

# Tema 3: Gestión de Usuarios

Programación y Administración de Sistemas (2023-2024)

Javier Sánchez Monedero

14 de mayo de 2024

## Tabla de contenidos

1	Objetivos y evaluación	1
2	Introducción	3
3	Usuarios	4
4	Grupos	18
5	Usuarios y grupos estándar	19
6	Referencias	21

## 1 Objetivos y evaluación

### Objetivos de aprendizaje

- Definir qué son **usuarios del sistema**, las características de los mismos y sus ficheros de configuración.
- Enumerar y explicar los campos del fichero `/etc/passwd` y `/etc/shadow`.
- Explicar las características que deberían tener las **contraseñas** para los usuarios.
- Explicar el mecanismo de *shadow passwords* y el mecanismo de **cifrado de contraseñas** que evita guardar las contraseñas del sistema en texto plano.
- Enumerar los **mecanismos de revocación de contraseñas**, las **restricciones** de tiempo en cuanto a la validez de las contraseñas y las **herramientas** de administración que permiten configurarlas.

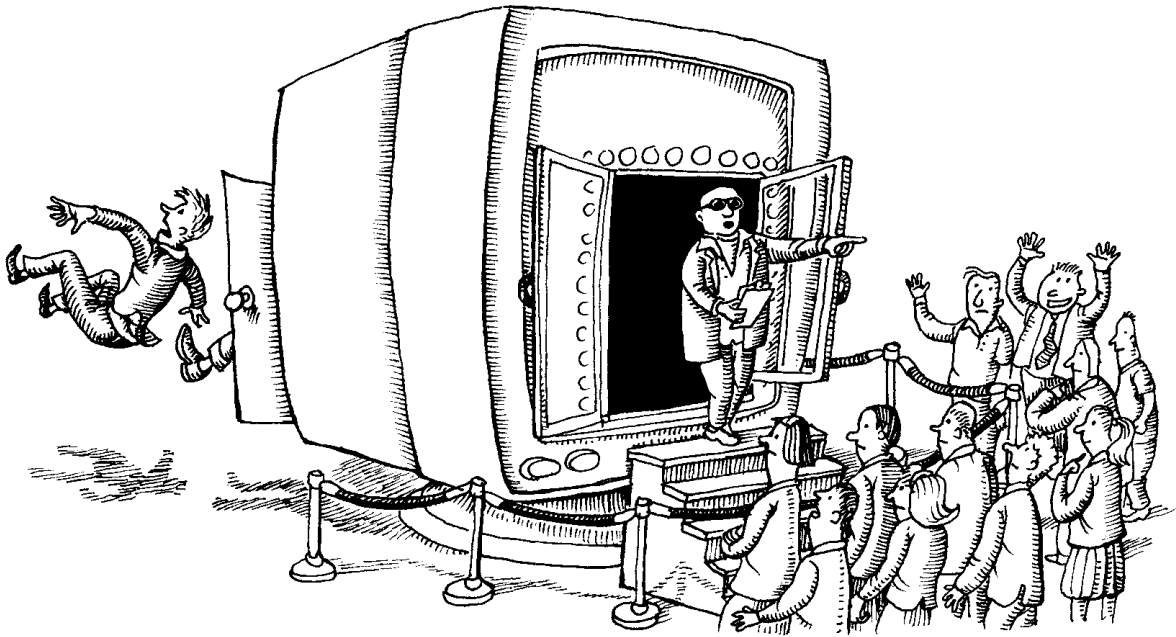
- Cambiar el **intérprete** de órdenes por defecto de los usuarios.
- Configurar **cuentas restrictivas** para usuarios especiales.
- Enumerar los pasos para **añadir un usuario** al sistema.
- Utilizar herramientas administrativas para **añadir o modificar cuentas** de usuario.
- Establecer el objetivo de los **grupos de usuarios**, identificar grupo primario y grupo activo de un usuario, enumerar y explicar los campos del fichero **/etc/group**.
- Configurar grupos con contraseñas.
- Utilizar las distintas herramientas administrativas para grupos.
- Identificar **usuarios y grupos estándar** en un sistema GNU/Linux.

