Módulo 5 – Sistema de archivos

5.1. Introducción al sistema de archivos

5.1.1. Definición de un sistema de archivos

Un **sistema de archivos** es la estructura y método que un sistema operativo utiliza para controlar cómo se almacenan y recuperan los datos en un dispositivo de almacenamiento (como discos duros, SSDs o dispositivos de almacenamiento extraíbles). Sin un sistema de archivos, los datos almacenados en el dispositivo serían un bloque continuo de información sin organización, lo que haría imposible saber dónde comienza o termina un archivo.

El sistema de archivos se encarga de gestionar y organizar los datos en bloques o sectores del dispositivo de almacenamiento, asignando a cada archivo una localización física concreta y proporcionando una manera eficiente de acceder a ellos.

- **Ejemplo en GNU/Linux**: Los sistemas de archivos más comunes en GNU/Linux incluyen ext4, XFS, Btrfs, entre otros, que permiten no solo almacenar datos, sino también gestionar metadatos (como permisos, fechas de creación y modificación) y realizar operaciones avanzadas como la compresión o snapshots.

5.1.2. Importancia en la gestión de datos y recursos

El sistema de archivos es fundamental en la **gestión de datos y recursos** del sistema operativo por varias razones:

- **Organización:** Permite la creación de una jerarquía de archivos y directorios que facilita la organización lógica de la información. Los usuarios pueden crear, modificar, mover y eliminar archivos de manera eficiente.
- **Acceso eficiente:** Los sistemas de archivos optimizan el tiempo de acceso a los datos, almacenando y recuperando la información de manera organizada. Esto incluye índices o estructuras de árbol (como en ext4) que permiten localizar rápidamente un archivo en un disco con millones de archivos.
- **Gestión de espacio:** El sistema de archivos administra el uso del espacio en disco. Distribuye los archivos en bloques físicos del dispositivo de almacenamiento y, en sistemas avanzados como Btrfs, permite comprimir archivos y gestionar volúmenes dinámicos.
- **Seguridad:** Proporciona mecanismos de seguridad a través de los permisos de acceso (lectura, escritura, ejecución) y características avanzadas como el SUID y el SGID. Además, sistemas de archivos como ext4 permiten características de **journal** (registro de transacciones) para evitar la corrupción de datos en caso de una falla del sistema.

- **Recuperación de fallos:** Muchos sistemas de archivos modernos como ext4 o Btrfs implementan funciones de **journaling** o **snapshots** que ayudan a evitar la pérdida de datos tras un fallo del sistema o una interrupción inesperada.

5.2. Sistemas de archivos soportados en GNU/Linux

GNU/Linux es compatible con una amplia gama de sistemas de archivos, cada uno diseñado para diferentes necesidades y casos de uso. Aquí detallamos los sistemas de archivos más comunes, destacando sus ventajas y desventajas para comprender mejor cuál es el adecuado en cada situación.

5.2.1. ext4 (Fourth Extended File System)

Descripción: ext4 es el sistema de archivos por defecto en muchas distribuciones de Linux y el sucesor de ext3. Está optimizado para la velocidad y la eficiencia en el manejo de archivos.

- **Ventajas:**
    - **Compatibilidad amplia**: Soportado de forma nativa por la mayoría de las distribuciones de Linux.
    - **Journaling**: Implementa un sistema de registro de transacciones (journal), que ayuda a proteger la integridad de los datos en caso de un fallo.
    - **Gran capacidad de almacenamiento**: Puede manejar sistemas de archivos de hasta 1 exabyte y archivos individuales de hasta 16 terabytes.
    - **Desfragmentación**: Mejorado en comparación con ext3, lo que permite reducir la fragmentación en disco.
- **Desventajas:**
    - **No es un sistema de archivos COW (Copy-on-Write)**: No permite snapshots nativos como Btrfs o ZFS.
    - **Limitado en características avanzadas**: Ext4 carece de ciertas funcionalidades avanzadas de gestión de volúmenes dinámicos y compresión de datos.

5.2.2. XFS (Extended File System)

Descripción: XFS es un sistema de archivos de alto rendimiento desarrollado por SGI, optimizado para archivos grandes y excelente en operaciones de lectura/escritura de archivos secuenciales.

- **Ventajas:**
    - **Rendimiento en archivos grandes**: Ideal para aplicaciones que manejan grandes volúmenes de datos, como bases de datos y procesamiento multimedia.

- **Alto rendimiento en Lectura/Escritura**: Gracias a su diseño para gestionar eficientemente grandes bloques de datos.
- **Gestión de espacio avanzada**: Permite la administración dinámica del espacio en disco, facilitando la expansión de volúmenes en tiempo real.

- **Desventajas:**
  - **Complejidad en recuperación de datos**: La recuperación puede ser difícil y lenta en caso de fallo.
  - **No permite reducir el tamaño de la partición**: XFS solo permite la expansión, pero no la contracción de particiones.
  - **Menor flexibilidad para escritorios de uso diario**: Aunque es excelente en servidores, puede no ser tan versátil en entornos de uso general.

