

Sumario

Introducción.....	3
Tipos de sistemas de archivos más utilizados.....	4
FAT (File Allocation Table).....	4
NTFS.....	4
EXT4.....	4
Estructura de directorios en Linux y Microsoft Windows.....	6
Estructura de directorios en GNU/Linux.....	6
Estructura de directorios de Microsoft Windows.....	9
Gestión de archivos GNU/Linux.....	9
Directorio actual.....	9
Cambiar de directorio.....	10
Rutas.....	10
Mostrar el contenido de un directorio.....	10
Caracteres comodín * y ?.....	10
Borrar directorios.....	10
Copiando ficheros y directorios.....	10
Moviendo objetos.....	11
Eliminando objetos.....	11
Creando ficheros.....	12
Mostrar contenido de un fichero.....	12
cat.....	12
more.....	13
less.....	13
head y tail.....	13
Redirecciones.....	14
Redirección de salida (stdout).....	14
Redirección de la salida de errores (stderr).....	14
Redirección de entrada (stdin).....	15
Expresiones regulares.....	16
Expresiones alternativas.....	16
Contenedores.....	17
Cuantificadores.....	19
Puntos de anclaje.....	19
Escapes.....	20
Ejercicios.....	20
Tuberías y filtros.....	21
wc (Word Count).....	22
tr (Traductor).....	23
cut (Cortar).....	23
grep.....	24
Sort (ordenar).....	24
Gestión de archivos por interfaz gráfica en Microsoft Windows.....	26
Gestión de archivos por la consola en Microsoft Windows.....	26
Raid.....	26
Particiones de disco.....	28

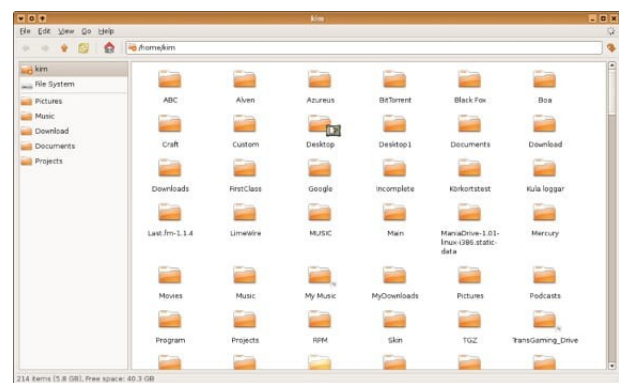
MBR.....	28
GPT.....	30
Estructura:.....	30
Ventajas:.....	30
Tipos de particiones.....	30
Gestión de almacenamiento por línea de comandos en Linux.....	32
Como se nombran los dispositivos de almacenamiento.....	32
Particionar un disco.....	32
Particionar disco mediante GUI.....	32
Ejemplo con Gparted.....	33
Particionar disco mediante terminal.....	33
Formatear particiones (Dar sistema de archivos).....	33
Desfragmentar.....	33
Chequeo.....	34
fsck.....	35
e2fsck.....	36
Gestión de almacenamiento por línea de comandos en Windows.....	36
Particionamiento de disco.....	36
Formatear particiones (Dar sistema de archivos).....	36
Búsqueda de información por líneas de comando.....	36
Búsqueda por nombre.....	37
Búsqueda por tipo.....	37
Búsqueda por fecha.....	37
Búsqueda por tamaño.....	38

Introducción.

El sistema de archivos o sistema de ficheros es el **componente del sistema operativo** encargado de **administrar y facilitar el uso de las memorias secundarias**.

Sus **principales funciones** son la **asignación de espacio a los archivos**, la **administración del espacio libre** y del **acceso a los datos guardados**.

Estructuran la información guardada en un dispositivo de almacenamiento de datos o unidad de almacenamiento, que luego será representada un **gestor de archivos**.



Por lo general **el acceso a los datos** se hace por **bloques** de un mismo tamaño, a veces llamados **sectores o clústers**.

El software del sistema de archivos es el encargado de la **organización de estos sectores** en archivos y directorios y mantiene un **registro de qué sectores pertenecen a qué archivos** y cuáles son los **sectores libres**.

La mayoría de los sistemas operativos manejan su propio sistema de archivos.

Los sistemas de archivos pueden realizar las siguientes **operaciones crear, mover, renombrar y eliminar** tanto **archivos como directorios**, aunque no son las únicas operaciones también pueden manejar accesos directos, enlaces ...

La seguridad (quien puede leer, escribir o ejecutar un fichero) a sistemas de archivos básicos puede estar basado en los esquemas de **lista de control de acceso** (access control list, **ACL**)

Las ACL hace décadas que demostraron ser inseguras, los sistemas operativos comerciales todavía funcionan con listas de control de acceso.

Tipos de sistemas de archivos más utilizados.

FAT (File Allocation Table).

Tabla de asignación de archivos, comúnmente conocido como FAT, es un sistema de archivos desarrollado para **MS-DOS**, así como el sistema de archivos principal de las versiones no empresariales de Microsoft Windows hasta windows XP.

FAT es relativamente sencillo. Es un formato de sistemas de archivos admitido prácticamente por todos los sistemas operativos existentes .

Se utiliza para **intercambio de datos** entre sistemas operativos de distintos fabricantes y sobre todo para **tarjetas de memoria y memory stick**.

Hay varias versiones de este sistema la más utilizada es FAT32 y sus principales limitaciones son:

- Imposibilidad de gestionar particiones superiores a 8 TB.
- No puede almacenar archivos de más de 4GB.
- Bajo rendimiento.
- Inseguro. Sin encriptación, sus atributos y permisos son muy limitados.

NTFS.

Está basado en el sistema de archivos HPFS de IBM/Microsoft, y también tiene ciertas influencias del formato de archivos HFS diseñado por Apple.

NTFS tiene un tamaño de clúster variable desde 512 bytes en adelante.

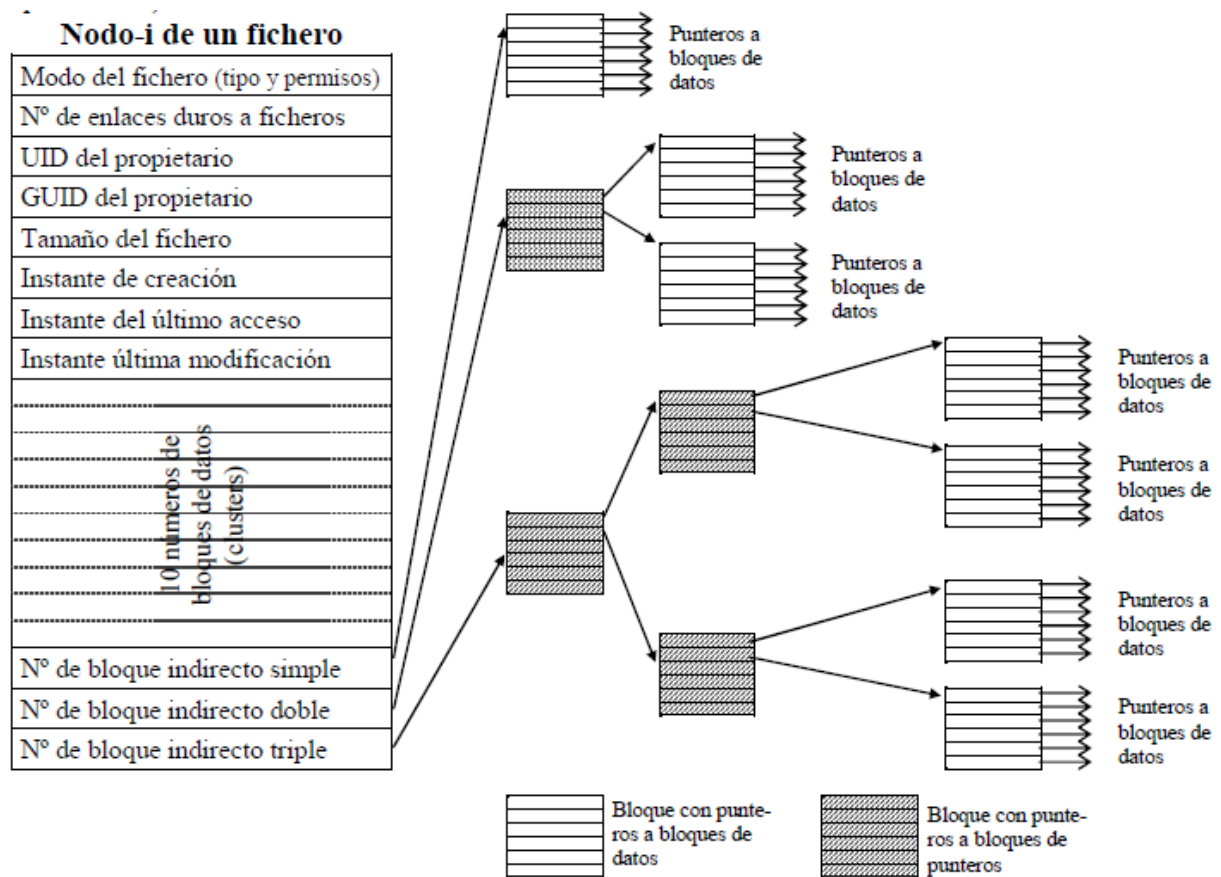
- Permite cifrado y compresión.
- Reduce la fragmentación y aumenta la velocidad de búsqueda de archivos con respecto a FAT32.
- Puede llegar a gestionar hasta 16EB y archivos de 16TB.
- Nombres de archivos de hasta 255 caracteres, usa UNICODE.

