Manual básico de Linux

María Bazán Medina

Contenido

1. Navegación y manipulación de directorios

1.1. `pwd` – Imprimir el directorio actual

Este comando muestra la ruta del directorio en el que estás actualmente.

```
$ pwd
/home/usuario
```

1.2. `ls` – Listar archivos y directorios

Muestra el contenido del directorio actual. Existen opciones para ver detalles adicionales.

```
$ ls
archivo1.txt  archivo2.txt  directorio1

$ ls -l   # Lista con detalles (permisos, dueño, tamaño)
total 4
-rw-r--r-- 1 usuario usuario  0 Oct 20 12:00 archivo1.txt
drwxr-xr-x 2 usuario usuario 4096 Oct 20 12:10 directorio1

$ ls -a   # Muestra archivos ocultos
.  ..  .archivo_oculto  archivo1.txt
```

1.3. `cd` – Cambiar de directorio

Navega entre directorios.

```
$ cd /home/usuario/documentos   # Cambia a "documentos"
$ cd ..   # Retrocede un directorio
$ cd ~    # Vuelve al directorio home del usuario
```

1.4. `mkdir` – Crear un directorio

Crea un nuevo directorio.

```
$ mkdir nuevo_directorio
```

1.5. `rmdir` – Eliminar un directorio vacío

```
$ rmdir nuevo_directorio
```

---

2. Manipulación de archivos

2.1. `touch` – Crear archivos vacíos

Crea un archivo vacío o actualiza la fecha de modificación de un archivo existente.

```
$ touch archivo_nuevo.txt
```

2.2. `cp` – Copiar archivos y directorios

Copia archivos de un lugar a otro.

```
$ cp archivo1.txt /ruta/destino/   # Copia archivo1.txt a otra ruta
$ cp -r directorio1 /ruta/destino/  # Copia recursiva (directorios)
```

2.3. `mv` – Mover o renombrar archivos

Sirve para mover archivos o renombrarlos.

```
$ mv archivo1.txt /ruta/destino/   # Mueve archivo1.txt a otra ruta
$ mv archivo1.txt archivo_renombrado.txt  # Renombra el archivo
```

2.4. `rm` – Eliminar archivos o directorios

Elimina archivos y directorios. Usa `-r` para eliminar directorios con contenido.

```
$ rm archivo1.txt   # Elimina un archivo
$ rm -r directorio1  # Elimina un directorio con todos sus
contenidos
```

---

3. Ver contenido de archivos

3.1. `cat` – Mostrar el contenido de un archivo

```
$ cat archivo1.txt
Este es el contenido de archivo1.txt
```

3.2. `less` – Leer archivos largos por secciones

Permite leer archivos grandes, paginando la salida.

```
$ less archivo1.txt
```

3.3. `head` y `tail` – Mostrar las primeras o últimas líneas de un archivo

```
$ head archivo1.txt  # Muestra las primeras 10 líneas
$ tail archivo1.txt  # Muestra las últimas 10 líneas
$ tail -n 5 archivo1.txt  # Muestra las últimas 5 líneas
```

---

4. Permisos y propiedad de archivos

4.1. `chmod` – Cambiar permisos de archivos o directorios

Los permisos de los archivos se dividen en lectura (r), escritura (w) y ejecución (x). Se pueden ajustar usando números o letras.

```
$ chmod 755 archivo1.txt  # Propietario: todos los permisos; otros:
lectura y ejecución
$ chmod u+x archivo1.txt  # Da permiso de ejecución al propietario
```

4.2. `chown` – Cambiar el dueño de un archivo o directorio

```
$ sudo chown usuario:grupo archivo1.txt  # Cambia el dueño del
archivo a "usuario"
```

---

5. Información del sistema

5.1. `df` – Mostrar espacio en disco

```
$ df -h
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        50G   20G   28G  42% /
```

5.2. `du` – Mostrar uso de espacio de directorios y archivos

```
$ du -sh carpeta1   # Muestra el tamaño de la carpeta
100M    carpeta1
```

