Listado de comandos Linux más utilizados

Comando	Función
ls	Lista el contenido de un directorio
pwd	Muestra la ruta del directorio de trabajo actual
cd	Cambia el directorio de trabajo
mkdir	Crea un nuevo directorio
rm	Elimina un archivo
cp	Copia archivos y directorios, incluido su contenido
mv	Mueve o renombra archivos y directorios
touch	Crea un nuevo archivo vacío
file	Comprueba el tipo de archivo
zip and unzip	Crea y extrae un archivo ZIP
tar	Archiva ficheros sin compresión en formato TAR

Listado de comandos Linux más utilizados

nano, vi y jed	Edita un archivo con un editor de texto
cat	Lista, combina y escribe el contenido de un archivo como salida estándar
grep	Busca una cadena dentro de un archivo
sed	Busca, sustituye o elimina patrones en un archivo
head	Muestra las diez primeras líneas de un archivo
tail	Imprime las diez últimas líneas de un archivo
awk	Busca y manipula patrones en un archivo
sort	Reordena el contenido de un archivo
cut	Secciona e imprime líneas de un archivo
diff	Compara el contenido de dos archivos y sus diferencias
tee	Imprime las salidas de los comandos en el Terminal y en un archivo
locate	Busca archivos en la base de datos de un sistema

Listado de comandos Linux más utilizados

find	Muestra la ubicación de un archivo o carpeta
sudo	Ejecuta un comando como superusuario
su	Ejecuta programas en el shell actual como otro usuario
chmod	Modifica los permisos de lectura, escritura y ejecución de un archivo
chown	Cambia la propiedad de un archivo, directorio o enlace simbólico
useradd y userdel	Crea y elimina una cuenta de usuario
df	Muestra el uso general de espacio en disco del sistema
du	Comprueba el consumo de almacenamiento de un archivo o directorio
top	Muestra los procesos en ejecución y el uso de recursos del sistema
htop	Funciona como **top** pero con una interfaz de usuario interactiva
ps	Crea una instantánea de todos los procesos en ejecución

Listado de comandos Linux más utilizados

uname	Imprime información sobre el núcleo, el nombre y el hardware de tu máquina
hostname	Muestra el nombre de host de tu sistema
time	Calcula el tiempo de ejecución de los comandos
systemctl	Gestiona los servicios del sistema
watch	Ejecuta otro comando de forma continua
jobs	Muestra los procesos en ejecución de un intérprete de órdenes con sus estados
kills	Finaliza un proceso en ejecución
shutdown	Apaga o reinicia el sistema
ping	Comprueba la conectividad de red del sistema
wget	Descarga archivos de una URL
curl	Transmite datos entre servidores utilizando URLs
scp	Copia de forma segura archivos o directorios a otro sistema

Listado de comandos Linux más utilizados

rsync	Sincroniza el contenido entre directorios o máquinas
lfconfig	Muestra las interfaces de red del sistema y sus configuraciones
netstat	Muestra la información de red del sistema, como enrutamiento y sockets
traceroute	Rastrea los saltos de un paquete hasta su destino
nslookup	Consulta la dirección IP de un dominio y viceversa
dig	Muestra información del DNS, incluidos los tipos de registro
history	Lista los comandos ejecutados anteriormente
man	Muestra el manual de un comando
echo	Imprime un mensaje como salida estándar
ln	Enlaza archivos o directorios
alias y unalias	Establece y elimina un alias para un archivo o comando
cal	Muestra un calendario en Terminal

Listado de comandos Linux más utilizados

apt-get	Gestiona las bibliotecas de paquetes de las distribuciones basadas en Debian

1. Comando ls

El comando **ls** lista los archivos y directorios de tu sistema. Ésta es la sintaxis:

```
ls [/directorio/carpeta/ruta]
```

Si eliminas la ruta, el comando **ls** mostrará el contenido del directorio de trabajo actual. Puedes modificar el comando utilizando estas opciones:

- **-R:** Lista todos los archivos de los subdirectorios.
- **-a:** Muestra todos los archivos, incluidos los ocultos.
- **-lh:** convierte los tamaños a formatos legibles, como **MB**, **GB** y **TB**.

2. Comando pwd

El comando **pwd** imprime la ruta de tu directorio de trabajo actual, como **/inicio/directorio/ruta**. Ésta es la sintaxis del comando:

```
pwd [opción]
```

Admite dos opciones. La opción **-L** o **–logical** imprime el contenido de las variables de entorno, incluidos los **enlaces simbólicos**. Por su parte, **-P** o **-physical** muestra la ruta real del directorio actual.

3. Comando cd

Utiliza el comando **cd** para navegar por los archivos y directorios de Linux. Para utilizarlo, ejecuta esta sintaxis con privilegios sudo:

```
cd /directorio/carpeta/ruta
```

Dependiendo de tu ubicación actual, requiere la ruta completa o el nombre del directorio. Por ejemplo, omite **/nombredeusuario** de **/nombredeusuario/directorio/carpeta** si ya estás dentro de él.
Si omites los argumentos, irás a la carpeta de inicio. Aquí tienes algunos atajos de navegación:

- **cd ~[nombredeusuario]:** va al directorio personal de otro usuario.
- **cd ..**: mueve un directorio hacia arriba.
- **cd-**: cambia al directorio anterior.

