UD 06.

LINUX: GESTIÓN DE ARCHIVOS

Sistemas Informáticos
DAW CFGS

Álvaro Maceda

a.macedaarranz@edu.gva.es

2022/2023

Versión:221113.2151

Licencia

Nomenclatura

A lo largo de esta unidad se utilizarán diferentes símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

ÿ Importante

ÿ Atención

ÿ Interesante

TABLA DE CONTENIDO

UT 06. LINUX: GESTIÓN DE ARCHIVOS

1.QUÉ ES UN ARCHIVO

En los sistemas informáticos, un archivo es simplemente una serie de bits ordenados. Como hemos visto en unidades anteriores, estos bits pueden representar cualquier tipo de información: texto, imágenes, programas, etc. Los archivos suelen almacenarse en sistemas de almacenamiento secundarios como discos duros o memorias USB.

Los archivos no se escriben secuencialmente en el disco: hay diferentes formas de administrar el espacio en disco, para diferentes propósitos: velocidad, confiabilidad, seguridad... Cada método conduce a una organización diferente del disco, lo que se denomina sistema de archivos.

Es importante tener en cuenta que los archivos son solo una serie de bits, y que son las aplicaciones las que interpretan su contenido como texto, imágenes, etc.

carpetas

Dado que un sistema puede requerir decenas de miles de archivos para funcionar, necesitamos una forma de organizarlos. La forma habitual de hacerlo es mediante el uso de carpetas. Las carpetas son contenedores donde podemos poner los archivos. También pueden contener otras carpetas, formando una estructura jerárquica. La información de la carpeta también se almacena en el sistema de archivos.

Ejemplo de una estructura de carpetas

2. SISTEMAS DE ARCHIVOS

No puede escribir un archivo en un disco. La mayoría de ellos presentan una interfaz que expone el disco como una serie de **sectores.** No puede escribir bits individuales en un disco, solo sectores completos. Los tamaños de sector más habituales son 4KB y 512 bytes.

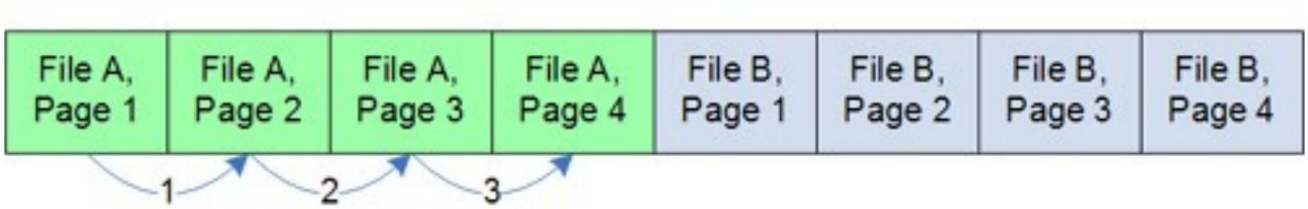
ÿ Un **sector** es la unidad más pequeña que un disco permite leer o escribir

Podríamos escribir los bytes de un archivo secuencialmente en el disco pero, si lo hiciéramos, podríamos tener problemas similares a los de la asignación de memoria, teniendo suficiente espacio dentro del disco pero fragmentado de alguna manera.

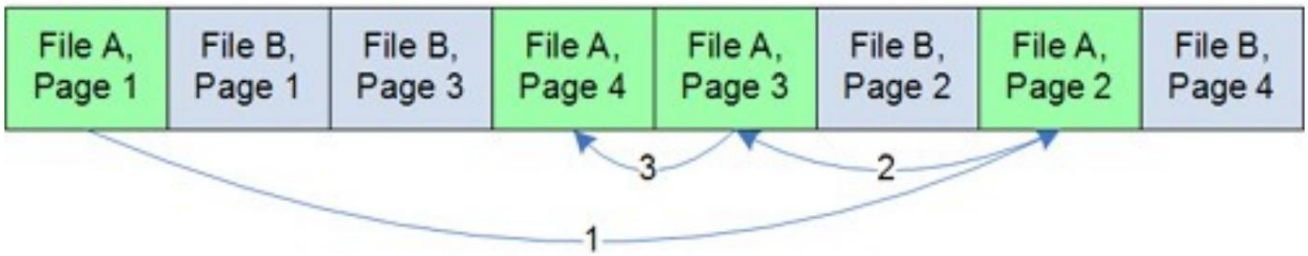
que no podremos almacenar archivos de cierto tamaño. Además, la estructura de carpetas también debe almacenarse en el sistema de archivos. Por lo tanto, los sistemas de archivos deben decidir cómo organizar los sectores del disco para almacenar información de manera eficiente.

Los diferentes sistemas de archivos tienen diferentes enfoques para organizar sus datos. Algunos sistemas de archivos son más rápidos que otros, algunos tienen funciones de seguridad adicionales y algunos admiten unidades de gran capacidad de almacenamiento, mientras que otros solo funcionan con unidades más pequeñas. Hay sistemas de archivos más robustos y resistentes a la corrupción de archivos, mientras que otros cambian la robustez por una mayor velocidad.

Lo que hacen la mayoría de los sistemas de archivos es algo similar a la paginación de la memoria: dividen el espacio del disco en bloques de cierto tamaño y el contenido del archivo se distribuye a través del disco utilizando tantos bloques como sea necesario.



Este sería un archivo almacenado secuencialmente



Un sistema de archivos puede dividir el archivo en varios bloques no contiguos en el disco

ÿ Un **bloque** o **clúster** es el tamaño mínimo que se puede leer o escribir en un archivo sistema

Por lo general, el tamaño del bloque es un múltiplo del tamaño del sector para mejorar el rendimiento.

2.1 Sistemas de archivos más comunes

Hay cientos de sistemas de archivos diferentes, pero algunos de ellos están muy difundidos y se utilizan en la mayoría de las computadoras:

- **FAT** (Tabla de Asignación de Archivos): Es relativamente simple desde el punto de vista técnico. Era el sistema de archivos predeterminado para Windows antes de Windows 2000, y hoy en día es compatible con todos los principales sistemas operativos. No puede funcionar con archivos o discos grandes.
- **exFAT** (Tabla de asignación de archivos extendida): diseñada por Microsoft, es capaz de administrar archivos más grandes que FAT. Es compatible con muchos sistemas operativos y dispositivos. Linux puede administrar este sistema de archivos, pero probablemente necesitará instalar algunos paquetes (generalmente exfat-fuse y exfat-utils)

- **NTFS** (Sistema de Archivos de Nueva Tecnología): Es el sistema de archivos usado por defecto para las computadoras Windows. Puede ser utilizado por máquinas Linux y Windows, y leído por computadoras Mac con controladores adicionales.

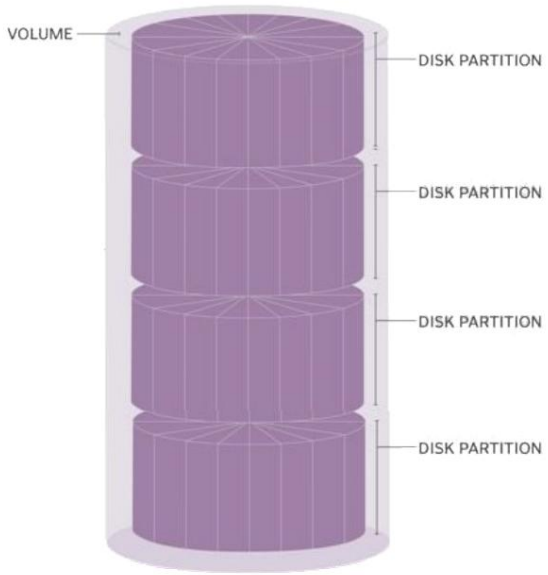
- **APFS** (Sistema de archivos de Apple): un sistema de archivos de propiedad que se utiliza en las computadoras Apple. tiene caracteristicas como instantáneas y encriptación.
- **ext4** (Sistema de archivos extendido): Es el sistema de archivos predeterminado para la mayoría de las distribuciones de Linux. Las computadoras Windows y Mac no pueden leerlo de forma predeterminada.
- **ZFS** (Sistema de archivos Zettabyte): es un sistema de archivos de nivel empresarial capaz de fusionar dispositivos, crear volúmenes RAID y agregar o eliminar discos del grupo de almacenamiento. Algunas distribuciones de Linux lo incluyen por defecto.

