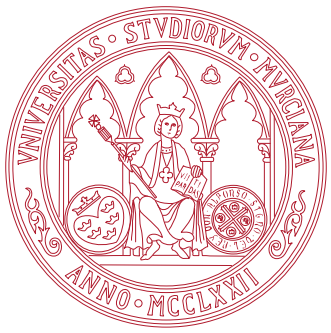

Universidad de Murcia

Facultad de Informática

Departamento de Ingeniería y Tecnología de Computadores

Área de Arquitectura y Tecnología de Computadores

Prácticas de

Introducción a los Sistemas Operativos

2º de Grado en Ingeniería Informática

Boletín de Prácticas 5 – Gestión de usuarios en Linux

Curso 2021/2022

Índice

1. Introducción

En este boletín se va a trabajar con la gestión de los usuarios, creando y configurando cuentas de usuarios mediante órdenes en modo texto. A pesar de la existencia de herramientas gráficas para estos propósitos, la utilización de estas órdenes es preferida por los administradores de sistema dada la mayor rapidez que ofrecen a la hora de gestionar determinadas situaciones (por ejemplo cuando hay que crear muchos usuarios a la vez).

2. Objetivos

Al terminar este boletín deberás ser capaz de:

- Crear cuentas de usuario.
- Controlar qué usuarios hay conectados al sistema y qué labores están realizando.
- Modificar parámetros de una cuenta de usuario: asignar nuevos grupos, cambiar el intérprete de órdenes asignado, etc.
- Gestionar las restricciones de tiempo asignadas a la contraseña y a la cuenta.
- Crear y modificar grupos de usuario.
- Distinguir entre grupo primario, grupos secundarios y grupo activo.
- Eliminar cuentas de usuario y de grupo.
- Gestionar los permisos básicos y especiales de los ficheros, siendo capaz de decidir cuándo es necesario un permiso concreto para solucionar un problema.
- Cambiar el propietario y grupo propietario de un fichero.

3. Órdenes utilizadas

A lo largo de este boletín, algunas de las órdenes que vamos a ver son:

- `su`: cambio de usuario.
- `who`: listar usuarios contectados.
- `useradd`, `usermod` y `userdel`: creación, modificación y eliminación de cuentas de usuario.
- `passwd`: asigna o cambia la contraseña de un usuario.
- `chfn`: cambia la información `GECOS` de un usuario.
- `finger`: muestra la información de un usuario.
- `chsh`: cambia el shell asignado a un usuario.
- `chage`: asigna los valores de envejecimiento de la contraseña y la cuenta de un usuario.
- `groupadd`, `groupmod` y `groupdel`: creación, modificación y eliminación de grupos.
- `id` y `groups`: identificación del usuario y grupos a los que pertenece.
- `newgrp`: cambia de grupo activo.
- `gpasswd`: administra configuración de grupos de usuarios.
- `chmod`, `chown` y `chgrp`: cambiar permisos, propietario y grupo propietario de un fichero.

4. Conceptos básicos

Un usuario es una entidad que ejecuta programas o posee ficheros. Normalmente es una persona que trabaja en el sistema (entra al sistema, edita ficheros, ejecuta programas, etc.), pero también puede ser una cuenta del sistema, como veremos al final del boletín. Las características principales de un usuario son:

- Nombre del usuario (logname o username): nos permite identificarlo de una forma sencilla.
- Identificador de usuario (UID): es un número que identifica al usuario. Es importante tener en cuenta que, internamente, el sistema usa el UID para identificar al usuario, y no su nombre.
- Grupos a los que pertenece: Los grupos son colecciones de usuarios que comparten ficheros o recursos del sistema. Características de un grupo:
  - Nombre del grupo o `groupname`.
  - Identificador del grupo (GID): Internamente el sistema identifica al grupo por este número.

Para cada usuario podemos encontrar dos tipos de grupos:

- Primario: el grupo especificado para este usuario en su definición (como veremos más adelante, las definiciones de todos los usuarios creados en el sistema se encuentran en el fichero `/etc/passwd`).
- Secundarios: los otros grupos a los que pertenece el usuario (como veremos más adelante, esta información se encuentra en el fichero `/etc/group`).

Toda esta información se puede consultar mediante la orden `id`, pasándole como parámetro el nombre del usuario. Así, por ejemplo si ejecutamos:

```
$ id javiercm
uid=503(javiercm) gid=501(profesor) grupos=501(profesor),526(webmaster)
```

vemos cómo el usuario de nombre `javiercm` tiene UID=503 y pertenece a dos grupos: `profesor`, con GID=501, y `webmaster`, con GID=526; siendo `profesor` su grupo primario.

En cada momento, un usuario tiene como grupo activo uno de los grupos a los que pertenece. Cuando el usuario entra al sistema, su grupo activo será su grupo primario, pero el usuario puede cambiar este grupo activo en cualquier momento mediante la orden `newgrp`. Cuando se ejecuta esta orden lo que realmente ocurre es que se lanza un nuevo shell, manteniendo el mismo usuario, pero cambiando el grupo activo de éste. En todo momento, un usuario puede consultar cuál es grupo activo ejecutando la orden `id` sin parámetros. Por ejemplo, si el usuario `javiercm` ejecuta:

```
$ id
uid=503(javiercm) gid=501(profesor) grupos=501(profesor),526(webmaster)
$ newgrp webmaster
$ id
uid=503(javiercm) gid=503(webmaster) grupos=526(webmaster),501(profesor)
$ exit
exit
$ id
uid=503(javiercm) gid=501(profesor) grupos=501(profesor),526(webmaster)
```

podemos apreciar que este usuario pertenece a dos grupos (`profesor` y `webmaster`), teniendo inicialmente como grupo activo `profesor`. Al ejecutar `newgrp webmaster`, se lanza uno nuevo shell

donde ahora su grupo activo es `webmaster`. Finalmente, cuando el usuario ejecuta `exit`, vuelve al shell inicial, con grupo activo `profesor`.

Sin embargo, es importante tener en cuenta, que si seguimos el mismo ejemplo pero ejecutando `id` con el nombre del usuario como parámetro, la información mostrada no cambia respecto a a la situación previa a `newgrp`, pues ésta corresponde a la información que está guardada en los ficheros `/etc/passwd` y `/etc/group`, es decir, el grupo primario y los diferentes grupos secundarios, tal como se dijo anteriormente, sin considerar cuál es su grupo activo en cada momento. Es decir:

```
$ id javiercm
uid=503(javiercm) gid=501(profesor) grupos=501(profesor),526(webmaster)
$ newgrp webmaster
$ id javiercm
uid=503(javiercm) gid=501(profesor) grupos=501(profesor),526(webmaster)
$ exit
exit
$ id javiercm
uid=503(javiercm) gid=501(profesor) grupos=501(profesor),526(webmaster)
```

Tal como se vio en clase de teoría, el UID de un usuario y los grupos a los que pertenece marcarán los permisos que tiene este usuario para acceder a ficheros o recursos. De esta manera, cuando un usuario crea un fichero, se establece como grupo propietario de este fichero el grupo activo del usuario en ese momento. Por otro lado, cuando un usuario solicita realizar una operación (lectura, escritura o ejecución) sobre un fichero del que no es el propietario, se le permitirá realizar esta operación si tiene el permiso correspondiente, al menos, para uno los grupos a los que pertenece dicho usuario. Por ejemplo, dado el usuario `juan`:

```
$ id juan
uid=1000(juan) gid=1000(juan) grupos=1001(empleados),1002(clientes)
```

y estos ficheros pertenecientes a diferentes usuarios y grupos:

```
-rw-rw----. 2 pepe   empleados    2 mar 30 11:44 f1
-rw-r-----. 2 paco   clientes     0 mar 30 11:44 f2
-rw-r-----. 1 jose   proveedores  5 mar 30 11:45 f3
```

entonces, el usuario `juan` podrá leer o escribir en el fichero `f1` (por pertenecer al grupo `empleados`) y podrá leer del fichero `f2` (por pertenecer al grupo `clientes`), pero no podrá acceder al fichero `f3` ni siquiera para leerlo (al no pertenecer al grupo `proveedores`).