4. Comando mkdir

Utiliza el comando **mkdir** para crear uno o varios directorios y establecer sus permisos. Asegúrate de que estás autorizado a crear una nueva carpeta en el directorio padre. Ésta es la sintaxis básica:

```
mkdir [opción] [nombre_directorio]
```

Para crear una carpeta dentro de un directorio, utiliza la ruta como parámetro del comando. Por ejemplo, **mkdir música/canciones** creará una carpeta **canciones** dentro de **música**. Aquí tienes varias opciones comunes del comando mkdir:

- **-p:** crea un directorio entre dos carpetas existentes. Por ejemplo, **mkdir -p Música/2024/Canciones** crea un nuevo directorio **2024**.
- **-m:** establece los permisos de la carpeta. Por ejemplo, introduce **mkdir -m777 directory** para crear un directorio con permisos de lectura, escritura y ejecución para todos los usuarios.
- **-v:** imprime un mensaje por cada directorio creado.

5. Comando rmdir

Utiliza el comando **rmdir** para **borrar un directorio vacío en Linux**. El usuario debe tener privilegios **sudo** en el directorio padre. Esta es la sintaxis:

```
rmdir [opción] nombre_directorio
```

Si la carpeta contiene un subdirectorio, el comando devolverá un error. Para forzar la eliminación de un directorio no vacío, utiliza la opción **-p**.

6. Comando rm

Utiliza el comando rm para eliminar permanentemente los archivos de un directorio. Ésta es la sintaxis general:

```
rm [nombrearchivo1] [nombrearchivo2] [nombrearchivo3]
```

Ajusta el número de archivos del comando según tus necesidades. Si encuentras un error, asegúrate de que tienes permiso de **escritura** en el directorio.
Para modificar el comando, añade las siguientes opciones:

- **-i:** pide una confirmación antes de borrar.
- **-f:** permite eliminar archivos sin confirmación.
- **-r:** borra archivos y directorios recursivamente.

7. Comando cp

Utiliza el comando **cp** para copiar archivos o directorios, incluido su contenido, de tu ubicación actual a otra. Tiene varios casos de uso, como:

- Copiar un archivo del directorio actual a otra carpeta. Especifica el nombre del archivo y la ruta de destino:

```
cp nombrearchivo.txt /inicio/nombredeusuario/documentos
```

- Duplicar varios archivos en un directorio. Introduce los nombres de los archivos y la ruta de destino:

```
cp nombrearchivo1.txt nombrearchivo2.txt nombrearchivo3.txt /inicio/nombredeusuario/documentos
```

- Copiar el contenido de un fichero a otro dentro del mismo directorio. Introduce el fichero de origen y el de destino:

```
cp nombrearchivo1.txt nombrearchivo2.txt
```

- Duplicar un directorio entero. Pasa la bandera **-R** seguida del directorio de origen y de destino:

```
cp -R /inicio/nombredeusuario/documentos /inicio/nombredeusuario/documentos_respaldo
```

8. Comando mv

Utiliza el comando **mv** para mover o renombrar archivos y directorios. Para mover elementos, introduce el nombre del archivo seguido del directorio de destino:

```
mv nombrearchivo.txt /inicio/nombredeusuario/documentos
```

Mientras tanto, utiliza la siguiente sintaxis para **renombrar un archivo en Linux** con el comando **mv**:

```
mv nombre_archivo_antiguo.txt nombre_archivo_nuevo.txt
```

9. Comando touch

El comando **touch** te permite crear un archivo vacío en una ruta de directorio específica. Esta es la sintaxis:

```
touch [opción] /inicio/directorio/ruta/archivo.txt
```

Si omites la ruta, el comando creará el elemento en la carpeta actual. También puedes utilizar **touch** para generar y modificar una marca de tiempo en la línea de comandos de Linux.

10. Comando file

El comando file te permite comprobar un tipo de archivo, ya sea texto, imagen o binario. Ésta es la sintaxis:

```
file nombrearchivo.txt
```

Para comprobar en bloque varios archivos, enuméralos individualmente o utiliza su ruta si están en el mismo directorio. Añade la opción **-k** para mostrar información más detallada e **-i** para mostrar el **tipo MIME** del archivo.

