Práctica 4

**File Systems**

**1. ¿Qué es un file system?**

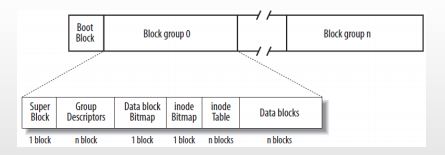
Un file system es una abstracción que permite la creación, eliminación, modificación y búsqueda de archivos y su organización en directorios.

* File System es la parte del SO que se encarga del manejo de los archivos.
* Un archivo es un conjunto de información o datos relacionados que (probablemente) vivar´a por mucho tiempo.
* Vista del usuario: conjunto de datos persistentes con un nombre
* Vista del SO: colecci´on de bloques de disco
* Tres componentes asociados con un archivo:
* Nombre
* Metadata
* Datos
* También administra el control de acceso a los archivos y el espacio en disco asignado a él.
* File systems operan sobre bloques de datos (conjunto consecutivo de sectores físicos).
* Archivos son almacenados en una estructura jerárquica de tipo árbol ( “/” en Linux, “C” en Windows).
* Define convenciones para el nombrado de los archivos.
* File systems usados en discos, CDs, etc.; otros proveen acceso por la red (NFS, SMB, etc.); otros son virtuales (procfs, sysfs).

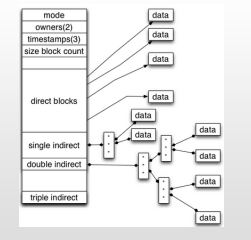
**2. Describa las principales diferencias y similitudes entre los file systems: FAT, NTFS, Ext(2,3,4), XFS, HFS+ y ZFS.**

**File Systems Ext2**

* Second Extended Filesystem (Ext2). Introducido en 1994.
* Primer sector de la partición no es administrado por Ext2
* Dividido en **“block groups”** de igual tamaño (excepto el último)
* **“Block groups”** reducen la fragmentación y aumentan la velocidad de acceso.
* **“Superblock”** y **“Group Descriptors Table”** replicados en block groups para backup.
* Cada bloque contiene su propio **“Group Descriptor”**
* **“Block Bitmap”** e **“Inode Bitmap”** indican si un bloque de datos o inodo est´a libre u ocupado.
* Tabla de inodos consiste de una serie de bloques consecutivos donde cada uno tiene un número predefinido de inodos. Todos los inodos de igual tama˜no: 128 bytes (por default).
* Cada archivo en el file system es representado por un inodo (index node).
* Inodos contienen metadata de los archivos y punteros a los bloques de datos.
* Metadata: permisos, owner, grupo, flags, tama˜no, n´umero bloques usados, tiempo acceso, cambio y modificación, etc.
* Nombre del archivo no se almacena en el inodo.
* Ext2 implementa directorios como un tipo especial de archivo cuyo bloques de datos almacenan el nombre del archivo y su inodo correspondiente (“dentry”).
* Atributos extendidos, como las ACLs, se almacenan en un bloque de datos (campo en el inodo llamado “i file acl)).
* Datos se almacenan en bloques de 1024, 2048 ó 4096 bytes.
* Esquema de un file system Ext2



* “Superblock” y “Group Descriptors” no es necesario replicarlo en todos los grupos.
* Solo “superblock” del grupo 0 se lee y se modifica
* Estructura de un inodo:

****

**Los inodos**, junto con los superbloques, son una estructura esencial en cualquier sistema de ficheros.

**El inodo es un metadato** encargado de almacenar información referente a cada fichero/directorio almacenado en cualquier sistema de ficheros (desde el punto de vista de un programador es un registro (register o struct)).

**El inodo** además de almacenar ciertos atributos del f/d (como el propietario) almacena referencias a los '**bloques**' de datos.

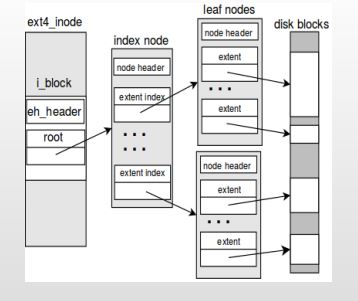
**El superbloque**, es otro tipo de estructura que no contiene información de un fichero o directorio, sino que almacena información sobre el FS(filesystem) al que pertenece. Es una estructura esencial, ya que, de querer realizar un búsqueda u otra operación sobre el FS es necesario acceder a ella, que es quien 'sabe' o 'conoce' cómo realizar las operaciones en ese tipo específico de FS. El Kernel es quien almacena una referencia a cada superbloque, de cada sistema de ficheros que haya sido registrado.

