

## UD 05\_01 – GNU/Linux

### Introducción. Comandos sobre ficheros. Usuarios y grupos.

## Contenido

1	INTRODUCCIÓN: GNU/LINUX Y SUS DISTRIBUCIONES .....	3
2	NOMBRES DE DISPOSITIVOS EN GNU/LINUX.....	4
3	PARTICIONES NECESARIAS PARA INSTALAR EL SISTEMA OPERATIVO .....	5
3.1	Al menos, la partición / .....	5
3.2	Partición de swap.....	5
4	ESTRUCTURA DE GNU/LINUX.....	6
4.1	El kernel.....	6
4.2	El shell.....	7
4.3	El sistema de archivos .....	8
4.3.1	La estructura de directorios en GNU/Linux.....	9
5	LA TERMINAL O CONSOLA .....	11
6	AYUDA DE GNU/LINUX Y DOCUMENTACIÓN .....	13
6.1	Comando man .....	14
7	SINTAXIS DE LAS ÓRDENES .....	16
8	COMANDOS PARA MANIPULAR FICHEROS Y DIRECTORIOS.....	17
8.1	Comando ls.....	17
8.2	Comando cd .....	17
8.3	Comando pwd .....	18
8.4	Comando mkdir.....	18
8.5	Comando rmdir .....	18
8.6	Comando mv .....	19
8.7	Comando cp .....	19
8.8	Comando rm.....	20
8.9	Comando chown/chgrp.....	20
8.10	Otros comandos relacionados.....	21
8.10.1	Comando df .....	21
8.10.2	Comando type .....	21
8.10.3	Comando stat .....	21
8.10.4	Comando diff.....	21
8.10.5	Comando du .....	22
8.10.6	Comando file.....	22
8.10.7	Comando whereis .....	22
8.10.8	Comando which.....	22
9	SUPERUSUARIO.....	23
9.1	Comando su .....	23
9.2	Comando sudo.....	24

<b>10 FICHEROS: INFORMACIÓN, PERMISOS, PROPIETARIO, GRUPO .....</b>	<b>27</b>
<b>10.1 Permisos de lectura, escritura y ejecución .....</b>	<b>27</b>
10.1.1 Permiso de lectura .....	27
10.1.2 Permiso de escritura .....	27
10.1.3 Permiso de ejecución .....	27
<b>10.2 Información sobre un fichero.....</b>	<b>28</b>
<b>10.3 Cambiar permisos de un fichero (chmod) .....</b>	<b>29</b>
<b>11 USUARIOS Y GRUPOS.....</b>	<b>31</b>
<b>11.1 Comandos para gestión de cuentas .....</b>	<b>31</b>
11.1.1 Comando useradd .....	31
11.1.2 Comando passwd .....	32
11.1.3 Comando chage .....	32
11.1.4 Comando usermod .....	33
11.1.5 Comando userdel.....	33
11.1.6 Comando groupadd.....	33
11.1.7 Comando groupmod.....	33
11.1.8 Comando groupdel.....	34
11.1.9 Comando gpasswd .....	34
11.1.10 Comandos adicionales para gestionar usuarios.....	34
<b>11.2 Ficheros de configuración de usuarios y grupos .....</b>	<b>36</b>
11.2.1 /etc/passwd .....	36
11.2.2 /etc/shadow.....	37
11.2.3 /etc/group .....	39
<b>12 VARIOS COMANDOS.....</b>	<b>40</b>
<b>12.1 Fecha y estadísticas .....</b>	<b>40</b>
12.1.1 Comando date.....	40
12.1.2 Comando cal.....	43
12.1.3 Comando time.....	43
12.1.4 Comando uptime .....	43
<b>12.2 Información de la memoria .....</b>	<b>43</b>
12.2.1 Comando top .....	43
12.2.2 Comando free.....	45
<b>12.3 Otra información del sistema.....</b>	<b>45</b>
12.3.1 Comando uname .....	45
12.3.2 Comando who .....	46
12.3.3 Comando w.....	47
12.3.4 Comandos hostname y dnsdomainname.....	47
<b>13 BIBLIOGRAFÍA .....</b>	<b>48</b>
<b>14 EJERCICIOS .....</b>	<b>49</b>
<b>14.1 Distribuciones de GNU/Linux .....</b>	<b>49</b>
<b>14.2 Manejo de comandos.....</b>	<b>49</b>
14.2.1 Cuadro resumen .....	49
14.2.2 Ayuda de GNU/Linux .....	49
14.2.3 Comandos para manipular ficheros y directorios .....	50
<b>14.3 Usuarios, grupos y contraseñas .....</b>	<b>52</b>
<b>14.4 Otros.....</b>	<b>53</b>
14.4.1 Comandos de fecha y estadísticas.....	53
14.4.2 Información de memoria .....	53
14.4.3 Información del sistema.....	53

## 1 INTRODUCCIÓN: GNU/LINUX Y SUS DISTRIBUCIONES

---

GNU/Linux, también conocido como Linux, es un sistema operativo libre tipo Unix; multiplataforma, multiusuario y multitarea. El sistema es la combinación de varios proyectos, entre los cuales destacan GNU (encabezado por Richard Stallman y la Free Software Foundation) y el núcleo Linux (encabezado por Linus Torvalds). Su desarrollo es uno de los ejemplos más prominentes de software libre: todo su código fuente puede ser utilizado, modificado y redistribuido libremente por cualquiera, bajo los términos de la GPL (Licencia Pública General de GNU) y otra serie de licencias libres.

El nombre GNU/Linux viene de que se utiliza el núcleo o kernel de Linux y herramientas desarrolladas desde el proyecto GNU. GNU también está desarrollando su propio núcleo, el Hurd (se empezó a desarrollar en 1990, pero se ha quedado estancado en varias ocasiones).

A GNU/Linux se le encuentra normalmente en forma de compendios conocidos como distribuciones o distros, a las cuales se les han adicionado selecciones de aplicaciones y programas para descargar e instalar las mismas. Son recopilaciones de software ya compilado y configurado que se basa en el núcleo Linux e incluye una serie de paquetes de software para dar respuesta a diferentes necesidades.

- Aquí puedes ver las distribuciones GNU/Linux a lo largo del tiempo: [https://upload.wikimedia.org/wikipedia/commons/1/1b/Linux\\_Distribution\\_Timeline.svg](https://upload.wikimedia.org/wikipedia/commons/1/1b/Linux_Distribution_Timeline.svg)

El propósito de una distribución es ofrecer GNU/Linux como un producto final que el usuario pueda instalar, cumpliendo con las necesidades de un grupo de usuarios o bien del público general. En general están compuestas mayoritariamente de software libre, aunque a menudo incorporan aplicaciones o controladores propietarios.

### Más información:

- GNU/Linux: <https://es.wikipedia.org/wiki/GNU%2FLinux>
- Unix: <https://es.wikipedia.org/wiki/Unix>
- Distribuciones GNU/Linux: [https://es.wikipedia.org/wiki/Distribuci%C3%B3n\\_Linux](https://es.wikipedia.org/wiki/Distribuci%C3%B3n_Linux)
- Algunas distribuciones:
  - Debian: <https://es.wikipedia.org/wiki/Debian>
  - Ubuntu: <https://es.wikipedia.org/wiki/Ubuntu>
  - Red Hat: [https://es.wikipedia.org/wiki/Red\\_Hat\\_Linux](https://es.wikipedia.org/wiki/Red_Hat_Linux)
  - openSuse: <https://es.wikipedia.org/wiki/OpenSUSE>

## 2 NOMBRES DE DISPOSITIVOS EN GNU/LINUX

Cuando el sistema GNU/Linux arranca, escanea el hardware, y cuando encuentra los discos y particiones les da un nombre. Estos nombres se utilizan para ser encontrados como archivos de dispositivos especiales dentro del directorio `/dev/` (que viene de devices o dispositivos).

Los discos tienen nombres como `hda`, `hdb`, `hdc`,... para dispositivos IDE; o `sda`, `sdb`, `sdc`, ... (**actualmente se usa `sdx` para dispositivos IDE, SATA y SCSI**).

Las **particiones** de `hda` serán `hda1`, `hda2`,... y las `sda` serán `sda1`, `sda2`, ... respectivamente, donde el número es el número de partición.

GNU/Linux representa la partición primaria como el nombre del dispositivo, más un número del 1 al 4 (salvo que se esté usando GPT, entonces se numerarán hasta el número de particiones que haya).

- Por ejemplo, la primera partición en la primera unidad sería `/dev/sda1`.
  - `/` indica el root o raíz del árbol de directorios de GNU/Linux.
  - **`dev`** indica el directorio donde se almacenan todos los dispositivos
  - **`/sda1`** se refiere a la primera partición en la unidad (`sd`) con la letra `a`.
- Las particiones lógicas son enumeradas empezando desde el número 5, así:
  - **`/dev/sda5`** será la primera partición lógica en el mismo disco.
  - **`/dev/sda6`** será la segunda partición lógica.
- Si tenemos otra unidad, se utilizará la letra `b`, así tendríamos `/dev/sdb`.
- Recuerda que la partición extendida, es decir, la partición primaria que contiene a las particiones lógicas, no es utilizable de por sí misma.

### Fuentes y más información:

- [https://es.opensuse.org/SDB:Fundamentos\\_sobre\\_particiones\\_sistema\\_s\\_de\\_archivos\\_y\\_puntos\\_de\\_montaje](https://es.opensuse.org/SDB:Fundamentos_sobre_particiones_sistema_s_de_archivos_y_puntos_de_montaje)
- <https://www.debian.org/releases/jessie/amd64/apcs04.html.es>
- [http://bibliaubuntu.a.freewiki.in/index.php/Montando\\_sistemas\\_de\\_archivos](http://bibliaubuntu.a.freewiki.in/index.php/Montando_sistemas_de_archivos)

## 3 PARTICIONES NECESARIAS PARA INSTALAR EL SISTEMA OPERATIVO

---

### 3.1 Al menos, la partición /

Es obligatorio montar al menos la partición / durante la instalación.

Además, es posible crear particiones dedicadas a otros directorios, como una partición para /home (ficheros de los usuarios), una para /etc, una para /root, etc. (o sólo para alguno/s de ellos, para los que nos interese). Esto podría facilitar labores de mantenimiento, por ejemplo, se podría reinstalar el sistema sin tocar los archivos de los usuarios.

### 3.2 Partición de swap

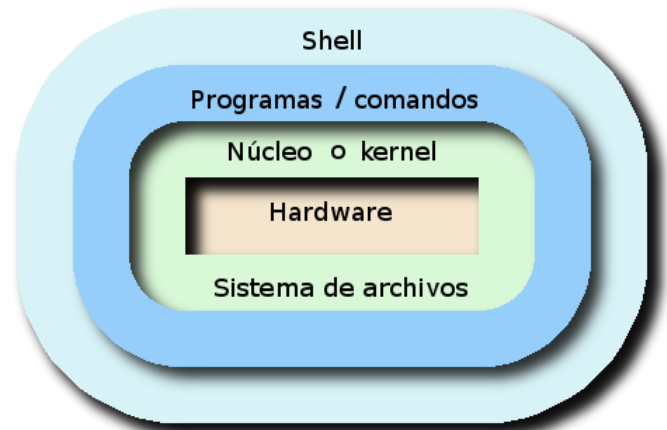
La partición de swap o de intercambio corresponde a la memoria virtual que veríamos en el tema sobre introducción a los sistemas operativos. A esta partición no se le asigna un nombre y un número como a las otras particiones, es sólo el sistema el que se encarga de utilizarla.

Típicamente, el tamaño que se le solía dar a una partición swap era el doble de la memoria RAM que se tenía instalada en el sistema, aunque esta cifra es simplemente una recomendación (lo que sí se recomienda es utilizar siempre swap, a menos que se tengan cantidades de memoria RAM cercanas al GB).

Hoy en día, si disponemos de más de 2 GB de RAM y no vamos a utilizar nuestro ordenador como máquina con mucha carga de procesos, la memoria swap no es necesaria. Si lo deseamos, podemos poner un fichero de intercambio de 1GB por ejemplo, aunque esa porción del disco posiblemente estará desaprovechada. Por ejemplo, en Ubuntu 18.04 se utiliza por defecto un fichero de intercambio en lugar de una partición swap, pero sería posible crear esta en lugar del fichero si se deseara.

