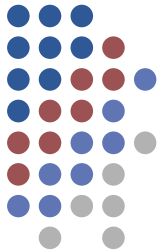
S.I.
Unidad 11

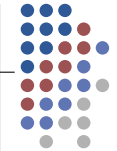
Administración de GNU/Linux

Permisos, usuarios y grupos



Unión Europea
Fondo Social Europeo
El FSE invierte en tu futuro

Permisos, usuarios y grupos

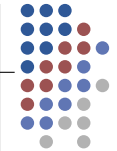


- Índice
  - Introducción
  - Identificación de usuarios conectados
  - Permisos
  - Administración de usuarios y grupos
  - Cambio de identificador
  - Ficheros de sesión

Permisos, usuarios y grupos

- Índice
  - Introducción
  - Identificación de usuarios conectados
  - Permisos
  - Administración de usuarios y grupos
  - Cambio de identificador
  - Ficheros de sesión

Introducción



- GNU/Linux = multiusuario
  - Varios usuarios trabajan en el sistema al mismo tiempo
    - A través de un solo equipo (consolas virtuales)
    - A través de varios equipos en red
- Cada usuario inicia la sesión con un nombre de usuario para que el sistema sepa quién acaba de entrar

Introducción

- **Grupo**
  - Conjunto de usuarios
  - Cada grupo tiene un nombre
  - Los usuarios pueden pertenecer a más de un grupo
  - Al crear un usuario, se genera un grupo con el mismo nombre que el usuario: será su **grupo principal**
- Cada archivo tiene un **propietario**
  - Por defecto, el usuario que creó el archivo
- Cada archivo pertenece también a un **grupo**
  - Por defecto, al grupo principal del usuario que creó el archivo

Introducción

- La identidad del usuario junto al grupo/s a los que pertenece determina los **derechos** de acceso a los ficheros
  - La forma en la que podrá hacer uso de ellos, si puede
- En este capítulo vamos a estudiar:
  - Los permisos de acceso a los ficheros del sistema
    - ¿Cómo podemos cambiar los permisos?
    - ¿Y el grupo y/o usuario al que pertenece un fichero?
  - La forma de gestionar los usuarios y los grupos
    - ¿Cómo añadimos, borramos o cambiamos usuarios y grupos?

Permisos, usuarios y grupos

- Índice
  - Introducción
  - Identificación de usuarios conectados
  - Permisos
  - Administración de usuarios y grupos
  - Cambio de identificador
  - Ficheros de sesión

Identificación de usuarios conectados

`tty`

- Muestra el controlador de terminal asignado y su ruta de acceso

```
$tty
/dev/tty4
```

- En este ejemplo, el usuario que ha ejecutado el comando *tty* se encontraba en la terminal número 4

Identificación de usuarios conectados

who

- Muestra **quién** está conectado al sistema

```
$who
ivens    tty1         2024-04-08 13:22
maria    tty3         2024-04-08 11:02
```

- En estos momentos, están conectados el usuario *ivens* en la terminal 1 y el usuario *maria* en la terminal número 3
- Además, podemos observar desde cuándo están conectados cada uno de ellos a dichas terminarles

Identificación de usuarios conectados

w

- `w` (en minúscula) es similar al comando *who*, pero además indica:
  - Qué está haciendo cada uno de los usuarios
  - Informa del tiempo que lleva…
    - En la sesión
    - Inactivo
    - Ejecutando procesos
    - Ejecutando ese comando

Identificación de usuarios conectados

w

```
$w
17:25:19 up  6:23,  4 users,  load average: 1,32, 1,14, 1,09
USER     TTY      FROM     LOGIN@   IDLE   JCPU   PCPU    WHAT
ivens    tty1     -        13:22    2:44m  0.32s  0.19s   -bash
maria    tty3     -        11:02    0.00s  0.20s  0.01s   who
```

- ***maria*** está en la terminal número 3 desde las 11:02 horas y no lleva inactivo ningún tiempo (***ivens*** lleva 2 minutos y 44 segundos inactivo)
- Está ejecutando el comando *who* (mientras que *ivens* no está ejecutando nada, ya que aparece que ejecuta el intérprete de comandos *bash*)
- Lleva 0.20 segundos en total ejecutando procesos y 0.01 segundos ejecutando el comando en curso (*who*)

Permisos, usuarios y grupos

- Índice
  - Introducción
  - Identificación de usuarios conectados
  - Permisos
  - Administración de usuarios y grupos
  - Cambio de identificador
  - Ficheros de sesión

Permisos

- Hemos visto que todo fichero tiene:
  - Un propietario (por defecto, el usuario que lo creó)
  - Un grupo al que pertenece (por defecto, el grupo principal del usuario creador)
- Partiendo de esto, el sistema asigna **permisos** de uso para:
  - El **usuario**
  - El **grupo**
  - **Otros**