Por último, como curiosidad, hay que tener en cuenta que si, mientras un usuario está conectado al sistema, el superusuario realiza algún cambio en la información de definición del usuario, como por ejemplo, añadir/eliminar este usuario a/de algún grupo, a este usuario no se le mostrarán estos cambios cuando ejecuta `id` sin parámetros. Aunque, por supuesto, dichos cambios ya son efectivos, pues han quedado guardados en los ficheros `/etc/passwd` y `/etc/group`. Por ejemplo, suponemos que el usuario `paco`, que pertenece únicamente al grupo `paco`, está conectado al sistema y ejecuta:

```
$ id
uid=1003(paco) gid=1003(paco) grupos=1003(paco)
```

Si en este momento, el superusuario añade el usuario `paco` al grupo `becario` y al grupo `alumno`, la información que puede consultar dicho usuario cuando ejecuta la orden `id` sin parámetros no habrá cambiado:

```
$ id
uid=1003(paco) gid=1003(paco) grupos=1003(paco)
```

En cualquier caso, siempre podrá consultar estos cambios ejecutando `id` pasándole como parámetro su propio nombre de usuario.

```
$ id paco
uid=1003(paco) gid=1003(paco) grupos=1003(paco),1000(alumno),1005(becario)
```

4.1. El superusuario

En sistemas operativos del tipo Unix, el usuario `root`, también llamado *supersusuario*, es el nombre convencional de la cuenta de usuario que posee todos los derechos. El usuario `root` puede hacer muchas cosas que están vedadas a un usuario común, tales como hacer modificaciones en la configuración del sistema, instalar programas, modificar configuraciones de servidores, administrar usuarios, etc. El superusuario siempre tiene como UID el valor 0.

No es recomendable utilizar el usuario `root` para una simple sesión de uso habitual, ya que pone en riesgo el sistema al garantizar acceso privilegiado a cada programa en ejecución. Es preferible utilizar una cuenta de usuario normal y utilizar la orden `su` para acceder a los privilegios de `root` en caso de ser necesario, como veremos más adelante.

4.2. Orden `who`

La orden `who` muestra un listado de los usuarios con sesiones abiertas. Por ejemplo:

```
$ who
jcamara  pts/0        2017-09-18 16:59 (77.210.244.55)
jgines   pts/3        2017-09-18 18:17 (185.154.58.181)
javiercm pts/4        2017-09-18 19:25 (132.red-79-154-79)
```

En este ejemplo, vemos como hay 3 usuarios conectados: el usuario `jcamara` está conectado a la terminal `pts/0`, `jgines` a la `pts/3` y `javiercm` a la `pts/4`. Igualmente, podemos ver la fecha y hora de inicio de cada conexión, así como la dirección IP desde donde se conectaron.

4.3. Orden `su`

La orden `su` ejecuta un nuevo shell, con los identificadores de usuario (UID) y grupo (GID) del nuevo usuario indicado, de tal manera que podremos trabajar en el sistema como si efectivamente fuésemos dicho usuario. Se operará con estos nuevos identificadores hasta salir con la orden `exit` del shell que se ha lanzado. Por defecto, se utilizará el usuario `root`.

Algunas opciones de la orden `su` son:

- Si se ejecuta sin opciones, el valor de la mayoría de las variables de entorno no varía, excepto las variables `HOME` y `SHELL`. En caso de haber cambiado a un usuario distinto del root, también cambian las variables `USER` y `LOGNAME`. Por ejemplo:

  ```
  login: javiercm
  password: ****

  $ su
  password: ****
  # echo $USER
  javiercm
  # echo $LOGNAME
  javiercm
  # exit

  $ su pepe
  password: ****
  $ echo $USER
  ```

```
pepe
$ echo $LOGNAME
pepe
# exit
```

- -l : se cambian todas las variables de entorno, para cualquier usuario, como si se hubiera hecho un login de nuevo. (En algunas versiones esta opción ha sido eliminada y han dejado sólo la siguiente). Por ejemplo:

```
login: javiercm
password: ****

$ su -l
password: ****
# echo $USER
root
# echo $LOGNAME
root
```

- - : equivalente a -l.

El superusuario puede usar esta orden para actuar fácilmente como otro usuario y así comprobar la configuración de su cuenta, resolver algún problema que pueda tener el usuario, etc.

Cuando esta orden la ejecuta un usuario normal, se le solicitará la contraseña del usuario al que quiere cambiar.

5. Gestión de las cuentas de usuarios

En esta sección, en primer lugar se describirán los ficheros que se utilizan para almacenar la información relativa a las cuentas de los usuarios. Cualquier modificación que se quisiese hacer sobre la configuración de usuarios se podría llevar a cabo editando y cambiando el contenido de estos ficheros directamente. Sin embargo, en general, resulta más recomendable realizar estos cambios mediante las órdenes que se verán en el apartado 5.2.

5.1. Ficheros de configuración de usuarios

La información de las cuentas de usuario se guarda en estos ficheros de configuración:

- `/etc/passwd`: Información sobre las cuentas de los usuarios definidos en el sistema.
- `/etc/shadow`: Contraseñas encriptadas e información de envejecimiento de las cuentas de usuario.

5.1.1. Fichero `/etc/passwd`

Este fichero contiene la lista de los usuarios definidos en el sistema. El formato de cada una de sus líneas es:

```
name:password:uid:gid:gecos:home:shell
```

donde:

- `name`: Es el nombre asignado al usuario (logname o username).