5.2.3. Btrfs (B-tree File System)

Descripción: Btrfs es un sistema de archivos de próxima generación, diseñado para ofrecer funcionalidades avanzadas como snapshots, compresión y gestión de volúmenes.

- **Ventajas:**
  - **Copy-on-Write (COW)**: Soporta snapshots nativos, lo que facilita la realización de copias de seguridad incrementales sin duplicar el espacio en disco.
  - **Compresión transparente**: Ahorra espacio en disco al comprimir datos automáticamente.
  - **Gestión de volúmenes y subvolúmenes**: Permite crear y gestionar subvolúmenes dentro de un mismo sistema de archivos.
  - **Detección y corrección de errores**: Implementa sumas de verificación para los datos y metadatos, lo que facilita la detección y corrección de errores en disco.
- **Desventajas:**
  - **Inestabilidad relativa**: Aunque ha mejorado, Btrfs aún tiene limitaciones de estabilidad en comparación con ext4 en ciertos escenarios.
  - **Rendimiento menor en archivos pequeños**: Puede ser más lento en operaciones que manejan numerosos archivos pequeños, como las cargas de trabajo de servidores de bases de datos.
  - **Fragmentación**: Debido a su diseño COW, puede experimentar mayor fragmentación que otros sistemas de archivos.

5.2.4. FAT32 y exFAT

Descripción: FAT32 y exFAT son sistemas de archivos diseñados por Microsoft, y son comunes en dispositivos extraíbles por su amplia compatibilidad con múltiples sistemas operativos.

- **Ventajas:**
    - **Alta compatibilidad**: FAT32 y exFAT son compatibles con la mayoría de los sistemas operativos, incluyendo Windows, macOS y Linux.
    - **Adecuado para dispositivos portátiles**: Ideal para unidades USB y tarjetas SD.
    - **Ligero**: Requiere menos recursos y es fácil de implementar en dispositivos con recursos limitados.
- **Desventajas:**
    - **Limitaciones en tamaño de archivo (FAT32)**: FAT32 no soporta archivos de más de 4 GB, lo cual es una limitación significativa en entornos modernos.
    - **Sin seguridad avanzada**: No soporta permisos avanzados o características de seguridad como el journaling.
    - **No es adecuado para sistemas críticos**: Su simplicidad lo hace inadecuado para entornos de servidor o almacenamiento a largo plazo.

5.2.5. NTFS (New Technology File System)

Descripción: NTFS es el sistema de archivos nativo de Windows, que ofrece funcionalidades avanzadas como permisos de archivos y journaling. Aunque es soportado en Linux, su rendimiento puede no ser óptimo en todos los casos.

- **Ventajas:**
    - **Permisos y seguridad avanzados**: Ofrece control de acceso granular y características avanzadas de seguridad.
    - **Journaling y recuperación de errores**: Ayuda a proteger la integridad de los datos en caso de fallos.
    - **Tamaño de archivo y partición elevado**: No tiene las limitaciones de tamaño de FAT32, soportando archivos y particiones muy grandes.
- **Desventajas:**
    - **Compatibilidad parcial en Linux**: Aunque Linux puede leer y escribir en NTFS, el soporte completo puede depender de herramientas como ntfs-3g.
    - **No diseñado para Linux**: Puede ser ineficiente o menos estable en Linux, especialmente en entornos de escritura intensiva.
    - **Menor flexibilidad**: No es tan flexible ni eficiente en Linux como ext4 o XFS, por lo que no es ideal para almacenamiento primario en Linux.

Resumen Comparativo

Sistema de Archivos	Principales Ventajas	Principales Desventajas	Casos de Uso Comunes
ext4	Journal, amplia compatibilidad, bajo riesgo	Sin COW, funciones avanzadas limitadas	General en Linux, escritorios y servidores
XFS	Alta eficiencia con archivos grandes	Difícil recuperación, solo expansión	Bases de datos, servidores multimedia
Btrfs	COW, snapshots, compresión, detección de errores	Fragmentación, estabilidad relativa	Backups, NAS, gestión de volúmenes
FAT32/exFAT	Amplia compatibilidad	Límite de 4 GB (FAT32), sin seguridad	USBs, dispositivos móviles
NTFS	Seguridad avanzada, journaling	Soporte limitado en Linux, rendimiento mixto	Doble boot, intercambio con Windows

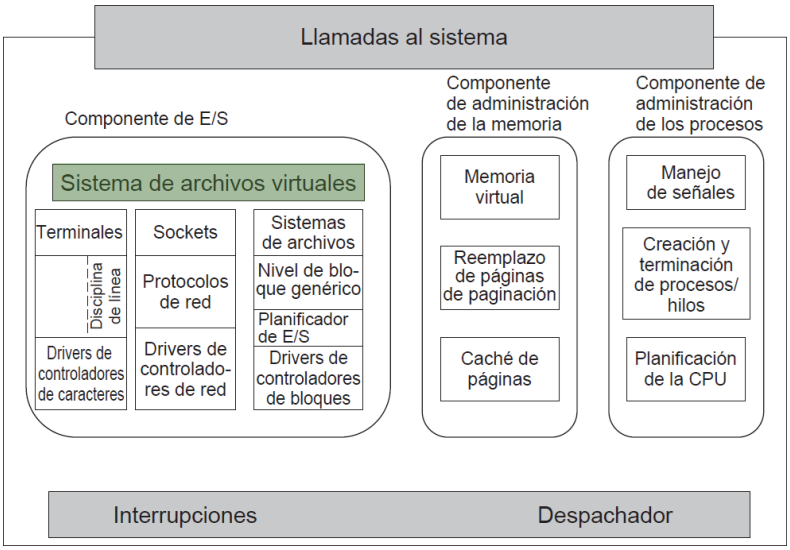
5.3. El Virtual File System (VFS)