Permisos

- **Usuario**
  - También conocido como **propietario**
  - Es el usuario que creó el fichero
- **Grupo**
  - **Grupo** al que pertenece el propietario
- **Otros**
  - Son los demás usuarios, el resto de usuarios que no son el propietario ni los que pertenecen al grupo

Permisos

- Además, para cada uno de los tres anteriores existen otros tres tipos de permisos básicos:
  - **Lectura (r)**
  - **Escritura (w)**
  - **Ejecución (x)**

Permisos

- **Lectura** (**r**)
  - Fichero
    - Se puede leer el contenido del mismo
      - `cat`, `more`, `less`, head,…
  - Directorio
    - Se puede listar el contenido de ese directorio
      - `ls`, `find`,…

Permisos

- Lectura (r)
- **Escritura (w)**
  - Fichero
    - Se puede cambiar el contenido
      - nano y guardar, redirigir hacia él,…
  - Directorio
    - Se pueden borrar y crear ficheros dentro de él
      - `cp`, `mv`, `rm`,…

Permisos

- Lectura (r)
- Escritura (w)
- **Ejecución (x)**
  - Fichero
    - Se puede ejecutar el fichero (debe ser de tipo ejecutable)
  - Directorio
    - Quien tiene el permiso, puede acceder a él (entrar)
      - `cd`

Permisos

- **¡Importante!**

Ficheros

Para que un fichero pueda ser **ejecutable (x)**

↓

Es indispensable que también tenga activado el permiso de **lectura (r)**

Directorios

Para que funcione el permiso de **lectura (r)** o **escritura (w)** sobre un directorio

↓

También debe estar activo el permiso de **ejecución (x)** sobre él

Permisos

- Identificación de permisos

```
$ls -l
drwxr-xr-x  6  ivens  ivens  4096  2024-04-01 20:22 Música
drwxr-xr-x  4  ivens  ivens  4096  2024-04-01 20:22 Programas
-rwxr-xr-x  1  ivens  ivens   528  2024-04-04 01:34 carta.txt
-rwxr-x---  4  ivens  dam    4096  2024-04-05 21:53 prueba
-rwxr-xr-x  1  root   root   9896  2024-04-04 01:34 retrato.jpg
```

- **Usuario propietario del fichero/directorio**
- **Grupo propietario del fichero/directorio**
- **Permisos del fichero/directorio**

Permisos

- Identificación de permisos

```
$ls -l
drwxr-xr-x  6  ivens  ivens  4096  2024-04-01 20:22 Música
drwxr-xr-x  4  ivens  ivens  4096  2024-04-01 20:22 Programas
-rwxr-xr-x  1  ivens  ivens   528  2024-04-04 01:34 carta.txt
-rwxr-x---  4  ivens  dam    4096  2024-04-05 21:53 prueba
-rwxr-xr-x  1  root   root   9896  2024-04-04 01:34 retrato.jpg
```

Tipo	Usuario			Grupo			Otros		
-	r	w	x	r	w	x	r	w	x

d = directorio
- = fichero
b = dispositivo de bloques
l = enlace simbólico
c = dispositivo de caracteres

Permisos

- Identificación de permisos

```
$ls -l pru*
-rwxr-x---  4  ivens  dam  4096  2024-04-05 21:53 prueba
```

- Usuario propietario = **ivens**
- Grupo propietario = **dam**
- Permisos

root puede leer, escribir y ejecutarlo **TODO**, independientemente de los permisos que tenga el archivo

-rwxr-x---

- Es un fichero (tiene un guión en el primer carácter)
- Usuario = **rwx** → Puede leer, escribir y ejecutar
- Grupo = **r-x** → Puede leer y ejecutar, pero no escribir
- Otros = **---** → No puede ni leer, ni escribir, ni ejecutar

Permisos

-rw-r--r--

- Fichero que puede ser leído por cualquiera, pero sólo modificado por su dueño

drwx------

- Directorio al que sólo puede acceder, listar y modificar su dueño

-r-xr-xr-x

- Fichero que puede ejecutar cualquier usuario

Permisos

-r-xr-x---

- Fichero que sólo el dueño y los de su grupo de usuarios pueden ejecutarlo

-r-x------

- Fichero que sólo el dueño del mismo puede ejecutarlo

---x------

- Fichero que nadie puede ejecutar, ya que, a pesar de estar marcado como ejecutable, el dueño no puede leerlo y, por tanto, no puede ejecutarlo

Permisos

- Equivalencias binario-decimal-permisos

PERMISOS	BINARIO	DECIMAL
---	000	0
--x	001	1
-w-	010	2
-wx	011	3
r--	100	4
r-x	101	5
rw-	110	6
rwx	111	7

Permisos

754

- Usuario = 7 → 111 → rwx
- Grupo = 5 → 101 → r-x
- Otros = 4 → 100 → r--

rwxr-xr--

660

- Usuario = 6 → 110 → rw-
- Grupo = 6 → 110 → rw-
- Otros = 0 → 000 → ---

rw-rw----

Permisos

- Manipulación de permisos

¿Quién puede cambiar los propietarios y permisos de un fichero o directorio?