11. Comandos zip y unzip

El comando **zip** te permite comprimir elementos en un archivo **ZIP** con la relación de compresión óptima. Ésta es la sintaxis:

```
zip [opciones] archivozip archivo1 archivo2....
```

Por ejemplo, este comando comprime **nota.txt** en **archivo.zip** en el directorio de trabajo actual:

```
zip archivo.zip nota.txt
```

Utiliza el comando **unzip** para **extraer el archivo comprimido**. Ésta es la sintaxis:

```
unzip [opción] nombre_archivo.zip
```

12. Comando tar

El comando tar archiva varios elementos en un archivo **TAR**, un formato similar al **ZIP** con compresión opcional. Ésta es la sintaxis

```
tar [opciones] [fichero_archivo] [archivo de destino o directorio]
```

Por ejemplo, introduce lo siguiente para crear un nuevo archivo **nuevoarchivo.tar** en el directorio **/inicio/usuario/documentos**:

```
tar -cvzf nuevoarchivo.tar /inicio/usuario/documentos
```

13. Comandos nano, vi y jed

Linux permite a los usuarios editar archivos utilizando un editor de texto como nano, **vi** o **jed**. Aunque la mayoría de las distribuciones incluyen **nano** y **vi**, los usuarios deben instalar **jed** manualmente. Todas estas herramientas tienen la misma sintaxis de comandos:

```
nano nombrearchivo
vi nombrearchivo
jed nombrearchivo
```

Si el archivo de destino no existe, estos editores crearán uno. Te recomendamos, por un lado, **nano** si quieres editar rápidamente archivos de texto. Por otro lado, utiliza **vi** o **jed** para scripts y programación.

14. Comando cat

Concatenar o **cat** es uno de los comandos Linux más utilizados. Lista, combina y escribe el contenido de los archivos en la salida estándar. Ésta es la sintaxis:

```
cat nombrearchivo.txt
```

Hay varias formas de utilizar el comando **cat**:

- **cat > filen.txt:** crea un nuevo archivo.
- **cat archivo1.txt archivo2.txt > archivo3.txt:** fusiona el **archivo1.txt** con **el archivo2.txt** y almacena el resultado en **el archivo3.txt**.
- **tac archivo.txt:** muestra el contenido en orden inverso.

15. Comando grep

La **expresión regular global** o comando **grep** te permite encontrar una palabra buscando en el contenido de un archivo. Este comando de Linux imprime todas las líneas que contienen las cadenas coincidentes, lo que resulta útil para filtrar archivos de registro de gran tamaño.

Por ejemplo, para mostrar las líneas que contienen **azul** en el archivo **blocdenotas.txt**, introduce:

```
grep blue blocdenotas.txt
```

16. Comando sed

El comando **sed** te permite encontrar, sustituir y eliminar patrones en un archivo sin utilizar un editor de texto.

Ésta es la sintaxis general:

```
sed [opción] 'script' archivo_entrada
```

El script contiene el patrón de expresión regular buscado, la cadena de sustitución y los subcomandos. Utiliza el subcomando **s** para reemplazar los patrones coincidentes o **d** para eliminarlos.

Al final, especifica el archivo que contiene el patrón a modificar. Aquí tienes un ejemplo de comando que sustituye el **rojo** de **colores.txt** y **tono.txt** por el **azul**:

```
sed 's/red/blue' colores.txt tono.txt
```

17. Comando head

El comando head imprime las diez primeras líneas de un archivo de texto o datos canalizados en tu interfaz de línea de comandos. Ésta es la sintaxis general:

```
head [opción] [archivo]
```

Por ejemplo, para ver las diez primeras líneas de **nota.txt** en el directorio actual, introduce:

```
head nota.txt
```

El comando **head** acepta varias opciones, como:

- **-n:** cambia el número de líneas impresas. Por ejemplo, **head -n 5** muestra las cinco primeras líneas.
- **-c:** imprime el primer número personalizado de bytes del archivo.
- **-q:** desactiva las cabeceras que especifican el nombre del archivo.

18. Comando tail

El comando **tail** muestra las diez últimas líneas de un archivo, lo que resulta útil para comprobar nuevos datos y errores. Ésta es la sintaxis:

```
tail [opción] [archivo]
```

Por ejemplo, introduce lo siguiente para mostrar las diez últimas líneas del archivo **colores.txt**:

```
tail -n colores.txt
```

19. Comando awk

El comando **awk** escanea patrones de expresiones regulares en un archivo para recuperar o manipular datos coincidentes. Ésta es la sintaxis básica:

```
awk '/regex pattern/{action}' archivo_entrada.txt
```

La acción puede ser operaciones matemáticas, sentencias condicionales como **if**, expresiones de salida como **print** y un comando de **borrado**. También contiene la notación **$n**, que se refiere a un campo de la línea actual.

Para añadir varias acciones, enuméralas según el orden de ejecución, separadas mediante punto y coma. Por ejemplo, este comando contiene sentencias matemáticas, condicionales y de salida:

```
awk -F':' '{ total += $2; estudiantes[$1] = $2 } END { average = total / length(estudiantes); print "Average:", average; print "Above average:"; for (estudiante en estudiantes) if (estudiantes[estudiante] > average) print estudiante }' puntuación.txt
```

20. Comando sort

El comando **sort** reordena las líneas de un archivo en un orden determinado. No modifica el archivo real y sólo imprime el resultado como salida del Terminal. Ésta es la sintaxis:

```
sort [opción] [archivo]
```

Por defecto, este comando ordenará las líneas por orden alfabético, de la A a la Z. Para modificar la ordenación, utiliza estas opciones:

- **-o:** redirige las salidas del comando a otro archivo.