- `password`: Tradicionalmente este campo guardaba la contraseña encriptada, pero en la actualidad guarda una `x` indicando que la contraseña se almacena en el fichero `/etc/shadow` por mayor seguridad.
- `uid`: Es el identificador asignado al usuario. Para los usuarios del sistema se reserva desde el 1 al 999, mientras que los valores a partir de 1000 se utilizan para usuarios normales, y el UID 0 está reservado para el usuario root. Es importante tener en cuenta que si dos usuarios comparten el mismo UID serán considerados como el mismo usuario para el sistema. Esta compartición sólo debería usarse en casos muy concretos, y con precaución. Por ejemplo, puede ser útil para tener a dos usuarios, con nombre y clave de acceso diferente, como administradores del sistema (UID=0).
- `gid`: Es el identificador del grupo primario al que pertenece el usuario.
- `gecos`: Contiene diversa información que identifica al usuario: nombre completo, teléfono, despacho, etc. El superusuario puede usar la orden `chfn` para cambiar esta información
- `home`: Contiene la ruta absoluta al directorio de trabajo del usuario. Cuando el usuario se autentica y entra al sistema, este es el directorio donde se le posiciona. En principio, es el único directorio donde el usuario podrá crear sus ficheros. El propietario ha de ser el usuario y el grupo propietario suele ser el grupo primario del usuario, aunque esto último se podría cambiar.
- `shell`: Contiene la ruta del intérprete de órdenes usado para ese usuario. Se ejecuta automáticamente cuando el usuario entra al sistema. El valor de este campo se puede cambiar con la orden `chsh`. Este intérprete podrá escogerse de entre los que aparezcan en el fichero `/etc/shells`, donde se indican los shells permitidos. Si se elimina un shell de este fichero, este shell quedaría como prohibido, con lo que no se podrá elegir como nuevo, pero los usuarios que ya lo tienen asignado lo seguirán usando sin problemas. Algunos puntos a considerar sobre la elección del intérprete de órdenes:
  - Si un usuario no tiene asignado ningún intérprete de órdenes, se usará el shell por defecto `/bin/sh`.
  - Si se desea que el usuario no pueda entrar al sistema se le debe asignar `/bin/false` o `/sbin/nologin`.
  - Se puede indicar que el shell de un usuario es un fichero ejecutable específico, convirtiendo, por tanto, esta cuenta en una **cuenta restringida**[1]. De esta manera, cuando este usuario entre al sistema se ejecuta ese fichero ejecutable, y, al finalizar la ejecución, el usuario sale del sistema automáticamente, sin llegar a hacer un login propiamente dicho. El usuario asociado tiene que tener los permisos necesarios para poder hacer la tarea asignada. Dos ejemplos podrían ser:
    - Usuario para realizar las copias de seguridad: como shell tiene asignado un ejecutable (guión shell) que hace esta tarea.
    - Usuario para apagar el equipo: como shell tiene asignado la orden: `/usr/bin/systemctl poweroff`.

Un ejemplo de línea de este fichero sería:
`pilar:x:1008:100:Pilar Gonzalez Ferez,3.45:/home/pilar:/bin/bash`

5.1.2. Fichero `/etc/shadow`

Las contraseñas encriptadas de los usuarios no se guardan en el `/etc/passwd`, sino en el fichero `/etc/shadow`, ya que éste es más seguro, pues sus permisos son `---------`, siendo el usuario propietario el `root` y el grupo propietario el `root`. Por otro lado, en este fichero también se guarda

[1] Las cuentas restringidas permiten limitar las acciones de los usuarios en el sistema, haciendo que tengan determinadas restricciones.

información para establecer restricciones de tiempo o envejecimiento respecto a la validez de cada cuenta de usuario y de su contraseña. El formato de cada una de sus líneas es:

```
name:password:changed:minlife:maxlife:warn:inactive:expired:unused
```

donde:

- `name`: nombre asignado al usuario (logname o username).
- `password`: contraseña del usuario. Puede haber una de las siguientes opciones:
  - Una contraseña encriptada.
  - '*','!!': La cuenta está bloqueada y no se puede usar.
- `changed`: fecha del último cambio de contraseña (expresada como el número de días desde el 1 de enero de 1970).
- `minlife`: número de días que han de pasar para poder cambiar la contraseña después de hacer un cambio.
- `maxlife`: número de días máximo que puede estar con la misma contraseña sin cambiarla después de haber hecho un cambio.
- `warn`: cuántos días antes de que la contraseña expire (`maxlife`) será informado sobre ello, indicándole que tiene que cambiarla.
- `inactive`: número de días que la contraseña seguirá siendo válida después de que expire. El usuario debería cambiarla la próxima vez que entre. Transcurrido este tiempo la cuenta se deshabilitará.
- `expired`: fecha en la que la cuenta expira y se deshabilita de forma automática (expresada como el número de días desde el 1 de enero de 1970).

5.2. Herramientas de gestión de usuarios

Las herramientas más importantes para gestionar la información almacenada en los ficheros descritos en la sección 5.1 son las órdenes:

- `useradd`: para crear nuevas cuentas de usuario.
- `usermod`: para modificar las características de una cuenta de usuario ya existente.
- `userdel`: para eliminar cuentas (por defecto no borra el directorio personal).
- `passwd`: para gestionar las contraseñas.
- `chage`: para gestionar las restricciones de tiempo de las cuentas.

5.2.1. Orden `useradd`

La orden `useradd` se utiliza para crear nuevas cuentas de usuario. Su sintaxis es:

```
useradd [opciones] usuario
```

Las opciones más importantes de esta orden son:

- `-d HOME_DIR`: El nuevo usuario se creará utilizando `HOME_DIR` como su directorio personal. Por defecto, si no se indica esta opción, se creará un directorio personal usando el nombre del usuario.

- `-g GROUP`: El grupo principal del usuario será `GROUP`. Este grupo debe existir previamente. Si no se especifica esta opción, dependiendo de la configuración establecida por defecto en el sistema, o bien el grupo principal del usuario será un nuevo grupo con el mismo nombre del usuario, o bien, será un grupo por defecto.
- `-G GROUP [,GROUP2,...[,GROUPN]]`: Para indicar una lista de grupos secundarios a los que también pertenece el usuario.
- `-m`: Para forzar la creación del directorio personal, en caso de que éste no exista previamente. Si no se especifica esta opción, la creación o no del directorio personal dependerá de la configuración establecida por defecto en el sistema[2].
- `-s SHELL`: Para que `SHELL` sea el shell inicial del usuario.
- `-c COMMENT`: Para establecer la información que identifica a un usuario (nombre completo, despacho, teléfono del despacho, teléfono de casa). Esta información es la que aparece en el campo de comentarios, `gecos`, del usuario.

Veamos varios ejemplos de uso de esta orden:

```
# useradd paco -m

# id paco
uid=1003(paco) gid=1003(paco) grupos=1003(paco)
# su -l paco
$ whoami
paco
$ pwd
/home/paco

# useradd pepe -m -G usuarios

# id pepe
uid=1004(pepe) gid=1014(pepe) grupos=1014(pepe),1013(usuarios)

# useradd juan -m -g profesores -G usuarios
# id juan
uid=1005(juan) gid=1012(profesores) grupos=1012(profesores),1013(usuarios)

# useradd jesus -m -d /home/jesusito
# su -l jesus
$ whoami
jesus
$ pwd
/home/jesusito
```

En estos ejemplos, en primer lugar, se crea el usuario `paco`, para el que se crea el grupo con su mismo nombre y se le asigna como grupo primario. A continuación, se ha creado el usuario `pepe`, asignándole `usuarios` como grupo secundario. Tras ello, se ha creado el usuario `juan`, que tendrá

[2] A la hora de establecer para una nueva cuenta sus características básicas iniciales (el shell por defecto que va a tener el usuario, si se le va a crear un directorio personal y donde se crearía en tal caso, las restricciones de tiempo de la contraseña, etc.) las órdenes de gestión de usuarios utiliza los valores por defecto que se encuentran en los ficheros `/etc/default/useradd` y `/etc/login.defs`.

profesores como grupo primario y usuarios como secundario. Finalmente, se ha creado el usuario jesus, que tendrá como directorio de trabajo /home/jesusito.

Cuando se crea un usuario nuevo de esta manera, sin asignarle contraseña directamente, su cuenta se encontrará bloqueada inicialmente, con lo que este usuario no podrá entrar al sistema. Para cambiar la contraseña de una cuenta, o asignársela inicialmente, se utiliza la orden passwd, cuyo uso describiremos más adelante.