- El único usuario que puede cambiar los **propietarios** de un fichero es el superusuario **root**
- Los únicos usuarios con derechos a cambiar los **permisos** de ficheros, sean cuales sean y sean quienes sea, son:
  - El superusuario **root**
  - El propietario del fichero
- Vamos a ver los comandos más usuales para el manejo de permisos y propiedad de ficheros

Permisos

- Manipulación de permisos
  - Existen unas opciones comunes a los tres siguientes comandos que vamos a ver:
    - **-R** Cambia de forma recursiva la propiedad/permisos de los directorios y sus contenidos
    - **-v** Describe en detalle los cambios de propiedad/permisos
    - **-c** Describe en detalle solo los archivos cuya propiedad/permisos cambia

Permisos

`chown [opciones] nuevo_usuario archivo`

- **Cambia el propietario** de un fichero
- **root** es el único que puede cambiar de propietario un fichero

Permisos

`chown [opciones] nuevo_usuario archivo`

```
#chown maria noticias.pdf
```

El fichero *noticias.pdf* pasaría a ser propiedad del usuario *maria*

Reiteramos: este comando sólo lo podría haber ejecutado ***root***

Permisos

`chgrp [opciones] nuevo_grupo archivo`

- Igual que el comando anterior, pero **cambia el grupo** al que pertenece el fichero
  - El usuario que lo invoca debe ser **root**

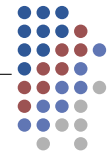
```
#chgrp contables balances.txt
```

El archivo *balances.txt* pasa a ser propiedad del grupo *contables*

Reiteramos: este comando sólo lo podría haber ejecutado ***root***

Permisos

`chmod [opciones] nuevos_permisos archivo`



- **Cambia los permisos** de un fichero
  - El usuario que lo invoca, es el **propietario** o es **root**
- Hay varias formas de cambiar los permisos
  - Utilizando la notación simbólica
  - Utilizando la notación decimal

Permisos

`chmod [opciones] nuevos_permisos archivo`

- Utilizando la notación simbólica
  - Usando los acrónimos de los permisos junto con:
    - "**u**" para usuario
    - "**g**" para grupos
    - "**o**" para otros
    - "**a**" para todos
  - Repitamos el ejemplo anterior:

```
$chmod u=rwx,g=rx,o=rx balances.txt
```


Permisos

`chmod [opciones] nuevos_permisos archivo`

- La forma "**relativa**" de utilizar *chmod* implica que no tenemos que especificar *todos* los permisos
  - Para ello utilizaremos:
    - Las letras "**u**", "**g**", "**o**", "**a**"
    - El símbolo más (**+**) o el menos (**-**) para indicar si activamos o eliminamos el permiso
    - La letra indicativa del permiso: **r**, **w**, **x**

```
$chmod o+w carta.txt
```

Habilita los permisos de escritura al resto de usuarios para el archivo *carta.txt*

Permisos

`chmod [opciones] nuevos_permisos archivo`

```
$chmod ug+w foto.jpg
```

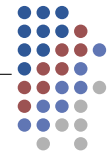
Habilitamos los permisos de escritura al usuario y al grupo para el archivo *foto.jpg*

```
$chmod a+rw libro.pdf
```

Habilita los permisos de lectura y escritura a todos para el archivo *libro.pdf*

Permisos

`chmod [opciones] nuevos_permisos archivo`



```
$chmod o-x fichero
```

Deshabilita los permisos de ejecución para el resto de usuarios de *fichero*

```
$chmod a-rwx paisaje.png
```

Elimina todos los permisos al fichero *paisaje.png*

Permisos

`chmod [opciones] nuevos_permisos archivo`

- Utilizando la notación decimal

```
$chmod 755 balances.txt
```

Tal como hemos visto anteriormente, se establecerían los permisos **rwxr-xr-x** para el archivo *balances.txt*

Permisos

- Cuando un usuario crea un fichero o un directorio lo hace con unos **permisos por defecto**
  - Lo hace el sistema automáticamente
- Si tenemos un fichero al que queremos que pueda acceder cualquiera y hacer con él lo que se quiera, le daremos los permisos 666
  - Recordamos que, aunque cada número puede llegar a 7, en un fichero únicamente colocaremos el permiso de ejecución si el fichero es **ejecutable**
    - *No tiene sentido darle permisos de ejecución a un fichero PDF, por ejemplo*
- Si hablamos de directorios, asignaremos como máximo el permiso 777 al mismo