- **-r:** invierte el orden de clasificación a descendente.
- **-n:** ordena el archivo numéricamente.
- **-k:** reordena los datos de un campo concreto.

21. Comando cut

El comando **cut** recupera secciones de un archivo e imprime el resultado como salida de Terminal. Ésta es la sintaxis:

```
cut [opción] [archivo]
```

En lugar de un archivo, puedes utilizar datos de la **entrada estándar**. Para determinar cómo secciona el comando la línea, utiliza las siguientes opciones:

- **-f:** selecciona un campo específico.
- **-b:** recorta la línea en un tamaño de byte especificado.
- **-c:** secciona la línea utilizando un carácter especificado.
- **-d:** separa las líneas en función de los delimitadores.

Puedes combinar estas opciones, utilizar un rango y especificar varios valores. Por ejemplo, este comando extrae del tercer al quinto campo de una lista separada por comas:

```
cut -d',' -f3-5 lista.txt
```

22. Comando diff

El comando **diff** compara el contenido de dos archivos y muestra las diferencias. Se utiliza para alterar un programa sin modificar el código. Éste es el formato general:

```
diff [opción] archivo1 archivo2
```

A continuación se indican algunas opciones aceptables:

- **-c:** muestra la diferencia entre dos archivos en un formulario contextual.
- **-u:** muestra la salida sin información redundante.
- **-i:** hace que el comando diff no distinga entre mayúsculas y minúsculas.

23. Comando tee

El comando **tee** escribe la entrada del usuario en la salida y los archivos de Terminal. Ésta es la sintaxis básica:

```
command | tee [opción] archivo1
```

Por ejemplo, lo siguiente hace ping a Google e imprime la salida en Terminal, **ping_result.txt** y el archivo **19092024.txt:**

```
ping google.com | tee ping_result.txt 19092024.txt
```

24. Comando locate

El comando **locate** te permite encontrar un archivo en el sistema de base de datos. Añade la opción **-i** para desactivar la distinción entre mayúsculas y minúsculas, y un asterisco (*) para buscar contenido con varias palabras clave. Por ejemplo:

```
locate -i escuela*nota
```

El comando busca archivos que contengan **escuela** y **nota**, independientemente de la mayúscula o minúscula que tengan.

25. Comando find

Utiliza el comando **find** para buscar archivos dentro de un directorio concreto. Ésta es la sintaxis:

```
find [opción] [ruta] [expresión]
```

Por ejemplo, para buscar un archivo llamado **archivo1.txt** dentro de la carpeta **directorio** y sus subdirectorios, utiliza este comando:

```
find /inicio -name archivo1.txt
```

Si omites la ruta, el comando buscará en el directorio de trabajo actual. También puedes buscar directorios utilizando lo siguiente:

```
find ./ -type d -name nombredirectorio
```

26. Comando sudo

El **superusuario do** o **sudo** es uno de los comandos más básicos de Linux. Ejecuta tu comando con permisos administrativos o de root. Esta es la sintaxis general:

```
sudo (comando)
```

Cuando ejecutes un comando sudo, Terminal te pedirá la contraseña de root. Por ejemplo, este fragmento ejecuta **useradd** con el privilegio de superusuario:

```
sudo useradd nombredeusuario
```

También puedes añadir una opción, como por ejemplo

- **-k:** invalida el archivo de marca de tiempo.
- **-g:** ejecuta comandos como un nombre o ID de grupo especificado.
- **-h:** ejecuta comandos en el host.

27. Comando su

El comando **su** te permite ejecutar un programa en el shell de Linux como un usuario diferente. Es útil para conectarse por **SSH** mientras el usuario root está deshabilitado. Esta es la sintaxis:

```
su [opciones] [nombre de usuario [argumento]]
```

Sin ninguna opción ni argumento, este comando se ejecuta a través de **root** y te pide que utilices los privilegios **sudo** temporalmente. Algunas opciones son

- **-p:** mantiene el mismo entorno de shell, formado por **HOME**, **SHELL**, **USER** y **LOGNAME**.
- **-s:** te permite especificar otro entorno shell para ejecutar.
- **-l:** ejecuta un script de inicio de sesión para cambiar de usuario. Requiere que introduzcas la contraseña del usuario.

Para comprobar la cuenta de usuario del shell actual, ejecuta el comando **whoami**:

28. Comando chmod

El comando **chmod** modifica los permisos de directorios o archivos en Linux. Ésta es la sintaxis básica:

```
chmod [opción] [permiso] [nombre_archivo]
```

En Linux, cada archivo está asociado a tres clases de usuario: **propietario**, **miembro de grupo** y **otros**.

También tiene tres permisos: **lectura**, **escritura** y **ejecución**.