5.3.1. Concepto y función del VFS en Linux

El **Virtual File System (VFS)** es una capa de abstracción dentro del kernel de Linux que permite al sistema operativo interactuar con múltiples sistemas de archivos de forma uniforme. En otras palabras, VFS actúa como una "interfaz virtual" que traduce las operaciones comunes sobre archivos (como abrir, leer, escribir, cerrar) a comandos específicos para el sistema de archivos subyacente, sin que el usuario o las aplicaciones necesiten conocer los detalles específicos de cada sistema. Esto hace de Linux un sistema operativo versátil y escalable. Además, simplifica la compatibilidad con dispositivos externos formateados en sistemas no nativos, haciendo de Linux una plataforma flexible tanto para usuarios domésticos como para entornos empresariales.

Principales funciones de VFS

- **Unificación de operaciones:** Permite a las aplicaciones realizar operaciones sobre archivos sin preocuparse por el sistema de archivos real en el que se encuentran. Así, se pueden realizar operaciones comunes en diferentes sistemas de archivos (como ext4, XFS, FAT32) sin necesidad de que la aplicación se adapte a cada uno.
- **Portabilidad y flexibilidad:** Gracias al VFS, Linux puede ser compatible con una gran variedad de sistemas de archivos, incluso aquellos diseñados para otros sistemas operativos, como NTFS o FAT. Esto es fundamental para escenarios de doble arranque y para la lectura y escritura en unidades extraíbles formateadas en diferentes sistemas de archivos.
- **Simplificación del kernel:** El VFS organiza y centraliza las operaciones de entrada/salida (I/O), aliviando al kernel de tener que implementar instrucciones específicas para cada sistema de archivos. En su lugar, solo se deben definir las operaciones comunes en VFS, y cada sistema de archivos proporciona una implementación de esas operaciones según sus propias reglas.



Ubicación, dentro del kernel, del VFS.

5.3.2. Estructura del VFS

Para entender cómo funciona, veamos sus estructuras principales:

- **Superbloque (superblock):** Almacena información general sobre el sistema de archivos montado, como su tamaño y tipo, y se carga cuando el sistema de archivos se monta.
- **Inodo (inode):** Cada archivo o directorio en Linux se representa mediante una estructura de datos llamada inodo, que contiene información como permisos, propietario, tamaño y punteros a los bloques de datos. VFS mantiene una tabla de inodos en memoria para acceder rápidamente a los archivos.
- **Dentry (directory entry):** Es una estructura que vincula el nombre del archivo con su inodo. Esto facilita la navegación en el sistema de archivos, permitiendo que se traduzcan nombres de archivos a sus respectivas ubicaciones físicas.
- **File:** Esta estructura representa los archivos abiertos en el sistema, almacenando información sobre su posición y estado en cada proceso que los abre.

5.3.3. Interacción con sistemas de archivos físicos

El VFS se encuentra entre las aplicaciones del usuario y los sistemas de archivos físicos, y cumple un papel de "traductor" en este proceso. La interacción funciona de la siguiente manera:

1. **Aplicaciones:** Cuando una aplicación realiza una operación (como abrir o leer un archivo), se envía una solicitud al sistema operativo. La aplicación no sabe (ni necesita saber) si el archivo está en un sistema ext4, XFS o FAT32, ya que el VFS se encarga de la traducción.
2. **VFS como interfaz común:** Al recibir la solicitud, el VFS determina el sistema de archivos al que pertenece el archivo y localiza el inodo correspondiente. Luego, envía la solicitud al controlador específico de ese sistema de archivos (por ejemplo, el controlador ext4) para ejecutar la operación de forma adecuada.

3. **Controladores de sistemas de archivos:** Los controladores de cada sistema de archivos implementan las operaciones específicas según las estructuras y particularidades del sistema (por ejemplo, el manejo de journal en ext4 o el sistema de compresión en Btrfs).
4. **Respuesta de los datos:** Finalmente, el controlador devuelve el resultado de la operación al VFS, que a su vez se lo transmite a la aplicación.