Permisos

- *Vamos a comprobar los permisos que asigna nuestro sistema cuando creamos un fichero y un directorio nuevo*
  - *En nuestro caso (**xubuntu**):*

```
$touch fichero
$mkdir dir1
$ls -ld fichero dir1
drwxrwxr-x   6  ivens  ivens  4096  2024-04-04 19:48 dir1
-rw-rw-r--   1  ivens  ivens   528  2024-04-04 19:48 fichero
```

  - *Comprobamos que los permisos por defecto son:*
    - *Para los directorios: `rwxrwxr-x` (775)*
    - *Para los ficheros: `rw-rw-r--` (664)*

Permisos

`umask [opciones] [valor]`

- El comando umask crea una "mascara" que **quita** los permisos que queremos que, por defecto, tengan los ficheros/directorios que se vayan a crear
- Pero ¿cómo se forma la máscara?
  - El cálculo de estos tres dígitos se realiza de forma idéntica que en *chmod*
  - La única diferencia que en esta orden hay que dar el número decimal de los permisos que queremos quitar

Permisos

`umask [opciones] [valor]`

```
777 – máscara = permisos por defecto de directorio
666 – máscara = permisos por defecto de fichero
```

- En nuestro caso, directorios:

```
777 – máscara = 775 (que equivale a rwx rwx r-x)
            máscara = 777 – 775
                máscara = 2
```

- Y para nuestros ficheros (debería coincidir):

```
666 – máscara = 664 (que equivale a rw- rw- r--)
            máscara = 666 – 664
                máscara = 2
```


Permisos

`umask [opciones] [valor]`

- *Ejercicio práctico: supón que queremos que el dueño de los ficheros del sistema tenga todos los permisos y para el grupo únicamente se le conceda el permiso de lectura. El resto, no tendrá ninguno. Calcula su máscara:*

```
666 – máscara = 640 (que equivale a rw- r-- ---)

            máscara = 666 – 640

                máscara = 26
```

Recuerda que el permiso de **ejecución** (**x**) sólo hay que colocarlo en los ficheros ejecutables, nunca por defecto

Permisos

`umask [opciones] [valor]`

- Ahora podemos establecer esta máscara de la siguiente manera:

```
$umask 26
```

- Todos los cambios que realicemos en la máscara tendrán únicamente efecto durante la sesión en curso

- La acción por defecto que realiza es mostrar la máscara actual

```
$umask
0026
```


Permisos

`umask [opciones] [valor]`

- *Ejercicio práctico: establece la máscara de modo que, cuando creemos un directorio, el propietario tenga todos los permisos, el grupo también, excepto los permisos de escritura, y el resto de usuarios sólo pueda listar su contenido*

```
777 – máscara = 754 (que equivale a rwx r-x r-x)
            máscara = 777 – 755
                máscara = 22
```

- *Ejecutaríamos:*

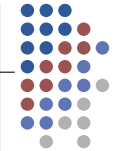
```
$umask 22
```

Recuerda que para que sea efectivo el permiso de **lectura**, debe estar activado el permiso de **ejecución**

Permisos, usuarios y grupos

- Índice
  - Introducción
  - Identificación de usuarios conectados
  - Permisos
  - Administración de usuarios y grupos
  - Cambio de identificador
  - Ficheros de sesión

Administración de usuarios y grupos

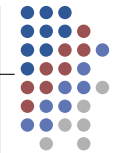


- GNU/Linux = multiusuario
  - Se permite a más de un usuario el uso del sistema al mismo tiempo y a la vez
- En este punto vamos a ocuparnos de la administración de todos ellos (usuarios y grupos)

Administración de usuarios y grupos

- En todos los sistemas GNU/Linux existe lo que se conoce como **UID** y **GID**
  - **UID** = Identificador de usuario, que vendría a ser como el DNI del usuario, un número identificativo único
  - **GID** = Identificador de grupo y tiene la misma misión que el UID, pero aplicada a los grupos
- Estos números están comprendidos entre un rango de **0** a **65534**
  - El ID **0** está reservado para **root**
  - Los ID **1** a **999** se reservan para cuentas del sistema
  - Los ID **1000** en adelante son para usuarios ordinarios

Administración de usuarios y grupos



id

- Este comando nos va a permitir saber cuál es nuestro **UID**, así como los grupos a los que pertenecemos (mostrándonos también su **GID**)

```
$id
uid=1000(ivens) gid=1000(ivens)
grupos=1000(ivens),4(adm),24(cdrom),
27(sudo),30(dip),46(plugdev),
120(lpadmin),128(sambashare)
```

- En este ejemplo podemos ver que mi usuario (*ivens*) tiene el UID=1000 y el GID=1000
- Además, pertenece a los grupos `ivens`, `adm`, `cdrom`, `sudo`, `dip`, `plugdev`, `lpadmin` y `sambashare`