## Evaluación

- Cuestionarios objetivos.
- Pruebas de respuesta libre.
- Tareas de administración.

## 2 Introducción

### Introducción



### Definición de **usuario**

- Persona que trabaja en el sistema, editando ficheros, ejecutando programas...
- **Pseudo-usuario:** entidad, que sin ser una persona, puede ejecutar programas o poseer ficheros (se les reserva identificadores de 0 a 499) y se utiliza típicamente para servicios y tareas automatizadas.

### Información mínima **usuario**

Características básicas de un usuario:

- Nombre de usuario (`logname` o `username`).
- Identificador de usuario (UID): el sistema trabaja, internamente, con el UID y no con el nombre de usuario.
- Identificadores de los grupos a los que pertenece (GIDs).

## Ficheros básicos

Ficheros de configuración:

- `/etc/passwd`: información de las cuentas de usuarios.
- `/etc/shadow`: *passwords* cifradas (*hash* de las contraseñas) e información de “envejecimiento” de las cuentas.
- `/etc/group`: definición de los grupos y usuarios miembros.
- `/etc/gshadow`: *passwords* de grupos cifradas.

Ejercicio: lista los permisos de todos estos ficheros y trata de entender la política de seguridad.

```
ls -l /etc/passwd /etc/shadow /etc/group /etc/gshadow
```

## 3 Usuarios

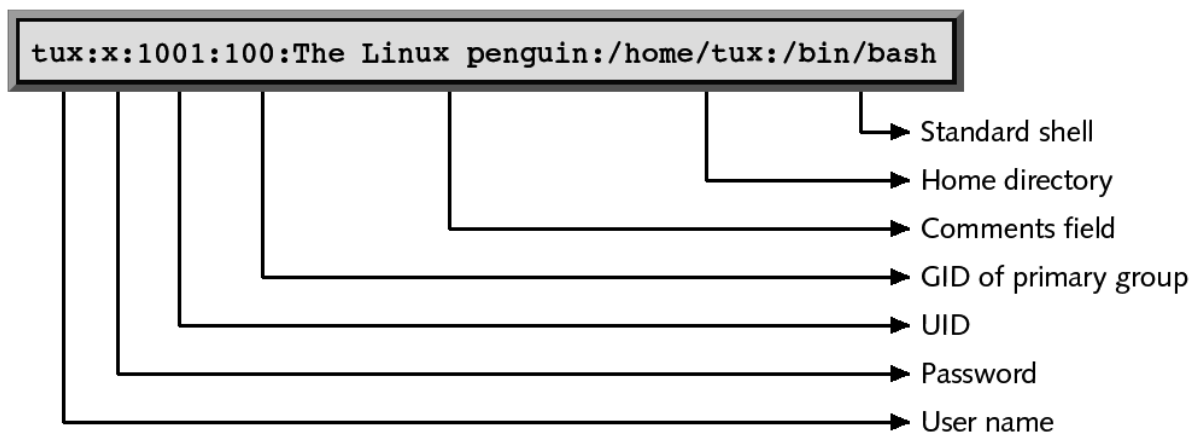
### Fichero `/etc/passwd`

- Contiene la lista de usuarios del sistema y sus contraseñas.
- Formato: `nombre:password:uid:gid:gecos:home:shell`.
  - `nombre` → Nombre del usuario, logname o username.
  - `password` → contraseña cifrada o:
    - \* “\*” o “!!” → la cuenta está desactivada o bloqueada.
    - \* “x” → las shadow están activas, la contraseña cifrada se guarda en `/etc/shadow`.
  - `uid` → identificador del usuario.
  - `gid` → identificador del grupo primario al que pertenece.
  - `gecos` → campo de información referente al usuario (nombre, teléfono, ...).
  - `home` → Path del directorio `$HOME` del usuario.
  - `shell` → Intérprete de órdenes.

## Fichero `/etc/passwd`

- El propietario del fichero es `root` y el grupo `root`.
- Los permisos del fichero son `rw-r--r--`.
- El programa `/usr/sbin/vipw` permite editar el fichero manualmente.
- El programa `pwck` verifica la integridad de `/etc/passwd` y `/etc/shadow`.
- Se permite el acceso al fichero `/etc/passwd` en modo lectura para poder leer información del usuario, pero no se debería permitir acceso a las *passwords* (aunque estén cifradas).

