UserEx17:r--
group::rwx
mask::rwx
other::r-x
```

Una vez se han impuesto los permisos ACL, vamos a crear 2 ficheros `.txt` dentro del directorio creado y vamos a comprobar si se han heredado los permisos ACL:

| `$ touch Fichero1.txt Fichero2.txt` | |
|---|---|
| `$ getfacl Ejercicio18/Fichero1.txt` | `$ getfacl Ejercicio18/Fichero2.txt` |
| `# file: Ejercicio18/Fichero1.txt`<br>`# owner: pablozar12`<br>`# group: pablozar12`<br>`user::rw-`<br>`group::rw-`<br>`other::r--` | `# file: Ejercicio18/Fichero2.txt`<br>`# owner: pablozar12`<br>`# group: pablozar12`<br>`user::rw-`<br>`group::rw-`<br>`other::r--` |

Como hemos podido observar, una vez hemos creado los ficheros y hemos comprobado sus permisos ACL se ha podido observar que no han heredado estos permisos.

Este es el comportamiento por defecto, mas para configurar una Lista de Control de Acceso para que herede de forma automática, se deben combinar las opciones `-d` y `-m` de la primitiva `setfacl`, lo cual provocará que nuevos archivos y directorios (y el contenido de los mismos) puedan heredarlos, ya que hasta ahora sólo se queda en el nivel del directorio afectado.

**(e19)** Experimenta con los permisos ACL y los básicos y explica las diferentes configuraciones de permisos efectivos que se alcanzan

Usaremos como ejemplo un fichero llamado `Fichero1.txt` y un usuario extra `Userex19`

```
$ touch Fichero1.txt
$ useradd Userex19
```

Actualmente, el fichero tiene los siguientes permisos:

```
-rw-r----- 1 pablozar12 pablozar12    0 mar 10 16:32 Fichero1.txt
```

Esto significa que el usuario propietario (`pablozar12`) tiene permisos de lectura y escritura (`rw`), el grupo (`pablozar12`) tiene permisos de lectura, y el resto de los usuarios no tienen permisos sobre el fichero.

A continuación, vamos a agregar permisos **ACL** adicionales:

- Damos permiso de lectura y ejecución al usuario `UserEx19`

```
$ setfacl -m u:Userex19:rx Fichero1.txt
```

- Damos permiso de escritura al grupo pablozar12

```
$ setfacl -m g:pablozar12:w Fichero1.txt
```

Ahora vemos los permisos ACL que tiene el fichero

```
$ getfacl Fichero1.txt
# file: Fichero1.txt
# owner: pablozar12
# group: pablozar12
user::rw-
user:Userex19:r-x
group::r--
group:pablozar12:-w-
mask::rwx
other::---
```

Los permisos efectivos combinan los permisos básicos y los ACL, los cuales se resumen así:

- El usuario dueño (`pablozar12`) tiene lectura y escritura
- El usuario `Userex19` tiene lectura y ejecución
- El grupo `pablozar12` tiene lectura y escritura
- Otros usuarios no tienen permisos sobre el fichero

**(e20)** Crear un directorio para colaboración en el que puedan colaborar varios usuarios que pertenezcan a un grupo determinado. Explica su funcionamiento proporcionando ejemplos.

- ¿Cambia la forma en que se muestran los permisos del directorio con el comando `ls`?

En primer lugar, crearemos un directorio que usaremos como directorio compartido:

```
$ sudo mkdir /home/pablozar12/DirectorioCompartido
```

Después crearemos un grupo para colaboradores:

```
$ sudo groupadd colaboradores
```

Una vez creado el grupo, asignamos este al directorio que queremos hacer colaborativo

```
$ sudo chown :colaboradores DirectorioCompartido
```

Asignado el grupo al directorio, procedemos a configurar los permisos del directorio

```
$ sudo chmod 2770 DirectorioCompartido
```

Agregamos los usuarios al grupo

```
$ sudo usermod –aG colaboradores Usuario1
$ sudo usermod –aG colaboradores Usuario2
```

Para verificar que está configurado el grupo correctamente, listamos los miembros del grupo

