



**Clase 03**

# Diseño y Programación Web

**Materia:**  
Sistemas Operativos

**Docente contenidista:** CARLASSARA, Fabrizio

**Revisión:** Coordinación

# Contenido

Sistema de archivos de Linux .....	04
El prompt .....	04
Sistema de archivos .....	05
Multiusuario .....	08
Comandos básicos .....	10
Obtener ayuda sobre comandos .....	10
Ver usuarios logueados .....	12
Ver ubicación del shell.....	13
Listar archivos .....	13
Limpiar pantalla .....	14
Apagar, reiniciar o suspender el sistema .....	15
Bibliografía .....	17
Para ampliar la información .....	17

# Clase 3



¡Te damos la bienvenida a la materia  
**Sistemas Operativos!**

En esta clase vamos a comenzar a adentrarnos en nuestro sistema operativo y empezar a probar algunos comandos.

## **En esta clase vamos a ver los siguientes temas:**

- Qué es el prompt.
- Cómo es el sistema de archivos de Linux.
- Cómo funciona el multiusuario.
- Diferencia entre terminal y shell.
- Comandos básicos que nos van a permitir: conocer más información de otros comandos, limpiar la pantalla, conocer quiénes se han logueado y apagar, reiniciar o suspender nuestro equipo.

# Sistema de archivos de Linux

Vamos a comenzar la clase discutiendo un poco en qué consiste la estructura del sistema de archivos de Linux, para entender un poco mejor dónde vamos a estar trabajando y qué cosas podemos encontrar.

## El prompt

En primer lugar, vamos a prender nuestra máquina virtual con CentOS y vamos a loguearnos con nuestro usuario root con la contraseña que elegimos.

Lo primero que vamos a ver es que hay un texto de bienvenida, una vez que nos logueamos, al que llamaremos prompt que tiene la forma: **[root@localhost ~]#**. Luego de este prompt vamos a notar que hay un cursor que está a la espera de que ingresemos comandos para poder trabajar con nuestro sistema operativo.

El formato del prompt tiene algunas implicaciones que tenemos que comenzar a interpretar:

- **root:** Esto indica el nombre de usuario actual que está utilizando la terminal. En este caso, "root" es el nombre de usuario del superusuario o de administrador, que tiene todos los privilegios en el sistema.
- **@:** Este carácter separa el nombre de usuario del siguiente componente, que es el nombre del sistema.
- **localhost:** Esto indica el nombre del sistema o el nombre de host. En este caso, "localhost" se refiere al nombre de host local, es decir, la misma máquina en la que estamos trabajando.
- **~:** Esta es la parte del prompt que muestra el directorio actual en el que nos encontramos. El símbolo "~" representa el directorio de inicio del usuario actual. Por lo tanto, en este caso, "~" indica que estamos en el directorio de inicio del usuario root.

Consideremos que un directorio no es más que una ubicación dentro del sistema de archivos del sistema operativo, pero el símbolo "~" puede resultar un tanto confuso ya que todavía no nos dice exactamente cuál es esa ubicación sino que sólo nos dice que estamos ahí.