EXT4

Es el Sistema de archivos más utilizado en Linux.

- Archivos de hasta 16TB.
- Volúmenes de hasta 1EB.

- Mejora el rendimiento.
- Reduce la fragmentación.
- La estructura fundamental para el funcionamiento de este sistema es el i-nodo.



Esctrucutura de directorios en Linux y Microsoft Windows.

Una vez instalado un sistema operativo este genera un serie de carpeta donde despliega el sistema operativo y prepara el entorno para que el usuario.

Se dice que el sistema de archivo tiene forma de árbol invertido.

Pero en cualquier caso distinguimos:

- **Directorio Raíz.** / en Linux y \ en Windows.
- **Directorio Actual.** Directorio en el que nos encontramos trabajando, en Linux puedes saber cual es con el comando `pwd`
- **Directorio Padre.** Es el que se encuentra por encima del actual en la estructura de directorio en forma de árbol. Para llegar a el desde el directorio actual se usa `cd ..`
- **Directorio Hijo.** Es el que se encuentra por debajo del actual en la esctructura de directrorio en forma de árbol.

La ruta de un fichero o directorio se puede indicar de dos forma:

- **Ruta absoluta.** Es la ruta de directorios que hay que seguir desde el directorio raíz para llegar al fichero o directorio.
Siempre empieza por / por ejemplo `/home/ricardo/datos.txt`.
En windows sería `\Archivos de programas\Visual\readme.txt`
- **Ruta relativa.** Es la ruta de directorios que hay que seguir desde el directorio actual hasta llevar al fichero o directorio.

https://www.youtube.com/watch?v=0NUorN_uadI

Estructura de directorios en GNU/Linux

- La mayoría de directorios de GNU/Linux siguen el estándar FHS, que establece el contenido y las funciones de los directorios. Los mas importantes son:
- **/:**
Es el directorio principal, la raíz o root, de él cuelgan todos los subdirectorios (incluso si están en particiones o discos diferentes). Sin duda es la más importante.
- **/bin:**
Es el directorio donde están los programas que emplea el sistema para labores administrativas como los comandos `cp`, `echo`, `grep`, `mv`, `rm`, `ls`, `kill`, `ps`, `su`, `tar`, etc.

- **/sbin:**

La S es de System, y como su nombre indica, se almacenan los programas que emplea el propio sistema operativo para tareas de arranque, restauración, etc.

Por ejemplo, fsck, mount, mkfs, reboot, swapon,...

- **/boot:**

Es el directorio de arranque, donde está el kernel de Linux que se cargarán durante el arranque, y también directorios y configuración del propio gestor de arranque.

- **/dev:**

Es un directorio muy especial donde se encuentran los dispositivos de bloques o caracteres, es decir, ficheros que representan la memoria, particiones, discos, dispositivos de hardware, etc.

En UNIX «todo» es un archivo, y no unidades como en Windows... Por ejemplo, el disco duro o particiones serán /dev/sda1, /dev/sda2, ... /dev/sdb1, etc.

- **/media o /mnt:**

Los directorios donde se establecen generalmente los puntos de montaje.

Cuando insertamos algún medio extraíble o recurso de red compartido, etc., que hayamos montado, estaría aquí si lo hemos puesto como punto de montaje.

El primero es más específico para medios que se montan de una forma temporal.

- **/etc:**

Aquí residen los ficheros de configuración de los componentes del sistema y otros programas instalados.

- **/home:**

Aquí se almacenan dentro de directorios separados (uno para cada usuario con su nombre), los ficheros personales.

Por ejemplo, /home/ricardo

- **/lib o /lib64:**

Se alojan las bibliotecas necesarias para los binarios presentes en el sistema. En /lib64 estarán las de las aplicaciones de 64-bit.

- **/opt:**

Se almacenarán los paquetes o programas instalados en el sistema que son de terceros.

Por ejemplo, si instalamos algún antivirus, Chrome, Arduino IDE, ... o ciertos programas, suelen instalarse aquí.

- **/proc:**

Directorio especial, más que un directorio es una interfaz por decirlo de un modo sencillo.

Y aquí el sistema nos presenta los procesos como directorios numerados con el PID.

Dentro de cada uno de ellos estará toda la información necesaria para la ejecución de cada proceso en marcha.

Además, encontrarás ficheros de los que extraer información importante, como `cpuinfo`, `meminfo`, etc.

Ejemplo `cat /proc/cpuinfo`

- **/root:**

Se puede asemejar a un `/home` pero exclusivo para el usuario `root` o privilegiado.

- **/svr:**

almacena ficheros y directorios relativos a servidores que tienes instalados en el sistema, como `web`, `FTP`, `CVS`, etc.

- **/sys:**

junto con `/dev` y `/proc`, es otro de los especiales. Y como `/proc`, realmente no almacena nada, sino que es una interfaz también. En este caso, son ficheros virtuales con información del kernel e incluso, se pueden emplear algunos de sus ficheros para configurar ciertos parámetros del kernel.

- **/tmp:**

Directorio para ficheros temporales de todo tipo.

Es empleado por los usuarios para almacenar de forma temporal ciertos ficheros, hay otro directorio para lo mismo en `/var/tmp`.

- **/var:**

Se trata de un directorio con directorios y ficheros que suelen crecer de tamaño, como bases de datos, logs, etc.

Es precisamente los logs o registros del sistema por lo que es más popular este directorio, y allí encontrarás muchísima información de todo lo que ocurre en el sistema: `/var/logs/`.

- **/usr:**

User System Resources, y actualmente almacena ficheros de solo lectura relativos a utilidades del usuario, como los programas instalamos mediante el gestor de paquetes.

No todas las distros siguen este esquema y puede haber pequeñas variaciones, pero si se adaptan al FHS.

Estructura de directorios de Microsoft Windows.

- **\Archivos de programa:**

Se encuentran los programas instalados, podemos encontrar otra con (x86) para las aplicaciones de 32bits en sistemas de 64bits

- **\PerfLogs:**

Puede contener registros de rendimiento del sistema en ficheros log.

- **\ProgramData.**

Carpeta oculta que contiene los datos de programas genéricos para todos los usuarios.

- **\Usuarios.**

Contiene una carpeta por cada usuario(contiene el perfil del usuario, compuesto por ficheros de configuración y las carpetas de Descarga, Documentos, Escritorio, ...) , además de Acceso público y Default (contiene el perfil base que se le transfiere a cada usuario cuando se crea).