```
$ sudo members colaboradores
Usuario1 Usuario2
```

Ahora los usuarios miembros del grupo pueden acceder al directorio "`DirectorioCompartido`" y trabajar juntos en archivos, compartir documentos y colaborar en proyectos del directorio.

Una vez hemos dejado todo configurado, si hacemos `ls –l ~`, veremos algo como:

```
drwxrws---  2 root       colaboradores 4096 mar 10 17:44 DirectorioCompartido
```

Tanto el usuario dueño del directorio (`root`) como el grupo de colaboradores tienen permisos de lectura, escritura y ejecución, excepto el resto de los usuarios, que no tienen ningún permiso sobre el directorio.

**(e21)** Crear un directorio de borrado restringido y crear/borrar archivos con diferentes usuarios. Explicar su funcionamiento proporcionando ejemplos.

- ¿Cambia la forma en que se muestran los permisos del directorio con el comando `ls`?

En primer lugar, crearemos un directorio que usaremos como directorio restringido

```
$ sudo mkdir /home/BorradoRestringido
```

Después crearemos un grupo para los usuarios restringidos:

```
$ sudo groupadd restringido
```

Una vez creado el grupo, asignamos este al directorio que queremos hacer colaborativo

```
$ sudo chown :restringido /home/BorradoRestringido
```

A continuación, configuramos los permisos del directorio

```
$ sudo chmod 1770 /home/BorradoRestringido
```

Una vez configurado el directorio y el grupo, creamos los usuarios y los asignamos al grupo deseado

```
$ sudo useradd –m Borrado1 –p$(openssl passwd 1234)
$ sudo useradd –m Borrado2 –p$(openssl passwd 1234)
$ sudo usermod –aG restringido Borrado1
$ sudo usermod –aG restringido Borrado2
```

Verificamos que los usuarios se encuentran correctamente dentro del grupo

```
$ sudo members restringido
Borrado1 Borrado2
```

Configurado todo, si hacemos `ls –l /home` veremos lo siguiente:

```
drwxrwx--T  2 root       restringido   4096 mar 10 19:57 BorradoRestringido
```

Tanto el usuario dueño del directorio (**root**) como el grupo de usuarios restringidos tienen permisos de lectura, escritura y ejecución, excepto el resto de los usuarios, que no tienen ningún permiso sobre el directorio.

A continuación, vamos a intentar eliminar un fichero previamente creado con dos tipos de usuario, uno que no esté en el grupo de usuarios restringidos y otro que sí que lo esté

- Usuario ajeno al grupo:

```
$ rm BorradoRestringido/Fichero1.txt
rm: no se puede borrar 'BorradoRestringido/Fichero1.txt': Permiso denegado
```

- Usuario perteneciente al grupo

```
$ sudo login Borrado1
Contraseña:
Último inicio de sesión: dom mar 10 20:29:43 CET 2024 en pts/2
$ bash
$ rm BorradoRestringido/Fichero1.txt
$
```

# Referencias


alonsojpd. (4 de Junio de 2021). *Usuarios, contraseñas y Grupos a Fondo en Linux*. Obtenido de proyectoa: https://proyectoa.com/usuarios-contrasenas-y-grupos-a-fondo-en-linux/

Kerrisk, M. (22 de Diciembre de 2023). *gpasswd(1) — Linux manual page*. Obtenido de man7.org: https://man7.org/linux/man-pages/man1/gpasswd.1.html

Kerrisk, M. (22 de Diciembre de 2023). *groupadd(8) — Linux manual page*. Obtenido de man7.org: https://man7.org/linux/man-pages/man8/groupadd.8.html

Kerrisk, M. (22 de Diciembre de 2023). *groupmod(8) — Linux manual page*. Obtenido de man7.org: https://man7.org/linux/man-pages/man8/groupmod.8.html

Kerrisk, M. (22 de Diciembre de 2023). *ls(1) — Linux manual page*. Obtenido de man7.org: https://man7.org/linux/man-pages/man1/ls.1.html

Kerrisk, M. (22 de Diciembre de 2023). *useradd(8) — Linux manual page*. Obtenido de man7.org: https://man7.org/linux/man-pages/man8/useradd.8.html

Kerrisk, M. (22 de Diciembre de 2023). *usermod(8) — Linux manual page*. Obtenido de man7.org: https://man7.org/linux/man-pages/man8/usermod.8.html