## Fichero `/etc/passwd`



## Contraseñas

- `passwd <nombre_usuario>`  $\Rightarrow$  asignar contraseña a un usuario (o cambiarla).
- Elección de una contraseña adecuada
- No utilizar:
  - [Las contraseñas más usadas](#)
  - Tu nombre, parte de él, o el de alguien cercano a ti.
  - Números significativos para ti o alguien cercano.
  - Nombre, nº, lugar o persona, relacionados con tu trabajo.
  - Nombres de gente famosa, lugares, películas, publicidad...

- Palabras que estén en el diccionario (ver `cracklib`).

## Contraseñas

- **Consejos** sin llegar a políticas absurdas:
  - Introducir 2 o más caracteres extras, símbolos especiales...
  - Escribir mal las palabras.
  - Utilizar mayúsculas y minúsculas, pero no de forma evidente.
  - Concatenar, embeber o mezclar 2 o más palabras.
  - Usar caracteres poco comunes: \$, &, #...

## Contraseñas

- La contraseña se debe cambiar cuando:
  - Se sospecha que alguien la ha podido conocer o averiguar.
  - Se sospecha que alguien ha conseguido el fichero con las contraseñas (`/etc/passwd` o `/etc/shadow`).
  - Un usuario se marcha del trabajo  $\Rightarrow$  cambiar todas las que conozca.
  - Un administrador del sistema se va  $\Rightarrow$  cambiar TODAS.
  - Un intruso ha conseguido entrar en el sistema.
- Periódicamente, se debe forzar a que los usuarios cambien sus contraseñas, incluido el administrador.
  - Por otro lado, si se obliga a los usuarios a cambiar su contraseña con demasiada frecuencia, lo normal es que elijan malas contraseñas, fáciles de adivinar...

## Herramientas de robustez de contraseñas

`pam_cracklib` y [pam\\_pwquality](#) son dos herramientas para forzar políticas de seguridad de contraseñas fuertes.

Puedes ver un mínimo de fortaleza de una contraseña escribiendo por teclado:

```
cracklib-check
```

ó así:

```
echo "1234" | cracklib-check  
echo "micontra" | cracklib-check  
echo "M1contra$" | cracklib-check
```

## Algunos recursos web

[dumbpasswordrules.com](http://dumbpasswordrules.com)

[haveibeenpwned.com](http://haveibeenpwned.com)