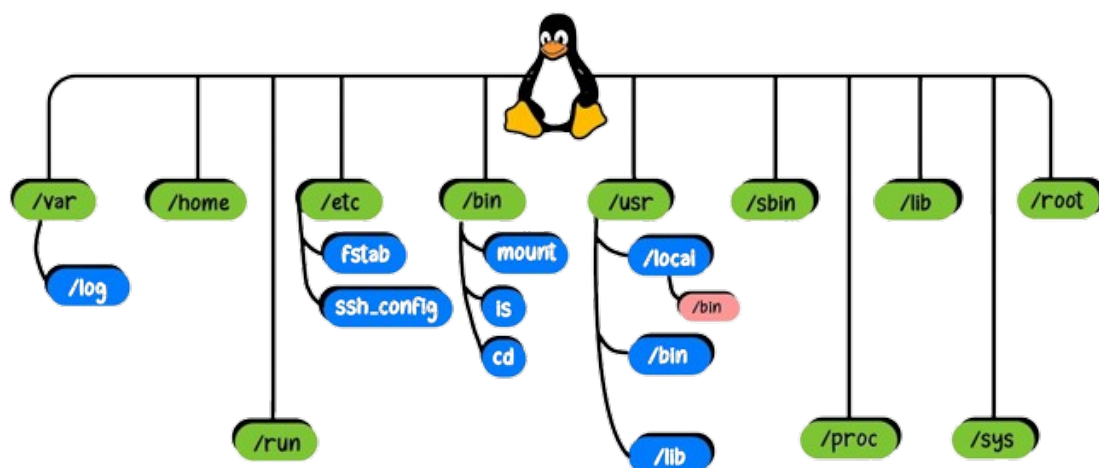
Pronto veremos cuál es esa ubicación concretamente, pero primero, analizaremos juntos algunos conceptos sobre cómo se estructura Linux.

## Sistema de archivos

Llamaremos sistema de archivos a la forma en la que un sistema operativo, en este caso Linux, estructura sus archivos y directorios. Nada en la forma en la que los ubica es casualidad sino que todo tiene un rol y por qué.

En primer lugar, vamos a tener que definir que existe algo llamado raíz o root del sistema que no es más que el lugar del cual todos los demás directorios surgen.

El sistema de archivos de Linux es algo como lo que muestra la figura de abajo:



En la raíz de todo el sistema es donde se encuentra el pingüino, y vemos cómo todos los demás directorios (precedidos por un "/") surgen de ahí. En la imagen podemos ver en verde algunos directorios principales como:

- **/bin:** Contiene programas binarios esenciales para el funcionamiento del sistema, accesibles para todos los usuarios.
- **/etc:** Contiene archivos de configuración del sistema, como archivos de configuración de red, usuarios, servicios y otros aspectos del sistema.
- **/home:** Directorio de inicio de los usuarios. Cada usuario tiene su propio subdirectorio aquí para almacenar sus archivos personales y configuraciones.

- **/lib o /lib64:** Contienen bibliotecas compartidas necesarias para los programas en /bin y /sbin, así como para otros programas del sistema.
- **/proc:** Un sistema de archivos virtual que proporciona información sobre los procesos en ejecución y otros datos del sistema en tiempo real.
- **/root:** El directorio de inicio del usuario root (superusuario). Este es el usuario con privilegios de administrador en Linux.
- **/run:** Un sistema de archivos temporal que contiene archivos de estado y de configuración de ejecución del sistema.
- **/sbin:** Contiene programas binarios esenciales para el funcionamiento del sistema, pero generalmente solo accesibles para el superusuario (root).
- **/sys:** Un sistema de archivos virtual que proporciona una interfaz para interactuar con el núcleo del sistema y configurar ciertos aspectos del hardware y del kernel.
- **/usr:** Contiene la mayoría de los programas y archivos no esenciales del sistema, como aplicaciones, bibliotecas, documentación y otros recursos compartidos.
- **/var:** Contiene datos variables del sistema, como archivos de registro, bases de datos de correo electrónico, archivos de caché y otros datos que pueden cambiar durante el uso normal del sistema.

Algunos otros directorios que no figuran en la imagen pero que suelen estar incluyen:

- **/boot:** Contiene los archivos necesarios para el arranque del sistema, como el kernel del sistema y los archivos de configuración del gestor de arranque.
- **/dev:** Contiene archivos especiales que representan dispositivos (por ejemplo, discos, particiones, impresoras) y permiten la comunicación entre el sistema y los dispositivos hardware.
- **/media y /mnt:** Directorios para montar dispositivos externos (por ejemplo, unidades USB, discos duros externos). /media se utiliza a menudo para montajes automáticos, mientras que /mnt es utilizado para montajes temporales o manuales.
- **/opt:** Directorio para la instalación de paquetes de software adicionales de terceros. Algunos programas de terceros se instalan en subdirectorios dentro de /opt.
- **/srv:** Contiene datos de servicios específicos del sistema, como archivos de sitios web, repositorios de datos y otros datos servidos por el sistema.