El sistema operativo debe ser capaz de interpretar la organización del sistema de archivos para poder trabajar con él. Si intentamos utilizar un sistema de archivos no compatible, no podremos leer ni escribir información. Por ejemplo, si intentamos usar un disco con el sistema de archivos ext4 en una máquina con Windows, no podremos hacerlo.

3.PARTICIONES

A veces no queremos usar el mismo sistema de archivos para todo el disco. Esto podría ser, por ejemplo, porque queremos usar diferentes sistemas operativos en la computadora. O, en el caso de UEFI, porque necesitamos una partición especial para arrancar el equipo. En ese caso, usamos particiones.

Una **partición** de disco es una región del disco duro que está separada de otras regiones similares. Las particiones permiten a los usuarios dividir un disco físico en espacios lógicos que se comportan más o menos como discos independientes.



Podemos dividir un disco en particiones, que se comportan como discos independientes.

3.1 La tabla de particiones

Hay un lugar especial al comienzo del disco para almacenar cómo se particiona el disco. Esta información se denomina **tabla de particiones.**

ÿ Puede romper su computadora al jugar con las particiones. Usa máquinas virtuales para experimentar con ellas.

Los principales estándares para almacenar la tabla de particiones son MBR y GPT.

ÿ Una partición puede ocupar todo el disco.

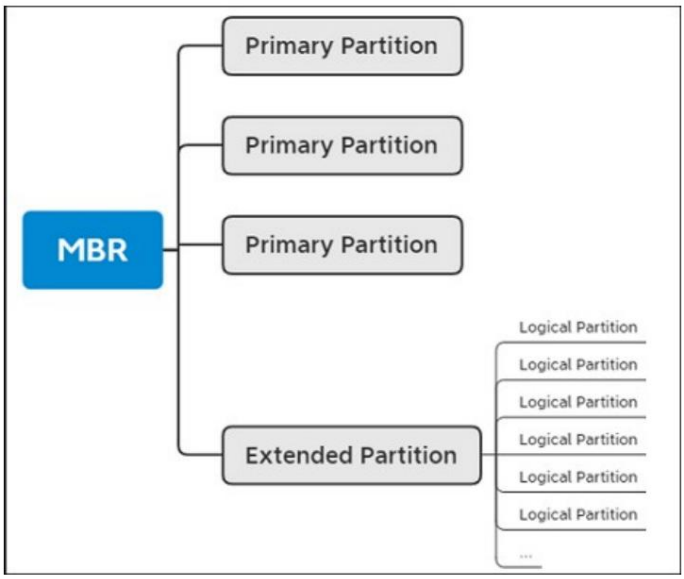
3.1.1 Registro de arranque maestro (MBR)

MBR significa Master Boot Record. Fue introducido por primera vez en 1983 con IBM PC DOS 2.0 Es un sistema heredado y ha sido reemplazado por GPT, pero todavía se usa en algunos dispositivos como unidades USB.

El MBR normalmente tiene un tamaño de 512 bytes, que era el tamaño anterior de un sector en un disco. Siempre está en el mismo lugar del disco: Cilindro 0, Cabeza 0, Sector 1 (los discos no mecánicos no tienen cilindros, cabezas o sectores, pero el MBR está en el primer espacio direccionable del disco). Además de la tabla de particiones, también incluye un programa que inicia el proceso de arranque del sistema operativo.

Las particiones MBR pueden ser de tres tipos: particiones primarias, particiones extendidas y particiones lógicas. La razón de esto es que MBR solo permite 4 particiones. Esta limitación se supera con particiones extendidas y lógicas.

Si necesita crear más de cuatro particiones, lo que puede hacer es crear una partición extendida. Dentro de esa partición puede crear tantas particiones lógicas como desee. Algunos sistemas operativos, como Windows, no pueden arrancar desde una partición lógica.



El tipo de partición se almacena en la tabla de particiones para que el sistema operativo sepa cómo administrar la partición. Los tipos de partición de las tablas de partición MBR están limitados a un byte. Puede ver todos los tipos de particiones MBR en este enlace:

https://en.wikipedia.org/wiki/Partition_type#List_of_partition_IDs

MBR es un sistema heredado y debe usarse solo para unidades pequeñas que no están destinadas a arrancar un computadora.

3.1.2 Tabla de particiones GUID (GPT)

GPT es parte del estándar de interfaz de firmware extenso de United (UEFI), el reemplazo propuesto para el BIOS. Fue diseñado para superar las limitaciones del estándar MBR. Las principales ventajas de GPT son:

- Puede crear un número ilimitado de particiones
- Puede manejar unidades de hasta 18 exabytes
- GPT es más seguro, ya que agrega sumas de verificación para evitar la corrupción de datos
- El proceso de arranque es más rápido si usa firmware GPT y UEFI
- Los tipos de partición se identifican como un GUID (como 21686148-6449-6E6F-744E 656564454649) en lugar de los tipos de partición 255 MBR. Puede consultar todos los tipos de partición aquí: https://en.wikipedia.org/wiki/GUID_Partition_Table#Partition_type_GUIDs

GPT utiliza el direccionamiento de bloques lógicos (LBA), una alternativa al direccionamiento histórico del sector de la culata, donde todo el espacio del disco se ve como un "flujo" de direcciones de almacenamiento consecutivas. El primer sector está reservado para la compatibilidad con MBR, por lo que la información de GPT comienza en LBA(1).

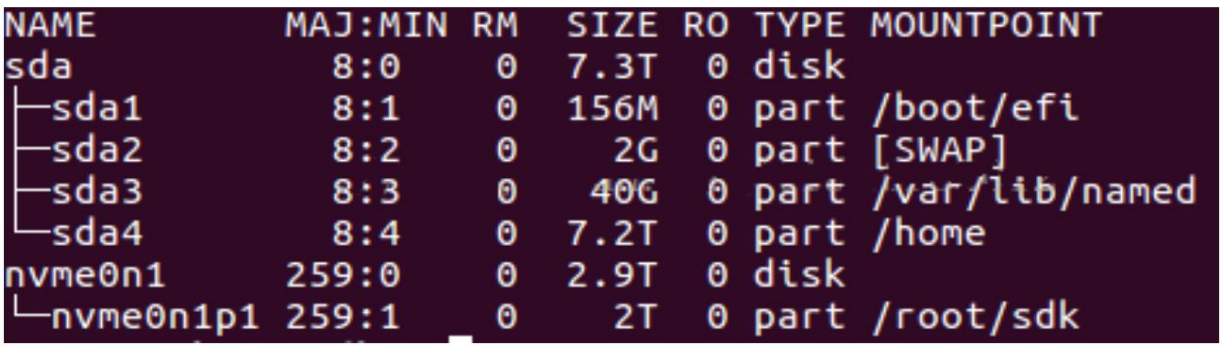
3.2 Administrar particiones con Linux

Para administrar particiones en Linux necesitamos permisos de administrador. Para la terminal, podemos abrir una consola raíz ejecutando sudo -i. Para las herramientas gráficas, el sistema nos pedirá nuestra contraseña o una contraseña de administrador.