Si un propietario quiere conceder todos los permisos a todos los usuarios, el comando tiene el siguiente aspecto:

```
chmod -rwxrwxrwx nota.txt
```

29. Comando chown

El comando **chown** te permite cambiar la propiedad de un archivo, directorio o enlace simbólico al nombre de usuario especificado. Esta es la sintaxis:

```
chown [opción] owner[:grupo] archivo(s)
```

Por ejemplo, para que **usuariolinux2** sea el propietario de **nombrearchivo.txt**, utiliza:

```
chown usuariolinux2 nombrearchivo.txt
```

30. Comandos useradd y userdel

Utiliza **useradd** para crear una nueva cuenta de usuario Linux y cambia su contraseña con el comando **passwd**. Éstas son las sintaxis:

```
useradd [opción] nombredeusuario
passwd nombredeusuario
```

Tanto el comando **useradd** como el **passwd** requieren privilegios sudo. Para eliminar un usuario, utiliza el comando **userdel**:

```
Userdel nombredeusuario
```

31. Comando df

Utiliza el comando **df** para comprobar el uso de espacio en disco de un sistema Linux en porcentaje y en kilobytes (**KB)**. Ésta es la sintaxis:

```
df [opciones] [archivo]
```

Si no especificas el elemento, este comando mostrará información sobre cada sistema de archivos montado. Estas son algunas opciones aceptables:

- **-m:** muestra información sobre el uso del sistema de archivos en **MBs**.
- **-k:** imprime el uso del sistema de archivos en **KBs**.
- **-T:** muestra el **tipo de** sistema de archivos en una nueva columna.

32. Comando du

Utiliza **du** para comprobar el consumo de almacenamiento de un archivo o directorio. Recuerda especificar la ruta del directorio cuando utilices este comando, por ejemplo:

```
du /inicio/usuario/documentos
```

El comando **du** tiene varias opciones, como:

- **-s:** muestra el tamaño total de la carpeta especificada.
- **-m:** proporciona información de carpetas y archivos en **MB.**

- **-k:** muestra la información en **KB**.
- **-h:** informa de la fecha de última modificación de las carpetas y archivos mostrados.

33. Comando top

El comando **top** muestra los procesos en ejecución y el estado del sistema en tiempo real, incluida la utilización de recursos. Ayuda a identificar los procesos que consumen muchos recursos, permitiéndote desactivarlos fácilmente.
Para ejecutar el comando, introduce **top** en tu **interfaz de línea de comandos (CLI)**.

34. Comando htop

El comando **htop** es un programa interactivo para supervisar los recursos del sistema y los procesos del servidor. A diferencia de **top**, ofrece funciones adicionales como el manejo con el ratón e indicadores visuales.
Ésta es la sintaxis del comando:

```
htop [opciones]
```

Admite opciones como:

- **-d:** muestra el retardo entre actualizaciones en décimas de segundo.
- **-C:** activa el modo monocromo.

- **-h:** muestra el mensaje de ayuda y salidas.

35. Comando ps

El comando ps crea una instantánea de todos los procesos en ejecución de tu sistema. Ejecutándolo sin una opción o argumento se listarán los procesos en ejecución en el intérprete de comandos con la siguiente información:

- ID único del proceso (**PID**).
- Tipo de terminal (**TTY**).
- Duración (**TIME**).
- Comando que lanza el proceso (**CMD**).

El comando **ps** acepta varias opciones, entre ellas:

- **-T:** muestra todos los procesos asociados a la sesión shell actual.
- **-u nombredeusuario:** lista los procesos asociados a un usuario concreto.
- **-A:** Muestra todos los procesos en ejecución.

Consejo profesional

Aprende a utilizar los comandos **top**, **htop** y **ps** para **comprobar los procesos en ejecución en un sistema Linux**.

36. Comando uname

El comando **uname** o **nombre unix** imprime información sobre tu máquina, incluyendo su hardware, el nombre del sistema y el núcleo Linux. Ésta es la sintaxis básica:

```
uname [opción]
```

Aunque puedes utilizarlo sin opción, añade lo siguiente para modificar el comando:

- **-a:** imprime toda la información del sistema.
- **-s:** muestra el nombre del núcleo.
- **-n:** muestra el nombre de host del nodo del sistema.

37. Comando hostname

Ejecuta el comando **hostname** para mostrar el nombre de host del sistema. Ésta es la sintaxis:

```
hostname [opción]
```

Puedes ejecutarlo sin opción o utilizar lo siguiente:

- **-a:** muestra el alias del nombre de host.
- **-A:** muestra el Nombre de Dominio Completamente Cualificado (FQDN) de la máquina.
- **-i:** muestra la dirección IP de la máquina.