- **\Windows.**

Contiene la instalación del sistema operativo.

- **\Windows\System32**

Contiene las librerías de enlace dinámico (DLL, son ficheros con código ejecutable, datos. Se utilizan por modularidad) para el sistema operativo.

- **\SysWOW64.**

Almacena DLL de 32bits en el caso de sistemas de 32bits

- **\WinSxS.**

Es el almacén de Windows, contiene archivos de la instalación, actualizaciones o Service Packs entre otros.

Gestión de archivos GNU/Linux

Directorio actual

Sintaxis **pwd**

- En el prompt del sistema también da esta información.
- El símbolo ~ representa nuestro directorio personal.



```
ricardo@AH512: /var/www
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
ricardo@AH512: /var/www$ pwd
/var/www
ricardo@AH512: /var/www$
```

Cambiar de directorio.

Sintaxis **cd [ruta]**

- **cd ~** nos lleva a nuestro directorio personal.
- **cd** nos lleva a nuestro directorio personal.
- **cd ..** nos lleva al directorio padre del actual, si estamos en el raíz nos deja en el raíz.

Rutas

Absolutas. Son las rutas a un fichero o directorio que parten desde el directorio raíz, siempre empiezan por el símbolo **/**.

Relativa. Son las rutas hasta un fichero o directorio partiendo desde el directorio actual. Nunca empiezan por el símbolo **/**.

Mostrar el contenido de un directorio.

Sintaxis **ls [opciones] [rutas_ficheros_o_directorios]**

Opciones:

- **-l** formato largo, da información de cada entrada.
- **-a** muestra los ficheros o directorios ocultos.
- **-h** muestra la información para humanos.

Caracteres comodín * y ?

Sintaxis **mkdir [opciones] directorio [directorio ...]**

Opciones:

- **-p** Crear directorios padres si es necesario.
- **-v** Salida detallada. (modo verboso).
- **-m 777** establecemos las opciones de seguridad.

Borrar directorios.

Sintaxis **rmdir [directorios] [opciones]**

Elimina los archivos o directorios, solo si están vacíos.

Copiando ficheros y directorios

El comando **cp** es un abreviatura de **copy** (copiar); permite copiar archivos y directorios.

Sintaxis

- **cp [Opciones] archivo_fuente directorio_destino**

- **cp [Opciones] archivo_fuente archivo_destino**

Opciones

- -a conserva todos los atributos de los archivos.
- -d copia un vínculo pero no el fichero al que se hace referencia.
- -i pide confirmación antes de sobrescribir archivos.
- -p conserva permisos del archivo origen: propietario, permisos y fecha.
- -R o -r copia los archivos y subdirectorios.
- -s crea enlaces en vez de copiar los ficheros.
- -u únicamente procede a la copia si la fecha del archivo origen es posterior a la del destino.
- -v muestra mensajes relacionados con el proceso de copia de los archivos.

Descripción

El comando cp copia un archivo a otro. También puede copiar varios ficheros en un directorio determinado.

Moviendo objetos

Mover un fichero consiste en cambiar de ubicación un fichero o directorio.

Cuando usamos mv primero comprueba si existe, si es así la acción por defecto es preguntar si queremos sobrescribir el destino. Si queremos evitarlo hemos de usar la opción -f. Con la opción -r hacemos la copia de forma recursiva (incluyendo los ficheros y directorios que contiene).

El uso es similar a cp con la diferencia de que no mantiene el original.

Sintaxis

- **mv [Opciones] fuente destino**

Opciones

- -f elimina los archivos sin solicitar confirmación.
- -v pregunta antes de sobrescribir los archivos existentes.
- -r mueve archivos y directorios que contiene

Descripción

El comando mv se puede utilizar para modificar el nombre o mover un archivo de un directorio a otro. Trabaja tanto con archivos como con los directorios.

Eliminando objetos

rm permite eliminar archivos y directorios, puede eliminar un directorio completo con la opción -r.

Sintaxis

rm [Opciones] archivos o directorio

Opciones

- -f elimina todos los archivos sin pedir confirmación
- -i pregunta antes de eliminar un archivo.
- -r elimina todos los archivos que se encuentran en un subdirectorio y por último borra el propio subdirectorio.
- -v muestra el nombre de cada archivo antes de eliminarlo.

Descripción

El comando **rm** se utiliza para borrar los archivos que se le especifiquen. Para eliminar un fichero ha de tener permiso de escritura en el directorio en el que se encuentra.

Creando ficheros

touch

Crea fichero vacío con el nombre especificado si no existe o lo “toca” (le cambia la fecha a la fecha actual“

Sintaxis

touch [archivo]

Opciones

no tiene

Descripción

Crea un un fichero si no existe y, si existe, modifica su atributo fecha y hora a la fecha y hora actual.

Mostrar contenido de un fichero.

cat

El comando **cat** (concatenate), es un comando que ha sido desarrollado para que ejecute básicamente tres funciones asociadas a los archivos de texto, estas son:

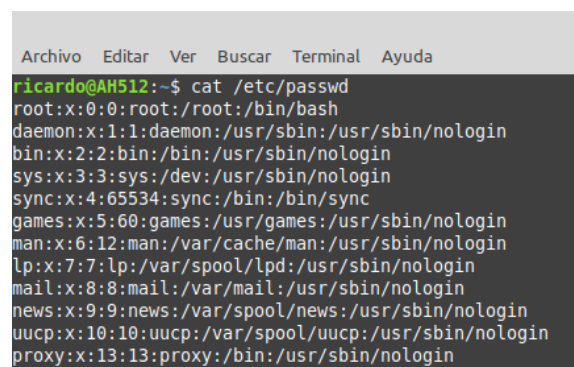
- Poder visualizarlos
- Realizar una combinación con copias de ellos
- Crear nuevos archivos.

Su sintaxis de uso es:

cat [opciones] [archivo] [-] [archivo]

Opciones:

-n, --number (numero)



```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
ricardo@AH512:~$ cat /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
sys:x:3:3:sys:/dev:/usr/sbin/nologin
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/usr/sbin/nologin
man:x:6:12:man:/var/cache/man:/usr/sbin/nologin
lp:x:7:7:lp:/var/spool/lpd:/usr/sbin/nologin
mail:x:8:8:mail:/var/mail:/usr/sbin/nologin
news:x:9:9:news:/var/spool/news:/usr/sbin/nologin
uucp:x:10:10:uucp:/var/spool/uucp:/usr/sbin/nologin
proxy:x:13:13:proxy:/bin:/usr/sbin/nologin
```

more

Al igual que 'cat', 'more' permite visualizar por pantalla el contenido de un fichero de texto, con la diferencia con el anterior de que 'more' pagina los resultados.

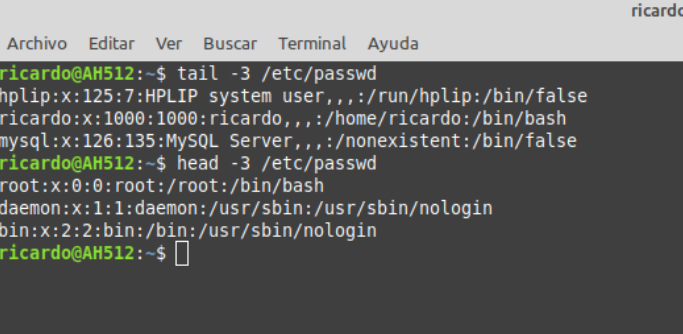
Primero mostrará por pantalla todo lo que se pueda visualizar sin hacer scroll y después, pulsando la tecla espacio avanzará de igual modo por el fichero.

less

El comando 'less' es el más completo de los tres, pues puede hacer todo lo que hace 'more' añadiendo mayor capacidad de navegación por el fichero (avanzar y retroceder) .

head y tail.