Para gestionar particiones primero necesitamos saber qué dispositivos tenemos conectados a nuestro sistema. Que se puede lograr con el comando:

```
lsblk -e7 --todos
```

Los parámetros -e7 --all son para excluir algunos tipos de discos especiales de la salida, como los discos RAM (puede probar lsblk directamente en su sistema para ver todos los dispositivos de bloque). Un ejemplo de salida de este comando podría ser:



En este ejemplo, el sistema tiene dos discos: sda y nvme0n1.

En Linux, los discos SCSI, SATA y USB tienen el identificador /dev/sdX, donde X es una letra que comienza con a. En este caso, tenemos un disco SATA identificado como /dev/sda. Las particiones en esos discos se identifican como /dev/sdXN, donde N es el número de partición dentro del disco. En este caso, el disco tiene cuatro particiones identificadas como sda1, sda2, sda3 y sda4 . Si tuviéramos otro disco SATA, tendrá el identificador / dev/sdb, y las particiones serán sdb1, sdb2, etc. Los números de partición pueden ser no contiguos. Esto suele ocurrir en el caso de particiones lógicas y extendidas en discos MBR.

Tenemos otro disco conectado como disco NVMe. Esos discos se identifican como /dev/nvmeN, donde N es un número. El n1 que sigue al identificador del disco es el número de espacio de nombres (una función que permite que los discos NVMe se dividan en diferentes discos) y, por lo general, será n1. Los identificadores de partición para discos NVMe son nvmeNn1pM, donde M es el número de partición.

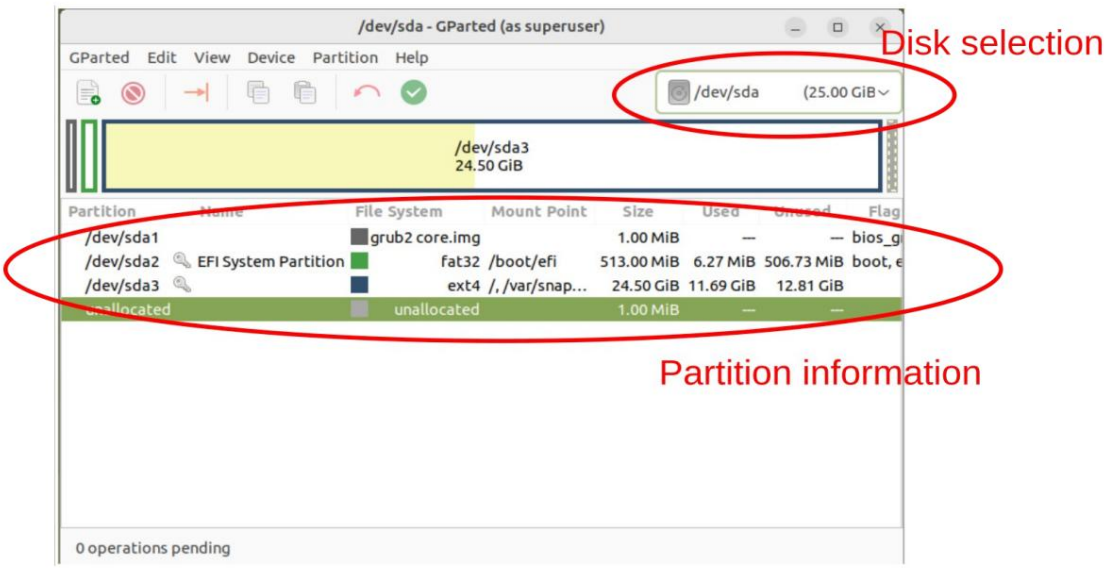
Las herramientas que veremos para gestionar particiones Linux son gparted y cfdisk.

3.2.1 Separado

gparted es una herramienta gráfica para administrar particiones de disco. En algunas distribuciones no viene instalado por defecto, necesitarás instalarlo con este comando:

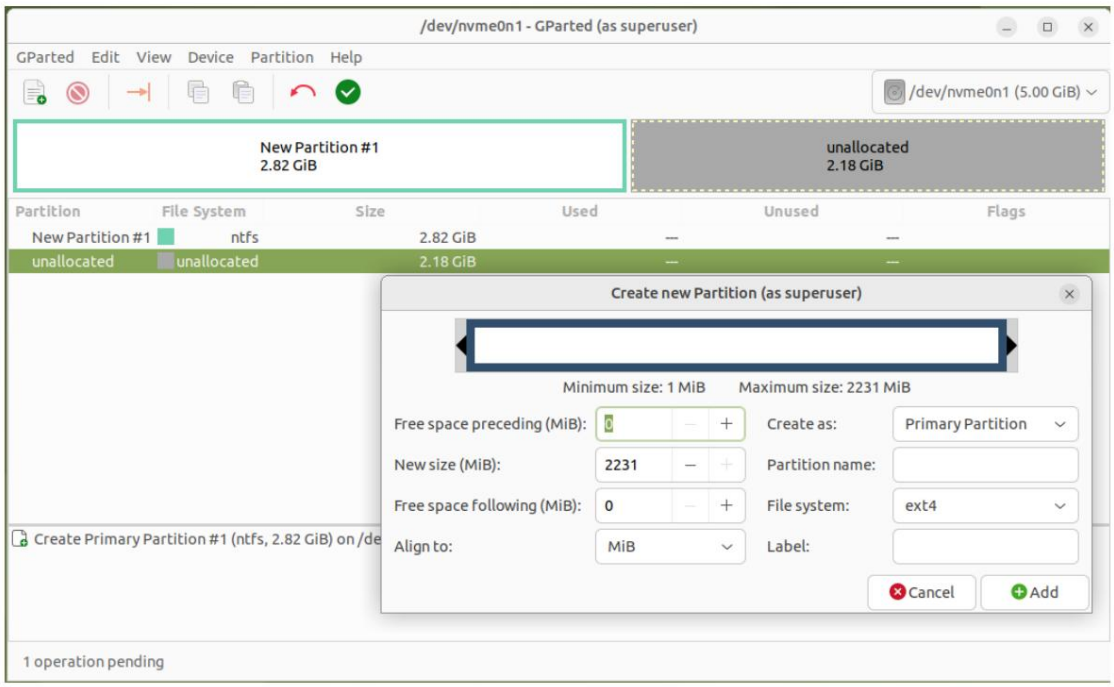
```
sudo apt install gparted
```

Cuando inicie gparted, seleccionará un disco y le mostrará información sobre las particiones en ese disco. Puede cambiar el disco usando el selector de disco en la parte superior derecha de la ventana:



Previamente, para administrar las particiones de unidades, deberá crear la tabla de particiones. Se puede hacer con la opción de menú *"Dispositivo/ Crear tabla de particiones" .* Borrará todos los datos del disco.