## 4 ESTRUCTURA DE GNU/LINUX

Para comprender el funcionamiento del sistema GNU/Linux es necesario comprender su estructura. Este sistema operativo está formado por varios componentes principales. Entre ellos, el **núcleo** (o kernel), el **shell**, el **sistema de archivos**, los **programas** y los **comandos**.



### 4.1 El kernel

El **kernel** es la parte del sistema operativo que sirve para **interactuar con el hardware**. Proporciona una serie de servicios que pueden ser utilizados por los programas, sin que estos tengan que preocuparse de cómo se gestiona el hardware.

En general, el **núcleo** es el encargado de gestionar la memoria, mantener el sistema de archivos, del manejo de interrupciones, manejo de errores, realización de los servicios de entrada/salida, asignación de los recursos de la CPU, comunicación entre procesos, etc.

El kernel se carga en memoria cuando el sistema se inicia y permanece en memoria hasta que el sistema se descarga por completo. Se diseña para ser lo más pequeño posible, permitiendo así que la memoria restante sea compartida entre todos los programas que se ejecutan en el sistema. Los programas se relacionan con el ordenador a través del núcleo mediante las **llamadas al sistema**.

El kernel suele estar escrito en **C**, con algo de lenguaje **ensamblador** que le facilita trabajar directamente con el hardware. Una de las ventajas de los sistemas Linux es que, al estar el código fuente del kernel disponible, es más sencillo desarrollar nuestros propios controladores y módulos del kernel.

El código ejecutable del núcleo de Linux (el que se carga en memoria al arrancar el ordenador) reside en un fichero denominado **vmlinux** (o en su homólogo comprimido, **vmlinuz**). Normalmente, este fichero se encuentra en el directorio **/boot** y suele tener como sufijo la versión del núcleo que generó el ejecutable.

En cada momento tenemos siempre dos versiones del núcleo: versión de producción y versión de desarrollo. La **versión de producción** es la versión estable en el momento. Esta versión es la que se debería utilizar para un uso

normal del sistema. Por su parte la **versión de desarrollo** es experimental y es la que usan los programadores para crear y verificar nuevas características. Estos núcleos suelen ser inestables y no deberían ser usadas en equipos en producción.

El kernel incluyendo su código fuente se puede conseguir en <http://www.kernel.org> (The Linux Kernel Archives).

## 4.2 El shell

El **shell** es la parte que permite al usuario comunicarse con el sistema. El hardware constituye la parte “trabajadora” del sistema. Está controlado por el núcleo o kernel, que realiza las funciones de más bajo nivel, suministrando recursos y atendiendo a los requerimientos del shell. Envolviendo al kernel se encuentra el shell, que recoge las peticiones del usuario y sus aplicaciones y las transmite previamente depuradas e interpretadas al kernel.

Puede estudiarse el shell desde dos puntos de vista: como **intérprete de comandos** y como **lenguaje de programación**, que combina mediante estructuras de control grupos de comandos almacenados en archivos llamados shell scripts o procedimientos shell.

Cuando el usuario introduce un comando, el shell, que es un programa en continua ejecución, analiza la línea y llama al programa o programas que realiza la función solicitada por el comando. Se ejecuta un shell para cada usuario que se conecta al sistema. Existen diferentes intérpretes de comandos:

- **SH:** Shell Bourne. Su indicativo por defecto es el símbolo dólar "\$". Este Shell está capacitado para redireccionar la salida y entrada standard, interpretar los metacaracteres, manejar variables y usar tuberías y filtros. Posee además su propio lenguaje de programación.
- **CSH:** C Shell. Tiene todas las características del SH, pero añade algunas específicas para los programadores de C. Su prompt de sistema queda representado por el símbolo "%".
- **BASH:** Bourne Again Shell. Este shell es una ampliación del SH, puesto que, entre otras características, permite edición de comandos ejecutados, tamaño ilimitado del histórico de comandos, control de trabajos y procesos, funciones, alias, cálculos aritméticos, etc. Este shell es el más utilizado en Linux.
- **Otros:** KSH: Korn shell. TCSH: Enhanced C Shell. ZSH: Z Shell. JSH: Shell Job.

El administrador del sistema debe adjudicar uno de estos shells a cada usuario. Este, a su vez, puede ejecutar cualquier shell siempre y cuando tenga autorización para ello. Cuando un usuario se conecta al sistema se inicia automáticamente un programa de shell denominado **shell de presentación**. Este shell se carga de forma automática cuando se accede al fichero

`/etc/passwd`, que contiene información de los usuarios incluido el shell estándar que usará cada usuario (veremos este fichero más adelante).

### 4.3 El sistema de archivos

Linux define una interfaz abstracta al nivel del kernel que incorpora varios sistemas de archivos diferentes. Linux define siete tipos de archivos, para determinar cuál es el tipo de un archivo se utiliza el comando `ls -l`. El primer carácter de la salida que genera `ls -l` determina el tipo de archivo.

Tipo de archivo	Símbolo
Archivos comunes	-
Directorios	d
Archivos de los dispositivos de caracteres	c
Archivos de los dispositivos de bloque	b
Conexiones del dominio local	s
Cauces designados	p
Enlace simbólico	l

- Un **archivo común** es un conjunto de bytes y Linux no impone una estructura determinada a su contenido. En Linux los **archivos ocultos** comienzan con un punto y para listarlos se debe incluir el parámetro `-a` en la orden `ls (ls -a)`.
- Un **directorio** contiene referencias a otros archivos. Se detallan algunos de los directorios estándar más importantes en el apartado 4.3.1.
- Los **archivos de dispositivo** permiten que el hardware del sistema se comunique con los periféricos. Los archivos de los dispositivos están caracterizados por dos números, el mayor le dice al kernel a qué controlador hace referencia el dispositivo y el menor le indica al controlador la unidad física a la que se ha de dirigir.
- Las **conexiones del dominio local** son conexiones que se establecen entre procesos en red y que permiten intercambiar información sin interferencias. Los **cauces designados** permiten que se establezca una comunicación entre dos procesos que se están ejecutando en el mismo ordenador.
- La diferencia entre los **enlaces físicos** y los **enlaces simbólicos** es que los enlaces físicos son referencias directas al contenido del archivo enlazado y los enlaces simbólicos son una referencia al nombre del archivo enlazado.



### 4.3.1 La estructura de directorios en GNU/Linux

En la mayor parte de las distribuciones GNU/Linux, los tipos específicos de archivos son almacenados juntos en subdirectorios estándar y los gestores de paquetes se usan para mantener una pista hacia el archivo que pertenece a cualquier programa. Es decir, al instalar un programa no se tendrán todos sus archivos en un único directorio perteneciente al programa, si no que se distribuirán según su función. Por ejemplo, los manuales del programa en /man o los archivos de configuración en /etc.

Ruta	Contenido
/	Directorio raíz del sistema
/bin	Comandos y programas esenciales
/boot	Kernel y archivos necesarios para cargarlo
/dev	Todos los dispositivos físicos del sistema (discos, impresoras..)
/etc	Archivos de configuración e inicio
/home	Directorio principal de cada uno de los usuarios
/lib	Librerías y partes del compilador de C
/media	Unidades físicas montadas (discos duros, DVDs, pen drives..)
/opt	Paquetes de software para aplicaciones opcionales
/proc	Información sobre los procesos cargados
/root	Directorio principal del superusuario
/sbin	Comandos para iniciar, reparar y recuperar el sistema
/srv	Datos servidos por el sistema
/sys	Área de trabajo del kernel y archivos de configuración
/tmp	Archivos temporales
/usr	Datos de usuario, contiene la mayoría de las utilidades y aplicaciones
/var	Archivos variables, como logs y temporales

Por curiosidad, puedes probar la distribución GoboLinux (<https://gobolinux.org/>), que utiliza otra jerarquía: cada programa tiene su propio árbol de directorios dentro del directorio /Programs. Para saber más: [https://gobolinux.org/at\\_a\\_glance.html](https://gobolinux.org/at_a_glance.html)

Más información:

- [https://es.wikipedia.org/wiki/Filesystem\\_Hierarchy\\_Standard](https://es.wikipedia.org/wiki/Filesystem_Hierarchy_Standard)
- <https://computernewage.com/2015/06/14/el-arbol-de-directorios-de-linux-al-detalle-que-contiene-cada-carpeta/>

### EJERCICIO:

Para ver el primer nivel del árbol de directorios del que dispones en tu ordenador (o máquina virtual), puedes abrir una terminal y escribir el siguiente comando:

```
tree -d -L 1 /
```

- tree lista los directorios de forma recursiva.
- La opción -d indica que sólo se listen los directorios.
- La opción -L 1 es para que sólo se muestre un nivel (si quisiéramos dos niveles, pondríamos un 2, etc.)
- La última opción utilizada es para indicar el directorio a partir del cual se tiene que listar. Si no se indica nada, se lista el directorio actual.

Prueba ahora con

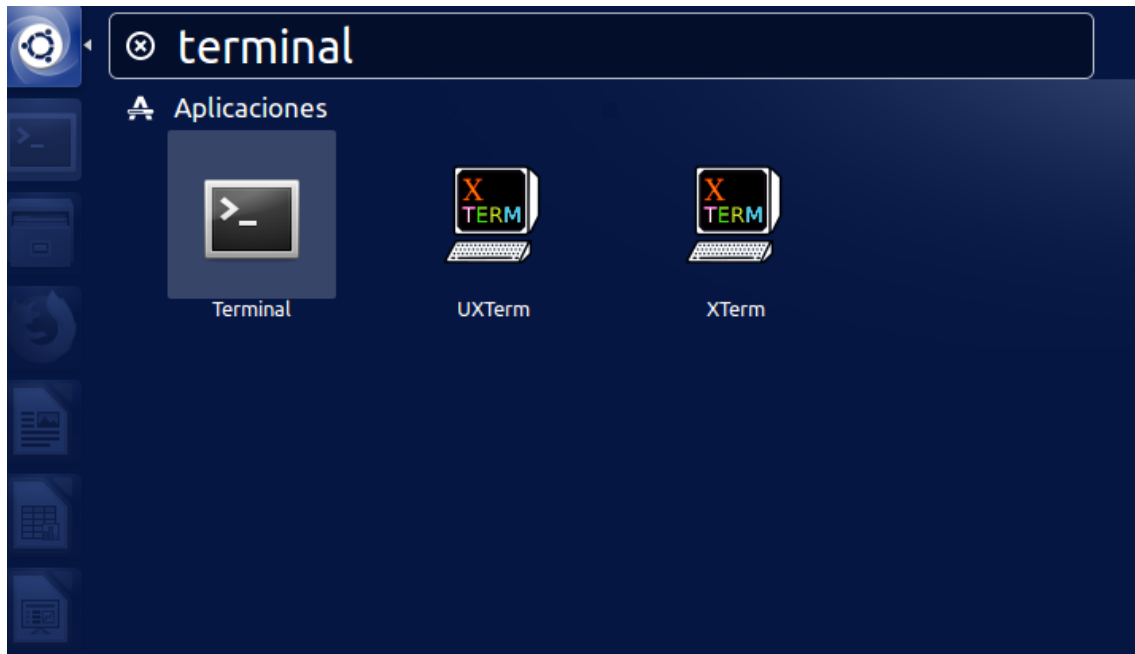
```
tree -d -L 2 /home
```

Listará el contenido del directorio /home, donde se encuentran los archivos de cada usuario. Observa que se muestra la información en forma de árbol.

Otro comando para listar el contenido de un directorio es `ls`, lo veremos con más detalle más adelante.