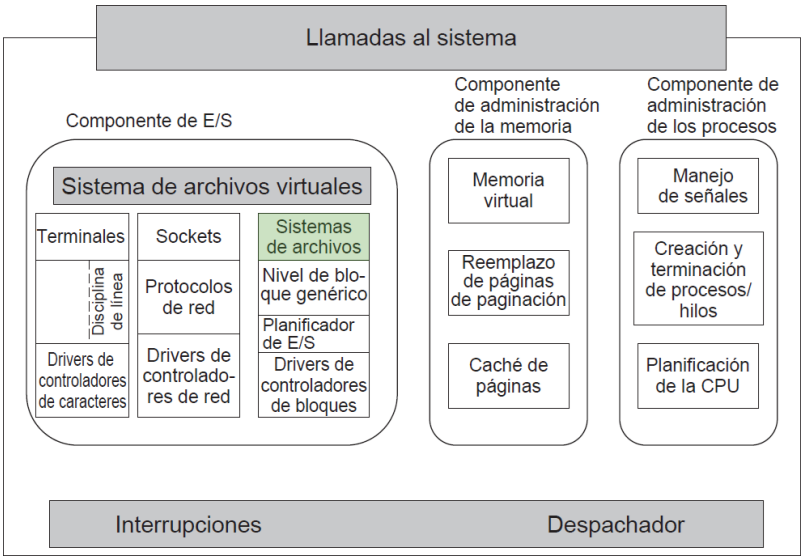
Ejemplo de uso

Supongamos que se desea abrir un archivo en una unidad formateada en NTFS en un sistema Linux:

- La aplicación envía una solicitud al VFS para abrir el archivo.
- El VFS verifica el sistema de archivos y delega la operación al controlador NTFS (utilizando, por ejemplo, `ntfs-3g` en Linux).
- El controlador NTFS localiza el archivo en la estructura NTFS y ejecuta la operación de apertura.
- El VFS devuelve el resultado a la aplicación, permitiéndole interactuar con el archivo como si estuviera en cualquier otro sistema de archivos.

5.4. Sistema de archivos ext4

A part**ir** de la versión 2.6.28 del kernel, **ext4** (Fourth Extended Filesystem) es el sistema de archivos utilizado por defecto en la mayoría de las distribuciones de GNU/Linux, por su estabilidad, eficiencia y soporte para sistemas de almacenamiento modernos. Diseñado para mejorar el rendimiento y la fiabilidad en comparación con ext3, ext4 incluye características avanzadas que permiten una administración de archivos y datos más efectiva.



Ubicación, dentro del kernel, del sistema de archivos.

5.4.1. Estructura básica de ext4

La estructura interna de ext4 se compone de varios bloques y estructuras de datos que garantizan la integridad y eficiencia del sistema de archivos. Vamos a desglosar sus elementos fundamentales:

- **Bloque de arranque (Boot Block):** Es el primer bloque de un sistema de archivos ext4 y contiene el **código de arranque** del sistema operativo. Aunque ext4 permite crear sistemas de archivos en cualquier partición, el bloque de arranque suele ubicarse en la primera partición y contiene instrucciones básicas para cargar el sistema operativo.
- **Superbloque:** El superbloque es una estructura crítica que almacena información esencial sobre el sistema de archivos, como:
    - Tamaño total del sistema de archivos.
    - Estado actual del sistema (montado o desmontado).
    - Número total de inodos y bloques de datos.
    - Información de journaling (registro de transacciones).

  Esta información es fundamental para que el sistema operativo interprete correctamente el sistema de archivos. Además, ext4 almacena copias del superbloque en varios lugares del disco para permitir una **recuperación rápida** en caso de errores.
- **Inodos:** Cada archivo y directorio en ext4 se representa mediante un inodo, que es una estructura de datos que almacena la **información de metadatos** del archivo, como:
    - Permisos de acceso y propietario.
    - Fechas de creación, modificación y acceso.
    - Tamaño del archivo.
    - Punteros a los bloques de datos donde se almacena el contenido del archivo.

  Los inodos permiten al sistema identificar y acceder a los archivos de manera eficiente, ya que ext4 utiliza una estructura de árbol para organizarlos.
- **Bloques de datos:** Son las unidades donde se almacena el contenido real de los archivos. Cada inodo contiene punteros a los bloques de datos correspondientes, lo que permite al sistema acceder a los datos de manera rápida. En ext4, los bloques de datos suelen tener un tamaño de 4 KB, aunque pueden ajustarse para optimizar el rendimiento según el uso del sistema de archivos.

5.4.2. Ventajas y características de ext4

Este sistema de archivos introduce varias mejoras significativas respecto a sus predecesores, especialmente en cuanto a rendimiento, escalabilidad y fiabilidad:

- **Journaling:** ext4 utiliza journaling para reducir la probabilidad de pérdida de datos en caso de fallos del sistema. El journaling guarda un registro de las operaciones de escritura en un área específica, permitiendo restaurar el sistema de archivos rápidamente en caso de fallo. Además, ext4 permite configurar el journaling en distintos niveles (modo journal, ordered, writeback), que ofrecen un equilibrio entre rendimiento y seguridad de los datos.
- **Soporte para archivos y sistemas de archivos grandes:** ext4 admite sistemas de archivos de hasta **1 exabyte** (10^{18} bytes, 10^{6} TB) y archivos individuales de hasta **16 terabytes**. Esta capacidad lo hace adecuado para servidores y sistemas de almacenamiento masivo.
- **Asignación de bloques extendida (Delayed Allocation):** ext4 utiliza una técnica de asignación de bloques retrasada, donde el sistema espera a tener toda la información del archivo antes de asignar los bloques de datos. Esto permite organizar los datos de manera óptima en el disco, reduciendo la fragmentación y mejorando el rendimiento.
- **Extents:** ext4 introduce los **extents**, que son grupos de bloques contiguos. En lugar de almacenar información de cada bloque individual, ext4 almacena el rango de bloques en un único descriptor, lo que reduce el tamaño de los metadatos y mejora la eficiencia al manejar archivos grandes y secuenciales.
- **Tiempo de montaje más rápido:** En sistemas grandes, ext4 reduce el tiempo necesario para montar el sistema de archivos, ya que optimiza la estructura de los metadatos, lo que permite acceder a grandes volúmenes de datos de forma más rápida.
- **Compatibilidad con ext2 y ext3:** ext4 es retrocompatible, lo que significa que es posible montar sistemas de archivos ext2 y ext3 en modo ext4 sin necesidad de conversión, facilitando la migración sin interrupciones en sistemas que utilicen versiones anteriores.

5.5. Estructura de directorios en GNU/Linux

La estructura de directorios en GNU/Linux se organiza siguiendo el **Estándar de Jerarquía de Sistema de Archivos (FHS, Filesystem Hierarchy Standard)**, que define una jerarquía lógica para los directorios y sus contenidos. Esta estructura es consistente en la mayoría de las distribuciones de Linux, lo cual facilita la navegación y administración del sistema.

5.5.1. Estándar de Jerarquía de Sistema de Archivos (FHS)

El FHS establece una disposición estándar de los directorios y archivos en el sistema, garantizando que tanto usuarios como aplicaciones sepan dónde encontrar o ubicar archivos específicos sin importar la distribución de Linux que utilicen. Este estándar categoriza los directorios en dos grandes grupos:

- **Directorios compartibles y no compartibles**:

  - *Compartibles*: Contienen archivos que pueden compartirse entre distintos sistemas (por ejemplo, `/usr y /home`).
  - *No compartibles*: Contienen archivos específicos de cada sistema, como los archivos de configuración en `/etc`.

- **Directorios variables y estáticos**:
  - *Variables*: Directorios que contienen datos que cambian frecuentemente, como logs y archivos temporales (por ejemplo, `/var`).
  - *Estáticos*: Contienen datos que rara vez cambian, como archivos binarios y de configuración de aplicaciones (por ejemplo, `/usr/bin`).

5.5.2. Descripción de directorios principales

A continuación, se detallan los directorios principales en GNU/Linux según la jerarquía definida por el FHS:

- **/ (Raíz)**: La raíz del sistema de archivos. Todos los demás directorios y archivos en Linux se encuentran bajo este directorio. Solo los administradores pueden escribir en la raíz, y su administración es fundamental para el buen funcionamiento del sistema.
- **/bin**: Contiene binarios esenciales del sistema, como comandos básicos (`ls`, `cp`, `mv`) necesarios para el arranque y recuperación del sistema. Estos programas están disponibles para todos los usuarios.
- **/boot**: Almacena archivos necesarios para el arranque del sistema, incluyendo el kernel, el cargador de arranque (GRUB, LILO), y archivos de configuración relacionados con el inicio.
- **/dev**: Contiene archivos de dispositivos que representan el hardware del sistema, como discos (`/dev/sda`), puertos serie, y dispositivos virtuales como `/dev/null`. Estos archivos permiten que el sistema y los usuarios interactúen con los dispositivos mediante operaciones de lectura y escritura.
- **/etc**: Almacena archivos de configuración del sistema y de aplicaciones. Los archivos en `/etc` son críticos para la configuración de servicios, redes, y aplicaciones del sistema. Ejemplos: `/etc/fstab` para el montaje de sistemas de archivos y `/etc/passwd` para las cuentas de usuario.
- **/home**: Directorio donde se almacenan los archivos personales de los usuarios del sistema. Cada usuario tiene su propia carpeta dentro de `/home` (`/home/usuario`) que contiene sus configuraciones personales y datos.
- **/lib**: Contiene bibliotecas esenciales compartidas utilizadas por los binarios en `/bin` y `/sbin`. Las bibliotecas en `/lib` permiten que los programas básicos del sistema se ejecuten correctamente.
- **/media**: Directorio utilizado para montar dispositivos de almacenamiento extraíbles, como unidades USB, discos duros externos y discos ópticos. Cada dispositivo montado se asigna a una subcarpeta en `/media`.