Luego puede seleccionar el espacio no asignado y crear una nueva partición usando la barra de herramientas o la opción de menú *"Partición/ Nuevo* ". Los cambios no se realizarán hasta que los confirme utilizando la barra de herramientas o *"Editar/ Aplicar todas las operaciones".*



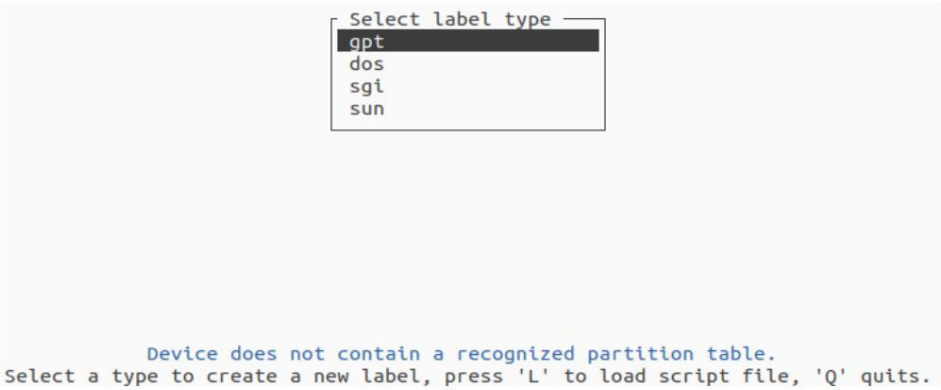
Creando una nueva partición. Ya hay una operación pendiente, la creación de la primera partición en el disco.

3.2.2 disco compacto

`cfdisk` es una herramienta de partición diseñada para ejecutarse desde la consola. Puede comenzar a particionar un disco con

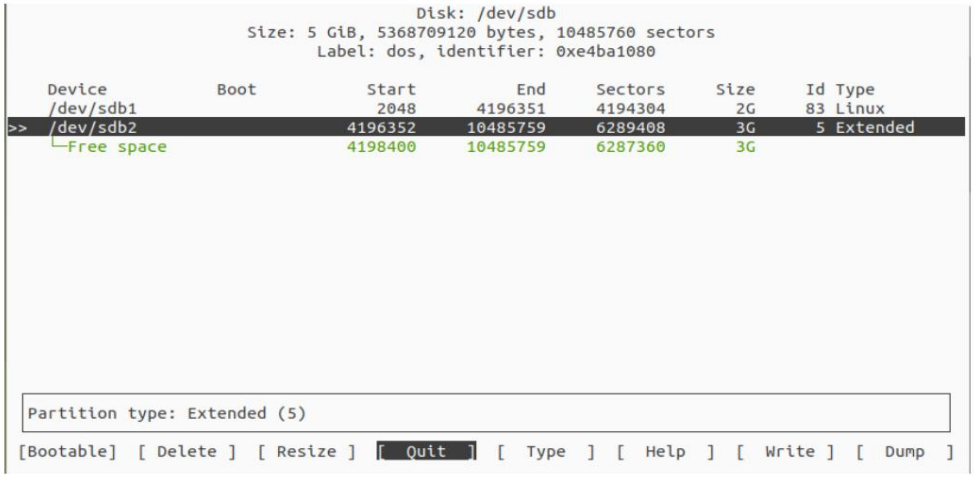
```
cfdisk <dispositivo>
```

Donde dispositivo es el dispositivo `/dev/XXXX` correspondiente al disco. Si el disco no tiene una tabla de particiones, mostrará una pantalla para seleccionar la tabla de particiones para ese disco:



Si necesita reconstruir la tabla de particiones usando un tipo diferente, puede usar otra herramienta como `fdisk <dispositivo>` o gparted

Una vez que se crea la tabla de particiones, puede agregar y eliminar particiones mediante una interfaz de texto:



Los cambios no se escribirán en el disco hasta que seleccione la opción *Escribir* .

> ÿ Es difícil y lento cambiar el tamaño de una partición una vez que se crea sin perder datos, así que elija sabiamente cuando particione un disco.

3.3 Formateo de particiones

Una vez que tenemos las particiones definidas, necesitamos formatearlas. Al formatear una partición, creamos un sistema de archivos en blanco listo para ser utilizado por el sistema operativo. En este paso es cuando decidimos que sistema de archivos tendrá esa partición.

> ÿ Todos los datos de la partición se borrarán cuando la formateemos

Puede formatear una partición en Linux con el comando mkfs, ejecutándolo como root (o con el comando sudo): mkfs -t <tipo de sistema de archivos> <dispositivo> <opciones> . Por ejemplo, para formatear la partición sda1 con un sistema de archivos ext4 debe ejecutar:

```
sudo mkfs -t ext4 /dev/sda1
```

Hay comandos individuales para formatear particiones con un sistema de archivos determinado, como mkfs.ext4 para formatear sistemas de archivos ext4, mkfs.fat para formatear sistemas de archivos FAT, etc.

Cada sistema de archivos se puede formatear con diferentes opciones, según los parámetros del sistema de archivos, lo que puede ayudar a ajustar el rendimiento del sistema de archivos. Con man mkfs.<type> puede obtener información sobre las opciones para cada sistema operativo. En este curso, estamos dejando los parámetros predeterminados al formatear un disco.

Puede ver todos los sistemas de archivos compatibles con una computadora Linux con:

```
cat /proc/sistemas de archivos | grep -v nodev
```

En Linux puede instalar soporte para otros sistemas de archivos. Por ejemplo, para instalar el soporte del sistema de archivos exFat, podría ejecutar:

```
sudo apt install exfat-fuse exfat-utils
```

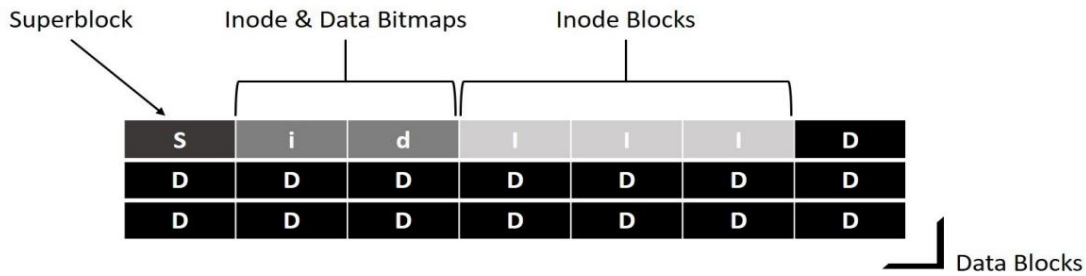
Herramientas como gparted también permiten formatear particiones mediante una interfaz gráfica.

4. SISTEMA DE ARCHIVOS LINUX

Todos los archivos en Linux están bajo el directorio raíz, /. Aquí es donde cuelgan todos los directorios y archivos del sistema. Linux usa en casi todas las distribuciones el sistema de archivos ext4. Ese será el que estudiaremos en este curso, aunque la mayoría de los comandos se pueden aplicar a otros sistemas de archivos gracias a la función de sistema de archivos virtual de Linux.

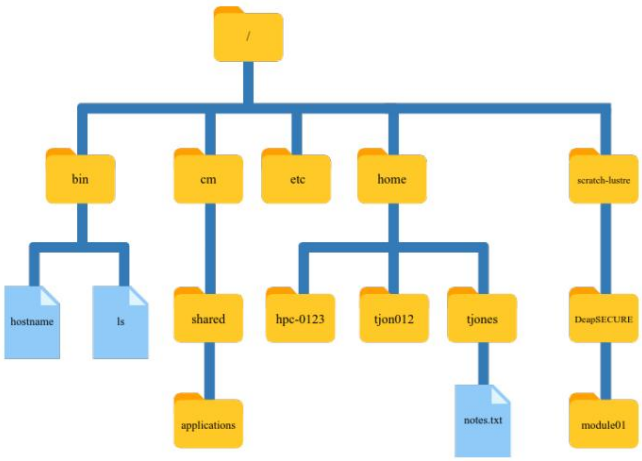
Algunos de los conceptos principales del sistema de archivos ext4 son:

- **Superbloque:** es esencialmente metadatos del sistema de archivos y define el tipo, tamaño, estado, etc. del sistema de archivos. El superbloque es muy crítico para el sistema de archivos y, por lo tanto, se almacena en múltiples copias redundantes para cada sistema de archivos.
- **Inodo:** un inodo es una estructura de datos que describe un objeto del sistema de archivos, como un archivo o un directorio. Su tarea principal es almacenar los bloques que utiliza un archivo. El inodo no contiene el nombre del archivo.
- **Tabla de inodos:** Los inodos se almacenan en una tabla al inicio del disco. Antes de eso, hay una estructura donde el sistema almacena qué bloques de datos e inodos ya se han utilizado.
- **dentry** (entrada de directorio): Una entrada de directorio vincula el nombre del archivo con su inodo correspondiente. Cuando cambiamos el nombre de un archivo, la entrada cambia, pero el inodo (los bloques donde se almacena el archivo) sigue siendo el mismo.