**File Systems Ext3**

* Ext3, Third Extended FileSystem, es la evolución de Ext2.
* Introducido en 2001. Disponible desde la versión de kernel 2.4.15.
* **Su principal mejora se basa en la incorporación del “journaling”, que permite reparar posibles inconsistencias en el file system**
* Es compatible con ext2.
* Tamaño máximo de archivo 2TB. Tamaño máximo de File System: 32 TB (ambos igual que en ext2).
* Cantidad máxima de subdirectorios: 32000.

**File Systems Ext4**

* Ext4, Fourth Extended FileSystem, es la evolución de Ext2.
* Introducido en 2006. Disponible desde la versión de kernel 2.6.19.
* Sistema de archivos de 64 bits (FS de 1EB, files de 16TB).
* Cantidad máxima de subdirectorios: 64000 (extendible).
* **Tamaño inodo: 256 bytes** (timestamps más precisos, ACLs).
* **Uso de extents:** descriptor que representa un rango contiguo de bloques físicos.
* Cada **extent** puede representar **2 a la 15 bloques** (128MB con bloques de 4KB, 4 extents por inodo) .
* Para archivos más grandes se utiliza un árbol de extents.
* Mejor alocación de bloques para disminuir la fragmentación e incrementar el throughput: **“persistent preallocation”**, **“delay and multiple block allocation”.**
* **Estructura de un inodo con extent:**

****

**Ext4** este tipo de sistema supone una mejora considerable respecto a su predecesor el ext3. Entre todas las mejoras, se destacan las siguientes:

* **Rendimiento y tasas de transferencia** bastante superiores a ext3.
* **Extensiones**: Se mejora la eficiencia de los descriptores de disco, reduciendo los tiempos de borrado de ficheros largos, además de otras ventajas.
* **Desfragmentación**: Aunque ext3 apenas se fragmenta, los ficheros almacenados siempre tienen cierta tendencia pequeña a estar fragmentados. ext4 añade soporte para la desfragmentación, que mejorará el rendimiento global.
* **Undelete**: ext4 soporta undelete (desborrado), herramienta para recuperar archivos que han sido borrados de forma accidental.

**File Systems XFS**

* Desarrollado por SGI (Silicon Graphics Inc.) para IRIX (su versión de UNIX)
* En 2000, SGI lo liberó bajo una licencia de código abierto.
* Incorporado a Linux desde la versión 2.4.25.
* Red Hat 7.0 lo incluye como su FS default (CentOS desde la versión 7.2)
* File System de 64 bits (16 EB max. file system, 8EB max. file size).
* FS dividió en regiones llamadas “allocation groups”. Uso de extents. Inodo asignados dinámicamente.
* Journaling (primer FS de la familia UNIX en tenerlo) .
* Mayor espacio para atributos extendidos (hasta 64KB).
* **Contra:** no es posible achicar un FS de este tipo.

Registro de Bitácora

XFS provee soporte para llevar un registro ([journaling](https://es.wikipedia.org/wiki/Journaling)), donde los cambios al sistema de archivos primero son escritos a un diario o *journal* antes de que se actualicen los datos del disco. El journal es un [buffer](https://es.wikipedia.org/wiki/Buffer_de_datos) circular de bloques del disco que no son parte del sistema de archivos. En XFS el registro (journal) contiene entradas 'lógicas' que describen a un alto nivel las operaciones que se están realizando, al contrario de otros sistemas de archivo con un registro (journal) 'físico', que guardan una copia de los bloques modificados durante cada transacción. Las actualizaciones del registro (journal) se realizan asincrónicamente para evitar una baja en el rendimiento. En el caso de una caída repentina del sistema, las operaciones inmediatamente anteriores a la caída pueden ser terminadas, garantizando así la consistencia del sistema. La recuperación se realiza automáticamente a la hora del montaje del sistema de archivos y la velocidad de recuperación es independiente del tamaño del sistema de archivos. Incluso si alguna información que fuese modificada inmediatamente antes de la caída del sistema no fuese escrita al disco, XFS se encarga de borrar todos los bloques de datos sin escribir, eliminando así cualquier compromiso de seguridad.