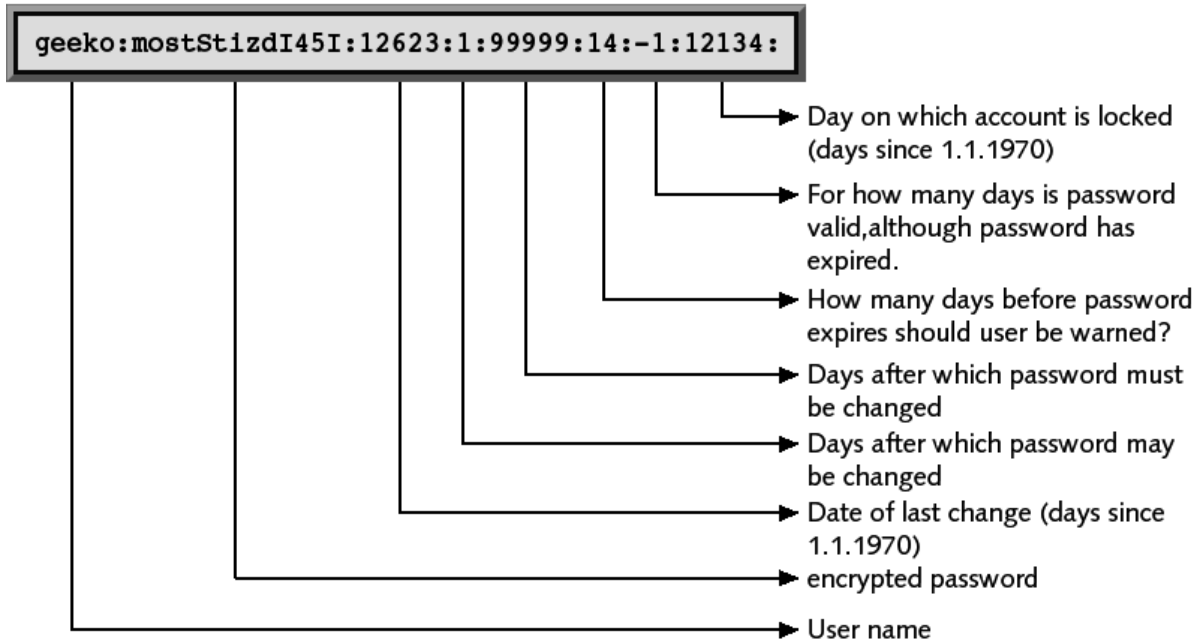
## Shadow passwords

- Permiten que las contraseñas cifradas no se guarden en el fichero `/etc/passwd` sino en `/etc/shadow` (más restringido).
- `/etc/shadow` tiene los permisos `rw-----`, y el usuario y grupo propietario es **root**.
  - En las últimas versiones, tiene permisos `rw-r----` y el grupo propietario es **shadow** (flexibilidad para comprobación de contraseña por usuarios que pertenezcan a dicho grupo).
- Este fichero guarda para cada usuario del sistema, la contraseña cifrada junto con su **información de envejecimiento**.
- Solo para aquellos usuarios que tengan una “x” en `/etc/passwd`.



- Por defecto, están activas y se actualizan automáticamente.

## Shadow passwords



## Shadow passwords

`nom:pass:changed:minlife:maxlife:warn:inactive:expired:unused`

- `nom` ⇒ nombre del usuario, *logname* o *username*.
- `pass` ⇒ contraseña cifrada.
  - `mkpasswd -method=sha-512` contraseña salt
- Comandos de **actualización**:
  - `pwconv` ⇒ crear y actualizar el fichero `/etc/shadow`.
  - `pwunconv` ⇒ desactivar los *shadow passwords*.

## Shadow passwords

- Para cifrar una contraseña, se utilizan **algoritmos criptográficos de generación de resumen** (función *hash*,  $H(\cdot)$ ).
  1. El mensaje en este caso es la **contraseña** ( $C$ ).
  2. **salt** ( $S$ ) es una palabra aleatoria que se concatena a los bytes de **contraseña**  $\rightarrow$  dificulta ataques con diccionarios y tablas de *hash* precomputadas; añade aleatoriedad al resumen.
  3. El sistema concatena  $C$  con  $S$ ,  $\{C, S\}$ , calcula el resumen  $F = H(\{C, S\})$  y almacena  $S$  y  $F$ .
  4. Cuando el usuario introduce una contraseña  $C'$ , se repite todo el proceso:  $F' = H(\{C', S\})$ .
  5. Si  $F = F'$ , entonces el usuario puede entrar al sistema.

## Shadow passwords

- Propiedades **deseables** de las funciones de resumen:
  - **Dado  $C$ , debe ser fácil calcular  $H(C)$**   $\rightarrow$  para que el coste computacional no sea alto.
  - **Dado  $H(C)$ , debe ser extremadamente difícil calcular  $C$**   $\rightarrow$  para que las contraseñas originales no se puedan conocer sabiendo el resumen (fugas de información).
  - **Dado  $C$ , debe ser muy difícil encontrar otro mensaje  $C'$  tal que  $H(C) = H(C')$**   $\rightarrow$  para que dos usuarios no terminen con la misma contraseña.
- Este tipo de funciones se denominan **funciones de dispersión de un solo sentido**.