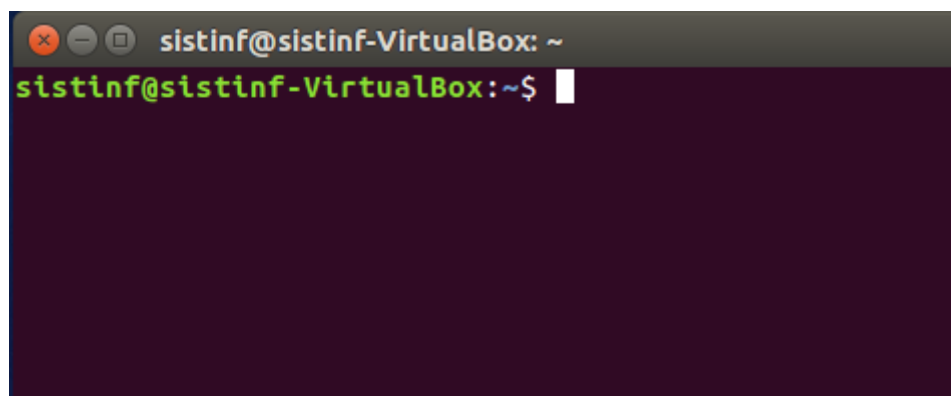
## 5 LA TERMINAL O CONSOLA

Para trabajar con ella, si estamos utilizando Ubuntu o LinuxMint por ejemplo, podemos escribir Terminal en el lanzador de aplicaciones:



La imagen de la terminal que se ve a continuación corresponde a la primera de ellas.

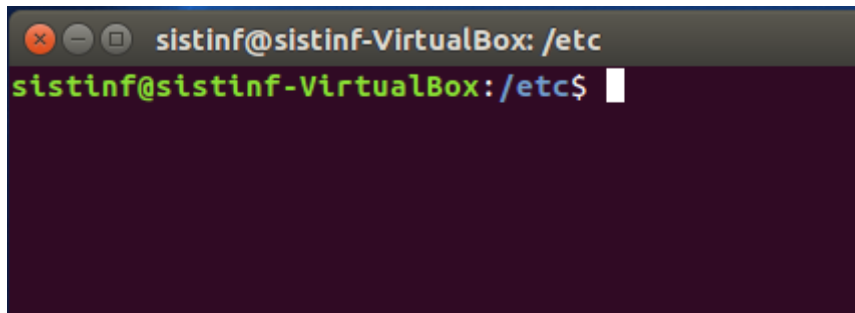
Al abrirse, veremos algo similar a lo siguiente, con información diferente según el usuario y el equipo que se esté utilizando, pero con la siguiente estructura, que se detalla más abajo.



- Las partes que se distinguen son las siguientes:  
usuario@equipo:directorio actual\$
  - usuario → sistinf
  - equipo → sistinf-VirtualBox
  - : separador

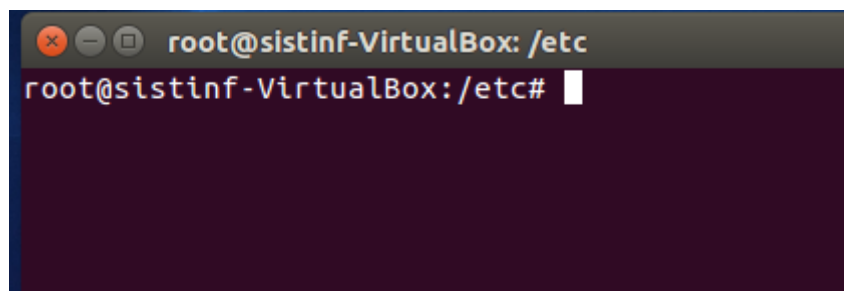
- ~ es la forma abreviada de referirse al directorio del usuario, en este caso /home/sistinf
- \$ es el carácter que indica que la terminal está a la espera de la introducción de una orden.

En la siguiente captura de pantalla, el directorio actual es /etc:



```
sistinf@sistinf-VirtualBox: /etc
sistinf@sistinf-VirtualBox: /etc$
```

Si el usuario es root, en lugar de \$, el carácter de espera es #:



```
root@sistinf-VirtualBox: /etc
root@sistinf-VirtualBox: /etc#
```

Vemos también que el nombre del usuario al principio de la línea ha cambiado.

(Para cambiar a root, se ha utilizado el comando `sudo su`. Se pedirá la contraseña, al teclearla no se verá nada, pero se está escribiendo. Para volver al usuario anterior, escribe `exit`. Es mejor volver al usuario normal, no se recomienda trabajar todo el tiempo como root, sólo cuando es necesario, ya que como puede realizar cualquier acción, es más fácil estropear algo (borrar, mover a donde no toca, cambiar configuraciones...) por error).

## 6 AYUDA DE GNU/LINUX Y DOCUMENTACIÓN

---

La documentación de Linux se encuentra dispersa en una gran variedad de fuentes, las más importantes son:

- Las páginas de ayuda accesible con el comando `man` (lo veremos con más detalle un poco más abajo).

- Documentos Texinfo accesibles con el comando `info`

```
info mkdir
```

- Ayuda breve de un comando añadiendo el parámetro `-h`, `--help` ó `-?`

```
mkdir --help
```

- Ayuda para los comandos integrados en el shell con `help`

```
help alias
```

- Descripción breve de un comando con `whatis`

```
whatis cp
```

- Buscar todo lo relacionado con una palabra con `apropos`

```
apropos delete
```

- Documentación de programas en el directorio `/usr/share/doc`

- HOWTOs, artículos sobre diferentes temas (por ejemplo: [www.tldp.org](http://www.tldp.org) )

- Documentación propia de cada distribución (por ejemplo: <https://help.ubuntu.com/lts/ubuntu-help/index.html> )

- En Internet de forma general existen multitud de sitios sobre GNU/Linux

## 6.1 Comando man

El manual en línea no es más que una serie de ficheros con un formato especial que contiene documentación de las órdenes UNIX.

```
Sintaxis: man [opciones] [comandos]
```

El manual se encuentra dividido en diferentes secciones:

```
La siguiente tabla muestra los números de sección del manual y los
tipos de páginas que contienen.

1  Programas ejecutables y guiones del intérprete de órdenes
2  Llamadas del sistema (funciones servidas por el núcleo)
3  Llamadas de la biblioteca (funciones contenidas en las bibliotecas
    del sistema)
4  Ficheros especiales (se encuentran generalmente en /dev)
5  Formato de ficheros y convenios p.ej. I/etc/passwd
6  Juegos
7  Paquetes de macros y convenios p.ej. man(7), groff(7).
8  Órdenes de admistración del sistema (generalmente solo son para
    root)
9  Rutinas del núcleo [No es estándar]
n  nuevo [obsoleto]
l  local [obsoleto]
p  público [obsoleto]
o  viejo [obsoleto]
```

Opciones:

- |                       |   |
|-----------------------|---|
| -f                    | Obtiene una pequeña descripción orientativa de lo que hace el comando. Este parámetro es equivalente al comando <code>whatis</code> .   |
| -s <i>num_sección</i> | Se accede a una sección en concreto del manual.   |
| -k <i>cadena</i>      | Busca en las páginas de manual las palabras reservadas que la contengan. Es equivalente al comando <code>apropos</code> .<br><br>Por ejemplo <code>man -k user</code> saca un listado de todos los comandos que contienen algo relativo a user. |

El propio manual dispone de su propio manual:

```
$ man man
```

Para salir del manual, pulsar la tecla Q.

Cuando se está dentro del manual, para buscar alguna palabra clave utilizaremos /palabra\_clave. Por ejemplo, estando dentro de la ayuda de bash, podemos buscar

### Ejemplos:

```
$ man -f time
```

```
time (1p) - time a simple command
time (3p) - get time
time (1) - time a simple command or give resource usage
time (7) - overview of time and timers
time (2) - get time in seconds
```

```
$ whatis time ls pwd
```

```
time (1p) - time a simple command
time (3p) - get time
time (1) - time a simple command or give resource usage
time (7) - overview of time and timers
time (2) - get time in seconds
ls (1p) - list directory contents
ls (1) - list directory contents
pwd (1p) - return working directory name
pwd (1) - print name of current/working directory
```

```
$ man -S 7 time
```

```
$ man -k password
```

```
$ apropos password
```

## 7 SINTAXIS DE LAS ÓRDENES

Las órdenes deben introducirse siguiendo unas normas determinadas. La estructura es la siguiente:

COMANDO [-OPCIONES] ARGUMENTOS

Se separan comando, opciones y argumentos con espacios en blanco.

Hay comandos que pueden utilizarse por sí solos, sin opciones ni argumentos, como `ls`.

Puede ser que usemos sólo opciones, pero no argumentos:

```
ls -l
```

Podemos usar argumentos, pero no opciones:

```
ls /etc
```

Podemos usar opciones y argumentos:

```
ls -l /etc
```

Es posible usar varias opciones:

```
ls -l -i -a /etc
```

O lo mismo de forma abreviada:

```
ls -lia /etc
```

Cada distribución puede tener sus propios parámetros para un mismo comando. En caso de duda, consultaremos la ayuda.



## 8 COMANDOS PARA MANIPULAR FICHEROS Y DIRECTORIOS.

### 8.1 Comando ls

El comando `ls` permite listar el contenido de un directorio. Ya lo hemos utilizado en ejemplos anteriores.

<b>Sintaxis:</b> <code>ls [opciones] [directorio   fichero]</code>
--

Algunas opciones:

- l muestra la salida en formato largo.
- R lista recursivamente un directorio.
- a lista además los ficheros ocultos (sus nombres comienzan con punto).
- h muestra el tamaño de los ficheros en forma más legible (Ej.: 16M, 4k, etc.), si no, están expresadas en bytes.
- i muestra el identificador del i-nodo asociado a cada elemento.

#### Ejemplos:

```
$ ls -hl /etc
$ ls -R /usr
$ ls -al
$ ls -ali ..
```

Un grupo de opciones que se suele utilizar es `lia` (`ls -lia`)

### 8.2 Comando cd

El comando `cd` se utiliza para cambiar el directorio actual.

<b>Sintaxis:</b> <code>cd [directorio]</code>
---

#### Ejemplos:

```
$ cd /tmp
$ cd      # sin parámetros cambia al directorio home del usuario.
$ cd ~    # cambia al directorio home del usuario.
$ cd ..   # cambia al directorio de nivel anterior.
```

Al hacer `cd ~` nos desplazamos al directorio que apunta la variable `$HOME`, por lo que sería equivalente a realizar un `cd $HOME`

El carácter `~` (virgulilla) se obtiene pulsando las teclas `Alt Gr + 4`.

La variable `$HOME` es una variable de entorno que almacena el directorio del usuario actual. Se puede visualizar su contenido con el comando `echo $HOME`

### 8.3 Comando pwd

El comando pwd indica la **ruta absoluta** (ruta completa desde /) del directorio en el cual nos encontramos actualmente.

**Sintaxis:** pwd

**Ejemplo:**

```
$ pwd
```

### 8.4 Comando mkdir

El comando mkdir se utiliza para crear directorios.

**Sintaxis:** mkdir [opciones] nombre(S)\_directorio(s)

Opciones

- m MODE Crea un directorio con los permisos de acceso dados
- p Crea los directorios padres que no existan en las rutas especificadas en el nombre del directorio

**Ejemplos:**

```
$ mkdir bin
```

```
$ mkdir /bin
```

```
$ mkdir -p docs/linuxdocs/howtos/pdf
```

En este último ejemplo, se crean los directorios intermedios si es necesario.

### 8.5 Comando rmdir

Elimina los directorios vacíos que se especifican como parámetro.

**Sintaxis:** rmdir [opciones] nombre(S)\_directorio(s)

Opciones

- p Elimina también los directorios predecesores que estén vacíos

**Ejemplos:**

```
$ rmdir bin
```

```
$ rmdir /bin
```

```
$ rmdir -p docs/linuxdocs/howtos/pdf #se eliminan los directorios  
intermedios
```

## 8.6 Comando mv

El comando `mv` mueve un fichero hacia otro, o varios ficheros hacia un directorio. Este permite a su vez renombrar ficheros o directorios.

**Sintaxis:**

```
mv [opciones] <fuente> <destino>
```

```
mv [opciones] <ficheros> <directorio>
```

Algunas opciones:

- i ejecuta el comando de forma interactiva, o sea, pregunta ante de sobrescribir el destino si existiera.
- u actualiza (upgrade) el destino con la fuente solo si este es más reciente.