Administración de usuarios y grupos

groups

- Nos indica a qué grupos pertenecemos

```
$groups
ivens adm cdrom sudo dip plugdev lpadmin sambashare
```


Administración de usuarios y grupos

- Vamos a ver unos **ficheros de configuración**:
  - **/etc/adduser.conf**
  - **/etc/passwd**
  - **/etc/shadow**
  - **/etc/group**

Administración de usuarios y grupos

- **/etc/adduser.conf**
  - Cuando añadimos un nuevo usuario, el sistema lee este archivo de configuración para saber qué debe crear
  - Veamos algunos de los parámetros más importantes:
    - **DSHELL=/bin/bash** Indica cual va a ser su intérprete de comandos
    - **DHOME=/home** Especifica dónde se deben crear los nuevos directorios de usuario

Administración de usuarios y grupos

- **/etc/adduser.conf**
  - **SKEL=/etc/skel** Cuando se crea un usuario se deben copiar una serie de estructuras de directorios y ficheros de configuración.
    Estos datos se encuentran en el directorio indicado
  - **EXTRA_GROUPS=""** Indica que cada usuario pertenecerá desde un principio a los grupos aquí indicados

Administración de usuarios y grupos

- **/etc/passwd**
  - Este fichero registra los usuarios existentes en el sistema
  - Originalmente, las contraseñas se almacenaban encriptadas en este fichero, pero esto ha cambiado para ser almacenadas en el fichero **/etc/shadow**
  - El fichero **/etc/passwd** contiene una línea por cada usuario del sistema
    - Cada uno de los campos está separado mediante el carácter "**:**"
    - Veamos los campos que componen cada una de las líneas…

Administración de usuarios y grupos

- **/etc/passwd**
  - Nombre de usuario
  - Contraseña
  - UID
  - GID
  - Nombre completo y descripción de la cuenta
  - Directorio personal
  - Shell

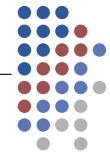
Administración de usuarios y grupos

- **/etc/passwd**
  - Un ejemplo de usuario registrado en el sistema:

```
ivens:x:1000:1000:Ivens Huertas,,,:/home/ivens:/bin/bash
```

  - Nombre de usuario
  - Contraseña
  - UID
  - GID
  - Nombre completo y descripción de la cuenta
  - Directorio personal
  - Shell

Administración de usuarios y grupos



- **/etc/passwd**
  - Si en el campo de la contraseña aparece una "**x**", indica que la contraseña estará almacenada en */etc/shadow*
    - A esta técnica se la conoce como **shadow passwords**
    - Si por el contrario la contraseña está almacenada en el propio fichero `/etc/passwd`, aparecerá encriptada

```
ivens:x:1000:1000:Ivens Huertas,,,:/home/ivens:/bin/bash
```

  - Para deshabilitar temporalmente un usuario y que no pueda entrar al sistema, basta con poner un asterisco (*****) donde está la "**x**" , el campo de la contraseña
    - El usuario no podrá entrar al sistema, pero sus ficheros personales aún permanecerán

```
ivens:*:1000:1000:Ivens Huertas,,,:/home/ivens:/bin/bash
```

Administración de usuarios y grupos

- **/etc/shadow**
  - El fichero /etc/passwd puede ser leído por los usuarios ordinarios, pero **/etc/shadow** sólo lo puede leer **root**
    - Es por ello que se hace la separación entre el primero y el segundo: por **seguridad**
    - Aunque las contraseñas estén cifradas, cualquiera que lo copiase podría dedicarse a descifrar las contraseñas en su tiempo libre
    - El traslado de las contraseñas cifradas al archivo **/etc/shadow**, accesible únicamente para el superusuario, añade una capa de protección

Administración de usuarios y grupos

- **/etc/shadow**

```
ivens:$1$KfNtvOJb$qFahz.f8zYyotjToffBFk0:18332:0:99999:7:3::
```

  - Usuario
  - Contraseña cifrada
  - Nº de días desde el 01/01/1970 hasta el último cambio de la contraseña
  - Nº de días que deben pasar hasta que se pueda cambiar la contraseña
  - Nº de días que deben pasar para que caduque la contraseña y deba ser cambiada

Administración de usuarios y grupos

- **/etc/shadow**

```
ivens:$1$KfNtvOJb$qFahz.f8zYyotjToffBFk0:18332:0:99999:7:3::
```

  - Nº de días de antelación con los que avisará el sistema de la caducidad de la contraseña
  - Nº de días con contraseña caducada antes de deshabilitar la cuenta
  - Nº de días entre el 01/01/1970 y el día en que se deshabilitó la cuenta *(en el ejemplo, está anulado al estar vacío)*
  - Campo reservado por si hace falta añadir alguna otra información futura *(en el ejemplo, está anulado al estar vacío)*