Grupo de Asignación

Los sistemas de archivos XFS están particionados internamente en *grupos de asignación*, que son regiones lineales de igual tamaño dentro del sistema de archivos. Los archivos y los directorios pueden crear grupos de asignación. Cada grupo gestiona sus inodos y su espacio libre de forma independiente, proporcionando escalabilidad y paralelismo — múltiples [hilos](https://es.wikipedia.org/wiki/Hilo_en_sistemas_operativos) pueden realizar operaciones de E/S simultáneamente en el mismo sistema de archivos.

**File System HFS+**

HFS Plus o HFS+ es un sistema de archivos desarrollado por Apple

Inc. para reemplazar al HFS (Sistema jerárquico de archivos). También es el formato usado por el iPod al ser formateado desde un Mac. HFS Plus también es conocido como HFS Extended y Mac OS Extended. HFS Plus es una versión mejorada de HFS, soportando archivos mucho más grandes (Bloques direccionables de 32 bits en vez de 16) y usando Unicode (En vez de Mac OS Roman) para el nombre de los archivos, lo que además permitió nombres de archivo de hasta 255 letras. HFS Plus permite nombres de fichero de hasta 255 caracteres de longitud UTF-16, y archivos n-bifurcados similares a NTFS, aunque casi ningún software se aprovecha de bifurcaciones con excepción de la bifurcación de los datos y de la bifurcación del recurso. HFS Plus también utiliza tabla de asignación de 32 bits, en lugar de los 16 bits de HFS. Ésta era una limitación seria de HFS, significando que ningún disco podría apoyar más de 65.536 bloques de la asignación sobre de HFS.

**Historia:** OS X fue lanzada en 1999. Represento un cambio radical con respecto a su antecesor (OS 9). Integro tecnologías desarrolladas por la empresa NeXT (entonces comprada por Apple). Fue un cambio total en la implementación del SO. Gradualmente adopto x86 como arquitectura, dejando de dar soporte a PowerPC. OS X es una evolución de OPENSET, el SO desarrollado por la empresa NeXT, basado en Darwin, Objective-C y multiusuario.

**File System ZFS**

ZFS es un sistema de archivos desarrollado por Sun Microsystems para su sistema operativo Solaris. El significado original era *‘Zettabyte File System’*, pero ahora es un acrónimo recursivo. ZFS es una nueva clase de sistema de ficheros que proporciona una administración simple, semántica transaccional, integridad de los datos, y una inmensa escalabilidad. ZFS no es una mejora incremental de la tecnología existente; es fundamentalmente un nuevo acercamiento a la administración/gestión de datos. ZFS presenta un modelo reunido de almacenamiento que elimina totalmente el concepto de volúmenes y los problemas asociados de particiones, del aprovisionamiento, del ancho de banda perdido y del almacenamiento inmovilizado. Los millares de sistemas de ficheros pueden escribirse en un pool de almacenamiento común, cada uno consumiendo solamente tanto espacio como realmente necesite.

ZFS proporciona snapshots ilimitados y clonaciones. Un snapshot es una copia inalterable de un punto en el tiempo de un sistema de ficheros, mientras que una copia es una copia

escribible de un snapshot. Las clonaciones proporcionan una manera extremadamente eficiente de almacenar muchas copias de datos sobre volúmenes compartidos, tales como espacios de trabajo (workspaces), instalaciones de software, y clientes diskless (sin discos). Las copias de seguridad (backups) y el restaurado (restore) ZFS son realizados a través

de snapshots. Cualquier snapshot puede generar un backup completo, y cualquier par de snapshot puede generar un backup incremental. Los backups incrementales son tan eficientes que pueden ser utilizados para los replicados remotos. No hay límites arbitrarios en ZFS. Puedes tener tantos ficheros como desees; offsetsde 64-bit del fichero; enlaces/links ilimitados, entradas de directorio, snapshots, etcétera.