- **/tmp:** Un directorio para archivos temporales que pueden ser eliminados de forma segura. Los archivos en este directorio suelen ser eliminados automáticamente cuando el sistema se reinicia.

Según lo que estuvimos mencionando arriba, se supone que el símbolo “~” representa que nos ubicamos en el directorio del usuario root que hemos visto, se encuentra directamente dentro del raíz del sistema. Podemos chequearlo escribiendo nuestro primer comando **pwd** en el prompt y deberíamos ver al apretar enter algo como: /root. Podemos verificar que efectivamente nos encontramos en la ubicación que mencionamos.

# Multiusuario

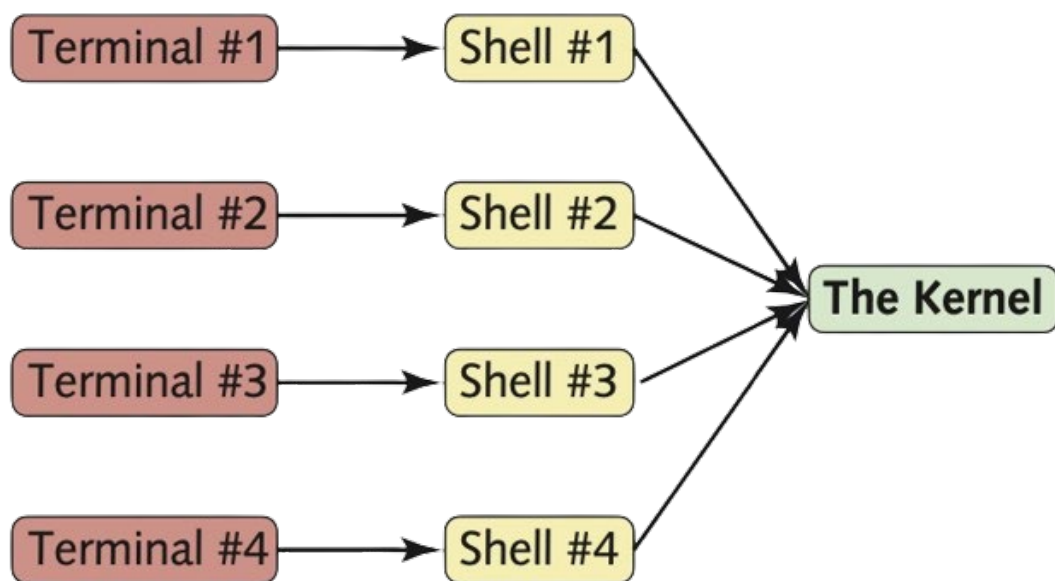
Recordemos que el sistema operativo no es más que una serie de programas que nos permite interactuar con el hardware de la computadora para darle algún uso particular. El responsable principal del sistema operativo que hace uso del hardware directamente, es el Kernel de Linux y la forma que tenemos en Linux de interactuar con el Kernel es primero loguearnos en la **terminal** que es lo primero que vemos cuando booteamos nuestro sistema operativo.

Una vez que nos logueamos con las credenciales correspondientes, nos recibe una interfaz de usuario llamada **shell** que es la responsable de tomar lo que escribamos, interpretarlo y darle órdenes al Kernel para procesar. La shell estándar que usan la mayoría de los Linux se llama BASH shell (Bourne Again Shell).

Antes hablamos de que Linux es un sistema operativo multiusuario. Esto se refiere a la capacidad del sistema operativo para permitir que múltiples usuarios accedan y utilicen el sistema simultáneamente. Eso implica que podemos tener múltiples personas logueándose en varias terminales, algunas pueden ser terminales de la misma computadora (la persona está sentada en la propia computadora) o pueden ser terminales remotas accedidas desde otro lugar a través de la red.