Los comandos head y tail permiten mostrar de forma parcial el contenido de un fichero. Como su nombre indica, head muestra las primeras líneas del fichero (la cabecera) y tail muestra las últimas líneas (la cola).



```
Archivo  Editar  Ver  Buscar  Terminal  Ayuda
ricardo@AH512:~$ tail -3 /etc/passwd
hplip:x:125:7:HPLIP system user,,,:/run/hplip:/bin/false
ricardo:x:1000:1000:ricardo,,,:/home/ricardo:/bin/bash
mysql:x:126:135:MySQL Server,,,:/nonexistent:/bin/false
ricardo@AH512:~$ head -3 /etc/passwd
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/usr/sbin/nologin
bin:x:2:2:bin:/bin:/usr/sbin/nologin
ricardo@AH512:~$
```

Redirecciones

Cada vez que se inicia un intérprete de órdenes, se abren automáticamente tres archivos:

stdin: archivo estándar de entrada, es de donde los programas leen su entrada

stdout: archivo estándar de salida, es a donde los programas envían sus resultados

stderr: archivo estándar de error, es a donde los programas envían sus salidas de error

Los operadores de redirección `>` | `>>` | `<` permiten cambiar los flujos E/S estándar.

stdin se identifica generalmente con el teclado.

stdout y **stderr** se identifican normalmente con la pantalla.

Redirección de salida (stdout)

Hay dos opciones:

- `orden > fichero`
- `orden >> fichero`

Se puede redireccionar la salida de cualquier orden a un determinado archivo en vez de hacerlo por la salida estándar stdout o pantalla.

Para obtener una redirección de salida, se utiliza el carácter mayor que, `>`.

Si el archivo al que redireccionamos no existe, el shell lo creará automáticamente, si, por el contrario ya existía, entonces se sobrescribirá la información, machacando el contenido original del archivo.

`date > prueba.txt` → Se almacena la fecha actual del sistema en el archivo prueba.txt

Si lo que queremos es añadir información a un archivo sin destruir su contenido, deberemos utilizar para la redirección el doble símbolo de mayor que, `>>`.

`ls >> prueba.txt` → Se almacena al final del fichero prueba.txt el contenido del directorio activo.

Redirección de la salida de errores (stderr)

`$ orden 2> fichero`

\$ orden 2>> fichero

Este tipo de redireccionamiento se utiliza para almacenar los errores producidos en la ejecución de un comando en un archivo específico, en vez de utilizar la pantalla.

Para ello se utilizan los caracteres 2> para crear y 2>> para añadir.

Ejemplo

cat fichero 2> errores.log

Si el comando cat fichero da un error, se crea un fichero llamado errores.log con el contenido del mensaje del error que se hubiera mostrado por pantalla.

Si el fichero errores existe, se destruye y se queda con la última información; si no existe, se crea.

Si **no queremos ver errores** en pantalla, podemos redireccionar la salida estándar de errores a /dev/null, es el cubo de basura en Linux. En el siguiente ejemplo, si no colocamos 2> /dev/null aparecerán errores de acceso en pantalla

find / -name *.c 2> /dev/null

Redirección de entrada (stdin)

\$ orden < fichero

cat < hotla.txt

Cualquier orden que lea su entrada en stdin puede ser avisada para que tome dicha entrada de otro archivo en lugar de tomarla por el teclado.

Esto se hace utilizando el carácter menor que, <. La redirección de entrada no produce ningún cambio en el archivo de entrada.

Ejemplo

mail pepe < listado2

Se envía el fichero listado2 por correo electrónico al usuario pepe.

Expresiones regulares

Las expresiones regulares son un medio para describir patrones de texto. Imaginemos que no sólo queremos buscar en un texto todas las líneas que contienen una palabra, como por ejemplo Barcelona, sino que sólo nos interesan las líneas que empiezan por la palabra Barcelona, pero no las que contengan la palabra en cualquier otra posición. Describir el patrón “Barcelona” es trivial, tan sólo hay que escribir “Barcelona”, pero ¿cómo podemos describir “La línea comienza por la palabra Barcelona”. Las expresiones regulares permiten describir este tipo de patrones de texto y muchos más por lo que nos serán de una gran utilidad.

En la web hay varios sitios en los que se pueden probar las expresiones regulares, como por ejemplo regex101.

Fichero con los ejemplos.

ospital	apellido_paciente	Biobanco	Tipo_Usher
Barcelona	Castells	Barcelona	Usher2A
Mallorca	Pagan	Barcelona	usher1b
Albacete	Blanca	Madrid	usher2
madrid	Cardosa	madrid	usher1a
almeria	Meliá	Valencia	sano
Valencia	Paniagua	Castellón	usher1B
Castellón	Sogorb	valencia	usher2B
castellon	Galiana	Barcelona	usher3
Granada	Bastella	Valencia	sana
Alicante	Carrícola	Barcelona	usher3A

https://bioinf.comav.upv.es/courses/unix/demo_data/origenes.txt

Expresiones alternativas

Una expresión u otra. Imaginemos que tenemos un fichero con los orígenes de los pacientes de un estudio sobre síndrome de Usher. Queremos seleccionar todos los pacientes que vienen de Valencia o Castellón. Podemos hacerlo utilizando la sintaxis para expresiones alternativas:

```
~$ grep -E 'Valencia|Castellón' origenes.txt
```

```
almeria Meliá Valencia sano
```

```
Valencia      Paniagua      Castellón      usher1B
```

```
Castellón     Sogorb  valencia      usher2B
```

```
Granada Bastella Valencia sana
```

La barra vertical (|) separa las expresiones alternativas. En este caso significa que la palabra encontrada puede ser Valencia o Castellón.

Contenedores

En muchas ocasiones nos interesa seleccionar patrones que pueden tener en una posición varias letras distintas. Por ejemplo podríamos describir el patrón “comienza por Usher y después tiene un número” o “comienza por usher y después tiene un par de letras”. Busquemos todas las líneas que tienen la palabra Usher o usher:

```
~$ grep -E --color '[uU]sher' origenes.txt
```

```
Hospital    apellido_paciente    Biobanco    Tipo_Usher
Barcelona    Castells            Barcelona    Usher2A
...
```

Al poner las letras entre corchetes indicamos que el carácter que aparezca en esa posición puede ser cualquiera de los caracteres indicados. En este caso la expresión coincidente podría ser Usher u usher.

Utilizando esta técnica podemos también indicar rangos de caracteres, como por ejemplo “aquellos con Usher tipo 0,1 ó 2”:

```
~$ grep -E --color '[uU]sher[0-2]' origenes.txt
```

```
Barcelona    Castells            Barcelona    Usher2A
Mallorca     Pagan    Barcelona    usher1b
...
```

Existen rangos y tipos de caracteres predefinidos que podemos utilizar como:

POSIX	ASCII	Significado
<code>[:alnum:]</code>	<code>[A-Za-z0-9]</code>	Caracteres alfanuméricos (letras y números)
<code>[:word:]</code>	<code>[A-Za-z0-9_]</code>	Caracteres alfanuméricos y “_”
<code>[:alpha:]</code>	<code>[A-Za-z]</code>	Caracteres alfabéticos
<code>[:blank:]</code>	<code>[\t]</code>	Espacio y tabulador

POSIX	ASCII	Significado
<code>[:space:]</code>	<code>[\t\r\n\v\f]</code>	Espacios
<code>[:digit:]</code>	<code>[0-9]</code>	Dígitos
<code>[:lower:]</code>	<code>[a-z]</code>	Letras minúsculas
<code>[:upper:]</code>	<code>[A-Z]</code>	Letras mayúsculas
<code>[:punct:]</code>	<code>[!'"#\$%&'()*+,-./:;<=>?@\^_`{\}~]</code>	Caracteres de puntuación

Por ejemplo, podríamos seleccionar cualquier dígito o letra:

```
~$ grep --color -E '[uU]she[[:alpha:]]' origenes.txt
```