**Historia:** BSD significa Berkeley Software Distribution. Es un SO derivado en

Unix creado por la universidad de Berkeley en 1977. En 1970 ATyT distribuye

el código fuente Unix a las universidades con fines educativos. En 1974, se

crea el PDP-11 basado en el Unix de ATyT. La primer versión fue el 1BSD en

1997, en 1978 nace 2BSD y recién en 1983 se lanza la 2.9BSD que se

considera un SO completo. En 1991 se intenta hacer un port de BSD a la

arquitectura 386 llamada BSD/386, base de las versiones libres. En el 1993 se

lanza la versión 4.2BSD y se anuncia la mascota. En el año 1994 se lanza

4.4BSD que fue la primera distribución sin nada de código ATyT. En el año

1995 se cesa el desarrollo de BSD por parte de la universidad de Berkeley.

Esto dio lugar a diversas distribuciones del SO.

**3. ¿Qué es el particionado? ¿Qué es el UUID? ¿Para qué se lo utiliza? (Hint: ver el comando blkid)**

Particiones son subdivisiones de un disco entero. El SO las presenta como un dispositivo de bloques (como si fuera el disco entero).

Están definidas en un área especial del disco llamado “partition table”. Cada partición contiene un “file system” específico o es una “raw partition”. Existen varios tipos de tablas de particiones. Dos más utilizadas:

* **Master Boot Record** (MBR)
* **Globally Unified Identifier Partition Table** (GPT)

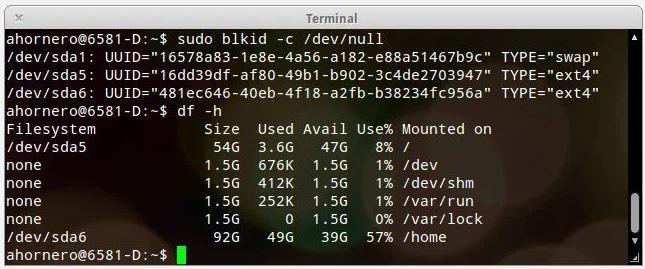
Como tal, el **UUID es** un código identificador estándar empleado en el proceso de construcción de software.

Un **identificador único universal** o *universally unique identifier* (**UUID**) es un número de 16 [bytes](https://es.wikipedia.org/wiki/Byte) (128 [bits](https://es.wikipedia.org/wiki/Bit)). Por tanto, el número de posibles UUID es de 1632, o unos 3,4 × 1038. En su [forma canónica](https://es.wikipedia.org/wiki/Forma_can%C3%B3nica), un UUID se expresa mediante 32 dígitos [hexadecimales](https://es.wikipedia.org/wiki/Hexadecimales), divididos en cinco grupos separados por guiones de la forma 8-4-4-4-12, lo que da un total de 36 caracteres (32 dígitos y 4 guiones). Por ejemplo:

**550e8400-e29b-41d4-a716-446655440000**

Un UUID puede ser usado también con un identificador específico "intencionalmente" y repetidamente usado para identificar la misma cosa en diferentes contextos

El comando BLKID sirve para poder obtener tanto el tipo de partición como el UUID asociada a cada identificador del nodo.



**4. ¿Es necesario tener un file system para acceder a una partición?**

Sí, es necesario para poder acceder y administrar la información guardada en dicha partición.

**5. ¿Qué es el área de swap en Linux? ¿Existe un área similar en Windows?**

Linux divide su memoria física RAM (memoria de acceso aleatorio) en capas de memoria llamadas páginas. El swapping es el proceso por el que una página de memoria se copia en un espacio del disco configurado previamente para ello, llamado espacio de swap (o de intercambio), para liberar esa memoria RAM. Los tamaños combinados de la memoria física y del espacio swap determinan la cantidad de memoria virtual disponible.

El espacio swap o de intercambio será normalmente una partición del disco, pero también puede ser un archivo.

[Microsoft Windows](https://es.wikipedia.org/wiki/Microsoft_Windows) usa un fichero de intercambio desde su versión 3.1 ([1992](https://es.wikipedia.org/wiki/1992)), la primera en usar [memoria virtual](https://es.wikipedia.org/wiki/Memoria_virtual). Lo implementa mediante un fichero situado en el directorio raíz (C:\) o en el de sistema (C:\WINDOWS\), y tiene por nombre:

* *386SPART.PAR* en Windows 3.1
* *WIN386.SWP* en Windows 3.11, 95 y 98
* *pagefile.sys* en Windows NT y sucesores

Este fichero tiene un tamaño variable (depende de la configuración) y no debe ser movido o borrado.