- **/mnt**: Se utiliza como un punto de montaje temporal para sistemas de archivos, por ejemplo, cuando un administrador desea montar manualmente una partición adicional o un dispositivo de red para tareas específicas.
- **/opt**: Almacena aplicaciones adicionales instaladas por el usuario o el administrador. En `/opt`, suelen ubicarse paquetes de software que no forman parte de la instalación estándar del sistema y que necesitan una estructura de directorios separada.
- **/proc**: Directorio virtual que contiene información sobre los procesos en ejecución y el estado del sistema. Los archivos dentro de `/proc` se generan dinámicamente por el kernel y permiten acceder a información del sistema, como el uso de memoria (`/proc/meminfo`) y CPU (`/proc/cpuinfo`).
- **/root**: Directorio personal del usuario root. Es el equivalente de `/home` para el administrador, pero, por seguridad, está fuera del directorio `/home`.
- **/run**: Contiene archivos de información sobre el sistema desde el último arranque, como sockets y archivos temporales necesarios para la inicialización de servicios.
- **/sbin**: Contiene binarios del sistema que son esenciales para la administración del sistema y que generalmente solo pueden ser ejecutados por el usuario root. Ejemplos de comandos en `/sbin` son `fdisk`, `mkfs` y `ifconfig`.
- **/srv**: Almacena datos específicos de servicios ofrecidos por el sistema, como sitios web en servidores web y repositorios de datos.
- **/tmp**: Directorio para almacenar archivos temporales. Cualquier usuario puede escribir en `/tmp`, y los archivos en este directorio se eliminan periódicamente.
- **/usr**: Almacena aplicaciones y archivos de usuario. Dentro de `/usr`, encontramos subdirectorios como:
    - **/usr/bin**: Contiene binarios de aplicaciones y comandos disponibles para todos los usuarios.
    - **/usr/lib**: Almacena bibliotecas compartidas utilizadas por programas en /usr/bin.
    - **/usr/local**: Reservado para software instalado manualmente, evitando así conflictos con los paquetes del sistema.
    - **/usr/share**: Contiene datos compartidos como documentación, archivos de configuración de ejemplo y otros datos no específicos del sistema.
- **/var**: Contiene archivos que cambian con frecuencia, como logs del sistema en `/var/log`, archivos de correos electrónicos en `/var/mail`, y directorios de almacenamiento temporal en `/var/tmp`.

5.6. Tipos de archivos en GNU/Linux

En GNU/Linux, los archivos se clasifican en varios tipos según su función y características. Esta organización es esencial para la administración del sistema y para entender cómo interactuar con distintos tipos de datos y dispositivos. A continuación, se detallan los tipos principales de archivos en GNU/Linux y cómo identificarlos.

5.6.1. Archivos normales

Los **archivos normales** son aquellos que contienen datos de usuario o del sistema. Pueden ser archivos de texto, imágenes, scripts, ejecutables, etc. Este tipo de archivo es el más común y se identifica con un guion – al comienzo de la salida del comando `ls -l`:

```
-rw-r--r-- 1 usuario usuario 1234 Oct 23 12:00 archivo.txt
```

Aquí, el – inicial indica que se trata de un archivo normal.

5.6.2. Directorios

Los **directorios** son carpetas que contienen otros archivos y subdirectorios. Un directorio se identifica con la letra `d` al inicio de la salida de `ls -l`. Los directorios se usan para organizar el sistema de archivos y crear una estructura jerárquica:

```
drwxr-xr-x 2 usuario usuario 4096 Oct 23 12:00 documentos
```

La `d` al principio indica que se trata de un directorio.

5.6.3. Archivos especiales

GNU/Linux utiliza archivos especiales para representar dispositivos de hardware y otras funcionalidades del sistema. Existen dos tipos principales de archivos especiales:

- **Dispositivos de bloque**: Estos archivos representan dispositivos de almacenamiento, como discos duros o memorias USB, que pueden leerse y escribirse en bloques. Los archivos de dispositivos de bloque suelen ubicarse en el directorio `/dev` y se identifican con una b al inicio de la salida de `ls -l`:

  ```
  brw-rw---- 1 root disk 8, 1 Oct 23 12:00 /dev/sda1
  ```

- **Dispositivos de carácter**: Representan dispositivos que transmiten datos como un flujo de caracteres, como teclados, impresoras, y puertos serie. Los archivos de dispositivos de carácter también se encuentran en `/dev` y se identifican con una `c`:

  ```
  crw-rw-r-- 1 root tty 4, 1 Oct 23 12:00 /dev/tty1
  ```

5.6.4. Enlaces (Links)

En GNU/Linux, los enlaces o links permiten crear "alias" o "accesos directos" a archivos o directorios. Existen dos tipos principales de enlaces:

- **Enlaces duros (Hard Links)**: Un enlace duro apunta directamente a un inodo específico, que es el identificador de datos de un archivo en el sistema de archivos. Los enlaces duros son prácticamente idénticos al archivo original, y el archivo solo se elimina cuando se eliminan todos sus enlaces duros. Los enlaces duros solo

pueden crearse dentro del mismo sistema de archivos y no pueden apuntar a directorios.

- **Creación de un enlace duro**:

```
ln archivo_original enlace_duro
```

- **Enlaces simbólicos (Soft Links o Symbolic Links)**: Un enlace simbólico es un archivo que apunta a la ruta de otro archivo o directorio. Funcionan como accesos directos y pueden crearse entre diferentes sistemas de archivos o hacia directorios. Sin embargo, si el archivo original es eliminado, el enlace simbólico queda roto y no apunta a nada.

  - **Creación de un enlace simbólico**:

```
ln -s archivo_original enlace_simbólico
```

Con el comando `ls -l` se visualizaría así:

```
lrwxrwxrwx 1 user user 12 Oct 23 12:00 symbolic_link -> original_file
```

La `l` inicial indica que se trata de un enlace simbólico, y la flecha (`->`) muestra el archivo al que apunta.