Birdview de un sistema de archivos ext4

El sistema de archivos de Linux es un árbol de carpetas, comenzando desde el directorio raíz / . Los directorios pueden contener otros archivos o directorios, formando una estructura recursiva.



Ejemplo de un árbol de directorios

Linux tiene varios subdirectorios importantes, que se enumeran a continuación. Esta estructura puede variar en algunas distribuciones, pero el esquema será muy similar.

- /bin contiene archivos de comandos ejecutables que pueden usar todos los usuarios. Aquí tenemos los programas que pueden ejecutar todos los usuarios del sistema. Puede ser un enlace a /usr/bin en algunas distribuciones.
- /sbin es para ejecutables para uso exclusivo del superusuario. Puede ser un enlace a /usr/sbin en algunas distribuciones.
- /home es un directorio donde se encuentran los directorios personales de los usuarios del sistema. Por ejemplo, para el usuario usuario1 sus archivos personales estarán en /home/usuario1
- /usr contiene utilidades y programas generales de usuario:
  - ÿ /usr/bin contiene programas de propósito general.
  - ÿ /usr/share contiene archivos independientes de la arquitectura que se pueden compartir.
  - ÿ /usr/etc contiene archivos de configuración utilizables globalmente.
  - ÿ /usr/share/doc contiene alguna documentación del sistema.
  - ÿ /usr/share/man contiene manuales.
  - ÿ /usr/include contiene encabezados C y C++.
  - ÿ /usr/lib contiene nuestras bibliotecas de programas.
  - ÿ /usr/sbin contiene los programas de administración del sistema.
  - ÿ /usr/src contiene el código fuente de nuestros programas.
- /dev contiene archivos de caracteres y bloques especiales asociados con dispositivos de hardware. Aquí encontramos todos los dispositivos físicos del sistema (todo nuestro hardware) como discos (/dev/sdX o /dev/nvmeXXX), cámaras web (/dev/videoN), etc.
- /lib contiene compiladores y bibliotecas del sistema. Contiene las bibliotecas necesarias para ejecutar los programas que tenemos en /bin y /sbin solamente.
- /proc contiene los archivos que reciben o envían información al kernel. Usted no debe modificar el contenido de este directorio.
- /etc contiene archivos de utilidades de administración y configuración.
- /var contiene archivos para el administrador. Este directorio contiene información variable, como como registros, datos del servidor, etc.
- /boot contiene los archivos de configuración de arranque del sistema, como GRUB.
- /media contiene todas las unidades físicas que hemos montado: discos duros, unidades de DVD, pen drives, etc.
- /opt se utiliza para admitir nuevos archivos creados después de la modificación del sistema. Es un punto de montaje desde el que se instalan paquetes de aplicaciones adicionales. Se puede usar para instalar aplicaciones que no vienen de los repositorios, por ejemplo, las que compilamos a mano.
- /tmp es donde se almacenan los archivos temporales. Se elimina en cada arranque del sistema.

4.1 Tipos de archivos

En Linux existen básicamente 5 tipos de archivos:

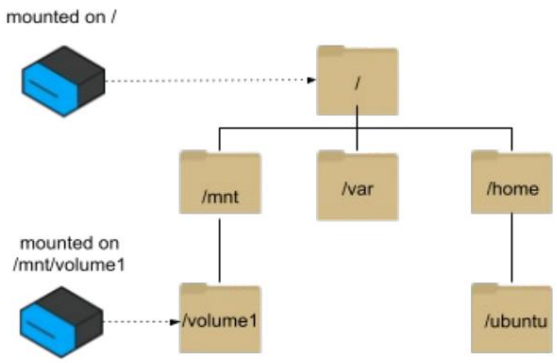
- **Archivos ordinarios.** Contienen la información con la que trabaja cada usuario.
- **Directorios.** Estos son archivos especiales que contienen referencias a otros archivos o directorios.
- **Enlaces físicos o duros.** Si dos entradas de directorio apuntan al mismo inodo, tenemos un enlace duro. Los dos archivos serán iguales, porque comparten los bloques de datos en el disco. El archivo no se eliminará hasta que eliminemos la última entrada de directorio de ese archivo.
- **Enlaces simbólicos.** También se utilizan para asignar un segundo nombre a un archivo. La diferencia con los enlaces duros es que los enlaces simbólicos solo hacen referencia al nombre del archivo original, por lo que si tenemos un enlace simbólico y borramos el archivo original, perderemos los datos.
- **Archivos especiales.** Suelen representar dispositivos físicos, como unidades de almacenamiento, impresoras, terminales, etc. En Linux, cada dispositivo físico conectado a la computadora está asociado con un archivo. Linux trata los archivos especiales como archivos ordinarios. También hay sockets y canalizaciones con nombre que se utilizan para la comunicación entre procesos.

4.2 Puntos de montaje

Una de las principales características del sistema de archivos de Linux es que permite al usuario crear, eliminar y acceder a archivos sin necesidad de saber exactamente dónde se encuentran. En Linux, no hay particiones físicas como c: o d:. Se accede a todos los archivos de la misma manera, independientemente de dónde se encuentren.

Si queremos usar un sistema de archivos en otra partición, necesitamos usar un **punto de montaje.** Un punto de montaje es un directorio (o archivo, pero no lo usaremos) en el que se hace accesible un nuevo sistema de archivos, directorio o archivo. Para montar un sistema de archivos o un directorio, el punto de montaje debe ser un directorio; y para montar un archivo, el punto de montaje debe ser un archivo.

Por lo general, un sistema de archivos, directorio o archivo se monta sobre un punto de montaje vacío, pero no es necesario. Si el archivo o directorio que sirve como punto de montaje contiene datos, no se puede acceder a esos datos mientras otro archivo o directorio los monta encima. En efecto, el archivo o directorio montado cubre lo que estaba previamente en ese directorio. Se puede volver a acceder al directorio o archivo original sobre el que se montó una vez que se deshace el montaje.



El directorio raíz está montado en la raíz del sistema de archivos y otra partición está montada en /mnt/volume1

Podemos montar un sistema de archivos usando el comando de montaje . Tomará el tipo de sistema de archivos, el dispositivo y el directorio. Por ejemplo, para montar la partición sdb1 con un sistema de archivos ext4 en /second_disk debemos ejecutar:

```
mount -t ext4 /dev/sdb1 /segundo_disco
```

Si no especificamos el tipo, mount intentará averiguar el tipo de sistema de archivos y actuará en consecuencia.

Puede pasar opciones para montar con el parámetro -o . Algunas opciones dependen del sistema de archivos. Por ejemplo, para abrir un sistema de archivos fat32 en modo de solo lectura:

```
montaje -o ro, grasa = 32
```

ÿ Para montar un dispositivo, no necesita especificar el dispositivo sino también la partición a la que desea acceder. En otras palabras, en lugar de /dev/sdb, monta /dev/sdb1 o /dev/sdb2, etc.