### Ejemplos:

```
$ mv mail.cf mail.cf.old # renombra un fichero
$ mv -i *.txt /tmp # mueve ficheros terminados en .txt al directorio /tmp
$ mv -u program.c src/ # actualiza el fichero destino si es menos reciente que el fuente
```

## 8.7 Comando cp

El comando `cp` permite copiar un fichero en otro, o varios ficheros en un directorio.

**Sintaxis:**

```
cp [opciones] <fuente> <destino>
```

```
cp [opciones] <ficheros> <directorio>
```

Algunas opciones:

- R copia recursivamente un directorio.
- i utiliza una forma interactiva (pregunta antes de sobrescribir el destino).
- l hace enlaces fuertes (duros) a los ficheros fuentes en lugar de copiarlos.
- s hace enlaces simbólicos a los ficheros fuentes en lugar de copiarlos.

### Ejemplos:

```
$ cp /etc/passwd . # copia un fichero en el directorio actual
$ cp -i /usr/bin/*sh /tmp # copia de forma interactiva (pregunta antes de sobrescribir) los ficheros terminados en sh en un directorio llamado /tmp
```

## 8.8 Comando rm

El comando rm se utiliza para borrar (desenlazar) ficheros

<b>Sintaxis:</b> rm [opciones] <ficheros   directorios>
---

Algunas opciones:

- r borra recursivamente un directorio.
- f borra forzosamente en caso de que no se tenga permiso de escritura en forma directa.
- i ejecuta el comando de forma interactiva.

### Ejemplos:

```
$ rm prueba
$ rm -i bin/*
$ rm -rf temp/
```

## 8.9 Comando chown/chgrp

El comando chown se utiliza para cambiar el dueño y el grupo de un fichero. Existe también el comando chgrp que se emplea de forma similar, pero para cambiar el grupo solamente. El dueño de un fichero solo lo puede cambiar el usuario root mientras que el grupo además de root, lo puede cambiar el propio dueño, siempre que pertenezca al nuevo grupo.

El propietario y el grupo de un fichero se puede comprobar con un ls -l

<b>Sintaxis:</b> chown [opciones] <dueño>[:grupo] <ficheros> chown [opciones] <dueño> <ficheros>
---

Opción:

- R en los directorios cambia el dueño y/o el grupo recursivamente.

### Ejemplos:

```
# chown pepe:pepe tesis/ # cambia el propietario de tesis a pepe y el grupo de tesis al grupo pepe
```

```
# chown -R root /tmp/oculto # cambia todos los ficheros que esten en el directorio oculto y coloca como propietario al usuario root
```

```
# chgrp ftp /usr/ftp
```

## 8.10 Otros comandos relacionados

### 8.10.1 Comando df

El comando nos informa del uso de disco en los distintos volúmenes. Así podemos saber cuánto espacio tenemos en cada volumen, y cuánto está usado y cuánto libre.

#### Ejemplo:

```
# df -h
```

### 8.10.2 Comando type

El comando type indica cómo interpreta el bash una orden. En caso de ser un comando mostraría el camino (path) de este, si fuera un alias, cuál es el comando original y para una función mostraría su código. Una misma palabra puede constituir un comando, un alias y una función. Bash interpreta con mayor prioridad la función, luego el alias y, por último, el comando.

<b>Sintaxis:</b> type [-all] [-path -type] [palabras]
---

#### Opciones:

-all	muestra todas las posibilidades de interpretación.
-path	muestra el camino completo al fichero que representa al comando, en caso de serlo.
-type	muestra la forma de interpretación asumida: alias, keyword, function, builtin y file.

#### Ejemplos:

```
$ type l
l is aliased to `ls -l -color'
```

```
$ type passwd
passwd is /usr/bin/passwd
```

```
# type -all rm
rm is aliased to `rm -i'
rm is /bin/rm
```

### 8.10.3 Comando stat

El comando stat muestra las características de un fichero. Por ejemplo: su nombre, permisos, tamaño en bytes, número del i-nodo que lo representa, las fechas de modificación y acceso, el tipo, el dueño, el grupo, etc.

#### Ejemplo:

```
$ stat /etc/shadow
```

### 8.10.4 Comando diff

El comando diff nos permite comparar ficheros de texto línea a línea, y nos indica en caso de que no sean iguales en qué líneas cambian. Dado que en Linux se realizan muchas configuraciones del sistema modificando ficheros de texto, es una orden interesante para saber si algo se ha modificado.

**Ejemplo:** # diff fichero1 fichero2

### 8.10.5 Comando du

El comando du permite conocer la longitud (expresada en kilobytes por defecto) de una jerarquía de ficheros a partir de un directorio.

**Sintaxis:** du [opciones] [ficheros | directorios]

Algunas opciones:

- h (human readable view) imprime las unidades de la forma más representativa (Ej. M para megabytes).
- s suma el tamaño de cada fichero/directorio sin profundizar recursivamente en estos últimos.
- c produce un total cuando se utilizan varios argumentos.

#### Ejemplos:

```
$ du
$ du -h *
$ du -hsc /usr /home
```

### 8.10.6 Comando file

Intenta clasificar los archivos de “lista-archivos”.

**Sintaxis:** file [opciones] lista-archivos

Opciones:

- f ARCHIVO Utiliza ARCHIVO como archivo de “lista-archivos”

#### Ejemplo:

```
$ file /*
```

### 8.10.7 Comando whereis

Devuelve el directorio donde se encuentra la orden que le pasamos como parámetro y la página correspondiente donde se encuentra en el manual. Se puede hacer una búsqueda más precisa utilizando parámetros.

**Sintaxis:** whereis [-bms] orden(es)

Opciones:

- m búsqueda de página de manual.
- s búsqueda de código fuente.

#### Ejemplo:

```
$ whereis vi Firefox
```

### 8.10.8 Comando which

Busca en los directorios especificados en la variable \$PATH del usuario el archivo que le especifiquemos. Como resultado visualiza en forma de camino absoluto el nombre del archivo. Si la búsqueda es infructuosa, nos avisa de ello.

**Sintaxis:** which lista-archivos

#### Ejemplo:

```
$ which vi Firefox
```

## 9 SUPERUSUARIO

### 9.1 Comando su

El comando `su` permite ejecutar un shell (u otro comando) cambiando los identificadores del grupo y del usuario actual. Si se le pasa - como primer argumento ejecuta el shell como un login shell, o sea, se creará un proceso de login tal y como ocurre naturalmente cuando un usuario se conecta al sistema. Si no se especifica el login del usuario, se asume root.

<b>Sintaxis:</b> <code>su [opciones] [usuario]</code>
---

#### Opciones

c comando	Ejecuta un comando con las credenciales de root.
s shell	Ejecuta el shell que le pasamos como parámetro siempre que se encuentre en el fichero /etc/shells

#### Ejemplos:

```
$ su -c "cat /etc/shadow" # ejecuta un comando con los privilegios de root (básicamente, realiza lo mismo que el comando sudo).
```

## 9.2 Comando sudo

En algunos sistemas, como puede ser Ubuntu, existe una orden denominada `sudo`, que viene a ser un `su -c` (es decir, permite ejecutar una orden como el usuario `root`).

<b>Sintaxis:</b> <code>sudo comando</code>
--

### Ejemplos:

```
$ sudo gedit /etc/shadow # ejecuta gedit /etc/shadow como si lo ejecutara el root.
```

```
$ sudo su # Nos transforma en el root hasta que escribamos exit.
```

```
$ sudo su - # Nos transforma en el root, y abre un nuevo shell para el root.
```

Se puede hacer una selección de qué usuarios pueden ejecutar qué cosas. El fichero de configuración es `/etc/sudoers`. Para modificar este fichero se puede utilizar el comando `visudo`. Simplemente hay que escribirlo en la línea de comandos (`sudo visudo`) y se abrirá el fichero `sudoers` que hay por defecto.

Es muy recomendable utilizar este comando para la edición de `sudoers` puesto que, cada vez que se ejecuta, además de lanzar el editor que hayamos configurado, bloqueará el archivo para evitar la edición concurrente y comprobará la sintaxis antes de proceder a guardar los cambios realizados.

En Ubuntu, en `sudoers` hay una línea que permite que cualquier usuario pueda ejecutar comandos con `sudo`:

```
%admin ALL=(ALL) ALL
```

Nota: para cambiar el editor por defecto: `sudo update-alternatives --config editor`

En otras distribuciones podemos encontrar que sólo el `root` puede ejecutar comandos de `root`. Si queremos que otros usuarios puedan utilizar `sudo`, deberemos introducir líneas como la segunda debajo de la que hace referencia a `root`:

```
root ALL=(ALL) ALL
```

```
nombre_usuario ALL=(ALL) ALL
```

La sintaxis es:

```
usuario lista_hosts = (lista_usuarios) lista_comandos
```

También podemos encontrar:

```
usuario lista_hosts = (lista_usuarios:lista_grupos) lista_comandos
```

En los ejemplos vistos, `ALL` significa sin restricciones.



## Artículo: Configurando sudo sin romperlo

Fuente: <http://www.ubuntu-es.org/node/1559>

Texto bajo licencia Creative Commons.

En Ubuntu todas las operaciones de administración que requieren el usuario root se hacen utilizando sudo (a no ser que activemos la cuenta root en cuyo caso ya podremos acceder con este usuario como en cualquier otra distribución), por eso puede ser crítico modificar erróneamente el archivo `/etc/sudoers` ya que puede dejarnos sin posibilidad de convertirnos en root para administrar nuestra máquina.

Una forma de cambiar la configuración sobre seguro es utilizar visudo que edita el archivo `/etc/sudoers` y antes de consolidar los cambios analiza la sintaxis del archivo detectando los posibles errores cometidos.

Voy a poner un par de ejemplos sobre la modificación con visudo.

1. Supongamos que tenemos creado (desde la instalación) el usuario `miusuario` y que hemos creado otro usuario (`otrousuario`). Queremos que el nuevo usuario sea también capaz de administrar el sistema.

- Editamos el fichero `sudoers` utilizando visudo

```
sudo visudo
```

- Con las flechas nos posicionamos en la línea donde aparece `miusuario`.

```
miusuario ALL=(ALL) ALL
```

- Cuando estamos sobre esta línea presionamos la tecla `Y` y a continuación la tecla `p` con lo que habremos duplicado la línea

```
miusuario ALL=(ALL) ALL
```

```
miusuario ALL=(ALL) ALL
```

- A continuación, nos ponemos sobre el principio del usuario (`miusuario`) en una de las dos líneas y presionamos las teclas `ce` y escribimos el otro usuario (`otrousuario`). Nos debería quedar algo como:

```
otrousuario ALL=(ALL) ALL
```

```
miusuario ALL=(ALL) ALL
```

- Grabamos el documento y salimos pulsando `<ESC>:wq<ENTER>`. Si hubiera algún error nos daría un aviso y no grabaría los cambios. Para salir sin grabar los cambios se haría con `<ESC>:q!<ENTER>`.

2. Queremos ejecutar un determinado comando (/usr/bin/micomando) sin necesidad de introducir nuestro password.

- Seguimos los pasos del ejemplo anterior para editar con visudo y duplicar la línea de nuestro usuario

```
miusuario ALL=(ALL) ALL
```

```
miusuario ALL=(ALL) ALL
```

- Nos posicionamos (con las flechas) en la segunda línea y dejamos el cursor sobre la letra A del último ALL. Tecleamos <ESC>C y escribimos NOPASSWD: /usr/bin/micomando pulsando <ESC> para salir del modo edición. Debería quedar de la siguiente forma

```
miusuario ALL=(ALL) ALL
```

```
miusuario ALL=(ALL) NOPASSWD: /usr/bin/micomando
```

- Grabamos el documento y salimos según lo indicado en el ejemplo anterior

Después de modificar (y grabar el archivo) entramos en un terminal y ejecutamos algún comando con sudo verificando que funciona como queremos. Si no nos da ningún error podemos cerrar la consola de root, si por el contrario tenemos algún error modificamos desde la consola de root (que no deberíamos haber cerrado) el archivo para corregir el fallo.