Comando *ln* y sus opciones

El comando `ln` permite crear enlaces duros y simbólicos. Sus opciones más comunes incluyen:

- **Enlace Duro**:

```
ln archivo_original enlace_duro
```

  Este comando crea un enlace duro llamado `enlace_duro` que apunta a `archivo_original`.

- **Enlace Simbólico**:

```
ln -s archivo_original enlace_simbólico
```

  La opción `-s` especifica que el enlace debe ser simbólico. El enlace `enlace_simbólico` apunta a `archivo_original`.

Comando *ls* para ver y distinguir los links

Para identificar y diferenciar los enlaces, el comando `ls` con la opción `-l` es muy útil, ya que muestra:

- La letra `l` al inicio de la línea para indicar un enlace simbólico.
- La `->` seguida del archivo o directorio al que apunta el enlace simbólico.

Un ejemplo de `ls -l` con varios tipos de archivos es el siguiente:

```
$ ls -l
drwxr-xr-x 2 usuario usuario 4096 Oct 23 12:00 directorio
-rw-r--r-- 1 usuario usuario 1024 Oct 23 12:00 archivo.txt
lrwxrwxrwx 1 usuario usuario 12 Oct 23 12:00 enlace_simbólico -> archivo.txt
```

En este ejemplo:

- `directorio` es un directorio.
- `archivo.txt` es un archivo normal.
- `enlace_simbólico` es un enlace simbólico que apunta a `archivo.txt`.

5.7. Montaje de sistemas de archivos

En GNU/Linux, el **montaje de sistemas de archivos** es el proceso mediante el cual el sistema operativo hace accesible un dispositivo de almacenamiento (como un disco duro, SSD o unidad USB) en una ubicación específica del sistema de archivos. Esto se realiza mediante el comando `mount`, que asocia el dispositivo con un directorio llamado **punto de montaje**.

5.7.1. Uso del comando *mount*

El comando `mount` permite al administrador del sistema montar dispositivos y sistemas de archivos en puntos específicos del sistema. La sintaxis básica es:

```
mount [opciones] dispositivo punto_de_montaje
```

- **dispositivo**: Es la ruta del dispositivo de almacenamiento, generalmente ubicado en el directorio `/dev` (por ejemplo, `/dev/sda1` para la primera partición del primer disco duro).
- **punto_de_montaje**: Es el directorio donde se montará el dispositivo. Una vez montado, los archivos y carpetas del dispositivo serán accesibles desde este punto.

Supongamos que deseas montar la partición `/dev/sdb1` en el directorio `/media/usb`.

1. **Crear el Punto de Montaje**: Primero, asegúrate de que el directorio de montaje exista. Si no, créalo:

```
sudo mkdir -p /media/usb
```

2. **Montar el Dispositivo**: Utiliza el comando `mount` para montar el dispositivo en el punto de montaje:

```
sudo mount /dev/sdb1 /media/usb
```

3. **Acceso a los Archivos**: Una vez montado, puedes acceder a los archivos de la unidad desde `/media/usb`.

5.7.2. Opciones comunes de *mount*

- **Especificar el tipo de sistema de archivos**: Si el sistema de archivos no se detecta automáticamente, puedes especificarlo con `-t`. Por ejemplo, para montar un dispositivo en formato NTFS:

```
sudo mount -t ntfs /dev/sdb1 /media/usb
```

- **Montaje de solo lectura**: Para montar un dispositivo en modo solo lectura, utiliza la opción `-o ro`:

```
sudo mount -o ro /dev/sdb1 /media/usb
```

- **Especificar opciones de montaje**: Puedes pasar opciones adicionales con `-o`, como permisos o límites de tamaño:

```
sudo mount -o uid=1000,gid=1000 /dev/sdb1 /media/usb
```

5.7.3. Verificación del montaje

Después de montar un dispositivo, puedes verificar su estado con los comandos `df` y `mount`.

Comando df

El comando `df` muestra el uso del espacio en disco de todos los sistemas de archivos montados. Es útil para verificar que el dispositivo se haya montado correctamente y para ver el espacio disponible.

```
df -h
```

- **-h**: Muestra el tamaño en un formato legible, usando KB, MB o GB.

Salida esperada:

```
Filesystem      Size  Used Avail Use% Mounted on
/dev/sda1        50G   10G   38G  21% /
/dev/sdb1       100G   20G   80G  20% /media/usb
```

Aquí, puedes ver que `/dev/sdb1` está montado en `/media/usb` y el espacio disponible en el dispositivo.

Comando *mount*

El comando `mount` sin argumentos muestra todos los sistemas de archivos actualmente montados en el sistema, incluyendo sus opciones de montaje. Esto permite verificar tanto los dispositivos montados como los puntos de montaje activos.

```
mount
```

Salida esperada:

```
/dev/sda1 on / type ext4 (rw,relatime)
/dev/sdb1 on /media/usb type ntfs (rw,relatime)
```

Esta salida muestra que `/dev/sdb1` está montado en `/media/usb` con el tipo de sistema de archivos NTFS y las opciones de solo lectura (`rw`).