Administración de usuarios y grupos

- **/etc/shadow**
  - La contraseña está cifrada mediante algoritmos que aseguran que nunca habrá dos contraseñas iguales, ni siquiera repitiendo la clave
  - Para deshabilitar un usuario y que no pueda entrar basta con poner un signo de exclamación de cierre (**!**) en el campo de la contraseña cifrada

```
root:!:18325:0:99999:7:::
```

Podemos comprobar que, en la familia Ubuntu, la cuenta de **root** está deshabilitada

Administración de usuarios y grupos

- **/etc/group**
  - Almacena los grupos registrados en el sistema
  - Al igual que los dos anteriores ficheros, encontramos una línea por cada grupo y sus campos están delimitados por el carácter "**:**"
  - Estos son los campos:
    - Nombre del grupo
    - Contraseña cifrada
    - GID
    - Usuarios que pertenecen al grupo

Administración de usuarios y grupos

- **/etc/group**
  - Un ejemplo de algunos grupos registrados en el sistema:

```
admin:x:121:ivens
ivens:x:1000:
cdrom:x:24:ivens
pulse:x:114:
```

  - Al igual que ocurría con **/etc/passwd**, si en el campo de la contraseña aparece una "**x**", indica que la contraseña estará almacenada en **/etc/gshadow**

Administración de usuarios y grupos

- Creación de usuarios
  - Los comandos que vamos a ver crean un grupo privado para cada nuevo usuario, con el mismo identificador de usuario (UID) y de grupo (GID)
  - Utilizaremos los siguientes comandos:
    - **adduser**
    - **useradd**

Administración de usuarios y grupos

adduser [opciones] usuario

- Permite crear un usuario sin más que especificar su login
  - El sistema irá pidiendo los datos necesarios para completar los campos
  - Si alguno de los campos, salvo la contraseña, se quiere dejar en blanco bastará con pulsar sucesivamente la tecla **Enter**

```
#adduser laura
```

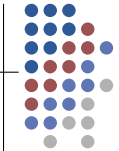
Administración de usuarios y grupos

`adduser [opciones] usuario`

- Disponemos de múltiples opciones, de las que destacaremos las siguientes:
  - **-d** Especifica el directorio personal del usuario
  - **-p** Establece la contraseña
  - **-G** Especifica el grupo al que va a pertenecer por defecto
  - **-e** Especifica cuándo expirará la cuenta, en formato de fecha YYYY-MM-DD

```
#adduser -d /home/laura -e 2025-12-31
         -p laurapass
         -G alumnos laura
```


Administración de usuarios y grupos



`useradd [opciones] usuario`

- Este comando está algo **en desuso** debido a que, para crear al completo el usuario, hay que realizar posteriormente a mano diversas tareas como son:
  - Editar a mano el fichero **/etc/passwd** y **/etc/group**
  - Crear la estructura de directorios del usuario
  - Establecer permisos a los directorios personales
  - Establecer las contraseñas y encriptarlas

Administración de usuarios y grupos

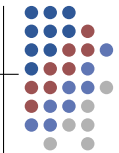
- Modificación de usuarios

`usermod [opciones] usuario`

- Se pueden variar las características de las cuentas de los usuarios mediante este comando:
  - **-l** Cambiar el nombre de usuario
  - **-d** Cambiar el directorio personal del usuario
  - **-p** Cambiar la contraseña
  - **-G** Cambiar su grupo por defecto
  - **-L** Deshabilitar cuenta
  - **-U** Habilitar cuenta
  - **-f** Volver la cuenta inactiva en un número determinado de días
  - **-e** Cambiar cuándo expirará la cuenta, en formato de fecha YYYY-MM-DD

```
#usermod -L -l maria laura
```


Administración de usuarios y grupos



`passwd [opciones] [usuario]`

- Este comando cambia la contraseña de usuarios y grupos
  - Cada usuario puede cambiar la suya cuando quiera
  - **root** puede cambiar la contraseña de cualquier usuario
- Si no se indica usuario, se estará cambiando la contraseña del usuario actual

Administración de usuarios y grupos

`passwd [opciones] [usuario]`

- Cambiarla desde la cuenta del usuario en cuestión

```
$passwd
Cambiando la contraseña de ivens
(actual) contraseña de UNIX:
Introduzca la nueva contraseña de ivens:
Vuelva a escribir la nueva contraseña de ivens:
passwd: contraseña actualizada correctamente
$
```


Administración de usuarios y grupos

- Eliminación de usuarios

`userdel [opciones] usuario`

- **-r** Borra todo rastro de directorio *home* del usuario borrado
- **-f** Fuerza el borrado en el caso de que el sistema no permita la eliminación por la existencia de ficheros que no son de propiedad del usuario

```
#userdel  -rf  maria
```