Ejecutar el comando de montaje sin parámetros, o con el parámetro -l , mostrará todos los sistemas de archivos montados. Puede filtrar el tipo de sistema de archivos con -t <tipo>. Por ejemplo, para mostrar todos los sistemas de archivos ext4 montados, puede ejecutar:

```
montar -t ext4
```

Para desmontar un sistema de archivos usamos el comando umount , especificando el directorio o el dispositivo. Para desmontar /dev/sbd1 previamente montado en /segundo_disco, podríamos ejecutar umount /dev/sdb1 o umount / segundo_disco.

4.2.1 tabulador

Puede configurar permanentemente dónde se debe montar una unidad conectada a una máquina editando el archivo /etc/ fstab. Ese archivo es un archivo de configuración que define cómo y dónde se montarán los principales sistemas de archivos, especialmente en el momento del arranque.

La sintaxis de una entrada fstab es:

```
[Dispositivo] [Punto de montaje] [Tipo de sistema de archivos] [Opciones] [Volcado] [Aprobado]
```

- **dispositivo:** El dispositivo/partición (por ubicación /dev o UUID) que contiene un sistema de archivos.
- **punto de montaje:** el directorio donde se montará la partición
- tipo de **sistema de archivos:** tipo de sistema de archivos
- **opciones: opciones** de montaje (las mismas que usaremos para el comando de montaje)
- **dump:** active o desactive la copia de seguridad del dispositivo/partición (el comando dump). Este campo generalmente se establece en 0, lo que lo deshabilita.
- **pass num:** controla el orden en el que fsck verifica el dispositivo/partición en busca de errores en el momento del arranque. El dispositivo raíz debe ser 1. Otras particiones deben ser 2 o 0 para deshabilitar la verificación. Por lo general, se establece en 0 para dispositivos no root.

Este es un ejemplo de un archivo fstab. Las líneas que comienzan con # son comentarios que no se procesan:

```
# <sistema de archivos> <punto de montaje> <tipo> <opciones> / /boot/efi <volcar> <pasar>
dev/sda1 ext4 errores=remontar-ro 0 1
UUID=B782-7AFB vfat umask=0077 0 1

/dev/sdb1 /segundo_disco ext4 /datos predeterminados, errores = volver a montar- 0 0
UUID=8dde8fbf-b7fb-... ext4 /red nfs4 ro predeterminados, errores = volver a 0 0
192.168.13.140:/dir1 montar-ro noauto, duro, bg, usuario 0 0
```

Este fstab especifica los siguientes montajes:

- El sistema de archivos raíz (/dev/sda1) montado en /
- Un sistema de archivos vfat con UUID B782... montado en /boot/efi
- La partición /dev/sdb1 montada en /second_disk
- La partición con UUID 8dd... montada en /data
- El sistema de archivos nfs de la red 192.168.13.140:/dir1 montado en /network

Es mejor usar UUID de partición en lugar de nombres de dispositivos porque los nombres de los dispositivos pueden cambiar si modificamos el hardware de la computadora (cambiando el puerto SATA donde está conectado un disco, por ejemplo, o agregando nuevo hardware) Para obtener el UUID de la partición podemos usar el comando niño negro

Si desea montar algo que está definido en fstab, solo necesita especificar el directorio. Por ejemplo, puede usar el montaje / segundo_disco para montar /dev/sdb1 en /segundo_disco con las opciones definidas en fstab.

5.GESTIÓN DE ARCHIVOS CON LINUX

Veremos aquí algunos de los comandos utilizados para administrar archivos y directorios en Linux.

> ÿ Linux distingue entre mayúsculas y minúsculas al nombrar archivos y directorios. Podemos tener un archivo llamado John y otro archivo llamado john en el mismo directorio

5.1 Navegación del sistema de archivos

5.1.1 Directorio actual

Desde el shell de Linux, podemos movernos por todo el sistema de archivos para encontrar un directorio o archivo específico.

El primer paso es saber dónde estamos en un momento dado. Para saber qué directorio es el directorio actual tenemos el comando pwd . Su nombre viene "imprimir directorio de trabajo" y su única tarea es mostrar en pantalla una línea que nos indica la ruta absoluta al directorio actual. Por ejemplo

```
$ pwd /
inicio/usuario
```

Esto nos dirá que estamos en el directorio /home/usuario

5.1.2 Desplazarse por directorios

Una vez que sabemos en qué directorio estamos, podemos comenzar a desplazarnos por el árbol de directorios con el comando cd . El nombre cd proviene de "cambiar directorio", y hace exactamente lo que parece: cambia el directorio actual al que especificamos en su argumento.

En Linux. podemos referirnos a archivos y directorios usando rutas absolutas o relativas

5.1.3 Caminos absolutos

Las rutas absolutas se forman comenzando en el directorio raíz y terminando en el punto al que queremos hacer referencia. Por ejemplo, si el nombre de nuestro usuario es john y queremos hacer referencia a su directorio de inicio, la ruta absoluta sería /home/john.

> ÿ Las rutas absolutas siempre comienzan con /

5.1.4 Caminos relativos.

Se forman utilizando los directorios especiales . y .. Por ejemplo, si estamos en /tmp/example/subdirectory entonces:

- . se refiere al directorio actual, /tmp/example/subdirectory
- .. se refiere al directorio principal, /tmp/example

Por lo tanto, podemos usar el comando cd con rutas absolutas y relativas. Por ejemplo, estamos en el directorio /usr/share/man y queremos ir a /usr/bin. Si usamos la ruta absoluta haremos:

```
cd/usr/bin
```

O usando una ruta relativa:

```
cd ../../papelera
```

> ÿ Puede tener más de una ruta relativa al mismo archivo. Por ejemplo: ../bin/file y ../bin/../bin/file se referirían al mismo archivo

El uso de cd sin parámetros cambiará el directorio actual al directorio de inicio del usuario.

5.2 Listado de archivos

Una vez que estemos en un directorio en particular, lo más probable es que queramos conocer el contenido del directorio. Para ello contamos con el comando ls . Su nombre proviene de "list" y eso es precisamente lo que hace, listar el contenido del directorio.

Si le pasamos un argumento, nos mostrará el contenido del directorio que le pasamos y si no, nos mostrará el contenido del directorio actual. El comando ls tiene una multitud de opciones que puede ver con man ls

Para mostrar la estructura del directorio podemos usar el comando de árbol . Mostrará los archivos y directorios bajo el directorio de trabajo.

5.3 Creación

Una vez que hemos visto cómo cambiar el directorio de trabajo y cómo listar el contenido de una carpeta concreta, pasamos a ver comandos que nos permitirán crear directorios y archivos

> ÿ Puede utilizar espacios como parte del nombre de un archivo o directorio. Para hacer eso, simplemente ponga el nombre entre comillas: mkdir "directorio con espacios"

5.3.1 Creación de directorios

El comando mkdir se usa para crear una carpeta o directorio cuyo nombre será el que le pasemos como argumento a este comando. Si, por ejemplo, ejecutamos el comando

```
$ mkdir directorio nuevo
```

el directorio newdirectory se creará en el directorio de trabajo actual. También podemos crear directorios en lugares distintos al que nos encontramos, utilizando rutas absolutas o relativas. Entonces, si queremos crear el directorio Música en /etc y estamos en /home/student, podemos usar rutas absolutas:

```
$ mkdir /etc/Música
```

o rutas relativas:

```
$ mkdir ../../etc/Música.
```

Con el comando mkdir , también podemos crear varios directorios en el mismo comando. Para ello, solo tenemos que introducir tantos argumentos como carpetas queramos crear. Así, el comando

```
$ mkdir Documentos Imágenes Música
```

creará esos 3 directorios en el directorio de trabajo actual.