## 10 FICHEROS: INFORMACIÓN, PERMISOS, PROPIETARIO, GRUPO.

---

### 10.1 Permisos de lectura, escritura y ejecución

#### 10.1.1 Permiso de lectura

Cuando un usuario tiene **permiso de lectura de un archivo** significa que puede leerlo o visualizarlo, bien sea con una aplicación o mediante comandos. Ejemplo, si tenemos permiso de lectura sobre el archivo examen.txt, significa que podemos ver el contenido del archivo. Si el usuario no tiene permiso de lectura, no podrá ver el contenido del archivo.

Cuando un usuario tiene **permiso de lectura de una carpeta**, significa que puede visualizar el contenido de la carpeta, es decir, puede ver los archivos y carpetas que contiene.

#### 10.1.2 Permiso de escritura

Cuando un usuario tiene **permiso de escritura sobre un archivo** significa que puede modificar su contenido, e incluso borrarlo. También le da derecho a cambiar los permisos del archivo mediante el comando `chmod` así como cambiar su propietario y el grupo propietario mediante el comando `chown` (veremos estos comandos a continuación). Si el usuario no tiene permiso de escritura, no podrá modificar el contenido del archivo.

Cuando un usuario tiene **permiso de escritura sobre una carpeta**, significa que puede modificar el contenido de la carpeta, es decir, puede crear y eliminar archivos y otras carpetas dentro de ella. Si el usuario no tiene permiso de escritura sobre la carpeta, no podrá crear ni eliminar archivos ni carpetas dentro de ella.

#### 10.1.3 Permiso de ejecución

Cuando un usuario tiene **permiso de ejecución de un archivo** significa que puede ejecutarlo. Si el usuario no dispone de permiso de ejecución, no podrá ejecutarlo, aunque sea una aplicación.

Los únicos archivos ejecutables son las aplicaciones y los archivos de comandos (scripts). Si tratamos de ejecutar un archivo no ejecutable, dará errores.

Cuando un usuario tiene permiso de ejecución sobre una carpeta, significa que puede entrar en ella.

## 10.2 Información sobre un fichero

Cuando se obtiene información sobre un fichero/directorio con el comando `ls`, existen diferentes campos que indican qué clase de permisos tiene el fichero/directorio.

Ejemplo:

```
[user@localhost]# ls -l  
  
-rwxr-x--- 1 pepito depart1 4348 Nov 24 16:19 test
```

En la primera columna se pueden ver una serie de letras y guiones `-rwxr-x---`, estas letras nos dicen quién en el sistema, y qué clases de permisos tiene el fichero `test`.

Estas letras están agrupadas en tres grupos con tres posiciones cada uno, más una primera posición que nos dice de qué clase de archivo se trata (los más normales (d) directorios, o (-) archivos de datos). En nuestro ejemplo la primera posición es (-) con lo cual el archivo `test`, es un archivo de datos (binario/ejecutable en este ejemplo).

El primer grupo de tres (`rw` en nuestro caso) nos dice qué clase de permisos tiene el dueño del fichero (u) (user/owner)

El segundo grupo de tres (`r-x` en nuestro caso) nos dice qué clase de permisos tiene el grupo del fichero (g) (group).

Y el último grupo de tres (`---` en nuestro caso) nos dice qué clase de permisos tienen todos los demás usuarios del sistema sobre este fichero (o) (others).

```
r :significa permiso para leer  
w :significa permiso para escribir  
x :significa permiso para ejecutar
```

La segunda columna es un número que normalmente indica el número de enlaces duros y blandos, así como entradas de directorio. Podemos encontrar diferencias entre las diferentes plataformas acerca del significado de este “link”.

La tercera columna `pepito`, nos dice quién es el dueño del fichero, (pepito en este caso).

La cuarta columna `depart1`, nos dice cuál es el grupo del fichero (depart1 en este caso).

La quinta columna `4348`, nos dice el tamaño del fichero.

La sexta columna `Nov 24 16:19`, nos dice cuál es la fecha y hora de la última modificación.

La séptima columna `test`, nos dice cuál es el nombre del fichero/directorio.

(se puede obtener información detallada de la información dada por `ls` con `info coreutils 'ls invocation'` )

Así pues, el fichero `test` de nuestro ejemplo tiene los siguientes permisos:

- `pepito` puede leer, escribir/modificar, y ejecutar el fichero `test`.
- Los usuarios pertenecientes al grupo `depart1` puede leer, y ejecutar, pero no escribir/modificar.
- Los demás usuarios no pueden hacer nada, ni leerlo, ni escribir/modificar, ni ejecutarlo.

### 10.3 Cambiar permisos de un fichero (`chmod`)

Para cambiar el **dueño** del fichero se utiliza el comando:

```
chown usuario fichero
```

Para cambiar el **grupo** del fichero se utiliza el comando:

```
chgrp grupo fichero
```

Para cambiar los **permisos** se utiliza el comando:

```
chmod permisos fichero
```

Los permisos se pueden especificar de diferentes maneras. Una forma es utilizar la notación `ugo+/-/=rwx`. `u` hace referencia al usuario propietario, `g` al grupo y `o` a otros. `+` añade los permisos especificados, `-` quita permisos y el signo `=` sirve para otorgar los permisos que se especifican y quitar los que no se especifican. Algunos ejemplos:

```
chmod ugo+rwx test (da permisos rwx a todos, user,group,others)

chmod ugo-x test (quita permiso x (ejecución) a todos, user,group,others)

chmod o-rwx test (quita permisos rwx a others)

chmod u=rwx,g=rx test (da permisos rwx a user, rx a group y no hace nada con others)

chmod a+wx test (da permisos wx a todos)

chmod +x test (da permisos x a todos, equivalentes a a+x)
```

Existe otro método que utiliza números, en vez de letras para asignar permisos, la siguiente tabla nos puede ayudar a comprender esta manera:

r	w	x	VALOR OCTAL
0	0	0	0
0	0	1	1
0	1	0	2
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7

1 significa activado y 0 desactivado, o sea 101, activa r y x, y desactiva w. Sabiendo esto, solo tenemos que usar el valor decimal para dar solo permisos de lectura y ejecución, veamos un ejemplo:

```
chmod 750 test  
  
- da permisos rwx al usuario (7=111)  
- da permisos r-x al grupo (5=101)  
- da permisos --- a los demás (0=000)
```

**Más información:**

<https://e-mc2.net/es/como-se-cambian-los-permisos-de-ficheros-y-directorios-en-linux>

[http://www.ite.educacion.es/formacion/materiales/85/cd/linux/m1/permisos\\_de\\_archivos\\_y\\_carpetas.html](http://www.ite.educacion.es/formacion/materiales/85/cd/linux/m1/permisos_de_archivos_y_carpetas.html)

## 11 USUARIOS Y GRUPOS

### 11.1 Comandos para gestión de cuentas

- Crear un usuario: `# useradd nombre_del_usuario`
- Cambiar contraseña: `# passwd nombre_del_usuario`
- Modificar usuario: `# usermod nombre_del_usuario`
- Borrar usuario: `# userdel nombre_del_usuario`
- Crear grupo: `# groupadd nombre_del_grupo`
- Modificar grupo: `# groupmod nombre_del_grupo`
- Borrar grupo: `# groupdel nombre_del_grupo`

#### 11.1.1 Comando useradd

Este comando crea una nueva cuenta de usuario en el sistema.

**Sintaxis:**

```
# useradd --showdefaults # muestra los valores por defecto
que se encuentran en el fichero /etc/default/useradd.
También se puede utilizar el parámetro -D
```

```
# useradd [opciones] usuario
```

Los parámetros de la segunda sintaxis son los siguientes:

-c comment	comentario de campo para la nueva cuenta.
-d home dir	directorio de trabajo del usuario.
-e expire date	fecha de expiración de una cuenta de usuario (AAAA-MM-DD)
-g initial group	grupo inicial al que pertenecerá el usuario.
-G group,[...]	grupos adicionales a los que pertenecerá el usuario.
-m	el directorio de trabajo del usuario debe ser creado si no existiese, y se copiarán dentro de este los archivos especificados en /etc/skel.
-s shell	Shell por defecto del usuario
-u uid	ID del usuario. Este debe ser único.

### Ejemplos:

```
# useradd -D #muestra las propiedades por defecto de los nuevos usuarios
# useradd pedro -G amigos #crea el usuario pedro y lo añade al grupo
suplementario amigos
# useradd nombre_usuario # crea un usuario con los parámetros por defecto
(/etc/default/useradd)
# useradd -s /bin/tcsh -d /home/pepito -c "Pepe Martínez" -m pepe
```

ATENCIÓN: observa que al crear usuarios con el comando `useradd`:

- no se solicita la creación de contraseña, por lo que el usuario no podrá iniciar sesión si no se le asigna una. Esto se hará con el comando `passwd`, descrito a continuación.
- si no se añade la opción `-m`, no se crea el directorio personal del usuario (esto impide que se inicie sesión en el entorno gráfico para ese usuario).

Existe otro comando de creación de usuarios, `adduser`, que por defecto solicita contraseña (ejecuta `passwd`) y crea el directorio personal del usuario.

#### 11.1.2 Comando `passwd`

El comando `passwd` permite cambiar el password de un usuario. También puede bloquear, desbloquear y deshabilitar una cuenta. Si se invoca sin argumentos se asume el usuario actual.

**Sintaxis:** `passwd [opciones] [login_user]`

### Ejemplos:

```
$ passwd # cambia la contraseña de mi usuario
# passwd pepe # coloca una contraseña nueva para pepe
# passwd -d pepe # deshabilita la cuenta del usuario pepe eliminando su
password del fichero /etc/shadow
# passwd -l pepe # bloquea la cuenta del usuario pepe poniendo un signo
! delante de su password en el fichero /etc/shadow
# passwd -u pepe # desbloquea la cuenta del usuario pepe
# passwd -S pepe # muestra información sobre el estado de la contraseña
```

#### 11.1.3 Comando `chage`

El comando `chage` nos permita gestionar la información sobre la caducidad de las contraseñas.

**Sintaxis:** `chage [opciones] usuario`

### Ejemplo (más ejemplos en el apartado 11.2.2):

```
# chage pepe
```



#### 11.1.4 Comando usermod

El comando usermod se emplea para modificar algunas propiedades de los usuarios como: el login, el directorio base, el shell que se inicia al conectarse, los grupos a los que pertenece, la fecha de expiración de la cuenta, etc. También bloquea y desbloquea una cuenta. Utiliza los mismos parámetros que useradd.

**Sintaxis:** usermod [opciones] <login>

##### Ejemplos:

```
# usermod -s /bin/csh pepe # coloca el shell csh para el usuario pepe
# usermod -a -G users,disk pepe # señala como grupos secundarios de pepe
a users y disk. ATENCIÓN: CON LA OPCIÓN -a SE AÑADEN LOS GRUPOS. SI NO
SE PONE, SE BORRAN LOS GRUPOS SECUNDARIOS QUE HUBIERA.
# usermod -e 2010-10-20 pepe # indica que la cuenta de pepe expirará el
20 de octubre del 2010
```

#### 11.1.5 Comando userdel

El comando userdel permite eliminar definitivamente un usuario del sistema.

**Sintaxis:** userdel [opciones] <login>

Opciones:

- r elimina archivos y directorios del HOME de ese usuario. Por defecto si este parámetro no se indica, se borra el usuario, pero no su directorio de trabajo.

El usuario no puede tener iniciada sesión para poder eliminarlo.

##### Ejemplo:

```
# userdel -r pepe # elimina al usuario pepe y borra su directorio base.
```

#### 11.1.6 Comando groupadd

Permite crear grupos.

**Sintaxis:** groupadd [opciones] grupo

##### Ejemplo:

```
# groupadd alumnos
```

#### 11.1.7 Comando groupmod

Permite modificar el nombre y el identificador de un grupo.

**Sintaxis:** groupmod [opciones] grupo

##### Ejemplo:

```
# groupmod -n estudiantes alumnos
```

### 11.1.8 Comando groupdel

El comando `groupdel` permite eliminar definitivamente un grupo del sistema.