## ¿Cómo se almacena el *hash* y *salt* en *shadows*?

¿Dónde se guarda la sal? El campo de la contraseña shadow tiene 3 subcampos: `$id$salt$hash` donde:

- `$1$` es MD5
- `$2a$` es Blowfish
- `$2y$` es Blowfish
- `$5$` es SHA-256
- `$6$` es SHA-512

Más información en:

- [Adding Salt to Hashing: A Better Way to Store Passwords \(con vídeo\)](#)
- [Password Storage Tier List: encryption, hashing, salting, bcrypt, and beyond](#)

## Shadow passwords: Algoritmos de hash

Dos algoritmos: MD5 y SHA

**MD5** (*Message-Digest algorithm 5*)

- Aplica funciones no lineales a los 17 segmentos de 32 bits de un bloque de 512 bits.
- Se obtiene un resumen de 128 bits.
- Obtener suma MD5 (GNU/Linux):

```
md5sum Fichero.ext > Fichero.md5
```

- Chequear suma MD5 (GNU/Linux) (se busca un fichero con el nombre correcto en la carpeta actual):

```
md5sum -c Fichero.md5
```

## Shadow passwords: Algoritmos de hash

**SHA** (*Secure Hash Algorithm*):

- Estándar del NIST.
- Parecido a MD5, pero genera resúmenes más grandes, que lo hacen más seguro contra ataques de fuerza bruta o del cumpleaños.
- Se pueden considerar 160, 224, 256, 384 o 512 bits para el resumen.
- Obtener suma SHA (GNU/Linux):

```
shasum [-anumBits] Fichero.ext > Fichero.sha
```

- Chequear suma SHA (GNU/Linux):

```
shasum -c Fichero.sha
```

- SHA-512 es el algoritmo utilizado por defecto en GNU/Linux para guardar la contraseña.

## Ejemplo SHA:

```
shasum /etc/passwd > resumen_pw.sha
cat resumen_pw.sha
shasum -c resumen_pw.sha
```

## Restricciones de tiempo (/etc/shadow)

- Introducir restricciones de tiempo o envejecimiento para la validez de la cuenta o de la contraseña.
  - **changed**  $\Rightarrow$  fecha del último cambio de contraseña.
  - **minlife**  $\Rightarrow$  n° de días que han de pasar para poder cambiar la contraseña.
  - **maxlife**  $\Rightarrow$  n° de días máximo que puede estar con la misma contraseña sin cambiarla.
  - **warn**  $\Rightarrow$  cuántos días antes de que la contraseña expire (**maxlife**) el usuario será informado sobre ello, indicándole que tiene que cambiarla.
  - **inactive**  $\Rightarrow$  n° de días después de que la contraseña expire en que la cuenta se deshabilitará si no ha sido cambiada.
  - **expired**  $\Rightarrow$  fecha en la que la cuenta expira y se deshabilita de forma automática.

## Restricciones de tiempo

- El fichero `/etc/login.defs` tiene los valores por defecto.
- Comando **chage** (administrador):
  - **chage -d ult\_día usuario**  $\Rightarrow$  último cambio de password.
  - **chage -m min\_días usuario**  $\Rightarrow$  n° de días que han de pasar para poder cambiar la contraseña.
  - **chage -M max\_días usuario**  $\Rightarrow$  n° de días máximo que puede estar con la misma contraseña sin cambiarla.
  - **chage -W warn\_días usuario**  $\Rightarrow$  establece un aviso de que la contraseña expira un número de días antes de que expire.
  - **chage -I inac\_días usuario**  $\Rightarrow$  n° de días después de que la contraseña expire que la cuenta se deshabilitará de forma automática si la contraseña no ha sido cambiada.

- `chage -E exp_días usuario`  $\Rightarrow$  fecha en la que la cuenta expira y se deshabilita de forma automática.

## Restricciones de tiempo

Supongamos que el usuario **pagutierrez** cambia su contraseña el 1 de marzo y **root** ejecuta estas órdenes:

```
chage -M 20 pagutierrez
chage -W 6 pagutierrez
chage -I 5 pagutierrez
chage -E 2023-10-30 pagutierrez
```

Los tiempos quedan fijados de la siguiente manera:

- El 14 de marzo **pagutierrez** recibirá el primer aviso para que cambie su contraseña.
- El 20 de marzo, debería haber cambiado su contraseña.
- Si no cambia la contraseña, como se ha fijado el tiempo de inactividad, la cuenta aún no se bloqueará.
- Si el 25 de marzo **pagutierrez** no ha cambiado su contraseña, la cuenta será bloqueada.
- La cuenta expira, pase lo que pase, el 30 de octubre.