5.2.2. Orden usermod

La orden usermod se utiliza para modificar las características de una cuenta de usuario ya existente. Su sintaxis es:

```
usermod [opciones] usuario
```

Las opciones más importantes de esta orden son semejantes a las de la orden useradd:

- -g GROUP: Para que GROUP sea el grupo principal del usuario.
- -G GROUP [,GROUP2,...[,GROUPN]]: Para fijar la lista de grupos secundarios a los que pertenece el usuario. Si se quiere añadir al usuario a nuevos grupos secundarios sin quitarlo de los que ya forma parte, hay que añadir la opción -a.
- -s SHELL: Para que SHELL sea el nuevo shell inicial del usuario. También se puede realizar este cambio de shell usando la orden chsh.
- -c COMMENT: Para cambiar información que identifica a un usuario. También se puede cambiar esta información usando la orden chfn.

Veamos algunos ejemplos de uso de esta orden:

```
# id paco
uid=1007(paco) gid=1015(paco) grupos=1015(paco)

# usermod paco -g profesores
# id paco
uid=1007(paco) gid=1012(profesores) grupos=1012(profesores)

# usermod paco -g paco
# id paco
uid=1007(paco) gid=1015(paco) grupos=1015(paco)

# usermod paco -G usuarios
# id paco
uid=1007(paco) gid=1015(paco) grupos=1015(paco),1013(usuarios)

# usermod paco -G profesores
# id paco
uid=1007(paco) gid=1015(paco) grupos=1015(paco),1012(profesores)

# usermod paco -G usuarios -a
# id paco
uid=1007(paco) gid=1015(paco) grupos=1015(paco),1012(profesores),
1013(usuarios)

# usermod paco -c Francisco,3.41,4821,900100100
```

```
# usermod paco -s /bin/sh

# finger paco
Login: paco                             Name: Francisco
Directory: /home/paco                   Shell: /bin/sh
Office: 3.41, x4821                     Home Phone: 900100100
Never logged in.
No mail.
No Plan.
```

En primer lugar, se establece `profesores` como grupo principal de `paco`, pero, a continuación, se vuelve a establecer `paco` como su grupo principal. Tras ello, se indica que el usuario `paco` tiene como grupo secundario `usuarios`. A continuación, se realiza la misma operación para el grupo `profesores`, pero como no se utiliza la opción `-a`, se elimina la pertenencia al grupo `usuarios`; por lo que, de cara a que tengamos ambos grupos, `profesores` y `usuarios`, como grupos secundarios de `paco`, se utiliza dicha opción `-a`. Finalmente, se cambia la información personal de este usuario, utilizando la opción `-c`, y su shell inicial, con la opción `-s`.

5.2.3. Orden `passwd`

Como se dijo anteriormente, cuando se crea un usuario nuevo, por ejemplo con la orden `useradd`, sin asignarle contraseña, su cuenta se encontrará bloqueada inicialmente, con lo que este usuario no podrá entrar al sistema. Sin embargo, aunque una cuenta esté bloqueada, sí puede ejecutar procesos que haya programado con herramientas de planificación. Para cambiar la contraseña de una cuenta, o asignársela inicialmente, se utiliza la orden `passwd` de esta manera:

- `passwd <nombre_usuario>`: la utiliza el `root` para asignar/cambiar la contraseña a un usuario. Dicha contraseña será solicitada tras iniciar la ejecución de esta orden. Por ejemplo:

  ```
  # passwd javiercm
  Cambiando la contraseña del usuario javiercm.
  Nueva contraseña: ******
  Vuelva a escribir la nueva contraseña: ******
  passwd: todos los tokens de autenticación se actualizaron
  exitosamente.
  ```

- `passwd --stdin <nombre_usuario>`: esta opción la utiliza el `root` para asignar/cambiar la contraseña a un usuario. La diferencia con el caso anterior, cuando se usaba la orden sin opciones, es que, con esta opción, la contraseña se leerá de la entrada estándar. Así, por ejemplo, se le puede hacer llegar la contraseña a esta orden mediante una tubería. Por ejemplo:

  ```
  # echo "clavesecreta" | passwd --stdin javiercm
  Cambiando la contraseña del usuario javiercm.
  passwd: todos los tokens de autenticación se actualizaron
  exitosamente.
  ```

- `passwd -e <nombre_usuario>`: la utiliza el `root` para obligar al usuario a cambiar su contraseña la próxima vez que entre al sistema.

- `passwd -l <nombre_usuario>`: la utiliza el `root` para bloquear la contraseña a un usuario.
- `passwd -u <nombre_usuario>`: la utiliza el `root` para desbloquear la contraseña a un usuario.
- `passwd`: la utiliza cualquier usuario para cambiar su propia contraseña.

La herramienta `passwd` hace un chequeo, previamente a realizar el cambio efectivo de contraseña, para asegurar que la contraseña elegida cumpla ciertos parámetros de seguridad (longitud mínima, validez, etc.).

Periódicamente, se debe forzar a que los usuarios cambien sus contraseñas de cara al mantenimiento de la seguridad general de éstas. Como veremos en la siguiente sección, una forma para llevar a cabo esta labor es gestionando apropiadamente la información de envejecimiento que se asocia a las cuentas.

5.2.4. Orden `chage`

Tal como se describió anteriormente, para las cuentas de los usuarios se pueden establecer restricciones de tiempo o envejecimiento respecto a la validez de su cuenta y de su contraseña. Estas restricciones se expresan mediante ciertos valores que se guardan en el fichero `/etc/shadow` para cada usuario.

Los valores de estas restricciones temporales los establece el administrador con las órdenes `chage` (o también puede ser con la orden `passwd`). El fichero `/etc/login.defs` tiene los valores por defecto.

Ejemplo: supongamos que el administrador ejecuta la siguiente orden de configuración en el verano de 2020:

```
# chage -M 20 -W 6 -I 5 -E 2020-10-30 pilar
```

Si el usuario `pilar` cambia su contraseña el 1 de octubre de 2020, los tiempos quedarían fijados de la siguiente manera:

- El usuario `pilar` tiene hasta el 21 de octubre, inclusive, para cambiar su contraseña de manera voluntaria, pues puede estar 20 días sin cambiarla desde que la fijó por última vez (el 1 de octubre). Por tanto, a las 12 de la noche del día 21 su contraseña habrá expirado.
- A partir del inicio del día 15 de octubre (6 días antes de la expiración de la contraseña), `pilar` empezará a recibir avisos para que cambie su contraseña.
- Si no cambia la contraseña antes de las 12 de la noche del día 21, a partir de ese momento el usuario no podrá acceder al sistema. Sin embargo, como se ha fijado un tiempo de inactividad de 5 días, la cuenta aún no se ha bloqueado completamente y el usuario todavía podrá cambiar su contraseña. Para ello, deberá iniciar sesión, momento en el cual el sistema le pedirá que cambie la contraseña. Si la cambia, el usuario podrá entrar al sistema como de costumbre. Si no, la sesión se cerrará.
- Si a las 12 de la noche del 26 de octubre (5 días tras la expiración de la contraseña) aún no la ha cambiado, la cuenta se bloqueará. Si el usuario quiere acceder al sistema, le tendrá que pedir al administrador que desbloquee la cuenta (por ejemplo, asignándole una nueva contraseña).
- En cualquier caso, aunque la contraseña no haya caducado, la cuenta expirará, y por tanto se bloqueará, al empezar el día 30 de octubre de 2020, independientemente de que la contraseña todavía sea válida o no.