5.3. `free` – Ver uso de memoria

```
$ free -h
               total        used        free      shared  buff/cache
available
Mem:            16G        8.4G        5.1G        520M        2.5G
6.9G
Swap:           2G          0B          2G
```

5.4. `top` – Monitorear procesos en tiempo real

```
$ top
```

5.5. `uname` – Mostrar información del sistema

```
$ uname -a   # Muestra información completa
Linux maquina 5.4.0-58-generic #64~18.04.1-Ubuntu SMP x86_64
GNU/Linux
```

---

6. Gestión de procesos

6.1. `ps` – Ver procesos en ejecución

```
$ ps aux   # Muestra todos los procesos
```

6.2. `kill` – Terminar procesos

```
$ kill 1234   # Termina el proceso con PID 1234
$ kill -9 1234  # Fuerza la terminación del proceso
```

6.3. `&` – Ejecutar un proceso en segundo plano

```
$ comando &   # Ejecuta el comando en segundo plano
```

6.4. `fg` y `bg` – Controlar procesos en segundo plano y primer plano

```
$ fg   # Trae un proceso en segundo plano al primer plano
$ bg   # Envía un proceso al segundo plano
```

---

7. Red y conexión

7.1. `ping` – Comprobar conectividad a una dirección IP o dominio

```
$ ping google.com
```

7.2. `ifconfig` – Mostrar configuración de red (se usa `ip` en distribuciones modernas)

```
$ ifconfig   # Muestra detalles de la red
```

---

8. Compresión y empaquetado

8.1. `tar` – Empaquetar o descomprimir archivos tar

```
$ tar -cvf archivo.tar carpeta/   # Crear un archivo .tar
$ tar -xvf archivo.tar   # Extraer un archivo .tar
$ tar -czvf archivo.tar.gz carpeta/   # Crear un archivo .tar.gz
$ tar -xzvf archivo.tar.gz   # Extraer un archivo .tar.gz
```

8.2. `zip` y `unzip` – Comprimir y descomprimir archivos zip

```
$ zip archivo.zip carpeta/   # Comprimir una carpeta en zip
$ unzip archivo.zip   # Descomprimir un archivo zip
```

---

9. Buscadores y filtros

9.1. `find` – Buscar archivos y directorios

```
$ find /ruta -name "*.txt"   # Buscar archivos .txt en una ruta
```

9.2. `grep` – Buscar patrones de texto dentro de archivos

```
$ grep "cadena" archivo.txt   # Buscar "cadena" en archivo.txt
$ grep -r "cadena" /ruta/   # Buscar recursivamente en un directorio
```

---

10. Otros útiles

10.1. `history` – Ver el historial de comandos ejecutados

```
$ history
```

10.2. `alias` – Crear un alias para un comando

```
$ alias ll="ls -l"   # Crea un alias para `ls -l`
$ unalias ll   # Elimina el alias
```

10.3. `sudo` – Ejecutar un comando como superusuario

```
$ sudo comando
```

11. Entradas y salidas en Linux

En Linux, todo es tratado como un archivo, incluidos los dispositivos de entrada y salida. Los programas que ejecutas en la terminal tienen tres flujos principales:

- **Entrada estándar (stdin)**: Recibe los datos de entrada. Por defecto, es el teclado.
- **Salida estándar (stdout)**: Muestra los resultados de la ejecución de un programa. Por defecto, se imprime en la pantalla (terminal).
- **Error estándar (stderr)**: Muestra los mensajes de error. También, por defecto, se muestra en la terminal.

Cada uno de estos flujos tiene un número asociado:

- **stdin**: `0`
- **stdout**: `1`
- **stderr**: `2`

11.1. Redirección de la Salida estándar (stdout)

Puedes redirigir la salida de un comando hacia un archivo en lugar de que aparezca en la pantalla. Esto se logra usando el operador `>`.

```
$ ls > lista_archivos.txt   # La salida de 'ls' se guarda en lista_archivos.txt
```

Este comando crea (o sobrescribe si ya existe) el archivo `lista_archivos.txt` con la salida del comando `ls`.