5.3.2 Creación de archivos

El comando touch se usa para cambiar la fecha de acceso y modificación de un archivo que le pasamos como argumento. Si dicho archivo no existe, creará un archivo vacío con el nombre que le hemos pasado como argumento.

Por ejemplo, si ejecutamos el comando

```
$ toque ejemplo.txt
```

creará un archivo llamado ejemplo.txt si no existía antes, o simplemente modificará su fecha de acceso y modificación a la fecha actual del sistema, si existiera antes. Al igual que con el comando mkdir , con el comando touch podemos crear múltiples archivos con un solo comando:

```
$ tocar fich1 fich2 fich2 fich3 fich3 fich4
```

creará los archivos fich1, fich2, fich3 y fich4

También podemos crear archivos por redirección, como vimos en la unidad anterior, o mediante un editor de texto como vi o nano.

> ÿ Si ejecuta vi, puede salir del editor escribiendo : y luego q!<enter>.

5.4 Eliminación

El comando rm se utiliza para borrar un archivo cuyo nombre será el que le pasaremos como argumento. Por ejemplo, para borrar el archivo llamado delete_me.txt, ejecutaremos el comando:

```
rm eliminar_me.txt
```

Si el archivo que queremos eliminar no existe, nos mostrará un error. Como hemos visto con otros comandos, podemos borrar varios archivos. Entonces, si ejecutamos rm tom dick harry, eliminará esos tres archivos del directorio de trabajo actual.

El comando rmdir se usa para eliminar un directorio. El directorio debe estar vacío. Si no, mostrará un mensaje diciendo que tiene contenidos. Entonces, si queremos eliminar una carpeta vacía que está en nuestro directorio de trabajo actual, ejecutaríamos rmdir the_directory.

El comando rm también se puede usar para eliminar directorios completos. Si el directorio contiene archivos y desea eliminar tanto el directorio como su contenido, deberá usar el comando rm con la opción -R:

```
rm -r directorio_no_vacío
```

Tenga cuidado al usar esta opción, ya que puede eliminar información importante.

ÿ sudo rm -fr / no eliminará el paquete de idioma francés

5.5 Copiar y mover

Para copiar archivos, usamos el comando cp. Copia el archivo que se le pasa como parámetro (origen) a otro cuyo nombre se pasa como segundo parámetro (destino). Por ejemplo, si ejecutamos el comando:

```
cp archivo1.txt copiararchivo1.txt
```

copiará el contenido del archivo file1.txt en copyfile1.txt, ambos en el directorio de trabajo actual.

Por el contrario, si en lugar de un archivo indicamos como destino un directorio, lo que ocurrirá es que creará una copia del archivo fuente en el destino especificado. Por ejemplo:

```
cp archivo2.txt /home/estudiante
```

creará un archivo con el mismo nombre que el original en el directorio /home/student.

En el comando cp , al igual que en el comando rm , tenemos la opción R para copiar directorios completos con todo su contenido. Así, si ejecutamos el comando:

```
cp -R /inicio/estudiante/directorio1 /tmp,
```

copiará todo el contenido de la carpeta directory1 al destino indicado y luego copiará todo el contenido de la carpeta directory1 al destino indicado, que en este caso es /tmp.

Si queremos mover archivos de un directorio a otro, podemos usar el comando mv. Por ejemplo, queremos mover el archivo moveme.txt, que se encuentra en el directorio actual, al directorio /tmp, ejecutaríamos:

```
mv moverme.txt /tmp
```

El comando mv tiene otra utilidad, que es cambiar el nombre de un archivo. Para ello, solo tenemos que cambiar el nombre del archivo que queremos renombrar y el nuevo nombre como argumentos, de la siguiente manera:

```
mv nombre antiguo.txt nombre nuevo.txt
```

mv detecta si el segundo argumento es un directorio. En ese caso, el archivo se moverá en lugar de renombrado

5.6 Visualización

Podemos mostrar archivos de texto desde la consola. También es posible mostrar información de otro tipo de archivos como imágenes, ejecutables, etc. mediante diferentes comandos, pero no nos vamos a ocupar de ellos en este apartado.

Los órdenes más comunes para mostrar archivos de texto son:

- cat: muestra el contenido de un archivo de texto.
- head: muestra las primeras líneas de un archivo de texto. Por ejemplo, head -n 3 file mostrará las tres primeras líneas del archivo.
- cola: muestra las últimas líneas de un archivo de texto. Por ejemplo, tail -n 4 file mostrará las últimas cuatro líneas del archivo.
- menos: muestra el contenido de un archivo, pero permite al usuario navegar por el contenido usando página arriba/página abajo, buscar contenido, etc.

También tenemos el comando info para mostrarnos información sobre los archivos. Por ejemplo, la codificación de caracteres, el tipo de datos que contiene e información específica en algunos casos, como imágenes o videos.

5.7 Vinculación

Para comprender cómo funciona la vinculación, primero debemos comprender cómo podemos encontrar los datos de un archivo en un sistema de archivos ext4.

> ÿ Veremos aquí que la vinculación es una característica de los sistemas de archivos ext4. Otros sistemas de archivos, como ntfs, pueden tener diferentes formas de vincular archivos.

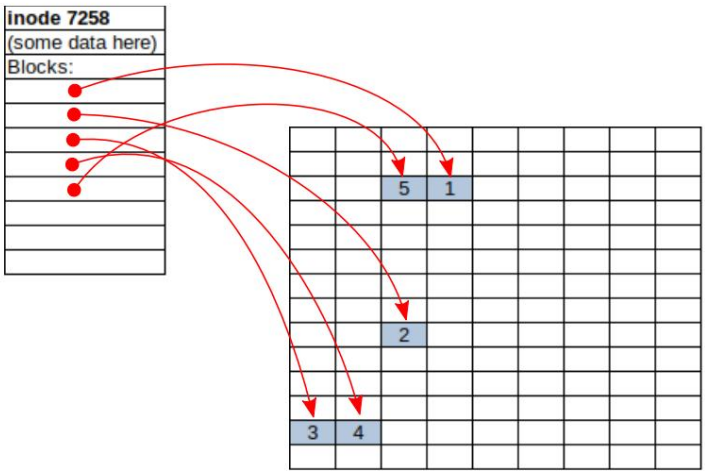
Para acceder a un archivo, normalmente lo hacemos a través de un directorio. Los directorios se componen de entradas de directorio o dentrys. Cada entrada de directorio almacena el nombre del archivo y el inodo que contiene los datos del archivo.
Por ejemplo, las entradas de un directorio podrían ser:

Directorio	
nombre	***inodo***
archivo1.txt	897
ejemplo de	1225
imagen.bmp	7258
...	

En la imagen de arriba podemos ver tres dentrys: un archivo llamado file1.txt cuyos datos están en el inodo 897, un archivo llamado image.bmp con el inodo 1225 y ejemplo con el inodo 7258.

El inodo contiene información sobre el archivo y los bloques donde se almacena el archivo en el disco.

Por ejemplo, el inodo 7258 podría ser algo como esto:



El archivo utilizará cinco bloques. No es necesario que los bloques sean contiguos u ordenados, porque el orden correcto se almacena en el inodo. Tenga en cuenta que el nombre del archivo no se almacena en el inodo, sino en el dentry. Podemos ver los inodos de los archivos en un directorio usando el parámetro -i con el comando ls :

```
user@ubuntu-mate:/tmp/links$ ls -lai
total 8
134437 drwxrwxr-x  2 user user 4096 Nov 13 09:50 .
131075 drwxrwxrwt 14 root root 4096 Nov 13 09:49 ..
134438 -rw-rw-r--  1 user user    0 Nov 13 09:50 file1.txt
134442 -rw-rw-r--  1 user user    0 Nov 13 09:50 file2.txt
```

El archivo file1.txt tiene el inodo 134438 y file2.txt tiene el inodo 13442. Los directorios también tienen inodos para almacenar las entradas del directorio.