En cualquiera de los casos mencionados arriba, Linux nos va a permitir que muchos usuarios desde distintas terminales accedan a un shell y puedan interactuar con el Kernel como se ilustra en la figura de abajo.





Localmente en una máquina con Linux, podemos acceder a distintas terminales con algunas combinaciones de teclas como Ctrl + Alt + F2 por ejemplo. Si Linux está corriendo en una computadora portátil, a veces es necesario agregar la tecla Fn a la combinación de teclas. Abajo tenemos una tabla a modo de resumen de las distintas terminales a las que podemos acceder de acuerdo al software de virtualización que usamos.

Nombre de terminal	Teclas VirtualBox	Teclas VMware
tty1	Ctrl + Alt + F1	Ctrl + Cmd + F1
tty2	Ctrl + Alt + F2	Ctrl + Cmd + F2
tty3	Ctrl + Alt + F3	Ctrl + Cmd + F3
tty4	Ctrl + Alt + F4	Ctrl + Cmd + F4
tty5	Ctrl + Alt + F5	Ctrl + Cmd + F5
tty6	Ctrl + Alt + F6	Ctrl + Cmd + F6

Vamos a notar que si hacemos cualquiera de esas combinaciones, vamos a obtener una nueva terminal para loguearnos, al hacerlo con nuestro usuario root, vamos a ver que nos informa el nombre de la terminal desde donde nos estamos conectando.

# Comandos básicos

Vamos ahora a presentar algunos comandos básicos que deberíamos conocer. Antes de eso es importante definir algunos detalles sobre el uso de comandos. En primer lugar, los comandos no son más que programas que estamos pidiendo a Linux que ejecute. Cada comando es invocado a través del nombre cuando lo escribimos en el shell pero luego es seguido de **opciones** y **argumentos**.

Las opciones vienen después del comando y suelen ser letras específicas anteceditas por un "-". Estos buscan alterar un poco la forma en la que el comando normalmente trabaja y son específicas para cada comando. Algunas veces es posible anteceder la opción por un "--" pero en vez de una letra, se usan palabras enteras.

Por otro lado, los argumentos también vienen después del comando pero no los antecede un "-" y se usan para ajustar un poco el comando para nuestras necesidades particulares.

Un ejemplo de todo esto podría ser el comando ls que sirve para listar archivos. Acompañado de la opción -a también nos lista archivos ocultos y, si agregamos un argumento como /etc/rpm le estamos pidiendo que nos liste los archivos ocultos dentro del directorio /etc/rpm.

Por otro lado, no todos los comandos requieren opciones y argumentos para funcionar. Algunos sólo requieren invocar el programa con el nombre y listo. Pero cuando usamos opciones y argumentos, es importante tener mucho cuidado con las mayúsculas y minúsculas ya que no se tratan de la misma forma un "a" que una "A" por ejemplo en una opción y podría resultar en algo completamente distinto.