**Sintaxis:** `groupdel [opciones] grupo`

#### Ejemplo:

```
# groupdel estudiantes
```

### 11.1.9 Comando gpasswd

El comando `gpasswd` permite administrar los grupos. Se puede utilizar para añadir y eliminar usuarios, señalar un administrador e indicar un password de grupo.

**Sintaxis:** `gpasswd [opciones] <grupo>`

#### Ejemplos:

```
# gpasswd -A pepe admin # señala como administrador del grupo admin al usuario pepe
```

```
$ gpasswd admin # cambia el passwd del grupo admin
```

```
$ gpasswd -a joe admin # añade el usuario joe al grupo admin
```

### 11.1.10 Comandos adicionales para gestionar usuarios

#### 11.1.10.1 Comando chsh

La orden `chsh` (*change shell*) se utiliza para cambiar el intérprete de comandos por defecto del usuario que actualmente tiene iniciada sesión. Para que un usuario pueda cambiar un shell es necesario que se encuentre dentro del fichero `/etc/shells`.

#### Ejemplo:

```
$ chsh
```

```
Changing login shell for alumno.
```

```
Contraseña:
```

```
Enter the new value, or press return for the default.
```

```
Login Shell [/bin/bash]: /bin/tcsh
```

```
Shell changed.
```

```
# chsh alumno
```

```
Changing login shell for alumno.
```

```
Enter the new value, or press return for the default.
```

```
Login Shell [/bin/tcsh]: /bin/bash
```

```
Shell changed.
```

#### 11.1.10.2 Comando chfn

Permite cambiar la información de contacto de un usuario. Esta incluye aspectos como: el nombre completo, la oficina de trabajo y los teléfonos. Se

almacena en el archivo de usuarios `/etc/passwd` en la sección de comentarios.

### Ejemplo:

```
#chfn pepe
```

Entre corchetes aparecerá la información actual. Si no se desea cambiar, pulsar intro.

#### 11.1.10.3 Comando *newgrp*

Por defecto, cuando iniciamos sesión con un usuario, pertenece al grupo con `gid` que indica el fichero `/etc/passwd`. Cualquier fichero que creamos pertenecerá a ese grupo a no ser que se lo cambiemos de forma manual con `chgrp`. Si deseamos que cualquier fichero que creamos de aquí hasta que cerremos sesión se cree para otro grupo podemos utilizar el comando `newgrp`.

### Ejemplo:

```
$ groups
dialout video users
$ newgrp dialout
```

#### 11.1.10.4 Otros

Comando	Descripción
# <code>whoami</code>	# nos muestra qué usuario somos
# <code>groups [usuario]</code>	# muestra los grupos a los que pertenece un usuario
# <code>id [usuario]</code>	# muestra el <i>uid</i> , <i>gid</i> del usuario del fichero <code>/etc/passwd</code> y los <i>gid</i> de los grupos a los que pertenece. Se puede utilizar con los parámetros <code>-u</code> , <code>-g</code> y <code>-G</code> respectivamente.
# <code>su [usuario]</code>	# sirve para convertirnos en otro usuario sin tener que salir de la sesión, así como para cambiar a ser un superusuario
# <code>who</code> # <code>users</code>	# muestra la lista de usuarios conectados al sistema
# <code>w [usuario]</code>	# muestra la lista de usuarios conectados del sistema y también lo que están haciendo
# <code>write [usuario]</code>	# comando que se utiliza para comunicarse entre los usuarios del sistema (para poder enviar mensajes tiene que estar activada la opción (ver comando <code>mesg</code> ))
# <code>wall [mensaje]</code>	# permite enviar un mensaje a todos los terminales de los usuarios dentro del sistema. [mensaje] es el nombre de un fichero de texto que contiene el mensaje.
# <code>mesg [y n]</code>	#permite activar o desactivar la opción de recibir mensajes
# <code>last</code>	# listado cronológico de los usuarios que han iniciado sesión en nuestro sistema.

## 11.2 Ficheros de configuración de usuarios y grupos

- `/etc/passwd`: contiene una lista de los usuarios y sus características
- `/etc/shadow`: contiene las contraseñas encriptadas de los usuarios
- `/etc/groups` o `/etc/group`: contiene una lista de los grupos
- `/etc/gshadow`: contraseñas cifradas de los grupos (normalmente no se usa este fichero)

### 11.2.1 `/etc/passwd`

En el fichero `/etc/passwd` tiene una entrada para cada uno de los usuarios que componen el sistema.

```
usuario1:x:500:501:usuario pepito:/home/usuario1:/bin/bash
```

Se trata de siete campos separados por dos puntos (:):

Campo	Descripción
usuario1	Nombre de la cuenta con el que se inicia sesión en el sistema (login)
x	Antiguamente este campo se utilizaba para almacenar la contraseña. En la actualidad se utiliza el fichero <code>/etc/shadow</code>
500	UID de esta cuenta (número identificador de usuario)
501	GID del grupo principal al que pertenece la cuenta
usuario pepito	Campo de comentarios sobre el usuario
/home/usuario1	Dirección absoluta del directorio de trabajo del usuario
/bin/bash	Dirección absoluta del intérprete de comando (shell) del usuario. <ul style="list-style-type: none"><li>- el programa que se debe ejecutar para ofrecerle un terminal de acceso al sistema a este usuario, normalmente es el bash ya que es el shell que utilizamos por defecto. Si no queremos que un usuario pueda tener acceso a estos terminales, bastaría con indicarle aquí un nombre falso, como por ejemplo <code>/bin/false</code> y de este modo este usuario no podría interactuar con el sistema aunque realizara correctamente el login.</li></ul>

Una serie de reglas a tener en cuenta sobre el contenido de este fichero:

- Un UID con valor 0 se reserva a root
- Los UID reservados para los usuarios suelen estar por encima de 1000 en adelante (en otras distribuciones lo hacen desde 100 o desde 500).
- Un GUID con valor 0 se reserva para root
- El GID del grupo principal está definido en el archivo `/etc/group` y este será el grupo por defecto cuando un usuario crea un fichero.

- Si ponemos `/bin/false` como intérprete de comando el usuario no podrá iniciar sesión.
- Con `chsh -l` se muestra un listado de los shells instalados en nuestro sistema operativo que se encuentran en un listado en el fichero `/etc/shells`

### 11.2.2 `/etc/shadow`

El fichero `/etc/passwd` debe tener permisos de lectura para todos y permisos de escritura solo para root. Con estos permisos, cualquiera que tenga acceso al sistema puede leer el contenido de estos ficheros e intentar descifrar la clave encriptada de las cuentas.

Para evitar esto se utiliza el fichero `/etc/shadow` para guardar las claves encriptadas, pero sólo root puede acceder a esa información.

El formato del fichero `/etc/shadow` es similar al siguiente:

`usuario:clave:último:mínimo:máximo:aviso:inactiva:expira:reservado`

Campo	Descripción
usuario	El nombre del usuario que debe coincide con el primer campo del fichero <code>/etc/shadow</code> .
clave	<p>La contraseña del usuario encriptada.</p> <p>Esta contraseña hoy en día se representa en tres partes distintas, separadas por el símbolo del dólar (\$).</p> <ul style="list-style-type: none"> <li>○ Id. Nos indica el método de encriptación que fue utilizado. Un valor de 1 indica MD5, un valor de 2 Blowfish, 3 es NT Hash, 5 es SHA-256 y 6 es SHA-512.</li> <li>○ Salt. Este valor se usa por los algoritmos de encriptación y puede ser de hasta 16 caracteres. Este valor se crea aleatoriamente en cada generación de contraseña.</li> <li>○ Hash. La contraseña en sí misma. MD5 usa 22 caracteres, SHA-256 usa 43, y SHA-512 usa 86.</li> </ul> <p><b>*</b> = password deshabilitado  <b>!</b> = password deshabilitado  <b>otra cosa</b> (de 13 a 24 caracteres) = password encriptado  <b>nada</b> (ni siquiera un espacio) o dos exclamaciones seguidas (!!) = cuenta sin password (no se pide password)</p>
último	Días transcurridos del último cambio de clave desde el día 1/1/1970.
mínimo	Días transcurridos antes de que la clave se pueda modificar, es decir, el número de días que deben transcurrir antes de que se le permita cambiar la contraseña al usuario
máximo	El número máximo de días en el que la contraseña es válida (una vez llegado el usuario es forzado a cambiar la contraseña).

aviso	Antes de que la contraseña caduque, avisaremos al usuario al llegar a este número de días.
inactiva	Una vez que la contraseña caduque esperaremos este número de días antes de bloquearla definitivamente.
expira	Es una fecha, en la cual la cuenta automáticamente quedará bloqueada. (Este número indica los días que han transcurrido desde el 1-1-1970).
reservado	Campo reservado.

Un ejemplo de una línea del fichero /etc/shadow podría ser:

```
julia:gEvm4sbKnGRlg:10627:0:99999:7:-1:-1:14375
```

#### Nota:

Para saber el valor 14375 a qué fecha corresponde podríamos hacer el siguiente cálculo:

14375 días = 14375 \* 24 \* 60 \* 60 segundos = 1242000000 segundos.

Este valor en formato UNIX lo convertiremos en formato legible mediante:

```
# date -d"1970/1/1 1242000000 sec"
```

Si queremos cambiar o mostrar estos valores de forma legible para un usuario podemos utilizar el comando `chage`.

Tiene dos sintaxis:

- Para hacer cambios:

```
# chage [-m mínimo] [-M máximo] [-d último] [-I inactiva]
[-E expira] [-W aviso] usuario
```

- Para consultar:

```
# chage -l usuario
```

Si no se pone ningún parámetro sino sólo el nombre del usuario, se hace de forma interactiva cada uno de los campos.

Con el parámetro `-l` se muestra, la información del usuario.

### Ejemplo:

```
# chage -m 5 -M 30 -W 7 -I 3 -E 2009-12-31 alumno
```

Aging information changed.

```
# chage -l alumno
```

Minimum: 5

Maximum: 30

Warning: 7

Inactive: 3

Last Change: May 11, 2009

Password Expires: Jun 10, 2009

Password Inactive: Jun 13, 2009

Account Expires: Dec 31, 2009

```
# more /etc/shadow
```

```
alumno:$2a$05$ZyWz/KbT3MzQBdE4omVMA:14375:5:30:7:3:14609:
```

### 11.2.3 /etc/group

Cada una de las líneas representa un grupo y responde al esquema:

```
grupo:x:GID:lista_usuarios
```

Campo	Descripción
grupo	es el nombre del grupo
x	contraseña: aparece una x; la contraseña se encuentra cifrada en /etc/gshadow. Si este campo aparece vacío, significa que el grupo no necesita contraseña.
GID	Identificador numérico del grupo (el cero se reserva para el grupo root)
lista_usuarios	lista de usuarios que pertenecen al grupo separados por comas. Lo habitual es poner sólo los usuarios cuyo gid del fichero /etc/passwd no es éste.

En principio no se suele poner contraseña a los grupos, pero si se deseara hacer por cualquier motivo de seguridad se puede hacer con el comando `gpasswd`. Sólo el administrador puede cambiar la contraseña de un grupo. La contraseña se puede almacenar en el fichero `/etc/gshadow` o en el segundo campo de este fichero (el marcado con una x).

## 12 VARIOS COMANDOS

### 12.1 Fecha y estadísticas

#### 12.1.1 Comando date

El comando date sirve para consultar y cambiar la fecha y hora del sistema. La forma más habitual de ejecutar este comando es mediante:

```
$ date
```

El comando date tiene bastantes opciones. Es recomendable revisar sus páginas de manual para entenderlos bien. Existen distintas opciones de la orden date que afectan al formato de salida. Colocando un campo determinado a continuación del operador %, precedido por el signo +, podemos obtener respuestas con el formato que deseemos.