11.1.1. Agregar a un archivo en lugar de sobrescribir

Si quieres agregar la salida a un archivo sin sobrescribir su contenido, puedes usar `>>`.

```
$ echo "Texto nuevo" >> archivo.txt   # Agrega "Texto nuevo" al final de archivo.txt
```

11.2. Redirección de la Entrada estándar (stdin)

La entrada estándar se puede redirigir para que un comando lea datos desde un archivo en lugar de desde el teclado. El operador usado es `<`.

```
$ wc < archivo.txt   # El comando 'wc' cuenta las líneas, palabras y caracteres de archivo.txt
```

En este ejemplo, el comando `wc` recibe la entrada desde el archivo `archivo.txt` en lugar de esperar a que ingreses texto desde el teclado.

11.3. Redirección de Error estándar (stderr)

De forma similar a la salida estándar, puedes redirigir los mensajes de error a un archivo. Esto es útil para separar los errores de la salida normal. Para redirigir el error estándar se utiliza `2>`.

```
$ ls /directorio_no_existente 2> error_log.txt
```

En este ejemplo, el comando `ls` intenta listar el contenido de un directorio que no existe, y el mensaje de error se guarda en el archivo `error_log.txt` en lugar de mostrarse en la pantalla.

11.4. Redirección de Salida estándar y Error estándar al mismo archivo

A veces, quieres redirigir tanto la salida estándar como los errores al mismo archivo. Para hacer esto, usa `&>` o bien puedes combinar `>` y `2>` de la siguiente forma:

```
$ comando > archivo.txt 2>&1
```

O usando `&>`:

```
$ comando &> archivo.txt
```

En ambos casos, se redirige tanto la salida estándar como los errores a `archivo.txt`.

11.5. Ejemplo de uso combinado de entradas y salidas

Puedes combinar las redirecciones para leer desde un archivo, escribir en otro y manejar errores, todo en un solo comando.

```
$ cat < archivo_entrada.txt > archivo_salida.txt 2> error_log.txt
```

Aquí:

- `cat` lee la entrada desde `archivo_entrada.txt` (en lugar del teclado),
- la salida se guarda en `archivo_salida.txt`,
- y los errores, si ocurren, se guardan en `error_log.txt`.

11.6. Tuberías (pipes) con `|`

Las tuberías (`|`) permiten que la **salida de un comando se use como entrada para otro comando**. Esto es útil para procesar datos en cadena.

```
$ ls | grep ".txt"
```

Aquí, `ls` lista el contenido del directorio y su salida es "canalizada" al comando `grep`, que busca archivos con la extensión `.txt`.

Otro ejemplo:

```
$ cat archivo.txt | sort | uniq > archivo_salida.txt
```

Este comando:

1. Muestra el contenido de `archivo.txt`,
2. lo ordena con `sort`,
3. elimina las líneas duplicadas con `uniq`,
4. y finalmente guarda el resultado en `archivo_salida.txt`.

11.7. Usar `/dev/null` para descartar salidas

`/dev/null` es un archivo especial que descarta cualquier cosa que se escriba en él. Puede ser útil si no te interesa la salida o los errores de un comando.

```
$ comando > /dev/null 2>&1
```

En este ejemplo, se descarta tanto la salida estándar como los mensajes de error, efectivamente "silenciando" el comando.

11.8. Uso de la redirección en scripts

Las redirecciones se pueden usar ampliamente en scripts para gestionar la entrada y la salida de manera automatizada, permitiendo procesar archivos, generar informes o manejar errores sin intervención manual.

Ejemplo de script con redirecciones:

```
#!/bin/bash

# Leer la entrada desde un archivo
cat < entrada.txt > salida.txt 2> errores.log

# Mostrar el contenido de salida.txt
cat salida.txt
```

Este script lee desde `entrada.txt`, guarda la salida en `salida.txt`, y si hay errores, los guarda en `errores.log`.