Si nos da igual que en una posición haya un carácter u otro podemos utilizar un punto (.). Por ejemplo, seleccionar aquellos individuos con subtipo A o B:

```
~$ grep --color -E '[Uu]sher.[[:alpha:]]' origenes.txt
```

```
Barcelona  Castells  Barcelona  Usher2A
Mallorca   Pagan  Barcelona  usher1b
...
```

Dentro de los contenedores resultan también bastante útiles las negaciones. Podemos, por ejemplo, seleccionar cualquier carácter que no sea un 2 o un 3 añadiendo “^” tras el primer corchete:

```
~$ grep --color -E 'sher[^23]' origenes.txt
```

```
Mallorca  Pagan  Barcelona  usher1b
madrid     Cardosa madrid usher1a
Valencia   Paniagua  Castellón  usher1B
```

Cuantificadores

Además de indicar qué caracteres queremos permitir podemos seleccionar cuantas veces deben aparecer. Si no añadimos nada que indique lo contrario se asume que el carácter debe aparecer una vez, pero podríamos pedir que el carácter aparezca un número distinto de veces:

- “?”, el carácter aparece ninguna o una vez. “usher1?” coincidiría con “usher” o “usher1”.
- “*”, cero, una o varias veces.
- “+”, al menos una vez.
- “{4}”, cuatro veces.
- “{4,10}”, entre 4 y 10 veces

```
~$ grep --color -E 'sher2.?' origenes.txt
```

```
Barcelona   Castells   Barcelona   Usher2A
Albacete    Blanca    Madrid     usher2
Castellón   Sogorb    valencia    usher2B
```

Puntos de anclaje

Además de poder indicar qué y cuántas veces queremos que algo aparezca podemos indicar dónde deseamos que lo haga. Los puntos de anclaje más utilizados son:

- “^”, inicio de línea
- “\$”, fin de línea
- “<”, principio de palabra
- “>”, fin de palabra
- “\b”, límite de palabra

Líneas que comienzan por B:

```
~$ grep --color -E '^B' origenes.txt
```

Barcelona	Castells	Barcelona	Usher2A
-----------	----------	-----------	---------

Líneas que terminan por B:

```
~$ grep --color -E 'B$' origenes.txt
```

Valencia	Paniagua	Castellón	usher1B
Castellón	Sogorb	valencia	usher2B

Escapes

Algunos caracteres tienen significados especiales, como ., \$, (,), [,], \ o ^ y si se quieren utilizar hay que escaparlos precediéndolos con \.

El campo de las expresiones regulares es muy amplio y esta pequeña introducción sólo ha pretendido mostrar algunas de las posibilidades de esta gran herramienta. El mejor modo de aprender a utilizarlas es intentar hacer uso de ellas en problemas concretos y echar una ojeada a algunos de los libros y tutoriales que se han escrito sobre ellas.

Ejercicios

1. Buscar las líneas en las que aparece la palabra bash en el archivo /etc/passwd.
2. Buscar en el archivo /etc/group todas las líneas que empiezan por m.
3. En el fichero anterior imprimir todas las líneas que no empiezan por m.
4. ¿Cuántos ficheros README hay en los subdirectorios de /usr/share/doc?
5. ¿Qué ficheros o directorios en /etc contienen un número en el nombre?

Trabajando con el fichero de los orígenes de los pacientes de Usher resolver las siguientes cuestiones.

6. Seleccionar los pacientes enviados por el hospital de Castellón.
7. Seleccionar los que vienen del Biobanco de Barcelona.
8. Buscar los pacientes que no tienen Usher3.
9. En el fichero de pacientes de Usher hay algunas líneas separadas por espacios y otras por tabuladores, cambiar todos los separadores a comas.

Disponemos de un fichero con una caracterización morfológica de unas plantas.

10.Recuperar aquellas plantas numeradas con 1 o con 3 dígitos.

11.Añadir ceros al identificador de las plantas para que tengan todos 3 dígitos.

12.Construir un nuevo fichero que incluya solo las filas sin datos faltantes.

Tuberías y filtros.

Podemos en una misma línea ejecutar dos o más comandos separados por el carácter tubería | , pasando salida estándar del primer comando con la entrada estándar del siguiente comando.

- Se usa para es filtrar datos utilizando comandos denominados filtros.
- Se pueden unir mas de dos comandos

orden1 | orden2

De esta forma, puedes ejecutar un comando, y su salida puede ser la entrada para otro, que a su vez puede servir de entrada a un tercero:

comando1 | comando2 | comando3,

Ejemplos:

```
more /etc/bash.bashrc
```

```
cat /etc/bash.bashrc | more
```

```
ls | more
```

```
less /etc/bash.bashrc
```

```
cat /etc/bash.bashrc | less
```

Obtiene los 10 últimos caracteres de fich.

```
cat -n /etc/bash.bashrc|head -n-2
```

Quita las 2 últimas líneas del fichero /etc/bash.bashrc

wc (Word Count)

Cuenta el número de caracteres, palabras y líneas de uno o más fichero(s), incluye espacios en blanco y caracteres de salto de línea.

Cuando se indica el nombre de más de un fichero wc proporciona los contadores individuales y su totalidad y se presentan los de cada fichero etiquetados con su nombre.

Si no se especifica ningún fichero, leerá de la entrada estándar. Su sintaxis es:

sintaxis

wc [opciones] [fichero(s)]

Donde las opciones pueden ser:

- -c visualiza sólo el número de caracteres del fichero.
- -l visualiza sólo el número de líneas del fichero.
- -w visualiza sólo el número de palabras del fichero.

Ejemplos:

cat fichero | wc -l

Cuenta las líneas de fichero.

ls -l /etc | grep -E '^- ' | wc -l

Cuenta el número de archivos de un directorio.

ls -l /etc | grep -E '^d' | wc -l

Cuenta el número de directorios de un directorio.

ls -la | wc -l

Cuenta las líneas del listado completo de ficheros.

tr (Traductor)

Traduce, comprime y borra caracteres de la entrada estándar, escribiendo el resultado en la salida estándar.

Sintaxis

tr [opciones] conjunto1 [conjunto2]

Opciones:

- **-c** opera sobre el complemento (sobre cada carácter que no coincida)
- **-d** borra caracteres de conjunto1, no traduce
- **-s** reemplaza cada sucesión de entrada de un carácter repetido del conjunto1 por una sola aparición de dicho carácter, muy útil para convertir varios espacios en uno solo.

Ejemplos:

```
cat vocales.txt | tr -dc [aeiou]
```

Borro todo lo que no sean vocales

cut (Cortar)

Se usa para cortar columnas en ficheros y pasar a la salida estándar las columnas o campos de la entrada estándar o del archivo especificado. Su sintaxis es:

Sintaxis:

cut [opciones] lista [fichero(s)]

Donde las opciones pueden ser:

- **-c** corta caracteres.
- **-f** corta campos.
- **-d** especifica el carácter de separación entre los distintos campos (delimitador). Por defecto el separador de campos es <TAB>

Ejemplo:

```
cat /etc/passwd | cut -f7 -d:
```

Nos quedamos con el campo 7 del fichero /etc/passwd

```
cat /etc/passwd | cut -c1
```

Nos quedamos con con la primera columna de caracteres del fichero /etc/passwd

grep

Es un filtro que permite buscar cadenas de caracteres (patrón) en los archivos que se le indica y muestra la línea del fichero que contiene el patrón. Su sintaxis:

Sintaxis

```
grep -E [opciones] patrón [fichero(s)]
```

Donde las opciones pueden ser:

- -c visualiza el nº total de líneas donde se localiza el patrón.
- -i elimina la diferencia entre mayúsculas y minúsculas.
- -l visualiza sólo el nombre de los ficheros donde se localiza el patrón.
- -n visualiza el nº de línea donde aparece el patrón, así como la línea completa.
- -v visualiza las líneas del fichero donde no aparece el patrón.
- -w obliga a que patrón coincida solamente con palabras completas
- -x obliga a que patrón coincida solamente con líneas completas

Lo mejor de este comando es que admite “Expresiones Regulares”

```
ls -ld [0-9]* | grep «^d»
```

Un ejemplo: Mostrar sólo los directorios que empiecen por un número

```
ls -ld [A-Z]* | grep «^d»
```

Directorios que comiencen por letra mayúscula → utiliza una expresión regular.