38. Comando time

Utiliza **time** para medir el tiempo de ejecución de los comandos. Esta es la sintaxis:

```
time [nombrecomando]
```

Para medir una serie de comandos, sepáralos utilizando punto y coma o doble ampersands (**&&**). Por ejemplo, mediremos el tiempo total de ejecución de los comandos **cd**, **touch** y **chmod**:

```
time cd /inicio/directorio/ruta; touch bashscript.sh; chmod +x bashscript.sh
```

39. Comando systemctl

El comando **systemctl** te permite gestionar los servicios instalados en tu sistema Linux. Esta es la sintaxis básica:

```
systemctl [nombrecomando] [nombre_servicio]
```

Para utilizar el comando, el usuario debe tener privilegios de **root**. Tiene varios usos, como iniciar, reiniciar y terminar un servicio. También puedes comprobar el estado de un servicio y sus dependencias.

El comando **systemctl** sólo está disponible en las distribuciones Linux con el **sistema de init Systemd**. Consulta nuestro artículo sobre cómo **listar y gestionar los servicios de linux** para saber más de los comandos de otros sistemas.

40. Comando Watch

El comando **watch** permite al usuario ejecutar continuamente otra utilidad en un intervalo específico e imprimir los resultados como salida estándar. Ésta es la sintaxis:

```
watch [opción] command
```

Es útil para controlar los cambios en la salida de los comandos. Para modificar su **comportamiento**, utiliza las siguientes opciones:

- **-d:** muestra las diferencias entre ejecuciones de comandos.
- **-n:** cambia el intervalo predeterminado de dos segundos.
- **-t:** desactiva la cabecera que contiene el intervalo de tiempo, el comando, la marca de tiempo y el nombre de host.

41. Comando jobs

El comando **jobs** muestra los procesos en ejecución de un intérprete de comandos con sus estados. Sólo está disponible en los shells **csh**, **bash**, **tcsh** y **ksh**. Ésta es la sintaxis básica:

```
jobs [opciones] jobID
```

Para comprobar el estado de los trabajos en el shell actual, introduce **jobs** sin ningún argumento en Terminal. El comando devolverá una salida vacía si tu sistema no tiene trabajos en ejecución. También puedes añadir las siguientes opciones:

- **-l:** lista los ID de proceso y su información.
- **-n:** muestra los trabajos cuyo estado ha cambiado desde la última notificación.
- **-p:** sólo muestra los ID de los procesos.

42. Comando kill

Utiliza el comando **kill** para terminar un programa que no responde utilizando su número de identificación (PID). Para comprobar el PID, ejecuta el siguiente comando:

```
ps ux
```

Para detener el programa, introduce la siguiente sintaxis:

```
kill [opcion_señal] pid
```

Existen 64 señales para terminar un programa, pero **SIGTERM** y **SIGKILL** son las más comunes. **SIGTERM** es la señal por defecto que permite al programa guardar su progreso antes de detenerse. Mientras tanto, **SIGKILL** obliga a los programas a detenerse y descartar el progreso no guardado.

43. Comando shutdown

El comando **shutdown** de Linux te permite apagar o reiniciar tu sistema a una hora determinada. Esta es la sintaxis:

```
shutdown [opción] [tiempo] "mensaje"
```

Puedes utilizar una hora absoluta en formato de 24 horas o una relativa como **+5** para programarlo en cinco minutos. El **mensaje** es una notificación enviada a los usuarios conectados sobre el apagado del sistema.
En lugar de apagar, reinicia el sistema utilizando la opción **-r**. Para cancelar un reinicio programado, ejecuta el comando con la opción **-c**.

44. Comando ping

El comando **ping** es uno de los más utilizados en Linux. Te permite comprobar si se puede acceder a una red o a un servidor, lo que resulta útil para solucionar problemas de conectividad. Aquí tienes la sintaxis:

```
ping [opción] [hostname_o_dirección_IP]
```

Por ejemplo, ejecuta lo siguiente para comprobar la conexión y el tiempo de respuesta a **Google**:

```
ping google.com
```

45. Comando wget

Utiliza el comando **wget** para descargar archivos de Internet mediante los protocolos **HTTP**, **HTTPS** o **FTP**. Esta es la sintaxis:

```
wget [opción] [url]
```

Por ejemplo, introduce lo siguiente para descargar la **última versión de WordPress**:

```
wget https://wordpress.org/latest.zip
```

46. Comando curl

El comando **curl** transfiere datos entre servidores. Su uso habitual es para recuperar el contenido de una página web en tu sistema utilizando su URL. Esta es la sintaxis:

```
curl [opción] URL
```

Sin embargo, puedes añadir varias opciones para modificar el comportamiento del comando **curl** para otras tareas. Algunas de las más populares son:

- **-o** o **-O:** descarga archivos desde una URL.
- **-X:** cambia el método HTTP GET por defecto.
- **-H:** envía una cabecera personalizada a la URL.
- **-F:** sube un archivo al destino especificado.