12. Estructura de directorios en Linux

En Linux, el sistema de archivos tiene una estructura jerárquica que comienza en la raíz del sistema representada por `/`. Desde ahí, se despliega un árbol de directorios que sigue un estándar conocido como **Filesystem Hierarchy Standard (FHS)**. Cada directorio tiene un propósito específico. A continuación, se describe cada uno de los directorios principales.

12.1. `/` – Raíz del sistema

Este es el directorio raíz de todo el sistema de archivos. Todos los demás directorios y archivos se encuentran bajo `/`. Es el punto de partida de la jerarquía del sistema.

```
$ ls /
bin  boot  dev  etc  home  lib  media  mnt  opt  proc  root  run
sbin  srv  sys  tmp  usr  var
```

12.2. `/bin` – Binarios esenciales

Este directorio contiene los **binarios ejecutables esenciales** necesarios para que el sistema funcione en modo monousuario o para tareas básicas, como el arranque. Aquí encontrarás comandos básicos como `ls`, `cp`, `mv`, y `cat`.

```
$ ls /bin
bash  cp  ls  mv  rm  touch  etc.
```

12.3. `/sbin` – Binarios de administración del sistema

Al igual que `/bin`, pero este contiene los binarios esenciales que solo pueden ser ejecutados por el superusuario (**root**). Son herramientas para la administración del sistema como `fdisk`, `ifconfig`, y `reboot`.

```
$ ls /sbin
reboot  shutdown  ifconfig  iptables
```

12.4. `/boot` – Archivos de arranque

Este directorio contiene los archivos necesarios para iniciar el sistema, como el **núcleo del sistema (kernel)** y el cargador de arranque (**GRUB** o **LILO**). Modificar los archivos aquí puede hacer que el sistema no arranque correctamente.

```
$ ls /boot
vmlinuz  initrd.img  grub/
```

12.5. `/dev` – Archivos de dispositivos

Aquí se encuentran los **archivos de dispositivos** que representan los dispositivos de hardware de la máquina, como discos duros, USB, terminales, etc. Los dispositivos en Linux

se tratan como archivos, y en este directorio se encuentran los controladores para acceder a ellos.

Ejemplos de dispositivos:

- `/dev/sda` – Primer disco duro.
- `/dev/tty` – Terminales.

```
$ ls /dev
sda  tty  null  zero  etc.
```

12.6. `/etc` – Archivos de configuración

Este directorio contiene **archivos de configuración** del sistema y las aplicaciones instaladas. Los archivos de configuración suelen ser ficheros de texto, y aquí se configuran aspectos como la red, usuarios, servicios, entre otros.

- `/etc/passwd` – Información de usuarios.
- `/etc/fstab` – Información de sistemas de archivos montados.
- `/etc/hostname` – Nombre de la máquina.

```
$ ls /etc
passwd  fstab  hostname  ssh/  network/
```

12.7. `/home` – Directorios personales de los usuarios

Este directorio contiene los directorios personales de los usuarios del sistema. Por ejemplo, el usuario **juan** tendrá su carpeta personal en `/home/juan`, donde podrá almacenar sus archivos, configuraciones personales, y proyectos.

```
$ ls /home
juan/  maria/  pedro/
```

Dentro de `/home/usuario/` encontrarás subdirectorios como:

- `Documents/` – Documentos personales.

- `Downloads/` – Descargas.
- `Desktop/` – Escritorio.

12.8. `/lib` – Bibliotecas esenciales

Este directorio contiene las **bibliotecas compartidas** (similares a las DLL en Windows) que son necesarias para que los programas en `/bin` y `/sbin` funcionen. Además, contiene los módulos del núcleo (kernel).

```
$ ls /lib
ld-linux.so.2  libc.so.6  modules/
```

12.9. `/media` – Puntos de montaje de medios extraíbles

Este directorio se utiliza para montar dispositivos extraíbles como **CDs**, **DVDs**, y **memorias USB**. Cuando insertas un dispositivo de este tipo, se monta automáticamente en un subdirectorio dentro de `/media`.

```
$ ls /media
usb_drive/  cdrom/
```