Sort (ordenar)

Se utiliza para ordenar líneas de un fichero o un flujo, puede usarse como comando o como filtro.

Puede ordenar alfabética o numéricamente, teniendo en cuenta que cada línea del fichero es un registro compuesto por varios campos, los cuales están separados por un carácter denominado separador de campo (tabulador, espacio en blanco, dos puntos...). Su sintaxis es:

Sintaxis

```
sort [opciones] [fichero/s]
```


Donde las opciones más interesantes son:

- -f ignora mayúsculas y minúsculas.
- -n ordena campos numéricos por valor no por caracteres.
- -r orden inverso.
- -u suprime líneas repetidas en el fichero de salida
- -t indica el delimitador de campos, por defecto el espacio en blanco.
- -kN indica el campo por el que ordenar (hasta el final)
- -kN,M indica el campo por el que ordenar (hasta el campo M)
- -kN,N indica que ordene sólo por el campo N (N=M)

ficheros nombres de los ficheros a ordenar.

Ejemplos:

```
ls -l | grep '^-' | sort -k5n
```

Lista solo ficheros y los ordena de forma numérica por la quinta columna, que en mi distro es el tamaño.

ls -l → Lista

grep ^- → filtra sólo los ficheros, que comienzan por –

-k5 → ordeno por el quinto campo/columna

-n → de forma numérica

```
sort -k3 -nt»» /etc/passwd
```

Ordena de forma numérica por el tercer campo (UID) el fichero de usuarios que tiene como separador :

Gestión de archivos por interfaz gráfica en Microsoft Windows.

Para llevar acabo la gestión de archivos utilizamos el Explorador de archivos.

Combinaciones	Descripción
Ctrl+x	Cortar el elemento seleccionado
Ctrl+c	Copiar el elemento seleccionado
Ctrl+v	Pegar el elemento seleccionado
Ctrl+z	Deshacer una acción
F2	Modificar el nombre del elemento
Ctrl+e	Seleccionar todos los elementos
Ctrl+d	Eliminar el elemento seleccionado y enviarlo a la papelera de reciclaje
Ctrl+click ratón sobre elementos	Seleccionar distintos elementos

Gestión de archivos por la consola en Microsoft Windows.

Comandos son:

CD

DIR

MD

RD

COPY

XCOPY

MOVE

DEL

TREE

TYPE

Raid

Tipos de Raid (esta en un pdf fuera)

Particiones de disco

Para poder **utilizar** una unidad de disco (ssd, memory stick, ...) debes realizar siempre **dos pasos** que son:

- Realizar una o varias **particiones** con el espacio que quieras utilizar del disco, se conoce como particionar el disco.
- Establecer es **sistema de archivos** (formatear) deseado en la partición o particiones creadas.

Pero antes debemos **elegir** el tipo de **esquema de particiones** deseadas utilizar entre las dos opciones existentes, **MBR** o **GPT**.

MBR o GPT son dos formas diferentes de almacenar la información sobre las particiones que hay en cada unidad y **MBR** la usa **BIOS** y **GPT** la usa **UEFI**, esta información entre otra será:

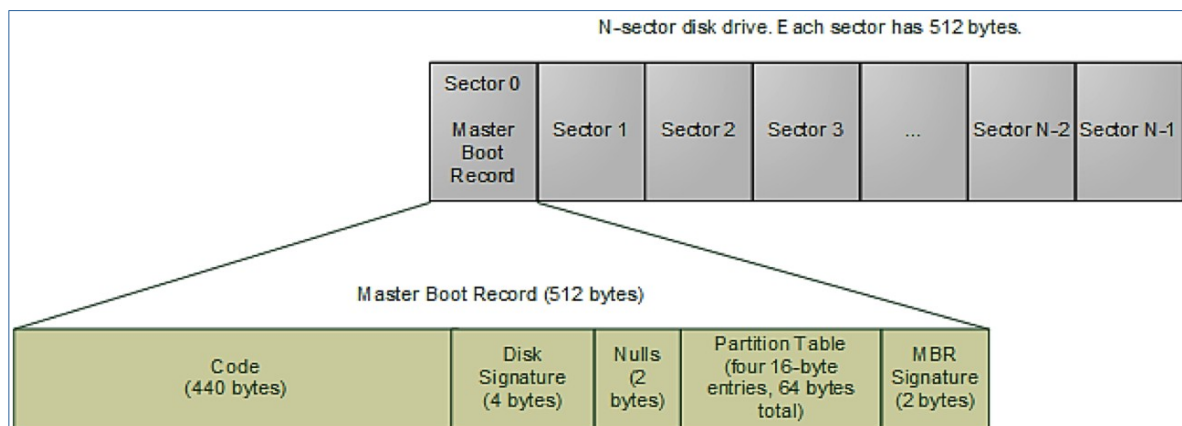
- Sector de inicio y tamaño de esta.
- Sistemas de archivo que contienen.
- Si es arrancable.



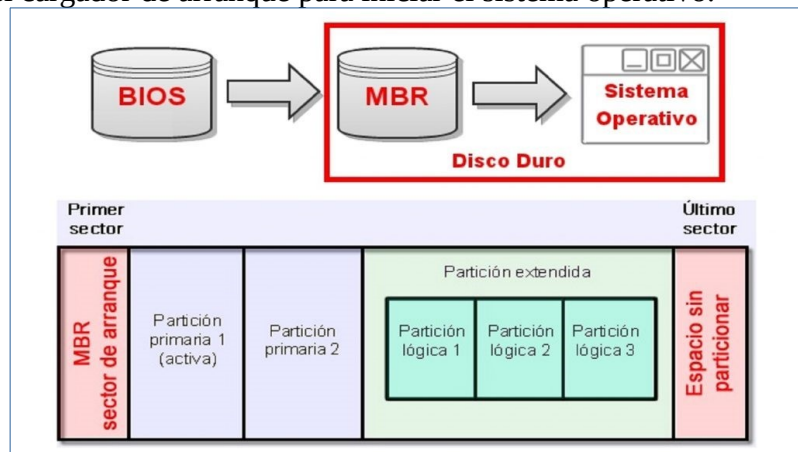
MBR.

Es el esquema más antiguo, fue desarrollado por IBM en 1983.

Se encuentra en el primer sector de cada disco, en los primeros bytes se encuentra el cargador de arranque del Sistema Operativo (sea el de **Windows** o **grub** para **Linux**), a continuación de esta tendremos la información de las particiones existentes en el disco.

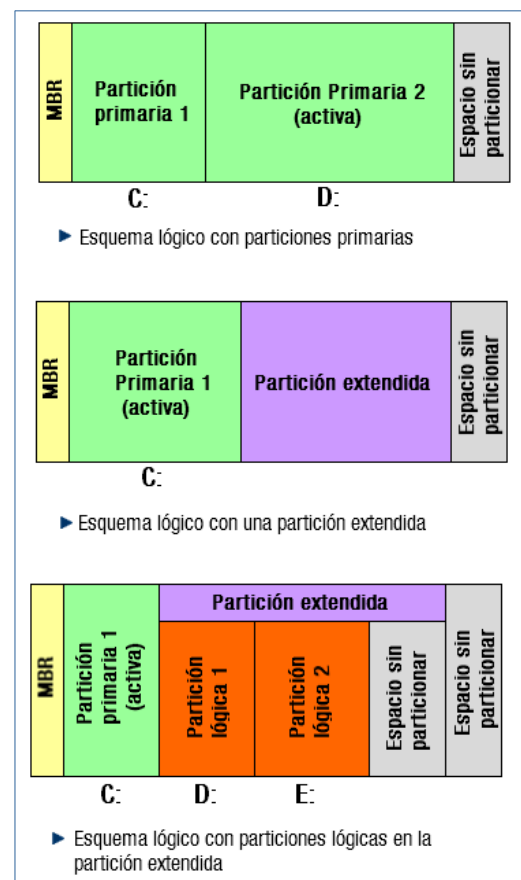


La **BIOS** busca el cargador de arranque para iniciar el sistema operativo.