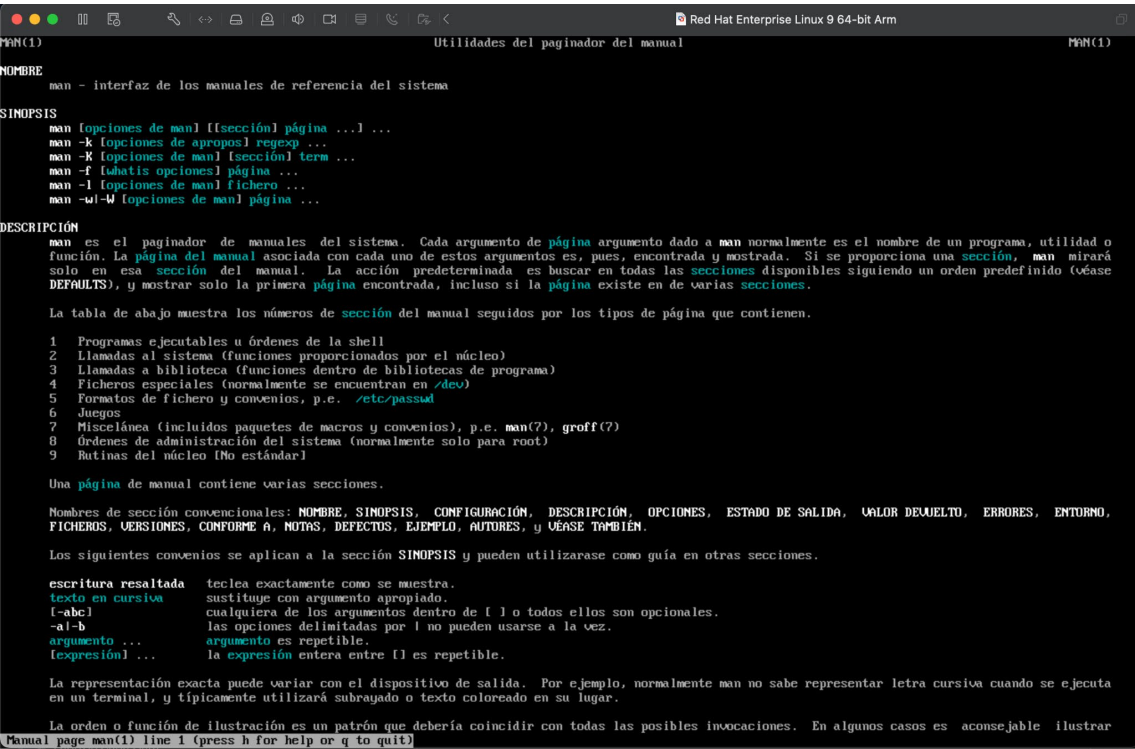
Con estas cosas en mente ahora podemos analizar algunos comandos comunes con algunas opciones y posibles argumentos.

## Obtener ayuda sobre comandos

Un comando elemental que tenemos que conocer y saber usar es el **man**. Este hace referencia a un manual de comandos de Linux y nos va a proporcionar descripciones, funcionamiento, opciones y argumentos posibles para los comandos que nos interesen. Este comando es una buena opción si no estamos seguros cómo usar un comando en particular.

Para usarlo, simplemente escribimos man seguido del comando que nos interesa y vamos a ver una larga descripción del comando. Podemos navegar en esta página con las flechas o la barra si queremos saltar páginas. Cuando queramos finalizar el comando simplemente usamos la letra q.