Administración de usuarios y grupos

- Creación de grupos

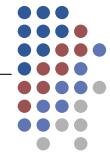
`groupadd [opciones] grupo`

- Este comando nos permitirá añadir grupos nuevos al sistema

```
#groupadd profesores
```

- En el ejemplo, se crea el grupo *profesores*

Administración de usuarios y grupos



- Añadir usuarios a grupos

`adduser usuario grupo`

- Utilizando el comando `adduser` que ya hemos visto, podemos añadir un usuario a un grupo

```
#adduser ivens profesores
```

- En el ejemplo, se añade el usuario *ivens* al grupo *profesores*

Administración de usuarios y grupos

- Modificación de grupos

```
groupmod [opciones] grupo
```

  - Este comando permite modificar grupos que ya están registrados en el sistema
  - Destacamos la siguiente opción:
    - **-n** Cambia el nombre del grupo

```
#groupmod -n docentes profesores
```

  - En el ejemplo, se modifica el nombre del grupo *profesores* por *docentes*

Permisos, usuarios y grupos

- Índice
  - Introducción
  - Identificación de usuarios conectados
  - Permisos
  - Administración de usuarios y grupos
  - Cambio de identificador
  - Ficheros de sesión

Cambio de identificador

```
su [opciones] [usuario]
```

- Corresponden a las siglas de "switch user" y permite cambiar de usuario sin necesidad de hacer un **logout**
  - Si no se especifica el nombre de usuario, se asume que éste será **root**
    - Recuerda que, en la familia Ubuntu, la cuenta **root** está deshabilitada (lo que comentamos ahora es para el caso en que la habilitemos o para sistemas con la cuenta habilitada desde un principio)

Cambio de identificador

```
su [opciones] [usuario]
```

```
$whoami
ivens
$pwd
/home/ivens
$su maria
Contraseña:
$whoami
maria
$pwd
/home/ivens
$exit
```

Podemos ver que *su* cambia de usuario (de **ivens** a **maria**), pero no nos cambia al directorio **home** de **maria**

Eso sí, esta variable de entorno es actualizada al nuevo usuario. Si hiciésemos **cd** a secas, nos llevaría a **/home/maria**

Cambio de identificador

`su [opciones] [usuario]`

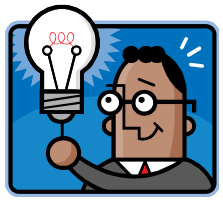
En sistemas donde tenemos que usar *su*,
la mejor y más segura forma de ser **root**, es mediante:
`$su -`

- Pero ¿qué hace el guión "-" detrás de *su*?
  - Eliminar casi todas las variables de entorno
  - Cambiar al nuevo directorio **`$HOME`** (en este caso, a **`/root`**)

Cambio de identificador

`su [opciones] [usuario]`

Recuerda que, como buena práctica informática,
debemos salir de la cuenta de `root` en cuanto hayamos terminado de realizar todas las tareas administrativas



Cambio de identificador

`sudo [opciones] [usuario] [comando]`

- El comando *sudo* otorga privilegios limitados de superusuario a usuarios específicos sin ser necesario dar la contraseña de superusuario
- En muchas de las nuevas distribuciones GNU/Linux (como Ubuntu, por ejemplo), hemos visto que el usuario **root** viene deshabilitado por defecto
  - Esto a priori es una buena medida de seguridad, pero requiere de una forma de poder administrar el sistema
  - Esta forma son los ***sudoers users***
  - Estos usuarios se administran en el fichero **`/etc/sudoers`** y ejecutan las tareas mediante el comando *sudo*

Cambio de identificador

`sudo [opciones] [usuario] [comando]`

- Resumiendo:
  - Deshabilitado el usuario root…
  - …damos a determinados usurios/grupos acceso a tareas de mantenimiento
  - Todo esto se determina en el citado fichero **`/etc/sudoers`**
- Si echamos un vistazo a este fichero de texto veremos usuarios y grupos junto a los privilegios que tienen concedidos

Cambio de identificador

`sudo [opciones] [usuario] [comando]`

- Los **usuarios** aparecen con su nombre tal cual
- Los **grupos** aparecen con el símbolo "**%**" delante del nombre del mismo

```
ivens ALL=(ALL) ALL
%admin ALL=(ALL) ALL
%sudo ALL=(ALL) ALL
```

Ojo: sólo podrán disponer de estos "poderes/privilegios", si usan el comando *sudo* delante del comando que vayan a ejecutar

- Se le otorgan al usuario ***ivens*** privilegios de superusuario desde cualquier equipo, para ejecutar comandos de todos los usuarios y para ejecutar todos los comandos
- Los mismos privilegios se le dan a cualquier usuario que esté en el grupo ***admin*** o en el grupo ***sudo***