Toda esta información se puede consultar utilizando la opción `-l` de la orden `chage` de esta manera:

```
# chage -l pilar
Último cambio de contraseña :oct 1, 2020
La contraseña caduca : oct 21, 2020
Contraseña inactiva : oct 26, 2020
```

```
La cuenta caduca : oct 30, 2020
Número de días mínimo entre cambio de contraseña : 0
Número de días máximo entre cambio de contraseña : 20
Número de días de aviso antes de que caduque la contraseña : 6
```

6. Gestión de grupos de usuarios

La información relativa a los grupos de usuarios se guarda en estos ficheros de configuración:

- `/etc/group`: Información sobre los grupos definidos y los usuarios miembros de los mismos. El formato de cada una de sus líneas es:

  ```
  name:x:gid:list_of_users
  ```

- `/etc/gshadow`: Información de seguridad de los grupos (grupo, contraseña, y también la relación de usuarios administradores y miembros). El formato de cada una de sus líneas es:

  ```
  name:password:list_of_administrators:list_of_members
  ```

De esta manera, como ocurría con las cuentas de usuarios, cualquier modificación que se quisiese hacer sobre la configuración de de sus grupos se podría llevar a cabo editando y cambiando el contenido de estos ficheros. Sin embargo, en general resulta más recomendable realizar estos cambios mediante las órdenes de gestión de grupos, siendo las más importantes:

- `groupadd grupo`: para crear un nuevo grupo.
- `groupdel grupo`: para eliminar un grupo
- `gpasswd`: para administrar la información contenida en los ficheros de configuración de grupos.

Tenemos que considerar que cada grupo puede tener usuarios administradores, usuarios miembros y, además, puede contar con una contraseña de acceso. La lista de usuarios miembros y administradores de un grupo la establece el usuario `root` de esta manera:

- `gpasswd -A usuarios grupo`: para definir la lista de usuarios (nombres de usuarios separados por comas) que serán administradores del grupo.
- `gpasswd -M usuarios grupo`: para definir la lista de usuarios miembros del grupo.

Si un usuario es definido como administrador de un grupo podrá utilizar a partir de ese momento esta orden para realizar diversas labores de gestión del grupo (obviamente, el usuario `root` siempre puede realizarlas, en cualquier caso):

- `gpasswd -a usuarios grupo`: para añadir un usuario al grupo.
- `gpasswd -d usuarios grupo`: para sacar a un usuario del grupo.

Veamos un ejemplo, con algunos usos de estas órdenes:

```
# groupadd estudiantes
# gpasswd -M paco,pepe estudiantes
# grep estudiantes /etc/group
estudiantes:x:1009:paco,pepe
# grep estudiantes /etc/gshadow
estudiantes:!::paco,pepe
# gpasswd -A paco estudiantes
# grep estudiantes /etc/gshadow
```

```
estudiantes:!:paco:paco,pepe

# su -l paco
$ gpasswd -a juan estudiantes
Añadiendo al usuario juan al grupo estudiantes
$ grep estudiantes /etc/group
estudiantes:x:1009:paco,pepe,juan
$ gpasswd -d pepe estudiantes
Eliminando al usuario pepe del grupo estudiantes
$ grep estudiantes /etc/group
estudiantes:x:1009:paco,juan
```

En este ejemplo, el superusuario crea el grupo `estudiantes`, al que añade a continuación a los usuarios `paco` y `pepe`. Tras ello, nombra a `paco` como administrador del grupo. A partir de ahora, vemos como el usuario `paco` ya puede administrar el grupo, añadiendo al usuario `juan` y eliminando del grupo al usuario `pepe`.

7. Otros aspectos sobre gestión de usuarios

7.1. Mensajes de bienvenida

El fichero `/etc/issue` contiene un mensaje que se mostrará cuando nos conectemos a un terminal antes de pedirnos el `login`. En el mensaje se pueden incluir caracteres de control que se sustituirán por su correspondiente valor cuando se muestre el mensaje. Por ejemplo, se puede incluir «`\l`», que se sustituirá por el nombre de la terminal de texto (`tty2`, `tty3`, etc.) en la que se muestre el mensaje.

Por otro lado, el fichero `/etc/motd` contiene un mensaje que se mostrará cuando un usuario entre en el sistema. Por defecto está vacío. Puedes modificar este fichero para incluir mensajes recordando cosas tales como que se limpien los directorios usados, no se haga un consumo excesivo del disco, o notificar la fecha de la próxima copia de seguridad, por ejemplo.

Hay que tener en cuenta que todo esto no se aplica cuando en modo gráfico abrimos una consola, ya que en este caso realmente no estamos haciendo un verdadero `login` en el sistema.

7.2. Ficheros de inicialización

Los ficheros de inicialización de un usuario son unos guiones shell que tiene dicho usuario en su directorio `$HOME`:

- `.bash_profile`: Este guión se ejecuta automáticamente cuando el usuario inicia su sesión, ya sea de manera local o remota desde otro ordenador.
- `.bashrc`: Una vez que el usuario está conectado a una máquina, este guión se ejecuta automáticamente cada vez que dicho usuario abre un nuevo terminal de intérprete de órdenes, o incluso si desde dentro de un shell se inicia una nueva instancia de intérprete de órdenes, por ejemplo tecleando `/bin/bash`.

Por otra parte, cada usuario también cuenta en su directorio `$HOME` con un fichero de finalización:

- `.bash_logout`: Este guión se ejecuta automáticamente cuando el usuario sale del sistema, es decir, al cerrar la sesión.

Alguna de las labores que pueden llevar a cabo los ficheros de inicialización son:

- Crear o modificar variables de entorno que el usuario puede usar o necesitar. Por ejemplo, con:

```
PATH=$PATH:/midir
EDITOR=gedit
export PATH EDITOR
```

se añade `/midir` a lista de directorios que contiene la variable `$PATH` y se crea una variable de nombre `EDITOR` con valor `gedit`. Finalmente, ambas variables se establecen como variables de entorno.

Esta labor se suele realizar en el `.bash_profile` de cara a que estas variables estén disponibles para cualquier programa que se ejecute en el sistema, ya sea desde un terminal de texto o utilizando su icono en el entorno gráfico.

- Establecer los alias. Un alias se utiliza para darle un nuevo nombre a una orden utilizada con unos determinados argumentos y opciones. Por ejemplo, si se establece el alias `llcol` de esta manera:

  ```
  alias llcol='ls -l --color=auto'
  ```

  cada vez que el usuario escriba `llcol`, se ejecutará `ls -l --color=auto`.

  Esta labor se suele realizar en el `.bashrc` para que afecte única y directamente al nuevo intérprete de órdenes que se va a abrir.

En el directorio `/etc/skel` hay unas versiones por defecto de estos ficheros. Cuando se utiliza la orden `useradd` para crear un usuario, se copian automáticamente todos los ficheros de `/etc/skel` al directorio `$HOME` del nuevo usuario. Obviamente, estos ficheros se pueden personalizar, según las necesidades del sistema o de cada usuario en particular, simplemente modificando su contenido como el de cualquier guión shell.

Frecuentemente, desde el guión `.bash_profile` se incluye una llamada para ejecutar también `.bashrc` de cara a tener, por ejemplo, todas las definiciones de alias disponibles desde el principio en todo el sistema. Tras iniciarse el sistema, el usuario podrá realizar modificaciones en su `.bashrc` particular que afectarán a los intérpretes de órdenes que abra desde ese momento.