5.7.4. Desmontar un sistema de archivos con umount

Para desmontar un sistema de archivos y hacer que el dispositivo sea seguro para extraer, usa el comando `umount` seguido del dispositivo o el punto de montaje:

```
sudo umount /media/usb
```

Una vez desmontado, el dispositivo ya no será accesible desde el punto de montaje y puede retirarse del sistema de forma segura.

5.7.5. Comparación de montaje de sistemas de archivos: Linux vs. Windows

En **GNU/Linux**, el montaje de sistemas de archivos permite integrar diferentes dispositivos de almacenamiento en una **única jerarquía de directorios**. Por el contrario, en **Windows**, cada dispositivo de almacenamiento se presenta como una **unidad independiente** identificada por una letra (como `C:`, `D:`, etc.), y no se requiere un proceso de montaje manual.

Jerarquía de Directorios

- **Linux**:
  - Todo el sistema de archivos se estructura en una única jerarquía que parte de la **raíz /**. Los dispositivos de almacenamiento adicionales se montan dentro de esta jerarquía, en puntos de montaje específicos (como `/mnt`, `/media`, o cualquier directorio que se elija).
  - Cuando se monta un dispositivo (por ejemplo, una unidad USB), no aparece como una unidad separada. En su lugar, se accede a él a través del punto de montaje donde se integró. Esto permite que todas las ubicaciones del sistema de archivos se organicen de forma coherente y evita fragmentación visual.
- **Windows**:
  - En Windows, cada partición o dispositivo de almacenamiento (como discos duros, particiones, y dispositivos USB) se asigna a una **letra de unidad** (`C:`, `D:`, etc.). Esta convención proviene de las primeras versiones de MS-DOS y se mantiene como estándar en la organización de Windows.
  - La letra `C:` generalmente se asigna al disco donde se encuentra instalado el sistema operativo, y las unidades adicionales se asignan de manera secuencial (`D:`, `E:`, etc.).
  - La organización de letras de unidad es independiente y no tiene un punto de inicio común, como la raíz `/` en Linux. Cada unidad actúa como un "isla" de archivos que se accede directamente, y no se integra a una jerarquía central.

Proceso de Montaje

- **Linux**:
  - En GNU/Linux, el proceso de montaje es explícito. Los dispositivos deben ser montados en un punto de montaje para ser accesibles. Esto se puede hacer manualmente con el comando `mount` o automáticamente a través de configuraciones en el archivo `/etc/fstab`.
  - Este enfoque permite control granular sobre los permisos, tipos de acceso (lectura/escritura) y opciones específicas para cada dispositivo montado.
  - Una ventaja es que Linux permite montar sistemas de archivos con permisos específicos y configuraciones de acceso, como modo de solo

lectura (`ro`) o lectura/escritura (`rw`), lo cual es muy útil en entornos multiusuario y servidores.

- **Windows**:
  - Windows no requiere un proceso explícito de montaje. Cuando se conecta un dispositivo de almacenamiento, el sistema operativo lo detecta y automáticamente le asigna una letra de unidad.
  - Sin embargo, esta automatización limita la personalización. Por ejemplo, las unidades son de lectura/escritura por defecto y los permisos avanzados no son configurables para cada partición o dispositivo desde el nivel de sistema de archivos.
  - En Windows, no es posible "montar" una unidad en otra ubicación dentro de otra unidad, lo que limita la flexibilidad para organizar dispositivos y sistemas de archivos.

Integración de Dispositivos y Flexibilidad

- **Linux**:
  - La flexibilidad del sistema de montaje permite a los usuarios montar unidades dentro de otras carpetas, como `/home`, para que las aplicaciones y usuarios accedan a los archivos de forma natural sin cambiar su ruta de acceso.
  - Linux permite la integración de varios sistemas de archivos dentro de su jerarquía unificada, lo que permite, por ejemplo, montar una partición de Windows (NTFS) en `/mnt/windows` y acceder a ella de manera fluida.
- **Windows**:
  - La estructura de letras de unidad hace que cada partición o dispositivo de almacenamiento sea independiente, lo que limita la posibilidad de organizar los recursos de almacenamiento de manera centralizada.
  - Aunque Windows permite compartir unidades en red y asignarlas a letras de unidad adicionales, no ofrece una integración real dentro de una jerarquía unificada. La falta de montaje explícito también significa que los usuarios tienen menos control sobre cómo y dónde acceden a cada dispositivo.

Resumen de Diferencias

Característica	Linux (Montaje)	Windows (Letras de Unidad)
Jerarquía	Única jerarquía a partir de /	Estructura independiente por letra de unidad
Proceso de Montaje	Manual (comando mount) o automático (fstab)	Automático (asigna letra de unidad)
Flexibilidad de Montaje	Monta en cualquier directorio	Solo asignación de letras de unidad
Control de Opciones	Opciones de permisos y acceso configurables	Opciones limitadas (solo lectura/escritura)
Integración de Dispositivos	Montaje en puntos específicos de la jerarquía	Unidades aisladas, sin jerarquía compartida