## Ejercicio restricciones de tiempo

Importante: no hacer esto sobre el único usuario de entrada al sistema.

1. Consulta la ayuda del comando `lslogin` y lista las últimas sesiones de los usuarios.
2. Consulta los parámetros de caducidad de algún usuario con: `chage -l usuario`.
3. Cambia la caducidad de contraseña de alguna cuenta y comprueba el resultado intentando iniciar sesión con esa cuenta.

## Ficheros de inicialización

- Directorio `/etc/skel/` ⇒ ficheros que se copian automáticamente a cada `$HOME`.
- Los **ficheros de inicialización** son *scripts shell* que realizan tareas como dar valor a variables, nombrar alias, realizar funciones específicas...
- Los ficheros dependen del intérprete de órdenes seleccionado: *Bourne shell* (**sh**), *Bourne again shell* (**bash**), *C shell* (**csh**)...
- Incluyen el PATH, variables de entorno, `umask`, funciones de inicialización, alias, var. del propio shell...
- Lo normal es que lean parte de su contenido de algún fichero global (`/etc/profile`, `/etc/bash.bashrc`)

## Ficheros de inicialización y cierre

Se ejecuta al hacer un *login* en el sistema por SSH o por terminal real:

- `.bash_profile` en **bash** / `.profile` en **bash** y **sh** / `.login` en **csh**:

Cada vez que se ejecuta una shell en **bash** aunque no conlleve *login*:

- `.bashrc` / `.cshrc` en **csh**:

Al salir del sistema el usuario (al finalizar la sesión):

- `.bash_logout` en **bash** / `.logout` en **C csh**

## Configuración de login de usuarios

Las política generales de gestión de contraseñas, creación de usuarios, `umask` por defecto, etc. se definen en: `/etc/login.defs`.

Puedes ver un resumen de todo esto en [Gestión de política de contraseñas en Linux: login.defs y pam\\_pwquality&pam\\_cracklib](#)

## Ejercicio ficheros inicialización

1. Prueba a añadir algún alias útil a tu fichero `.bashrc`. Por ejemplo un alias para `rm` que pida siempre confirmación antes de borrar.
2. Prueba a cambiar tu *prompt* y a hacerlo permanente si te gusta:

```
export PS1='\e[33;1m\u@\h: \e[31m\W\e[0m$ '
```

3. Prueba a poner una máscara por defecto para todos los usuarios de nueva creación. Por ejemplo una restrictiva que no deje hacer nada al resto del grupo y usuarios.
4. Añade un saludo con una vaca (necesitarás `sudo apt install cowsay`).

## Selección de intérprete de órdenes

- En el último campo del fichero `/etc/passwd`, se establece el **intérprete de órdenes** que se ejecuta al entrar al sistema.
- En el fichero `/etc/shells` se indican los *shells* permitidos.
- Un usuario puede cambiar su *shell* con `chsh`:
  - ¡Ojo! Si se prohíbe un *shell*, no se podrá elegir con `chsh`, pero los usuarios que ya lo tenían asignado lo podrán seguir usando.
- 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 puede asignar `/bin/false` o `/sbin/nologin`.
- También se puede establecer como *shell* un **fichero ejecutable**:
  - Cuando el usuario entre al sistema se ejecuta, y, al finalizar la ejecución, el usuario sale del sistema (no llega a hacer login).

## Cuentas restrictivas

Las cuentas restrictivas permiten limitar las acciones de los usuarios en el sistema.

Se pueden crear de dos formas:

- Asignar como *shell* un fichero ejecutable que realice una tarea determinada, y al terminar se sale del sistema:
  - Usuario para hacer copias de seguridad: como *shell* tiene un *script* que hace esa tarea.
  - Usuario para apagar el sistema: ejecuta la orden **shutdown**.
- Los usuarios restrictivos de este tipo tienen que tener los permisos necesarios para poder hacer la tarea asignada. Estos permisos se asignan a nivel de identificador de usuario.
  - Para apagar el sistema, se necesitan permisos de administración.