7.3. Cuentas del sistema

Tal como dijimos al principio del boletín, un usuario es una entidad que ejecuta programas o posee ficheros. Normalmente es una persona que trabaja en el sistema (entra al sistema, edita ficheros, ejecuta programas, etc.), pero también puede ser una cuenta del sistema. Una cuenta del sistema ejecuta programas o posee ficheros para que determinados servicios funcionen, pero no está asociada a una persona. Por ejemplo, la cuenta del sistema `apache` ejecuta los servicios necesarios de un servidor Web (en concreto, el demonio `httpd`).

8. Propietarios y permisos

Tal como vimos en el primer boletín, el permiso para acceder a los ficheros está organizado en dos grados de propiedad: usuario y grupo.

- Usuario propietario: Con la orden `chown` se cambia el usuario propietario. Esta orden solamente puede ejecutarla el superusuario. La sintaxis de esta orden es:

  ```
  chown [OPCIÓN]... [PROPIETARIO][:[GRUPO]] FICHERO...
  ```

  Una de las opciones más interesantes es `-R`, que ocasiona que la orden opere de forma recursiva sobre ficheros y directorios. Ejemplos:

  - El siguiente ejemplo establece al usuario `pepito` como propietario del fichero `datos.txt`.

    ```
    # chown pepito datos.txt
    ```

  - El siguiente ejemplo establece al usuario `pepito` como propietario del directorio `apuntes` y de todos sus ficheros y subdirectorios, recursivamente.

```
# chown -R pepito apuntes
```

- El siguiente ejemplo establece `pepito` y `alumnos` como propietario y grupo, respectivamente, del fichero `datos.txt`.

  ```
  # chown pepito:alumnos datos.txt
  ```

- Grupo propietario: Con la orden `chgrp` se cambia el grupo propietario. El propietario del fichero puede ejecutar esta orden pero tiene que pertenecer al nuevo grupo. Por supuesto, el superusuario también puede ejecutarla. La sintaxis más habitual de esta orden es la siguente:

  ```
  chgrp [OPCIÓN]... GRUPO FICHERO...
  ```

  De nuevo, a través de la opción `-R` se establece que la orden actúe de forma recursiva, cambiando el grupo de ficheros y directorios. Ejemplos:

  - El siguiente ejemplo establece `alumnos` como grupo propietario del fichero `datos.txt`:

    ```
    # chgrp alumnos datos.txt
    ```

  - El siguiente ejemplo establece `alumnos` como grupo propietario del directorio `apuntes` y de todos sus ficheros y subdirectorios, recursivamente:

    ```
    # chgrp -R alumnos apuntes
    ```

  Sabemos también que los permisos de un fichero están organizados en 3 bloques:

  - Para el usuario propietario (u).
  - Para el grupo propietario (g).
  - Para el resto de usuarios, que no son ni el usuario ni miembros del grupo (o).

  El significado de los distintos permisos referidos a un fichero es el siguiente:

  - r: Permiso para ver el contenido del fichero.
  - w: Permiso para modificar el contenido del fichero.
  - x: Permiso para ejecutar este fichero.

  Mientras que para un directorio:

  - r: Permiso para listar el contenido de este directorio.
  - w: Permiso para crear/eliminar ficheros en el interior de este directorio.
  - x: Permiso para entrar en este directorio.

  Los permisos de un fichero los puede cambiar el propietario y el superusuario mediante la orden `chmod`, tal y como se explicó en el primer boletín de prácticas. Además de la representación de los permisos como tres dígitos octales que vimos entonces, nos podemos referir a estos también indicando el bloque o bloques sobre los que queremos operar (u, g, o), la operación que queremos llevar a cabo (activación, +, o desactivación, –, de permisos) y el permiso o los permisos sobre los que actuar (r, w, x) De nuevo, con la opción `-R` podemos hacer que la orden actúe recursivamente. Veamos un par de ejemplos:

  - El siguiente ejemplo desactiva el permiso de lectura del fichero `datos.txt` para los usuarios del grupo propietario y para el resto de usuarios:

    ```
    # chmod go-r datos.txt
    ```

  - El siguiente ejemplo activa el permiso de escritura para los usuarios del grupo propietario del directorio `apuntes`, así como de todos los ficheros y subdirectorios que éste contega, recursivamente:

    ```
    # chmod -R g+w apuntes
    ```

8.1. Permisos especiales

Además de estos permisos básicos, que ya se habían analizado anteriormente, existen una serie de permisos «especiales» que nos permitirán dar respuesta a ciertas situaciones concretas:

- t (sticky bit): `chmod +t directorio`

  Si establecemos este permiso para un directorio concreto, si un usuario tiene permiso de escritura en el directorio, puede crear ficheros en su interior, pero sólo puede borrar los que le pertenecen. Por ejemplo, el directorio `/tmp` tiene los permisos `drwxrwxrw`**`t`**. Sólo tiene sentido si tiene permisos de escritura para todos (`drwxrwxrwx`). Si un directorio no tiene activado el bit t, si un usuario tiene permiso de escritura sobre un directorio, puede borrar cualquier fichero, aunque no le pertenezca.

- suid: `chmod u+s fichero`

  Si activamos este bit en un fichero ejecutable se produce un cambio de dominio a nivel de usuario, de manera que durante la ejecución el usuario efectivo del proceso es el propietario del fichero y no el usuario que lo ejecutó. Por ejemplo, supongamos que tenemos el ejecutable `gestorbd` que lee cuando se ejecuta el fichero de texto `basedatos`:

  ```
  -rwxr-xr-x  root root  /opt/bin/gestorbd
  -rw-------  root root  /opt/datos/basedatos
  ```

  Según esos permisos el usuario `pepito` puede ejecutar `gestorbd` pero no puede leer `basedatos` de ninguna manera. El usuario `pepito` sí podrá leer `basedatos` mediante la ejecución de este ejecutable si `gestorbd` tuviese activo el bit suid:

  `-rw`**`s`**`r-xr-x root root /opt/bin/gestorbd`

- sgid: `chmod g+s fichero`

  Si activamos este bit en un fichero ejecutable, se produce un cambio de dominio a nivel de grupo, de manera que durante la ejecución el grupo efectivo del proceso es el grupo propietario del fichero y no el grupo del usuario que crea el fichero. Por ejemplo, si el fichero ejecutable `gestorbd` lee el fichero `basedatos` y tenemos estos permisos:

  ```
  -rwxr-xr-x  root  root  /opt/bin/gestorbd
  -rw-rw----  root  root  /opt/datos/basedatos
  ```

  Los miembros del grupo `alumno` pueden ejecutar `gestorbd` pero no podrán leer `basedatos` de ninguna manera. En cambio sí podrán leer `basedatos`, usando `gestorbd`, si los permisos fuesen:

  `-rwxr-`**`s`**`r-x root root /opt/bin/gestorbd`

- sgid: `chmod g+s directorio`

  Si activamos este bit para un directorio, al crear un fichero en su interior, el grupo propietario del nuevo fichero es el grupo del directorio y no el grupo activo del usuario que ejecuta la orden.