Este sistema ha ido bien durante muchos años aunque tenía varias limitaciones que con el tiempo han hecho que se quedara obsoleto.

- Solo funciona para unidades de hasta 2TB de tamaño.
- Solo admite cuatro particiones primarias, aunque esto se podía solventar usando particiones extendidas, que solo eran capaces de almacenar particiones lógicas las cuales si podían almacenar ya información.
- Falta de seguridad.



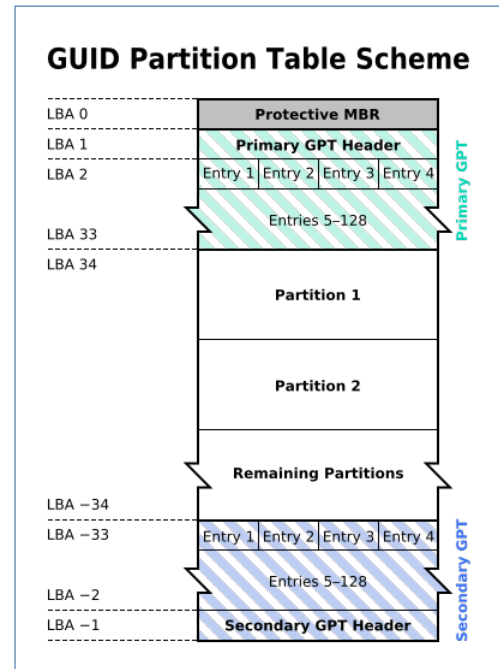
GPT

La **tabla de particiones GUID (GPT)** es un estándar para la colocación de la tabla de particiones en un disco duro físico.

Es **parte del estándar** Extensible Firmware Interface (**EFI**) propuesto por Intel para reemplazar el viejo BIOS del PC. La GPT sustituye al Master Boot Record (MBR) usado con BIOS.

Estructura:

- **Protector MBR:** Garantiza la compatibilidad con otros estilos de partición anteriores.
- **Tabla de partición GUID primaria:** cabecera GPT y entradas de partición.
- **Particiones:**
- **Tabla de partición GUID secundaria:** copia de seguridad de la cabecera GPT y las entradas de partición.



Ventajas:

- **Particiones primarias ilimitadas:** Los sistemas operativos establecen un límite. Por ejemplo, el valor de Windows asciende a 128.
- **Protección mediante sumas de verificación CRC32:** las sumas de verificación garantizan la integridad de la cabecera GPT, ya que permiten detectar los sectores defectuosos que la dañan, entre otras cosas.
- **Identificación clara de particiones y medios de almacenamiento:** Todas las particiones y medios de almacenamiento obtienen un número de identificación único.
- **Copia de seguridad de la cabecera:** Respaldada por una copia de seguridad idéntica, minimiza el riesgo de perderlos en caso de un fallo del hardware.
- **Compatibilidad con sistemas anteriores:** el llamado **Protective Master Boot MBR** del sector 0, el primer bloque de datos de un disco duro GPT, asegura que casi todos los sistemas operativos, servicios y herramientas diseñados para la partición MBR funcionen también con la GPT.

Para mas información sobre la estructura de la **Header GPT** (cabecera GPT) o **Entry partitions** (entrada partición).

En la siguiente url: <https://www.ionos.es/digitalguide/servidores/configuracion/en-que-consiste-la-particion-gpt/>

Tipos de particiones

Hay tres tipos de particiones.

- Primaria.
 - Como máximo pueden haber 4 en MBR y en GPT lo determina el Sistema Operativo.
 - Pueden albergar información (siempre que se le establecido un sistema de ficheros).
- Partición extendida.
 - No pueden albergar información.
 - Tienen solo sentido en MBR.
 - Se crearon para albergar particiones lógicas.
 - Se usan para eliminar el límite de 4 particiones primarias.
- Partición lógica.
 - Pueden haber hasta 15.
 - Tienen solo sentido en MBR.
 - Pueden albergar información (siempre que se le establecido un sistema de ficheros).
 - Tienen en algunos sistemas operativos ciertas limitaciones, como no poder albergar un sistema operativo arrancable.

Como se nombran los dispositivos de almacenamiento.

Los dispositivos se encuentran en el directorio /dev y son ficheros. Los más importantes para trabajar con dispositivos son:

- **/dev/hdxy:**

Para discos de tipo IDE, donde x es el una letra que indica el disco e y la partición de ese disco.

- **/dev/sdxy.**

Para discos de tipo SATA, USB, memorias FLASH, donde x es el una letra que indica el disco e y la partición de ese disco

Ejemplos

- **/dev/sdb.** Segundo disco SATA o dispositivo USB, ...
- **dev/sdc1.** Primera partición del tercer disco SATAo dispositivo USB

Particionar un disco.

Tenemos dos formas con la consola o mediante una herramienta con GUI.

Para las explicaciones:

1. Vamos a realizar estas tareas sobre una instalación mínima de Ubuntu 20.04 en una maquina virtual sobre Vmware.
2. Una vez instalado y configurado Ubuntu realizaremos una **SnapShot** (una instantanea) con el nombre de **Nuevo**, de esta forma podremos volver al estado inicial después de cada ejemplo.
3. Usaremos Vmware, ya que, nos permite añadir los discos duros con la maquina encendida (en caliente)

Particionar disco mediante GUI.

Particionar disco mediante terminal.

En el siguiente enlace tenéis 5 videos con una todos los pasos para realizar una práctica guiada de como de como realizar las particiones tanto con gui, como terminal, tanto para esquema de particiones MBR y GPT.

https://www.youtube.com/playlist?list=PLPcwuebUwBNI9rNSOZrTsT_Ci5ChXZxBV

Formatear particiones (Dar sistema de archivos)

(Esta en un PDF fuera)

Desfragmentar.

La desfragmentación es el proceso conveniente mediante el cual se acomodan los archivos en un disco para que no se aprecien fragmentos de cada uno de ellos, de tal manera que quede contiguo el archivo y sin espacios dentro del mismo.

Al irse escribiendo y borrando archivos continuamente en el disco duro, los fragmentos tienden a no quedar en áreas continuas, así, un archivo puede quedar "partido" en muchos pedazos a lo largo del disco, se dice entonces que el archivo está "fragmentado".

Al tener fragmentos de incluso un archivo esparcidos por el disco, se vuelve ineficiente el acceso a los archivos. Los fragmentos de uno o varios archivos es lo que hace factible la desfragmentación.

Utilizamos el comando

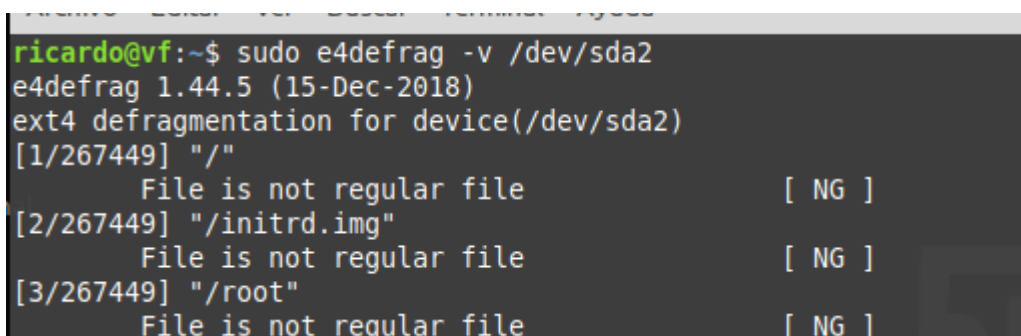
`e4defrag [opciones]`

- c

Muestra un recuento de la fragmentación actual con respecto a la fragmentación ideal.