## Cuentas

Usando el *shell* restrictivo `/bin/rbash`:

- **rbash** es un enlace simbólico a `/bin/bash` (**rbash** es equivalente a `/bin/bash -r`).
- Este intérprete se comporta como un intérprete normal, salvo que el usuario no puede hacer determinadas tareas, como:
  - Cambiar de directorio.
  - Establecer o modificar los valores de `$PATH` o `$HOME`.
  - Especificar nombres u órdenes que contengan `/`.
  - Usar redirección.
  - Utilizar la orden **exec** para reemplazar el *shell* por otro programa.
- A estos usuarios hay que limitarles los ficheros que pueden ejecutar, copiándolos a un directorio y que su `PATH` sea sólo ese directorio. En otro caso, con un `PATH` “normal”, es casi como si no tuviesen restricciones.



## Añadir nuevos usuarios al sistema (I)

Pasos a realizar (del 1 al 7, automatizados con herramientas):

1. Decidir el nombre de usuario, el UID, y los grupos a los que va a pertenecer (grupo **primario** y grupos **secundarios**).
2. Introducir los datos en los ficheros `/etc/passwd` y `/etc/group` (poniendo como contraseña “\*”).
3. Asignar un *password* a la nueva cuenta.
4. Si las *shadow* están activas, escribir la contraseña.
5. Establecer los parámetros de **envejecimiento** de la cuenta.
6. Crear el directorio `$HOME` del nuevo usuario, establecer el propietario y grupo correspondiente y los permisos adecuados.
7. Copiar ficheros necesarios por defecto (`.bash_profile`, `.bashrc`...) desde `/etc/skel/`.

## Añadir nuevos usuarios al sistema (II)

Pasos opcionales:

8. Establecer otras facilidades: *quotas*, *mail*, permisos, etc.
9. Ejecutar cualquier tarea de inicialización propia del sistema.
10. Probar la nueva cuenta.

## Herramientas para crear/modificar cuentas de usuario

Las herramientas de creación de cuentas de usuario suelen realizar todas las tareas básicas del proceso, a excepción de las específicas (*quotas*, impresión, etc.).

- **adduser** o **useradd**  $\Rightarrow$  crear cuentas de usuario, o modificar cuentas ya existentes. Toma los valores por defecto de `/etc/default/useradd` y de `/etc/login.defs`. **useradd** se salta algunos pasos.
- **usermod**  $\Rightarrow$  modificar cuentas.
- **deluser** o **userdel**  $\Rightarrow$  eliminar cuentas (por defecto no borra el directorio `$HOME`).
- **newusers**  $\Rightarrow$  crea cuentas de usuarios utilizando la información introducida en un fichero de texto (en *batch*), que ha de tener el formato del fichero `/etc/passwd` (no copia los ficheros de inicialización).

- `users-admin`  $\Rightarrow$  herramienta en modo gráfico.

## 4 Grupos

### Grupos

- **Grupos:** colecciones de usuarios que comparten recursos o ficheros del sistema. Características: nombre del grupo o `groupname` e identificador del grupo (GID)  $\Rightarrow$  internamente el sistema identifica al grupo por este número.
- Objetivo: Garantizar permisos concretos para un conjunto de usuarios, sin tener que aplicarlos a cada uno.
- El fichero de configuración es `/etc/group`, con el formato: `nombre:x:gid:lista de usuarios`
  - `nombre`  $\Rightarrow$  nombre del grupo.
  - `gid`  $\Rightarrow$  identificador del grupo.
  - `lista de usuarios` que pertenecen al grupo, sep. por “,”.

`pas:x:519:pagutierrez,jsanchezm`

### Grupos

- Tipos de grupos:
  - Primarios  $\Rightarrow$  grupo especificado en `/etc/passwd`.
  - Secundarios  $\Rightarrow$  otros grupos (indicados en `/etc/group`).
- Funcionamiento de los grupos:
  - Al crear un fichero se establece como grupo propietario el **grupo activo** del usuario en ese momento.
  - Grupo activo  $\Rightarrow$  grupo primario (salvo que usemos `newgrp`).
  - Al determinar los permisos sobre un fichero, se usan todos los grupos del usuario.