  Supongamos que el usuario `pepito` sólo pertenece al grupo `alumno` (y no al grupo `gestor`). Si se tiene el directorio:

  `drwxrw`**`s`**`rwx juanito gestor /buzon`

  y `pepito` ejecuta:

  ```
  $ cp propuesta.pdf /buzon
  ```

  entonces el fichero copiado pertenecerá al grupo `gestor` (y no a `alumno`):

  ```
  rw-rw-r--  pepito gestor  /buzon/propuesta.pdf
  ```

9. Ejercicios

1. Crea el usuario `iso1` con la orden `useradd`.

   (Nota: no asignes contraseña al usuario al crear su cuenta. De esta manera, se deja la cuenta bloqueada, por el momento, hasta que se le asigne una contraseña posteriormente.)

   Una vez creado el usuario, resuelve las siguientes cuestiones:

   1.1 ¿Se ha creado el directorio `$HOME`?

   1.2 ¿Qué grupo primario se le ha asignado a este usuario?

   1.3 ¿Se han copiado los ficheros de inicialización al directorio de trabajo del nuevo usuario?

   1.4 ¿Qué se ha añadido en los ficheros `/etc/passwd` y `/etc/shadow`?

2. Usando la orden `passwd`, asígnale una contraseña al usuario `iso1`. A continuación, comprueba si ha cambiado la información referente a este usuario en los ficheros `/etc/passwd` y `/etc/shadow`.

3. Edita los ficheros `/etc/motd` y `/etc/issue` y cambia los mensajes que tienen. En el primero de ellos pon el mensaje "`Mensaje del día: Hoy, a las 23:55, se reiniciará el sistema`". En el segundo el mensaje debe ser "`Facultad de Informática. Centro de Cálculo`". (Recuerda que estos mensajes sólo se visualizan en los terminales en modo texto. Por ejemplo, puedes acceder uno de estos terminales mediante la combinación de teclas CTRL+ALT+F3. Para volver al terminal gráfico en el que te encontrabas tienes que usar CTRL+ALT+F2).

4. En el directorio `/etc/skel`, están los ficheros de configuración iniciales que se copian a los directorios `$HOME` de los usuarios cuando se crean sus cuentas. Realiza las modificaciones que sean oportunas para que:

   - Al crear un nuevo usuario, se le copie a su `$HOME` un fichero de texto llamado `horario.txt` que contenga lo siguiente "`Las salas de prácticas están abiertas todos los días`". (Este fichero se copiará al `$HOME` del usuario, pero no se mostrará ni nada por el estilo).
   - Cada vez que el usuario entre al sistema, se ha de ejecutar automáticamente la orden `who` para saber quién hay conectado.

5. Crea ahora el usuario `iso2` con la orden `useradd` y comprueba si se le han copiado los ficheros creados en el ejercicio 4.

   Nota: no asignes contraseña al usuario al crear su cuenta. De esta manera, se deja la cuenta bloqueada, por el momento, hasta que se le asigne una contraseña posteriormente.

6. Usando la orden `passwd`, asígnale una contraseña al usuario `iso2`.

7. Entra al sistema con `iso2` y comprueba que se le ha copiado el fichero `horario.txt` y que al entrar al sistema se le ejecuta la orden `who`.

8. Para el usuario `iso2`, establece los siguientes parámetros de tiempo:

   - El número mínimo de días entre cambios de contraseña es 2.
   - El usuario puede mantener la misma contraseña durante, como mucho, 60 días.
   - Una semana antes de que su contraseña expire, el sistema debe empezar a informarle.
   - Si 15 días después de haber expirado la contraseña aún no ha sido cambiada, la cuenta se debe bloquear.
   - La cuenta no debe ser accesible a partir del 24 de diciembre del presente año.

Comprueba que se han configurado correctamente estos parámetros de tiempo para esta cuenta.

9. Como usuario `iso2`, intenta cambiar la contraseña asignada. Cumpliendo las restricciones de tiempo, el sistema no te lo debe permitir.

10. Como `iso2`, selecciona como nuevo shell alguno de los que tiene permitidos (ver fichero `/etc/shells`). A continuación, entra al sistema con este usuario y comprueba si te ha asignado el nuevo shell.

11. Como `root`, cambia el shell asignado a `iso1`, seleccionando `/bin/false`. No uses la orden `chsh` para resolver este ejercicio.

12. Con el usuario `iso1`, intenta entrar al sistema. ¿Puedes? ¿Por qué?

13. Como superusuario cambia la información que se tiene guardada sobre el usuario `iso2` y que se puede ver al ejecutar `finger iso2`. La información que debes cambiar es la referente a su nombre completo, indicando que es "Pepe Pérez", la referente a su número de despacho, indicando que es el "3.41" y la referente a su número de teléfono del despacho, indicando que es el "4821". A continuación:

    - Comprueba en qué campo del fichero `/etc/passwd` se almacenan los datos introducidos y qué formato se sigue para guardarlos.
    - Comprueba que se visualizan correctamente utilizando la orden `finger`.

14. Usando la herramienta `useradd`, crea un nuevo usuario, llamado `apagar`, que apague el sistema haciendo uso de la orden `/usr/bin/systemctl poweroff`. El usuario no debe iniciar ninguna sesión interactiva; simplemente, cuando este usuario introduzca su nombre y contraseña para entrar al sistema, la máquina se apagará.

    Asígnale una contraseña y comprueba si apaga la máquina.

    Si no funciona como se esperaba, es muy probable que te estés equivocando al asignarle el UID a ese usuario. Piensa qué usuario es el único que puede ejecutar esta orden. Esto te indicará el UID que tienes que asignarle al nuevo usuario.

    Es muy posible que este usuario sólo funcione en modo texto. Pruébalo, por tanto, en una consola de texto.

15. Con `groupadd` crea un nuevo grupo, `admin`. A continuación, haz que el usuario `iso2` pertenezca al mismo, como uno de sus grupos secundarios (usa la orden `usermod`). Comprueba el resultado usando la orden `id`.

16. Con `groupadd` crea un nuevo grupo, `ssoo`. A continuación, haz que el usuario `iso2` pertenezca al mismo, como uno de sus grupos secundarios (usa la orden `gpasswd`). Comprueba el resultado usando la orden `id`.

17. Crea un usuario `iso3` con `useradd` asignándole como grupo primario el grupo `admin` y haciendo que, además, pertenezca a los grupos `ssoo` y `users`. No se debe crear ningún grupo nuevo (ten en cuenta que, por defecto, `useradd` crea un nuevo grupo con el mismo nombre que el usuario creado). Asígnale contraseña.

18. Entra al sistema como el usuario `iso3` y realiza las siguientes labores:

    - Comprueba, con `id`, cuál es el grupo activo del usuario.
    - Crea el fichero `prueba01`, por ejemplo ejecutando "`touch prueba01`", y comprueba cuál es su grupo propietario.
    - Con la orden `newgrp`, haz que el nuevo grupo activo sea `users`. Comprueba con `id` que ha cambiado el grupo activo. Crea el fichero `prueba02` y observa cuál es el grupo asignado al mismo.

- Comprueba que `newgrp` realmente lo que hace es lanzar un nuevo intérprete de órdenes. Para ello, ejecuta `exit` para finalizar ese intérprete y ahora ejecuta `id` de nuevo para comprobar que has vuelto al intérprete original, donde el grupo activo vuelve a ser su grupo primario.

19. Borra los usuarios `iso1` y `iso2` con la orden `userdel`. A continuación, elimina aquellos directorios `$HOME` que no se hayan borrado.