-v

Muestra el recuento para cada archivo antes y después de la desfragmentación.



```
ricardo@vf:~$ sudo e4defrag -v /dev/sda2
e4defrag 1.44.5 (15-Dec-2018)
ext4 defragmentation for device(/dev/sda2)
[1/267449] "/"
      File is not regular file          [ NG ]
[2/267449] "/initrd.img"
      File is not regular file          [ NG ]
[3/267449] "/root"
      File is not regular file          [ NG ]
```

Chequeo

Dentro de un sistema de archivos los más importante son los datos que contiene, si se produce un error en un dispositivo y no hay copia de seguridad puede ser un desastre.

En algunos casos esos datos pueden ser recuperados.

Los discos pueden tener los siguientes problemas:

- Malware.
- Fallos electrónicos por humedad o condensación.
- Fallos el suministro eléctrico, como picos de tensión, cortes frecuentes de luz o fluctuaciones.
- Fallos físicos (caídas, vibraciones, golpes) o mecánicos (por el uso).

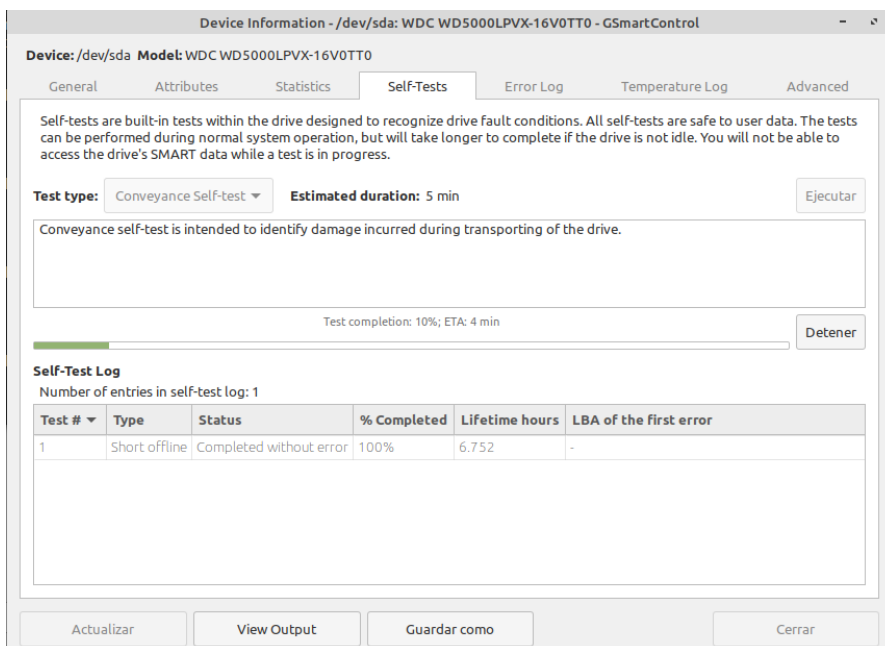
Los discos incorporan la tecnología SMART (siglas de Self Monitoring Analysis and Reporting Technology), consiste en la capacidad de detección de fallos del disco duro.

La detección con anticipación de los fallos en la superficie permite al usuario el poder realizar una copia de su contenido, o reemplazar el disco, antes de que se produzca una pérdida de datos irrecuperable.

Este tipo de tecnología tiene que ser compatible con el BIOS del equipo, estar activada y además que el propio disco duro sea compatible.

Tenemos diversas herramientas para acceder a los datos de SMART

- GSmartControl para linux.
- CrystalDiskInfo para Windows.



También tenemos herramientas para reparar fallos en el sistema de archivos como son:

- fsck
- e2fsck.

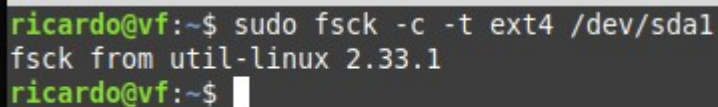
fsck

Si un sistema de archivos desarrolla una inconsistencia, se recomienda verificar su integridad.

Esto se puede completar a través de la utilidad del sistema llamada fsck (comprobación de coherencia del sistema de archivos) Esta comprobación se puede hacer automáticamente durante el tiempo de arranque o ejecutarse manualmente.

El comando Fsck debe ejecutarse con privilegios de superusuario o root . Puedes usarlo con diferentes argumentos. Algunas de las opciones más importantes:

- A: Chequea los sistemas de archivos indicados en /etc/fstab.
- C: Muestra una barra de progresos.
- V: Modo verboso
- a: repara sin pedir confirmación.
- r: pide confirmación para reparar.
- t : Indicamos el tipo de sistema de archivos ext4,fat, ntfs, ...



```
ricardo@vf:~$ sudo fsck -c -t ext4 /dev/sda1
fsck from util-linux 2.33.1
ricardo@vf:~$
```

e2fsck

La herramienta e2fsck permite escanear el sistema de archivos de particiones ext2 ext3 ext4 verificando que no existan errores.

Pero es necesario que el sistema este desmontado por lo que si es el sistema de archivos donde esta el sistema operativo hay que arrancar con un live_cd y ejecutar el comando.

Gestión de almacenamiento por línea de comandos en Windows

Particionamiento de disco.

(Esta en un PDF fuera)

Formatear particiones (Dar sistema de archivos)

(Esta en un PDF fuera)

Búsqueda de información por líneas de comando.

El comando más común utilizado para encontrar y filtrar archivos en Linux es a través del comando find.

La sintaxis básica

find <startingdirectory> <options>

El argumento <startingdirectory> es el punto de origen de donde deseas iniciar la búsqueda.

El segundo argumento es el filtro que deseas usar para buscar tu archivo. Este podría ser el nombre, tipo, fecha de creación o de modificación del archivo, etc.

Búsqueda por nombre

Usamos la opción -name y el nombre del fichero también se pueden utilizar * y ? (no confundir con expresiones regulares).

La opción -iname hace que la búsqueda sea indistinta de mayúsculas y minúsculas.

```
find / -name "my-file"
```

```
find / -iname "my-file"
```

```
find / -name "my*"
```

```
find / -iname "my?file"
```

Búsqueda por tipo

Usamos el argumento -type. Linux ofrece a los usuarios las siguientes opciones para buscar archivos por tipo:

- f – archivo normal
- d – directorio o carpeta
- l – enlace simbólico
- c – dispositivos de caracteres
- b – dispositivos de bloque

`find / -type d -name "nuevo"`

Búsqueda por fecha

Buscar archivos en función de su fecha de acceso y los registros de fecha de modificación, las siguientes:

Tiempo de acceso (-atime) – Fecha más reciente en que el archivo fue leído o escrito.

Tiempo de modificación (-mtime) – Fecha más reciente en que se modificó el archivo.

Hora de cambio (-ctime) – Fecha más reciente en que se actualizaron los metadatos del archivo.

`find / -atime 1`

Esto encontrará todos los archivos a los que se accedió hace un día desde el momento actual.

`find / -mtime +2`

Esto listará todos los archivos que tienen un tiempo de modificación de más de dos días.

`find / -ctime -1`

Para buscar todos los archivos cuyos metadatos de inodo se actualizaron hace menos de un día

Búsqueda por tamaño

La sintaxis básica para buscar archivos por tamaño es:

`find <startingdirectory> -size <size-magnitude> <size-unit>`

Puedes especificar las siguientes unidades de tamaño:

- c – bytes
- k – kilobytes
- M – megabytes
- G – gigabytes
- b – trozos de 512 bytes

.

`find / -size 10M`

Esto buscará en tu sistema archivos que tengan exactamente 10 megabytes de tamaño.

`find / -size +5G`

El comando anterior listará todos los archivos de tu disco que tengan más de 5 Gigabytes de tamaño.