Parámetro	Significado
x	Fecha en formato local (30/12/09)
d	Día del mes con dos dígitos (06)
e	Día del mes con un dígito (6)
m	Mes en formato numérico (05)
b	Mes en formato abreviado (may)
B	Mes en formato largo (mayo)
y	Año, dos últimos dígitos (09)
Y	Año, todos los dígitos (2009)
w	Día de la semana en formato numérico (0..6) 0 es domingo
u	Día de la semana (1..7) 1 es lunes
a	Día de la semana en formato abreviado (mié)
A	Día de la semana en formato largo (miércoles)
r	Hora en formato AM-PM (11:17:24 PM)
H	Hora en formato 0..23 (24 H)
I	Hora en formato 01-12
M	Minuto
S	Segundo
s	Segundos desde 1970-01-01 00:00:00 UTC
t	tabulación
n	Nueva línea



Cuando tengamos un campo de texto podemos ponerlo en mayúsculas si anteponemos el carácter `^` delante. Si saliera en mayúsculas el valor y quisiéramos ponerlo en minúsculas antepondríamos el carácter `#`.

### Ejemplos:

```
$ date +"Hoy es %A"
```

```
Hoy es miércoles
```

```
$ date +"Hoy es %^A"
```

```
Hoy es MIÉRCOLES
```

```
$ date +"La hora actual es %T"
```

```
La hora actual es 20:02:42
```

```
$ date +"La hora actual es %H:%M:%S"
```

```
La hora actual es 20:02:42
```

```
$ date +"%A, %-d de %B del %Y"
```

```
miércoles, 6 de mayo del 2017
```

```
$ date +"%A, %e de %B del %Y"
```

```
miércoles, 6 de mayo del 2017
```

```
$ date +"Son las %H:%M%nHoy es %x"
```

```
Son las 20:02:42
```

```
Hoy es 06/05/17
```

El comando `date` también nos permite obtener la hora utilizando un formato natural. Puede utilizarse el formato `--date=""` o `-d""`.

### Ejemplos:

```
$ date --date="today" # nos da la fecha de hoy
```

```
$ date --date="2 days ago" # nos da la fecha de hace dos días
```

```
$ date --date="3 months 1 day" # nos da la fecha que será en 3 meses y un día.
```

```
$ date -d"next month" # nos da la fecha del próximo mes.
```

```
$ date --date="25 Dec" # nos da el día que será el 25 de Diciembre del año actual
```

```
$ date -d"2017-05-06" # nos da el día del 6 de mayo del 2017
```

Pueden resultar útiles los siguientes comandos:

```
$ date +%s # Convierte la hora actual en formato Unix, número de segundos transcurridos desde el 1/1/1970 a las 00:00:00 UTC.
```

`date` también se puede utilizar para convertir fechas (en el ejemplo, 6 de mayo del 2017) en formato Unix. Veremos una posible aplicación para el fichero `/etc/shadow` en un apartado posterior.

```
$ date -d"2017/05/06 1:00" +%s
```

```
1241564400
```

Si deseamos realizar el caso contrario, una fecha UNIX (en nuestro ejemplo 1241634781 segundos) pasarla a formato legible. Para ello hay que poner a la derecha el formato que deseamos.

```
$ date -date='@1491634781'  
sáb abr 8 08:59:41 CEST 2017
```

Otra forma de obtener lo mismo:

```
$ date -d"1970/01/01 1491634781 sec"  
sáb abr 8 08:59:41 CEST 2017
```

Y si queremos cambiar el formato de fecha obtenido:

```
$ date -d"1970/01/01 1491634781 sec" +"%d/%B/%Y %H:%M:%S"  
08/abril/2017 08:59:41
```

Otra utilidad del comando date es para actualizar la fecha y la hora del sistema. Obviamente hemos de ser administradores. Para ello se usa el parámetro `--set="" 0 -s""`.

```
# date --set="08/abril/2017 07:59:41 AM" # ajusta la fecha del sistema  
# date --set="+3 minutes" # adelante la hora del sistema en 3 minutos.  
# date --set="14:20:58" # Ajusta la hora  
# date --set="14 apr 2017" # Ajusta la fecha (fíjate en que el mes está  
en formato inglés. Si no se especifica una hora se pone las 0:00).
```

Este comando está ligado a la variable LANG y según el idioma que tengamos definido mostrará el idioma en un lenguaje u otro. Si queremos saber todos los idiomas que soporta Linux podemos ejecutar el comando

```
$ locale -a
```

### Ejemplos:

```
$ LANG=fr_FR  
$ date +%B  
avril
```

```
$ LANG=zu_ZA  
$ date +%B  
uMbasa
```

```
$ LANG=en_UK  
$ date +%B  
April
```

```
$ LANG=it_IT  
$ date +%B  
aprile
```

### 12.1.2 Comando cal

Muestra un calendario. Si no se le asigna ningún parámetro, muestra un calendario del año actual.

<b>Sintaxis:</b> cal [[[día] mes] año]
--

Si se indica un solo parámetro se muestra un calendario de ese año. Si se ponen dos, el primero es un mes y el siguiente un año.

#### Ejemplos:

```
$ cal # calendario del año en curso
$ cal 2017 # calendario del año 2017
$ cal 5 2017 # calendario de mayo de 2017
```

### 12.1.3 Comando time

El comando time sirve para cronometrar cuanto tiempo tarda en ejecutarse un comando cualquiera, y que recursos consume. Puede ser una orden interesante para comprobar el rendimiento de los sistemas.

#### Ejemplo:

```
# time cat /etc/passwd
```

### 12.1.4 Comando uptime

El comando uptime nos indica el “uptime” del sistema. En informática se conoce como uptime el tiempo que un equipo lleva “levantado”, es decir cuánto tiempo lleva el sistema funcionando. En un servidor es muy importante que este uptime sea lo más elevado posible.

El comando uptime nos devuelve cuánto tiempo lleva el equipo levantado, el número de usuarios que están conectados, y la media de las cargas de sistema para los últimos 1, 5, y 15 minutos.

```
# uptime
10:56PM funcionando 5:49, 1 user, carga promedio: 0,20, 0,17, 0,07
```

## 12.2 Información de la memoria

### 12.2.1 Comando top

Si se están buscando procesos que acaparan los ciclos de la CPU, ps (que veremos más adelante) no es la mejor opción, ya que, como simplemente muestra una instantánea del estado actual, no se conseguirá obtener demasiada información acerca de la carga del sistema actual. Para ayudar en esta tarea Linux dispone de la herramienta top. Top es un controlador de procesos que actualiza la presentación para proporcionar el estado actual.

El programa proporciona una extensa información sobre el sistema y los procesos que corren en él. La línea superior muestra la hora, el tiempo que lleva la máquina conectada, el número de procesos y los detalles del estado junto con la CPU, memoria y la carga de la memoria de intercambio.

La línea de estado top contiene información sobre los procesos individuales. La línea de estado posee columnas para los procesos ID (PID), el uso de memoria en formato de porcentaje (%MEM), el proceso ID padre (PPID), el tiempo consumido por la CPU como porcentaje (%CPU) y el nombre del comando (COMMAND). Pero puede decirse a top lo que se desea ver pulsando las siguientes teclas:

Tecla	Acción
<H>	Ayuda en línea
<F>	Seleccionar las columnas a mostrar. Con las letras adecuadas se puede especificar el contenido de la línea de estado.
<U>	seguido del nombre de usuario presenta los procesos para ese usuario
<Shift-R>	invierte la salida mostrando los procesos más frugales en lugar de los acaparan a la CPU
<Shift-Z>	Se podrá cambiar el aspecto del programa. Las teclas <W> y <A> permiten elegir entre un número de esquemas de color predefinidos, aunque también pueden pulsarse las letras y números apropiados que permitan definir un esquema de color personalizado.
<1>	Si tenemos más de un procesador, desglosa la carga de los mismos.
<W>	Guarda las preferencias actuales de top en el perfil del usuario (~/.toprc)
<Q>	se sale del programa y se regresa al shell

Campos que se muestran en top:

- **PID:** Identificador del proceso.
- **USER:** Usuario que ha ejecutado el proceso.
- **PR:** Prioridad del proceso en el sistema.
- **NI:** Valor nice del proceso (si son negativos; tienen mayor prioridad, si son positivos; menos prioridad).
- **RES:** Memoria RAM ocupada por el proceso.
- **%CPU:** Porcentaje ocupado de la CPU
- **TIME+:** Cuanto tiempo lleva el proceso en el sistema.
- **COMMAND:** Nombre del proceso y sus parametros.
- **PPID:** PID del proceso padre.
- **UID:** ID del usuario que ha ejecutado el proceso.

### 12.2.2 Comando free

El comando free nos informa del consumo de memoria del sistema, tanto en memoria real como en memoria virtual (swap).

**Sintaxis:** free [opciones]

#### Opciones

-b,-k,-m,-g	Muestra la salida en bytes, KB, MB, or GB
-t	Muestra el total de RAM + swap
-s	Actualiza cada número de segundos que le indiquemos
-c	Utilizado con el parámetro anterior, actualiza el número de veces que le indiquemos.

#### Ejemplos:

**\$ free**

```
total used free shared buffers cached
Mem: 247972 233188 14784 0 3384 86840
-/+ buffers/cache: 142964 105008
Swap: 506008 29984 476024
```

**\$ free -t**

```
total used free shared buffers cached
Mem: 247972 233188 14784 0 3384 86840
-/+ buffers/cache: 142964 105008
Swap: 506008 29984 476024
Total: 753980 263172 490808
```

**\$free -s 20 -c 5** # Actualiza la información cada 20 segundos 5 veces

## 12.3 Otra información del sistema

### 12.3.1 Comando uname

El comando uname nos da información sobre el sistema, como el nombre del kernel, su versión, el nombre de la máquina, etc.

**Sintaxis:** uname [opciones]

#### Opciones:

-a	Todo acerca de la máquina que estamos utilizando
-m	Tipo de procesador que estamos utilizando.
-n	Nombre de la máquina.
-r	Versión del kernel de Linux.
-s	Nombre del Sistema Operativo.
-p	Fabricante del procesador.
-v	Fecha del kernel

## Ejemplos:

```
$ uname -a
Linux pepito 2.6.27.19-3.2-pae #1 SMP 2009-02-25 15:40:44 +0100 i686
i686 i386 GNU/Linux
$ uname -m
i686
$ uname -n
pepito
$ uname -r
2.6.27.19-3.2-pae
$ uname -s
Linux
$ uname -p
i686
$ uname -v
#1 SMP 2009-02-25 15:40:44 +0100
```

### 12.3.2 Comando who

El comando `who` muestra los usuarios conectados al sistema ya sea local o remotamente.

<b>Sintaxis:</b> <code>who [opciones] [fichero] [am i]</code>
---

Sin argumentos `who` muestra los logins de los usuarios conectados, a través de qué terminal lo han hecho y en qué fecha y hora.

Algunas opciones:

-H	imprime un encabezamiento para las columnas.
-w	indica si está activada o no la posibilidad de recibir mensajes por parte de cada usuario conectado (+ indica que sí, - que no y ?, desconocido).
-q	sólo muestra los logins de los usuarios conectados y la cantidad total de ellos.

## Ejemplos:

```
$ who
$ who am i # equivalente al comando whoami o who am I
```

### 12.3.3 Comando w

El comando `w` muestra también los usuarios conectados al sistema además de lo que están haciendo (proceso que ejecutan en ese momento) y otras informaciones.

<b>Sintaxis:</b> <code>w [opciones] [usuario]</code>
--

Sin argumentos este comando muestra una primera línea con:

- la hora en que arrancó el sistema
- cuánto tiempo lleva funcionando
- cuántos usuarios hay conectados (sin incluir las sesiones gráficas)
- tres porcentajes de carga de la CPU: durante el último, los 5 y los 15 minutos anteriores.

A continuación de esa información se muestra una tabla cuyas columnas representan:

- el login de cada usuario conectado
- a través de qué terminal lo ha hecho, desde qué host
- a qué hora
- el tiempo de inactividad (idle time)
- la cantidad de segundos de CPU que han empleado todos los procesos que ha ejecutado ese usuario (JCPU)
- el tiempo (PCPU)
- nombre del comando que ejecuta actualmente.

#### Ejemplo:

```
$ w
```

### 12.3.4 Comandos `hostname` y `dnsdomainname`

El comando `hostname` nos devuelve el nombre de equipo (host) y nos permite cambiarlo.