20. Usando la orden `userdel`, elimina el usuario `iso3`, consiguiendo que se borre directamente también su directorio `$HOME`.

21. Intenta ahora borrar el usuario `apagar` con `userdel`. ¿Qué ocurre? ¿Por qué? Fuerza el borrado del usuario, incluyendo el borrado de sus ficheros.

22. Borra los grupos `apagar`, `admin` y `ssoo`; usando `groupdel`.

23. Tenemos dos usuarios contratados a media jornada. Uno, llamado `matutino`, viene sólo por las mañanas, y el otro, `vespertino`, que viene sólo por las tardes, continúa el trabajo de matutino. El jefe quiere que, aunque sean personas diferentes y cada uno use su propio login, en realidad sean el mismo usuario efectivo y tenga los mismos permisos, de forma que de cara al sistema sean en realidad uno sólo. ¿Cómo podemos realizar esto?

24. Si no deseo tener dos usuarios con el mismo UID, ¿puedo hacer que compartan el mismo directorio `$HOME`? ¿Es posible conseguir que compartan los ficheros que hay dentro de ese directorio?

25. Como superusuario, crea tu propio directorio temporal (con el mismo comportamiento que `/tmp`) en `/pruebas`: es decir, todos los usuarios pueden escribir en él y borrar sus ficheros, pero no pueden borrar los ficheros de otros usuarios. Muestra a través de un ejemplo que el funcionamiento es el esperado.

26. Al ejecutar la siguiente secuencia de instrucciones, ¿funciona la última instrucción? ¿Cómo se podría conseguir que funcionase utilizando el bit `setuid`?

    Las primeras instrucciones las tienes que ejecutar como `root`, las que están indicadas inicialmente con el carácter `#`, mientras que la última es con el usuario `alumno`.

    ```
    # cp /bin/cat /home/alumno/micat
    # echo 'na na na' > /tmp/fichero
    # chmod go-rwx /tmp/fichero
    # su -l alumno
    $ /home/alumno/micat /tmp/fichero
    ```

27. Ejecuta como usuario `root` las siguientes instrucciones,

    ```
    # mkdir /practicas
    # chgrp mail /practicas
    # cd /practicas
    # touch p1.txt
    ```

    a) ¿Con qué usuario y grupo se crea el fichero `p1.txt`?

    b) Cambia el grupo del fichero `p1.txt` a `mail` para que coincida con el del directorio donde se encuentra.

28. ¿Qué habría que hacer para que cuando se crea un fichero dentro del directorio `/practicas` dicho fichero perteneciese al grupo `mail` directamente, sin tener que ejecutar la orden `chgrp` después de crear el fichero? Crea el fichero `/practicas/p2.txt` para comprobar que ahora tiene el grupo `mail` directamente.

29. Mediante una única orden, establece, para el directorio `/practicas` y para todos los ficheros que hay en su interior, que el usuario propietario sea `alumno` y el grupo propietario sea `users`.

Apéndice

A. Otros mecanismos de protección

Además del mecanismo de seguridad tradicional basado en permisos, usuarios, grupos y programas con bits SETUID y SETGID, Linux dispone también de otros mecanismos de seguridad más sofisticados y flexibles. A continuación describimos dos de los más utilizados, ACLs y SELinux, siendo SELinux el que viene activado por defecto en sistemas Fedora. La descripción es breve pues sólo pretende dar a conocer los mecanismos para que el alumno que quiera pueda indagar más sobre ellos.

A.1. Listas de control de acceso (ACLs)

Linux implementa ACLs como las vistas en teoría, es decir, permite asociar a un fichero una lista de usuarios y grupos junto con los permisos de acceso de cada uno. Para poder usar ACLs, es necesario que el sistema de ficheros tenga soporte y que se haya montado con la opción `acl`. Sistemas de ficheros como Ext2/3/4, ReiserFS y Btrfs implementan ACLs.

La orden `ls -l` indica qué ficheros tienen ACLs asociadas mostrando un signo «+» a la derecha de los permisos. Por ejemplo:

```
[root@localhost media]# ls -l fichero
-rw-r-xr--+ 1 root root 0 jul 25 00:03 fichero
```

Podemos obtener la ACL asociada a un fichero con la orden `getfacl`:

```
[root@localhost media]# getfacl fichero
# file: fichero
# owner: root
# group: root
user::rw-
user:internet:r-x
group::r--
mask::r-x
other::r--
```

y modificarla con la orden `setfacl`:

```
[root@localhost media]# setfacl -m "u:internet:r-x" fichero
[root@localhost media]# setfacl -m "g:bin:rw-" fichero
[root@localhost media]# getfacl fichero
# file: fichero
# owner: root
# group: root
user::rw-
user:internet:r-x
group::r--
group:bin:rw-
mask::rwx
other::r--
```

También podemos usar la orden `setfacl` para borrar una entrada particular de un usuario o grupo, o borrar toda la lista, como en los dos siguientes ejemplos:

```
[root@localhost media]# setfacl -x "u:internet" fichero
[root@localhost media]# getfacl fichero
# file: fichero
# owner: root
```

```
# group: root

user::rw-
group::r--
group:bin:rw-
mask::rw-
other::r--

[root@localhost media]# setfacl -b fichero
[root@localhost media]# getfacl fichero
# file: fichero
# owner: root
# group: root
user::rw-
group::r--
other::r--
```

A.2. SELinux

Security-Enhanced Linux (SELinux) es un módulo de seguridad del núcleo de Linux que proporciona soporte para políticas de seguridad basadas en *control de acceso obligatorio* o MAC (*Mandatory Accesss Control*). Con SELinux, los programas de usuario y los servidores (demonios) del sistema se confinan para que sólo puedan acceder a ciertos ficheros y a ciertos recursos de red. Al limitar los privilegios al mínimo necesario para que algo funcione, se reduce o elimina la capacidad de estos programas y demonios para producir cualquier tipo de daño en el sistema si su seguridad se ve afectada (por ejemplo, por un fallo de programación o una mala configuración).

SELinux es independiente del mecanismo de control de acceso tradicional de Linux y no posee el concepto de superusuario. Esto último hace que incluso programas ejecutados por el superusuario tengan ciertas restricciones, a diferencia de lo que ocurre en un sistema sin SELinux, donde un proceso de superusuario no tiene ninguna restricción y, por tanto, puede comprometer todo el sistema si presenta fallos de seguridad. Al ser mecanismos independientes, lo que un proceso pueda hacer con cierto recurso dependerá tanto de SELinux como de los permisos que haya establecidos en el recurso y el usuario/grupo al que pertenezca el proceso.

SELinux asigna a cada proceso un *contexto* formado por un *papel* (o role), un *nombre de usuario* y un *dominio* (o tipo). El nombre de usuario no tiene relación con ningún nombre de usuario definido en el sistema, aunque a veces coinciden. Podemos usar la opción `-Z` de `ps` para ver el contexto que tiene cada proceso.

Los ficheros, puertos de red y ciertos elementos hardware también tienen un contexto formado por un nombre, un papel (o role, que rara vez se usa) y un tipo. También aquí podemos usar la opción `-Z` de `ls` para ver estos contextos. La orden `ls -l` nos indicará que un fichero tiene un contexto asociado mostrándonos un punto a la derecha de los permisos.

Teniendo en cuenta todo lo anterior, lo que un proceso pueda hacer con un recurso dependerá del contexto del proceso, del contexto del recurso y de lo que las reglas de política de SELinux establezcan.