La imagen de abajo muestra el resultado de correr el comando man man, que nos devuelve una descripción completa de este mismo comando.



```
MAN(1)
Utilidades del paginador del manual
MAN(1)

NOMBRE
man - interfaz de los manuales de referencia del sistema

SINOPSIS
man [opciones de man] [[sección] página ...] ...
man -k [opciones de apropos] regex ...
man -K [opciones de man] [sección] term ...
man -f [whatis opciones] página ...
man -l [opciones de man] fichero ...
man -w [-w] [opciones de man] página ...

DESCRIPCIÓN
man es el paginador de manuales del sistema. Cada argumento de página argumento dado a man normalmente es el nombre de un programa, utilidad o
función. La página del manual asociada con cada uno de estos argumentos es, pues, encontrada y mostrada. Si se proporciona una sección, man mirará
solo en esa sección del manual. La acción predeterminada es buscar en todas las secciones disponibles siguiendo un orden predefinido (véase
DEFAULTS), y mostrar solo la primera página encontrada, incluso si la página existe en de varias secciones.

La tabla de abajo muestra los números de sección del manual seguidos por los tipos de página que contienen.

1 Programas ejecutables u órdenes de la shell
2 Llamadas al sistema (funciones proporcionados por el núcleo)
3 Llamadas a biblioteca (funciones dentro de bibliotecas de programa)
4 Ficheros especiales (normalmente se encuentran en /dev)
5 Formatos de fichero y convenios, p.e. /etc/passwd
6 Juegos
7 Miscelánea (incluidos paquetes de macros y convenios), p.e. man(7), groff(7)
8 Órdenes de administración del sistema (normalmente solo para root)
9 Rutinas del núcleo (No estándar)

Una página de manual contiene varias secciones.

Nombres de sección convencionales: NOMBRE, SINOPSIS, CONFIGURACIÓN, DESCRIPCIÓN, OPCIONES, ESTADO DE SALIDA, VALOR DEVUELTO, ERRORES, ENTORNO,
FICHEROS, VERSIONES, CONFORME A, NOTAS, DEFECTOS, EJEMPLO, AUTORES, y VÉASE TAMBIÉN.

Los siguientes convenios se aplican a la sección SINOPSIS y pueden utilizarse como guía en otras secciones.

escritura resaltada   teclea exactamente como se muestra.
texto en cursiva      sustituye con argumento apropiado.
[-abc]               cualquiera de los argumentos dentro de [ ] o todos ellos son opcionales.
-a[-b]               las opciones delimitadas por | no pueden usarse a la vez.
argumento ...         argumento es repetible.
[expresión] ...       la expresión entera entre [ ] es repetible.

La representación exacta puede variar con el dispositivo de salida. Por ejemplo, normalmente man no sabe representar letra cursiva cuando se ejecuta
en un terminal, y típicamente utilizará subrayado o texto coloreado en su lugar.

La orden o función de ilustración es un patrón que debería coincidir con todas las posibles invocaciones. En algunos casos es aconsejable ilustrar
[Manual page man(1) line 1 (press h for help or q to quit)]
```

En el resultado de este comando vamos a ver que en la primera línea tenemos el nombre del comando con un número entre paréntesis como MAN(1) en el ejemplo de arriba. Este número también nos da información sobre el comando en particular. Cada número corresponde a una sección para clasificar los comandos, particularmente el (1) hace referencia a comandos que cualquier usuario puede usar, mientras que el (8), hace referencia a comandos que solo pueden ser usados por el usuario root.

Sección del manual	Descripción
1	Comandos que cualquier usuario puede usar
2	Llamadas del sistema de Linux
3	Rutinas de bibliotecas

4	Archivos de dispositivos especiales
5	Formatos de archivos
6	Juegos
7	Misceláneos
8	Comandos que sólo puede usar el usuario root
9	Rutinas del kernel de Linux
n	Comandos nuevos no categorizados

## Ver usuarios logueados

Algunas secciones atrás, vimos que Linux es un sistema operativo multiusuario. Esto implicaría que cualquier persona con un usuario podría estar usando el sistema operativo sin que nos demos cuenta.

Afortunadamente, existen comandos que nos proporcionan ese tipo de información si necesitamos saber qué usuarios están logueados actualmente en el sistema operativo.

En primer lugar, tenemos el comando **whoami** que nos proporciona el nombre de usuario con el cual estamos trabajando actualmente. Este comando es útil si tomamos una máquina con un usuario logueado o donde el prompt no nos proporcione información clara en cuanto a quién está logueado actualmente. La respuesta de Linux es simple y concreta como vemos abajo:

```
(root@localhost ~)# whoami
root
(root@localhost ~)#
```

Por otro lado, si queremos ver información sobre todos los usuarios logueados, podemos usar el comando **who** que nos proporciona qué usuarios están logueados, en qué terminal y a qué hora y fecha lo hicieron.

```
(root@localhost ~)# who
root    tty1      2024-03-13 14:32
root    tty2      2024-03-13 14:34
root    tty3      2024-03-13 17:30
(root@localhost ~)#
```