47. Comando scp

El comando **scp** copia de forma segura archivos o directorios entre sistemas a través de una red. Ésta es la sintaxis:

```
scp [opción] [origen nombreusuario@IP]:/[directorio y nombre de archivo] [destino nombreusuario@IP]:/[directorio de destino].
```

Para una máquina local, omite el nombre de host y la dirección IP. Utiliza las siguientes opciones para modificar el comportamiento de la copia:

- **-P:** cambia el puerto para copiar. El valor por defecto es **22**.
- **-l:** limita el ancho de banda del comando **scp**.
- **-C:** comprime los datos transferidos para hacerlos más pequeños.

48. Comando rsync

El comando **rsync** te permite sincronizar archivos o carpetas entre dos destinos para asegurarte de que tienen el mismo contenido. Esta es la sintaxis:

```
rsync [opciones] origen destino
```

Si tu destino u origen es una carpeta, introduce la ruta del directorio como **/inicio/directorio/ruta**. Para sincronizar un servidor remoto, utiliza su nombre de host y su dirección IP, como **host@185.185.185.185**. Este comando tiene varias opciones:

- **-a:** activa el modo de archivo para conservar los permisos de los archivos, las fechas y otros atributos.
- **-v:** muestra información visual sobre el archivo transferido.
- **-z:** comprime los datos del archivo transferido para reducir su tamaño.

49. Comando ifconfig

El comando **ifconfig** te permite listar y configurar la interfaz de red de tu sistema. En las **distribuciones Linux** más recientes, es equivalente al comando **ip**. Ésta es la sintaxis básica:

```
ifconfig [interfaz] [opción]
```

Ejecutándolo sin argumentos muestra información sobre todas las interfaces de red de tu sistema. Para comprobar una interfaz concreta, añade su nombre como argumento sin opción. Para una tarea más específica, utiliza las siguientes opciones:

- **-s:** resume las interfaces de red y su configuración. Esta opción va antes del nombre de la interfaz.
- **up** y **down:** activa y desactiva una interfaz de red.
- **inet** e **inet6:** asigna una dirección IPv4 e IPv6 a una interfaz de red.
- **netmask:** especifica la máscara de subred a utilizar con una dirección IPv4.

50. Comando netstat

El comando **netstat** se utiliza para mostrar la información de red de tu sistema, como sockets y enrutamiento.

Ésta es la sintaxis del comando:

```
netstat [opción]
```

Utiliza varias opciones para modificar la información mostrada. Algunas de las más comunes son:

- **-a:** muestra los sockets en escucha y cerrados.
- **-t:** muestra las conexiones TCP.
- **-u:** lista las conexiones UDP.
- **-r:** muestra las tablas de enrutamiento.
- **-i:** muestra información sobre las interfaces de red.
- **-p:** lista los nombres de los programas y los ID de los procesos.

- **-c:** emite continuamente información de la red para su supervisión en tiempo real.

51. Comando traceroute

El comando **traceroute** rastrea la ruta de un paquete cuando se desplaza a otro host a través de una red. Te da información sobre los routers implicados y el tiempo de recorrido. Ésta es la sintaxis:

```
traceroute [opción] destino
```

Puedes utilizar un dominio, un nombre de host o una dirección IP como destino. Añade las siguientes opciones para una monitorización más detallada de los paquetes:

- **-m:** establece el máximo de saltos de cada paquete.
- **-n:** impide que el comando resuelva direcciones IP a nombres de host para un rastreo más rápido.
- **-I:** cambia los paquetes UDP por defecto a UCMP.
- **-w:** añade un tiempo de espera en segundos.

52. Comando nslookup

El comando **nslookup** consulta un servidor DNS para averiguar el dominio asociado a una dirección IP y viceversa. Ésta es la sintaxis:

```
nslookup [opciones] dominio-o-ip [servidor]
```

Si no especificas el servidor DNS a utilizar, **nslookup** utilizará el resolver por defecto de tu sistema o proveedor de servicios de Internet. Este comando admite varias opciones, algunas de las más utilizadas son:

- **-type=:** consulta información específica, como el tipo de dirección IP o el registro MX.
- **-port=:** establece el número de puerto del servidor DNS para la consulta.
- **-retry=:** repite la consulta un número determinado de veces en caso de fallo.
- **-debug:** activa el modo de depuración para proporcionar más información sobre la consulta.

53. Comando dig

El comando **dig** o **groper de información de dominio** recopila datos DNS de un dominio. A diferencia de **nslookup**, es más detallado y versátil. Ésta es la sintaxis:

```
dig [opción] objetivo [tipo_consulta]
```

Sustituye el **objetivo** por un nombre de dominio. Por defecto, este comando sólo muestra el tipo de registro **A**. Cambia **query_type** para consultar un tipo concreto o utiliza **ANY** para consultarlos todos. Para ejecutar una búsqueda DNS inversa, añade la opción **-x** y utiliza la dirección IP como destino.