También podemos ver el inodo de un archivo usando el comando stat:

```
user@ubuntu-mate:/tmp/links$ stat file1.txt
  File: file1.txt
  Size: 0               Blocks: 0          IO Block: 4096   regular empty file
Device: 803h/2051d      Inode: 134438      Links: 1
Access: (0664/-rw-rw-r--)  Uid: ( 1000/    user)   Gid: ( 1000/    user)
Access: 2022-11-13 09:50:25.600195191 +0100
Modify: 2022-11-13 09:50:25.600195191 +0100
Change: 2022-11-13 09:50:25.600195191 +0100
 Birth: 2022-11-13 09:50:25.600195191 +0100
```

5.7.1 Enlaces duros

Un enlace duro es una entrada que apunta al mismo inodo del archivo enlazado.

Podemos crear enlaces duros con `ln <archivo original> <archivo enlazado>` Por ejemplo, si ejecutamos:

```
ln archivo1.txt vinculado.txt
```

La nueva información del directorio será:

```
user@ubuntu-mate:/tmp/links$ ls -ial
total 8
134437 drwxrwxr-x  2 user user 4096 Nov 13 10:06 .
131075 drwxrwxrwt 14 root root 4096 Nov 13 09:49 ..
134438 -rw-rw-r--  2 user user    0 Nov 13 09:50 file1.txt
134442 -rw-rw-r--  1 user user    0 Nov 13 09:50 file2.txt
134438 -rw-rw-r--  2 user user    0 Nov 13 09:50 linked.txt
```

Como puede ver, el inodo de file1.txt y linked.txt es el mismo (134438). Ambos archivos apuntan a los mismos bloques en el disco, por lo que estaremos modificando la misma información si accedemos a través de archivo1.txt o a través de vinculado.txt.

También podemos ver que el recuento de enlaces (la columna anterior al primer usuario) ha aumentado. Cada inodo almacena el número de enlaces duros de ese archivo. Los datos en el disco no se eliminarán hasta que se elimine el último archivo que apunta a ese inodo.

Los enlaces duros no necesitan estar en el mismo directorio, podemos enlazar cualquier archivo que queramos.

No podemos vincular directorios, solo archivos.

5.7.2 Enlaces blandos

Los enlaces suaves funcionan de manera completamente diferente a los enlaces duros. Al crear un enlace suave, se crea un nuevo "archivo". Este archivo almacena la ruta al archivo vinculado. Cuando intente acceder al enlace, el sistema operativo detectará que está accediendo a un archivo enlazado y redirigirá la solicitud al destino del enlace.

Para crear un enlace suave usamos el comando ln con el parámetro -s

```
ln -s archivo1.txt soft_linked.txt
```

Esto creará un enlace suave a file1.txt en el mismo directorio:

```
user@ubuntu-mate:/tmp/links$ ls -lai
total 8
134437 drwxrwxr-x  2 user user 4096 Nov 13 10:14 .
131075 drwxrwxrwt 14 root root 4096 Nov 13 09:49 ..
134438 -rw-rw-r--  2 user user    0 Nov 13 09:50 file1.txt
134442 -rw-rw-r--  1 user user    0 Nov 13 09:50 file2.txt
134438 -rw-rw-r--  2 user user    0 Nov 13 09:50 linked.txt
134443 lrwxrwxrwx  1 user user    9 Nov 13 10:14 soft_linked.txt -> file1.txt
```

Como puede ver, el enlace suave tiene un nuevo inodo. Si eliminamos el destino del enlace, el enlace se “romperá” y ya no podremos acceder a los datos:

```
user@ubuntu-mate:/tmp/links$ ls -ali
total 8
134437 drwxrwxr-x  2 user user 4096 Nov 13 10:22 .
131075 drwxrwxrwt 14 root root 4096 Nov 13 09:49 ..
134442 -rw-rw-r--  1 user user    0 Nov 13 09:50 file2.txt
134438 -rw-rw-r--  1 user user    0 Nov 13 09:50 linked.txt
134443 lrwxrwxrwx  1 user user    9 Nov 13 10:14 soft_linked.txt -> file1.txt
```

Tenga en cuenta que los mismos datos todavía están disponibles a través de linked.txt (era un enlace fijo de file1.txt), pero el enlace suave apunta a file1.txt y ese archivo ya no existe, por lo que el enlace está roto.

Podemos enlazar directorios por software, a diferencia de los enlaces duros.

6. SISTEMA DE ARCHIVO VIRTUAL Y SISTEMAS DE ARCHIVO SIN DISCO

Para acceder a los archivos, Linux implementa el Sistema de archivos virtual (también conocido como VFS). Es un componente del núcleo que maneja todas las llamadas al sistema relacionadas con archivos y sistemas de archivos. VFS es una interfaz genérica entre el usuario y un sistema de archivos en particular. Esto significa que podemos acceder a diferentes sistemas de archivos de manera uniforme utilizando la API proporcionada por el VFS. La comunicación con el hardware y las E/S son manejadas por VFS.

Eso significa que podemos crear sistemas de archivos que no sean compatibles con los dispositivos de disco, siempre que proporcionemos las llamadas a la API que requiere el VFS. Por ejemplo, podemos crear sistemas de archivos que se almacenan en la memoria o se accede a ellos a través de la red.

Desde un punto de vista funcional, los sistemas de archivos se pueden agrupar en:

- sistemas de archivos de disco (ext3, ext4, xfs, fat, ntfs, etc.)
- sistemas de archivos de red (nfs, smbfs/cifs, ncp, etc.)
- sistemas de archivos virtuales (procfs, sysfs, sockfs, pipefs, etc.)

Los sistemas de archivos de red son "discos" que otra computadora pone a disposición a través de la red.
Existen diferentes sistemas de archivos de red como nfs o samba. De todos modos, gracias al VFS podemos montarlos y utilizarlos como si los archivos estuvieran en nuestro propio ordenador. Veremos esto más adelante en el curso.

Los sistemas de archivos virtuales son sistemas de archivos en los que los datos no se almacenan en un disco. Por ejemplo, podemos crear un disco RAM (un disco donde los datos se almacenan en la memoria RAM de la computadora) con este comando:

```
sudo mount -t tmpfs -o size=512M myramdisk /tmp/ramdisk
```

donde el nombre del dispositivo será myramdisk y se montará en /tmp/ramdisk. Tomará memoria de la memoria de la computadora y los datos se perderán cuando la computadora se apague, pero será extremadamente rápido.

7.MATERIAL COMPLEMENTARIO

Sistemas de archivos:

- https://www.howtogeek.com/196051/htg-explains-what-is-a-file-system-and-why-are-there-tantos-de-ellos/

- https://kb.wisc.edu/helpdesk/page.php?id=11300

MBR y GPT

- https://www.softwaretestinghelp.com/mbr-vs-gpt/

Particiones de disco

- https://docs.fedoraproject.org/en-US/fedora/latest/install-guide/appendixes/Disk_Partitions/

Inodos de Linux:

- https://www.youtube.com/watch?v=qXVbNIMG28I
- https://www.youtube.com/watch?v=6KjMlm8hhFA

Enlaces duros vs enlaces blandos:

- https://www.youtube.com/watch?v=4-vye3QFTFo