12.10. `/mnt` – Montaje temporal

Este directorio es tradicionalmente utilizado para montar sistemas de archivos de manera temporal. Se usa a menudo por los administradores del sistema para montar particiones de discos u otros sistemas de archivos manualmente.

```
$ ls /mnt
backup_disk/
```

12.11. `/opt` – Paquetes opcionales

Aquí se instalan las **aplicaciones adicionales** o paquetes de software de terceros que no forman parte del sistema base. Por ejemplo, si instalas un programa comercial, es posible que sus archivos se coloquen en `/opt`.

```
$ ls /opt
google/  vmware/
```

12.12. `/proc` – Información del sistema y procesos

Este directorio es especial porque no contiene archivos "reales". Es un **sistema de archivos virtual** que refleja el estado del sistema y los procesos en tiempo real. Contiene información sobre el kernel, el hardware, y los procesos en ejecución.

- `/proc/cpuinfo` – Información sobre el procesador.
- `/proc/meminfo` – Información sobre el uso de memoria.
- `/proc/<PID>` – Información de un proceso específico.

```
$ ls /proc
cpuinfo  meminfo  1/  2/  3/
```

12.13. `/root` – Directorio personal del usuario root

Este es el **directorio personal del superusuario (root)**. A diferencia de los usuarios normales que tienen sus directorios en `/home`, el directorio personal de root se encuentra directamente en `/root`.

```
$ ls /root
.backup/  .bashrc  .ssh/
```

12.14. `/run` – Información de runtime

Este directorio contiene archivos e información sobre los procesos y el sistema que se utilizan mientras el sistema está en funcionamiento. Es un sistema de archivos temporal y su contenido suele ser borrado al reiniciar.

```
$ ls /run
systemd/  lock/  utmp  user/
```

12.15. `/srv` – Datos de servicios

El directorio `/srv` (service) contiene **datos específicos para servicios del sistema** como servidores web o FTP. Aquí se almacenan los archivos que sirven estos servicios a otros sistemas.

- `/srv/www/` – Archivos del servidor web.
- `/srv/ftp/` – Archivos del servidor FTP.

```
$ ls /srv
www/  ftp/
```

12.16. `/sys` – Información del sistema y hardware

Al igual que `/proc`, este es un **sistema de archivos virtual** que proporciona información sobre el **hardware** del sistema y otros elementos del kernel. Es utilizado por el kernel de Linux para interactuar con dispositivos conectados al sistema.

```
$ ls /sys
block/  class/  devices/  firmware/
```

12.17. `/tmp` – Archivos temporales

Este directorio se usa para almacenar **archivos temporales** que se crean y eliminan durante las operaciones normales del sistema. Su contenido suele ser eliminado cada vez que el sistema se reinicia.

```
$ ls /tmp
tempfile.txt  tmp_script.sh
```

12.18. `/usr` – Utilidades y aplicaciones de usuario

Este es uno de los directorios más grandes. Contiene los **programas de usuario** y **utilidades** que no son necesarios para el funcionamiento básico del sistema. En `/usr` se encuentra la mayoría de las aplicaciones instaladas por los usuarios y las bibliotecas compartidas.

Subdirectorios importantes:

- `/usr/bin/` – Programas ejecutables (no críticos).
- `/usr/sbin/` – Binarios de administración del sistema (para root).
- `/usr/lib/` – Bibliotecas compartidas.
- `/usr/share/` – Archivos compartidos (documentación, iconos, etc.).

```
$ ls /usr
bin/  lib/  sbin/  share/
```

12.19. `/var` – Archivos variables

El directorio `/var` (de "variable") contiene **archivos que cambian frecuentemente** como logs, colas de correo, archivos de spool de impresión y archivos temporales creados por aplicaciones en ejecución.

Subdirectorios comunes:

- `/var/log/` – Archivos de log del sistema.
- `/var/spool/` – Colas de impresión o correo.
- `/var/tmp/` – Archivos temporales que pueden sobrevivir a reinicios.

- `$ ls /var`
- `log/  spool/  tmp/`