54. Comando history

Introduce **history** para listar los comandos ejecutados anteriormente. Te permite reutilizar los comandos sin reescribirlos. Para utilizarlo, introduce esta sintaxis con privilegios sudo:

```
history [opción]
```

Para volver a ejecutar una utilidad concreta, introduce un signo de exclamación (**!**) seguido del número de lista del comando. Por ejemplo, utiliza lo siguiente para volver a ejecutar el comando **255**:

```
!255
```

Este comando admite muchas opciones, como

- **-c:** borra la lista del historial.
- **-d offset:** borra la entrada del historial en la posición **OFFSET**.
- **-a:** añade líneas de historial.

55. Comando man

El comando **man** proporciona un manual de usuario de cualquier utilidad de la Terminal de Linux, incluyendo sus nombres, descripciones y opciones. Consta de nueve secciones:

- Programas ejecutables o comandos shell

Comandos diferentes de Linux

- Llamadas al sistema
- Llamadas a la biblioteca
- Juegos
- Archivos especiales
- Formatos de archivo y convenciones
- Comandos de administración del sistema
- Rutinas del núcleo
- Varios

Ésta es la sintaxis del comando:

```
man [opción] [número_sección] nombre_comando
```

Si sólo utilizas el nombre del comando como parámetro, Terminal muestra el manual de usuario completo. Aquí tienes un comando de ejemplo para consultar la sección **1** del manual del comando **ls**:

```
man 1 ls
```

56. Comando echo

El comando **echo** muestra una línea de texto como salida estándar. Ésta es la sintaxis básica del comando:

```
echo [opción] [cadena]
```

Por ejemplo, puedes mostrar **Tutoriales Hostinger** introduciendo:

```
echo "Tutoriales Hostinger"
```

Este comando admite muchas opciones como:

- **-n:** muestra la salida sin la nueva línea final.
- **-e:** activa la interpretación de los siguientes escapes de barra invertida:
- **\b:** Elimina los espacios entre un texto.
- **\c:** no produce ningún otro resultado.

57. Comando ln

El comando **ln** te permite crear enlaces entre archivos o directorios para simplificar la gestión del sistema. Esta es la sintaxis:

```
ln [opción] [origen] [destino]
```

El comando creará el archivo o directorio de destino y lo enlazará con el origen. Por defecto, crea un enlace duro, lo que significa que el nuevo elemento se conecta al mismo bloque de datos que el origen.

58. Comandos alias y unalias

El comando alias indica al shell que sustituya una cadena por otra, lo que te permite crear un acceso directo a un programa, nombre de archivo o texto. Esta es la sintaxis:

```
alias nombre=cadena
```

Por ejemplo, introduce lo siguiente para que **k** sea el alias del comando **kill**:

```
alias k='kill'
```

Este comando no da ninguna salida. Para comprobar el alias asociado a un comando, ejecuta lo siguiente:

```
alias nombre_comando
```

Para eliminar un alias existente, utiliza el comando **unalias** con la siguiente sintaxis:

```
unalias [nombre_alias]
```

59. Comando cal

El comando **cal** muestra un calendario en el Terminal de Linux. Mostrará la fecha actual si no especificas el mes y el año. Ésta es la sintaxis:

```
cal [opción] [mes] [año]
```

El mes está en la representación numérica del **1 al 12**. Para modificar la salida del comando, añade las siguientes opciones:

- **-1:** muestra el calendario en una sola línea.
- **-3:** muestra el mes anterior, el actual y el siguiente.
- **-A** y **-B:** muestra el número especificado de meses posteriores y anteriores al actual.
- **-m:** inicia el calendario con el lunes en lugar del domingo.

60. Comando apt-get

apt-get es una herramienta de línea de comandos para manejar las bibliotecas de la Herramienta de Paquetes Avanzados (APT) en Linux basado en Debian, como Ubuntu. Requiere privilegios **sudo** o de **root**.
Este comando de Linux te permite gestionar, actualizar, eliminar e instalar software, incluidas sus dependencias. Ésta es la sintaxis principal:

```
apt-get [opciones] (comando)
```

Estos son los comandos más comunes que se utilizan con **apt-get**:

- **update:** sincroniza los archivos del paquete desde sus fuentes.
- **upgrade:** instala la última versión de todos los paquetes instalados.
- **check**: actualiza la caché de paquetes y comprueba las dependencias rotas.

Trucos y consejos sobre comandos Linux

Aquí tienes algunos consejos para utilizar los comandos de Linux y el Terminal para mejorar la eficacia de la gestión de tu sistema:

- Añade la opción **-help** para listar el uso completo de un comando.
- Utiliza el comando **exit** para cerrar Terminal.
- Introduce el comando **clear** para limpiar la pantalla del Terminal.
- Pulsa el botón **Tab** para autocompletar después de introducir un comando con un argumento.
- Utiliza **Ctrl + C** para terminar un comando en ejecución.
- Pulsa **Ctrl + Z** para pausar un comando de trabajo.
- Utiliza **Ctrl + A** para desplazarte al principio de la línea.
- Pulsa **Ctrl + E** para ir al final de la línea.
- Separa varios comandos utilizando **punto y coma** (**;**) o **doble ampersands** (**&&**).

Extraído de Hostinger tutoriales