Si nos interesa saber aún más sobre los usuarios logueados podemos usar el comando **w** que nos brinda todo lo anterior pero además nos dice cuánto tiempo tienen de inactividad, cuánto tiempo y porcentaje

de recursos del procesador fueron usados por ese usuario y el último comando o proceso que usaron.

```
[root@localhost ~]# w
18:18:44 up 3:46, 3 users, load average: 0.05, 0.01, 0.00
USER      TTY      LOGIN@  IDLE   JCPU   PCPU WHAT
root     tty1      14:32    1.00s  0.11s  0.01s w
root     tty2      14:34    3:12m  0.03s  0.03s -bash
root     tty3      17:30   48:01  0.00s  0.00s -bash
[root@localhost ~]#
```

## Ver ubicación del shell

Algo que veremos en las próximas clases es que muchos comandos dependen fuertemente de la ubicación en la que se estén ejecutando, obteniendo a veces resultados inesperados o errores cuando no tenemos muy en claro dónde está parada nuestra shell en el sistema de archivos.

En muchas ocasiones el prompt va a proporcionarnos esa información, pero en algunos casos, no es lo suficientemente clara o es inexistente. Cuando no contamos con esa información, siempre podemos usar el comando **pwd** que nos va a devolver la ubicación actual del shell dentro del sistema de archivos.

## Listar archivos

Un comando muy común de usar es el **ls**. Este nos permite ver todos los archivos que se encuentran en el directorio en el que nos encontramos actualmente. La sintaxis del comando es: **ls [opciones] [directorio]**. El argumento directorio nos permite especificar qué directorio del sistema queremos inspeccionar y listar sus archivos. En cuanto a las opciones, mencionamos algunas de las más comunes abajo:

- **-a**: muestra los archivos ocultos que van a ser anteceditos con un ".".
- **-l**: muestra más información sobre los archivos como: permisos, dueños del archivo, tamaño, fecha y hora de modificación.
- **-S**: lista los archivos y directorios de más a menos pesado.
- **-t**: lista los archivos por fecha de modificación, mostrando los más recientes primero.

Por ejemplo, abajo mostramos el resultado de ejecutar el comando **ls -l /** que nos mostraría información adicional sobre los archivos que se encuentran en el raíz del sistema.

```

[root@localhost ~]# ls -l /
total 24
dr-xr-xr-x.  2 root root    6 ago  9  2021 afs
lrwxrwxrwx.  1 root root    7 ago  9  2021 bin -> usr/bin
dr-xr-xr-x.  6 root root 4096 mar  5 21:11 boot
drwxr-xr-x. 19 root root 3328 mar 13 17:23 dev
drwxr-xr-x. 81 root root 8192 mar 13 17:23 etc
drwxr-xr-x.  2 root root    6 ago  9  2021 home
lrwxrwxrwx.  1 root root    7 ago  9  2021 lib -> usr/lib
lrwxrwxrwx.  1 root root    9 ago  9  2021 lib64 -> usr/lib64
drwxr-xr-x.  2 root root    6 ago  9  2021 media
drwxr-xr-x.  2 root root    6 ago  9  2021 mnt
drwxr-xr-x.  2 root root    6 ago  9  2021 opt
dr-xr-xr-x. 299 root root    0 mar 13 17:22 proc
dr-xr-x---.  3 root root   163 mar 13 18:28 root
drwxr-xr-x. 28 root root   800 mar 13 17:23 run
lrwxrwxrwx.  1 root root    8 ago  9  2021/sbin -> usr/sbin
drwxr-xr-x.  2 root root    6 ago  9  2021 srv
dr-xr-xr-x. 12 root root    0 mar 13 17:23 sys
drwxrwxrwt. 11 root root 4096 mar 13 17:23 tmp
drwxr-xr-x. 12 root root   144 mar  5 21:08 usr
drwxr-xr-x. 19 root root 4096 mar  5 21:11 var
[root@localhost ~]# _

```

Donde vamos vemos información distinta en cada columna.

- El primer carácter nos dice qué tipo de archivo es ("d" para directorio, "l" si es un link a otro directorio, "-" si es un archivo, "s" si es un archivo socket).
- Inmediatamente después vienen los permisos que cada usuario tiene para ese archivo. Esto último será un tema que trataremos en algunas clases.
- Número de links a ese archivo.
- Nombre del usuario dueño de ese archivo.
- Nombre del grupo dueño de ese archivo.
- Tamaño en bytes que pesa ese archivo.
- Última modificación y tiempo que tuvo el archivo.
- Nombre del archivo o directorio.