El comando `dnsdomainname` hace lo mismo, pero para un nombre FQDN (Fully Qualified Domain Name).

#### Ejemplos:

```
# hostname
# dnsdomainname
# hostname pepito # Hace que el nombre de equipo pase a ser pepito.
```

Hay que tener mucho cuidado al cambiar el nombre de equipo, ya que es muy posible que tengamos programas y comandos que están preparados para

trabajar y encontrar nuestro equipo con el nombre actual (sudo por ejemplo). Nos podemos encontrar que estos programas funcionen mal si cambiamos el nombre del equipo. El nombre del equipo se almacena en el fichero `/etc/hostname`.

Para que funcione tanto el nombre de la máquina como el FQDN es necesario además añadir una línea en el fichero `/etc/hosts` donde se especifique lo siguiente:

```
127.0.0.1 pepito.oficina.local pepito
```

Hasta que no se reinicia la máquina no tienen efecto los cambios.

## 13 BIBLIOGRAFÍA

---

- Introducción a Linux: [https://es.wikibooks.org/wiki/Introducci%C3%B3n\\_a\\_Linux](https://es.wikibooks.org/wiki/Introducci%C3%B3n_a_Linux)
- Comandos: <https://www.guia-ubuntu.com/index.php?title=Comandos>
- El shell: <https://www.guia-ubuntu.com/index.php/Terminal>



## 14 EJERCICIOS

### 14.1 Distribuciones de GNU/Linux

1. ¿Qué es una distribución GNU/Linux?
2. A partir del enlace siguiente, indica qué tres grandes distribuciones se desarrollaron en un primer momento y en qué años:  
[https://upload.wikimedia.org/wikipedia/commons/1/1b/Linux\\_Distribution\\_Timeline.svg](https://upload.wikimedia.org/wikipedia/commons/1/1b/Linux_Distribution_Timeline.svg)

En la imagen puedes observar que de esas distribuciones derivan la mayoría de las distribuciones actuales. Indica de qué distribuciones derivan openSuse, Fedora y Ubuntu, así como los años en los que aparecen.

3. Familiarízate con el entorno gráfico de Ubuntu (o la distribución que hayas instalado). Localiza el explorador de archivos, el gestor de software para instalar y desinstalar aplicaciones, prueba a modificar la apariencia del escritorio... Si lo necesitas, puedes ayudarte de la guía del escritorio de Ubuntu para explorar esas y otras opciones:  
<https://help.ubuntu.com/lts/ubuntu-help/index.html>
4. Ampliación: prueba a instalar alguna otra distribución de GNU/Linux: openSuse, Linux Mint, Debian, Fedora, Red Hat... o alguna de las que encuentras en la imagen enlazada en el apartado 2.

### 14.2 Manejo de comandos

ATENCIÓN: **GNU/Linux distingue entre mayúsculas y minúsculas**. Por ejemplo, si queremos ejecutar el comando `ls` y tecleamos `LS`, el sistema no lo entenderá como un comando. O podríamos tener dos directorios en una misma ubicación, uno llamado `dir` y otro `Dir`.

#### 14.2.1 Cuadro resumen

Comienza a hacer un cuadro resumen de comandos. Puedes incluir el comando, una breve descripción y algún ejemplo de uso con opciones comunes.

#### 14.2.2 Ayuda de GNU/Linux

- A) Obtén, mediante una sola instrucción, ayuda de los comandos `cp`, `df`, `ifconfig` y `passwd`.
- B) ¿Cómo obtener en una instrucción la sección 5 del manual del comando `passwd`?
- C) ¿Cómo se sale del manual?
- D) Obtén una pequeña descripción orientativa de lo que hace el comando `passwd`.

### 14.2.3 Comandos para manipular ficheros y directorios

A) ¿Cuál es tu directorio de usuario o directorio HOME? ¿Existe algún archivo oculto en tu directorio de usuario?

B) Comprueba quién es el propietario del fichero `/etc/passwd`.

Crea en tu directorio de arranque un subdirectorio denominado *copia* y copia en él el archivo `/etc/passwd`. ¿Quién es ahora el propietario del archivo que has copiado? ¿Cuál es el grupo?

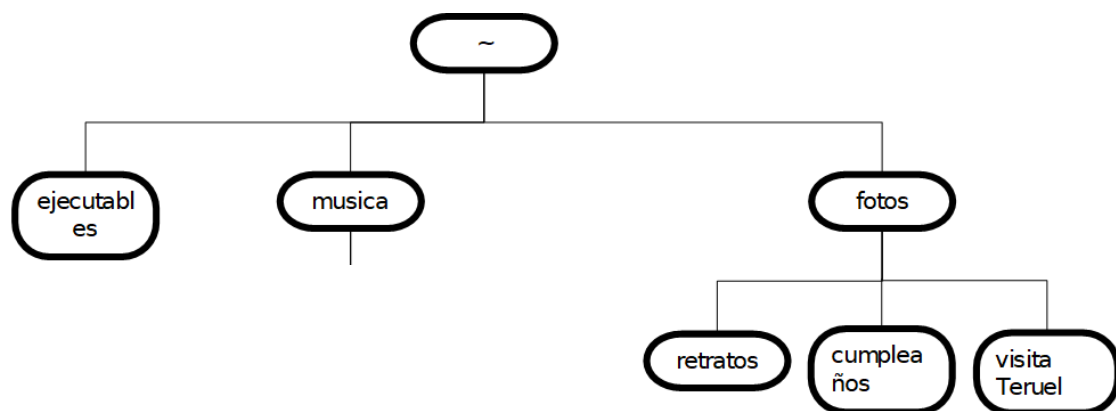
C) Cambia el nombre del archivo *passwd* del directorio *copia* por el de *palabras\_clave*.

D) En el directorio de usuario crea un subdirectorio denominado *.oculto* ¿Qué ocurre si intentas visualizar el nuevo subdirectorio? ¿Qué opción de `ls` debes utilizar para poder visualizarlo? Copia en este directorio el fichero `/etc/hosts`.

E) ¿Puedes cambiar el nombre de un directorio utilizando la orden `mv`? Compruébalo

F) Crea un subdirectorio en el directorio de arranque denominado *tmp*. Copia en este directorio el archivo `/etc/group` con el nombre de grupo. Cambia el propietario y grupo al archivo grupo de este directorio. NOTA: para hacer esto último, deberás tener otro usuario y grupo en el sistema. Puedes crear un grupo con el comando `groupadd nombre_grupo` y un usuario perteneciente al grupo anterior con el comando `useradd nombre_usuario -g nombre_grupo`

G) Dada la siguiente estructura de directorios:



Y teniendo en cuenta que:

- El signo ~ (virgulilla) hace referencia al directorio home del usuario o directorio de inicio del usuario.
- Para escribir nombres que incluyen espacios, se debe escribir la barra \ detrás de cada palabra y un espacio. Por ejemplo: `cd visita\ Teruel`
- Para crear varios directorios dentro de un mismo directorio, te pueden venir bien las llaves y los elementos de dentro separados por comas: {uno,dos,tres}

Realiza los ejercicios siguientes:

- a) Crea, en una sola instrucción, la estructura de directorios anterior. Intenta hacer el ejercicio primero con pocos directorios. Cuando veas que sabes manejar el comando, realiza el árbol completo en una sola instrucción.
- b) Copia los ficheros **cara.jpg** de la carpeta **retratos**, el fichero **tarta.jpg** de la carpeta **cumpleaños** y **paisaje.jpg** de la carpeta **Visita Teruel** a la carpeta **fotos**, estando ubicados en el directorio de inicio del usuario.

**Nota:** para hacer este apartado, crea primero los ficheros que sea necesario. Puedes hacerlo por ejemplo con el comando touch:

```
touch tarta.jpg
```

Crearé un fichero vacío, pero a nosotros ahora no nos importa el contenido.

- c) Lista los ficheros del directorio `/etc` ordenados por fecha de forma descendente.
- d) ¿Cómo podríamos saber el tipo de fichero que es **cara.jpg**?
- e) ¿Dónde se encuentra el ejecutable del comando `ls`? ¿Y sus páginas de manual?
- f) ¿Cuánto espacio tenemos en los discos duros de nuestro ordenador? ¿Qué espacio libre nos queda en la partición `/dev/sda3` (Bytes, Gigas o Megas)?
- g) ¿Cuánto espacio ocupa el directorio actual? ¿Y el directorio `/sbin`?

## 14.3 Usuarios, grupos y contraseñas

Te puede interesar echar un vistazo antes a los ejemplos recogidos en este artículo:

<https://nebul4ck.wordpress.com/2015/06/10/modificar-cuentas-de-usuario-y-grupos-en-linux/>

### Ejercicios:

1. Crea un usuario de nombre `alumno1` que tenga el UID 1200, pertenezca al grupo `alumnosFP`, tenga como comentario "Alumno grupo tardes". Además, tendrá como shell `/bin/bash`, y su propio directorio personal.
2. Cuando el usuario intenta acceder al sistema, ¿qué es lo que ocurre? ¿Qué podrías hacer si fueras el administrador para investigar y solventar el problema?
3. Añade un grupo con el nombre `erasmus` con GID 1300.
4. Añade `alumno1` al grupo `erasmus` editando el fichero `/etc/group`.
5. Modifica la información de cambio de contraseña de `alumno1`. No se puede cambiar la contraseña antes de 10 días, y es obligatorio cambiar la contraseña cada 30 días. Indica los diferentes métodos que puedes emplear.
6. ¿Qué realiza el comando `pwck`? ¿Para qué lo emplearías?
7. Modifica el fichero `sudoers` para que tu usuario pueda ejecutar el comando `useradd` sin necesidad de escribir la contraseña.
8. Comprueba que en la distribución de GNU/Linux que estás utilizando hay un grupo llamado `sudo`. En el fichero `sudoers` habrá una línea similar a esta:

```
#Allow members of group sudo to execute any command
%sudo ALL=(ALL:ALL) ALL
```

El usuario que has creado en el punto 1, `alumno1`, ¿puede ejecutar comandos con privilegios de administrador? Invierte esta situación. (Puedes ver los grupos a los que pertenece un usuario con los comandos `id` o `groups`)

## 14.4 Otros

### 14.4.1 Comandos de fecha y estadísticas

- A. Muestra el calendario del mes en que naciste.
- B. Cuánto tiempo tarda en ejecutarse el comando **du /etc** en tu máquina.
- C. ¿Cuánto tiempo lleva encendida tu máquina? ¿Cuántos usuarios han iniciado sesión?
- D. Muestra la hora actual en formato UNIX.
- E. Muestra sólo las fechas (día/mes/año con cuatro dígitos) de las siguientes fechas
  - Hace dos semanas.
  - Dentro de tres años.
  - Hace tres años.
  - El 3 de abril del presente año
- F. Muestra la hora con el siguiente formato (lo que está entre corchetes se tiene que actualizar automáticamente por el sistema).
  - Día de la semana: <lunes>
  - Semana del año en la que nos encontramos
  - 10-53-03
  - Día del año en el que nos encontramos.
  - Fecha de hoy en formato inglés, d/m/y
- G. Cambia la fecha y la hora del sistema a 3 de marzo del 2015, 10:54.
- H. Convierte, en formato UNIX, la fecha 3 de marzo del 2015, 10:54.
- I. Muestra, en una sola instrucción, la fecha de hoy en alemán.

### 14.4.2 Información de memoria

- A. Muestra los procesos que se están ejecutando en este momento. Filtra los resultados para mostrar sólo los procesos del usuario root. Muestra la carga de procesos por procesador.
- B. ¿Cuánta memoria tenemos libre en MB? Actualiza la información cada 3 minutos y que se muestre como máximo 7 veces.

### 14.4.3 Información del sistema

- A. ¿Qué versión del kernel tienes instalada?
- B. ¿Qué usuarios han iniciado sesión en este momento en nuestra máquina? ¿Cómo se llama el usuario con el estoy interactuando con el sistema?
- C. Cambia el nombre tu máquina a **pruebas** (conde XX es el número de tu máquina) y el dominio al que pertenecerá será **test.local**. ¿Qué ficheros has tenido que modificar? ¿Cómo comprobamos que todo es correcto?