## Grupos: contraseñas (opcional)

Esta es una funcionalidad con poco uso en la actualidad.

Los grupos pueden tener contraseña  $\Rightarrow$  `/etc/gshadow`:

- Si un usuario sabe la contraseña de un grupo, puede usarlo sin pertenecer a él con la orden `newgrp`.
- Información en `/etc/gshadow`: grupo, contraseña, usuarios administradores (pueden cambiar la contraseña y los miembros) y miembros (idea parecida al `/etc/shadow`).

## Grupos: herramientas

- `addgroup grupo`  $\Rightarrow$  crear un nuevo grupo.
- `groupmod grupo`  $\Rightarrow$  modificar un grupo existente.
- `delgroup grupo`  $\Rightarrow$  eliminar un grupo.
- `groups [usuario]`  $\Rightarrow$  grupos a los que pertenece un usuario.
- `id [usuario]`  $\Rightarrow$  lista el identificador del usuario y los grupos a los que pertenece.
- `grpck`  $\Rightarrow$  chequea la consistencia del fichero de grupos.

## Grupos: herramientas para contraseñas (opcional)

- `newgrp grupo`  $\Rightarrow$  cambiar de grupo activo (lanza un shell)
- `gpasswd grupo`  $\Rightarrow$  asignar una contraseña a un grupo:
  - Si el usuario no pertenece al grupo, pero el grupo tiene contraseña, se le solicita y pasa a ser su grupo activo.
- `gpasswd -a user grupo`  $\Rightarrow$  añadir un usuario a un grupo.

# 5 Usuarios y grupos estándar

## Rangos del UID

- $UID \in [0, 99]$ : Usuarios que representan al propio SO.
- $UID \in [100, 499]$ : Usuarios especiales que representan servicios o programas.
- $UID \geq 1000$ : Usuarios normales.

## Algunos usuarios y grupos estándar

- **Usuarios estándar:**
  - `root`  $\Rightarrow$  Cuenta del administrador (0).
  - `bin` (utilidades comunes de usuarios, 2), `daemon` (ejecución de demonios, 1), `lp`, `sync`, `shutdown`, etc.  $\Rightarrow$  Tradicionalmente usados para poseer ficheros o ejecutar servicios
  - `mail`, `news`, `ftp`  $\Rightarrow$  Asociados con herramientas o facilidades.
  - `postgres`, `mysql`, `xfs`  $\Rightarrow$  Creados por herramientas instaladas en el sistema para administrar y ejecutar sus servicios.
  - `nobody` o `nfsnobody`  $\Rightarrow$  Usado por NFS y otras utilidades, usuario sin privilegios.

## Algunos usuarios y grupos estándar

- **Grupos estándar:**
  - `root`, `sys`.
  - `bin`, `daemon`, `adm`, `lp`, `disk`, `mail`, `ftp`, `nobody`, etc.
  - `kmem`  $\Rightarrow$  Grupo propietario de los programas para leer la memoria del kernel.
  - `user` o `users`  $\Rightarrow$  Grupo de los usuarios normales (no siempre se usa).

## Ejercicio usuario y grupo para Apache

1. Instala apache en Debian con: `apt install apache2`.
2. Examina qué usuario y grupo utiliza el proceso y carpetas y las características de estos.
3. Supón que un usuario no administrador del sistema operativo va a ser el administrador web de esta instancia de Apache. ¿Qué estrategia de permisos y grupos se te ocurren para que sea práctico y seguro? (puedes buscar en internet). Este [hilo en stackoverflow](#) sobre establecer permisos adecuados y seguridad en Apache es interesante.

## 6 Referencias

### Referencias

Evi Nemeth, Garth Snyder, Trent R. Hein, Ben Whaley and Dan Mackin. UNIX and Linux System Administration Handbook, 5th Edition. Capítulo 8: *User Management*. Addison-Wesley. 2018.

Aleen Frisch. Essential system administration. Capítulo 6. *Managing users and groups*. O'Reilly and Associates. Tercera edición. 2002.

[Privilege Escalation on Linux](#)

[Linux Privilege Escalation: Three Easy Ways to Get a Root Shell](#)