Cambio de identificador

`sudo [opciones] [usuario] [comando]`

- *Ejemplo:*
  - Supongamos que queremos eliminar un fichero y no disponemos de permisos para borrarlo con nuestro usuario corriente `ivens` Haríamos lo siguiente:

```
$rm /home/maria/trabajo.pdf
rm: Permiso denegado
```

  - El sistema mostraría un mensaje de error
  - Es lógico: somos el usuario `ivens` y no tenemos permisos para eliminar ese fichero
    (está en el directorio personal del usuario *maria*)

Cambio de identificador

`sudo [opciones] [usuario] [comando]`

- Pero podemos utilizar el comando *sudo* para que se nos otorguen privilegios limitados de superusuario y ejecutar esta acción:

```
$sudo rm /home/maria/trabajo.pdf
[sudo] password for ivens:
```

El fichero se borra sin problema

Cambio de identificador

`sudo [opciones] [usuario] [comando]`

- Vemos que, anteponiendo *sudo* delante del comando administrativo a ejecutar, el sistema nos permitirá ejecutarlo
- Pero hay que tener en cuenta dos cosas muy importantes:
  - Nuestro usuario debe tener **asignados los permisos de superusuario** en **`/etc/sudoers`**
  - Cuando el sistema nos pide la contraseña, **es la contraseña del propio usuario, no la de `root`**

Cambio de identificador

`sudo [opciones] [usuario] [comando]`

- Una cosa que muchos usuarios acostumbrados a *su* detestan de *sudo*, es el tener que escribirlo cada línea, una vez por cada comando
  - No hay problema: con la opción `-i` cargamos una shell de `root`

```
$sudo -i
[sudo] password for ivens:
#whoami
root
```

Vemos que, una vez introducida nuestra contraseña, estaremos trabajando desde el shell de `root` hasta que salgamos de él mediante `exit` o `logout`

Cambio de identificador

Recordamos:

En sistemas donde tenemos que usar *su*,
la mejor y más segura forma de ser `root`, es mediante:

`$su -`

Ejemplo: Debian

En sistemas donde tengamos que usar `sudo`, la mejor opción es usar:

`$sudo comando`

Pero si hay que ejecutar muchos comandos, usaremos:

`$sudo -i`

Ejemplo: Ubuntu

Permisos, usuarios y grupos

- Índice
  - Introducción
  - Identificación de usuarios conectados
  - Permisos
  - Administración de usuarios y grupos
  - Cambio de identificador
  - Ficheros de sesión

Ficheros de sesión

- Existen algunos ficheros que son de cierto interés cuando trabajamos con usuarios en Ubuntu, ya que son ficheros que realizan una serie de funciones cuando dichos usuarios **inician sesión**
- Otros, sin embargo, son leídos cuando el usuario **cierra sesión**
- Concretamente, hablaremos de tres ficheros ubicados en la carpeta personal del usuario:
  - **`~/.bashrc`**
  - **`~/.profile`**
  - **`~/.bash_logout`**

Recuerda que, al comenzar por **punto**, son ficheros ocultos

Ficheros de sesión

- `~/.profile` y `~/.bashrc`
  - Estos dos scripts son leídos cuando el usuario **accede al sistema** a través de un **terminal** en modo gráfico o modo texto
    - Cuando el usuario inicia sesión, el sistema leerá el fichero **`.profile`** del usuario para inicializar variables de entorno a los valores acordados en él
    - Cuando el usuario abre un terminal, el sistema operativo localiza en su carpeta personal el fichero **`.bashrc`**
  - *Probad a visualizar el contenido de ambos ficheros*

Ficheros de sesión

- `~/.profile` y `~/.bashrc`
  - La utilidad de estos ficheros la podemos encontrar, por ejemplo, en la creación de *alias* o estableciendo máscaras de permisos mediante *umask*
    - Ya comprobamos que, una vez se cerraba la sesión (o cerrábamos el terminal), se perdían los alias o las máscaras de permisos creadas en dicha sesión
    - Es una buena idea escribir los alias dentro del fichero **`.bashrc`** para tener los alias, por ejemplo, siempre disponibles

Ficheros de sesión

- `~/.bash_logout`
  - Cuando el usuario cierre sesión, este será el fichero que se lea
  - Suele utilizarse para tareas de limpieza al cerrar sesión

Ficheros de sesión

- Cada usuario del sistema tiene su propia copia de estos ficheros
- Si necesitamos que se inicien ciertos comandos para todos los usuarios del sistema, disponemos de dos ficheros destinados a ello:
  - **`/etc/bash.bashrc`**
  - **`/etc/profile`**

Son el equivalente a los ficheros personales, pero aplicados a **todos los usuarios**

Necesitamos permisos de **`root`** para poder editarlos

Ficheros de sesión

- Otro fichero interesante es `~/.bash_history`
  - Contiene el historial de todos los comandos que ha ejecutado el usuario
  - Al lanzar el comando *`history`*, realmente estamos mostrando el contenido de este fichero