## Limpiar pantalla

Cuando la terminal queda muy llena de texto por la cantidad de comandos que ejecutamos, podemos usar el comando **clear** que simplemente nos deja la terminal limpia para que no haya texto de más que nos confunda.

# Apagar, reiniciar o suspender el sistema

Habrán notado que en este sistema operativo sin interfaz gráfica, no tenemos ningún menú con un botón para apagar o reiniciar el sistema. Los comandos para poder hacer eso son bastante sencillos pero como detalle, sólo pueden ser ejecutados desde el usuario root que afortunadamente ya somos.

Hay cuatro comandos comunes que podemos usar:

- **poweroff:** apaga el sistema inmediatamente.
- **reboot:** reinicia el sistema inmediatamente.
- **halt:** suspende el sistema inmediatamente.
- **shutdown:** hace todo lo anterior de acuerdo a las opciones que le indiquemos.

Como el comando shutdown es el más completo, vamos a centrarnos en ese. En principio, la sintaxis para usarlo es **shutdown [opciones] [tiempo] [mensaje]**.

En cuanto a opciones, podemos mencionar:

- **-H:** para suspender el sistema.
- **-P:** para apagar el sistema.
- **-r:** para reiniciar el sistema.
- **-c:** cancela un apagado programado.

En cuanto al argumento tiempo, tiene que ver con si queremos ejecutar la opción ahora o programarla para más adelante. Algunas opciones podrían ser:

- 5:00 para que el comando se ejecute a las 5 AM.
- +4 para que el comando se ejecute dentro de 4 minutos.
- now para que se ejecute ahora.

Si no indicamos opción el comando se interpreta por defecto como una orden de apagado. Si tampoco se indica tiempo, se programa un apagado dentro de un minuto.

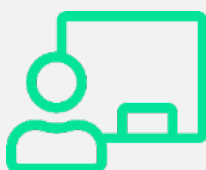
Por último, el argumento de mensaje nos da la opción de, siempre que estemos usando el argumento de tiempo también, enviar un mensaje a todos los usuarios logueados cuando el comando se ejecute. Este argumento debe ir entre comillas dobles ("").



Hemos llegado así al final de esta clase en la que vimos:

- Que es el prompt.
- Cómo es el sistema de archivos de Linux.
- Cómo funciona el multiusuario.
- Diferencia entre terminal y shell.
- Comandos básicos que nos van a permitir: conocer más información de otros comandos, limpiar la pantalla, conocer quiénes se han logueado y apagar, reiniciar o suspender nuestro equipo.

Aún hay mucho por delante y muchos comandos por explorar. ¡El siguiente paso será comenzar a conocer algunos comandos para manipular archivos!



Te esperamos en la **clase en vivo** de esta semana.  
No olvides realizar el **desafío semanal**.

**¡Hasta la próxima clase!**



# Bibliografía

Eckert, J. W. (2020). Linux+ and LPIC-1: Guide to Linux Certification. Cengage.

## Para ampliar la información

Ubuntu. (s.f.). Directorios y sistemas de archivos. Recuperado de <https://help.ubuntu.com/kubuntu/desktopguide/es/directories-file-systems.html>

Didweb.gitbooks.io (s.f.). Título del capítulo: ls - Comandos Linux. En Comandos Linux (p. [número de página si está disponible]). GitBook. Recuperado de <https://didweb.gitbooks.io/comandos-linux/content/chapter1/directorios/ls.html>

Didweb.gitbooks.io (s.f.). Título del capítulo: Apagar equipo - Comandos Linux. En Comandos Linux (p. [número de página si está disponible]). GitBook. Recuperado de <https://didweb.gitbooks.io/comandos-linux/content/chapter1/boot/apagar-equipo.html>