**6. ¿Qué función cumple el directorio lost+found en Linux?**

[*http://unix.stackexchange.com/questions/18154/what-is-the-purpose-of-the-lostfound-folder-in-linux-and-unix*](http://unix.stackexchange.com/questions/18154/what-is-the-purpose-of-the-lostfound-folder-in-linux-and-unix)

*Si ejecuta fsck, la comprobación del sistema de archivos y el comando de reparación, podría encontrar fragmentos de datos que no están referenciados en cualquier lugar del sistema de archivos. En particular, fsck podría encontrar datos que se parece a un archivo completo, pero no tiene un nombre en el sistema - un inodo con ningún nombre de archivo correspondiente. Estos datos se sigue utilizando hasta el espacio, pero no es accesible por medios normales.*

*Si usted le dice fsck para reparar el sistema de archivos, que a su vez, estos archivos borrados casi nuevo en los archivos. La cosa es que el archivo tenía un nombre y ubicación una vez, pero que la información ya no está disponible. Así depósitos fsck el archivo en un directorio específico, llamado lost + found (después de la pérdida de propiedad y encontrado).*

*Los archivos que aparecen en lost + found son típicamente archivos que ya estaban disociados (es decir, su nombre había sido borrado), pero aún abiertos por algún proceso (por lo que los datos no se borran aún) cuando el sistema se detuvo súbitamente (kernel panic o fallo de alimentación) . Si eso es todo lo que sucedió, estos archivos se tenía planeado para la eliminación de todas formas, no es necesario que se preocupan por ellos.*

*Los archivos también pueden aparecer en lost + found porque el sistema de archivos se encontraba en un estado inconsistente debido a un error de software o hardware. Si ese es el caso, es una manera para que pueda encontrar los archivos que se han perdido, pero que la reparación del sistema administrado de salvar. Los archivos pueden o no contienen datos útiles, e incluso si lo hacen, pueden ser incompletos o fuera de fecha; todo depende de qué tan grave fue el daño del sistema de archivos.*

*En muchos sistemas de ficheros, los perdidos + found directorio es un poco especial porque preasigna un poco de espacio para fsck para depositar los archivos allí. (El espacio no es para los datos del archivo, que fsck hojas en su lugar;. Que es para las entradas de directorio, que fsck tiene que recuperar) Si por accidente borra lost + found, no vuelva a crear con mkdir, el uso mklost + found si está disponible.*

**7. En Linux, ¿dónde se almacena el nombre y los metadatos de los archivos?**

**data**

Los datos son el contenido del archivo, en este caso, los 16 bytes que componen la lista del mercado de elvis (13 caracteres visibles y 3 invisibles de "retorno" que indican el final de la línea). En Linux como en Unix, cada contenido de archivo se almacena como una serie de bytes.

**metadatos**

Además de su contenido, en Linux cada archivo tiene información adicional asociada a éste. Todo el cuaderno anterior estuvo enfocado en dicha The Linux Filesystem 2 información, es decir, el propietario de archivo, el grupo propietario y los permisos. Información, tal como la última vez que el archivo fue modificado o leído, también se almacena. Muchos de estos metadatos son reportados al ejecutar el comando ls -l. En Linux (Unix), toda la información adicional asociada al archivo (con la excepción importante que pronto veremos) se almacena en una estructura llamada inodo.

**filename**

El nombre de archivo es la excepción a la regla. Aunque el nombre del archivo podría considerarse como metadatos asociados con el archivo, éste no se almacena directamente en el inodo. En su lugar, el archivo se almacena en una estructura llamada dentry, (el término dentry es un acortamiento para directory entry, y como veremos más adelante, la estructura está asociada a los directorios). En esencia, el nombre de archivo asocia un nombre con un inodo.   
------o------  
Los Inodos contienen metadata de los archivos y punteros a los bloques de datos. El nombre del archivo **no se almacena en el inodo**,el filesystem implementa directorios como un tipo especial de archivo cuyo bloques de datos almacenan el nombre del archivo y su inodo correspondiente (“dentry”).

**8. Seleccione uno de sus file systems (una partición) y conteste usando el comando dumpe2fs:  
¿Qué información describe el comando dumpe2fs?**

dumpe2fs muestra la información de super bloque y grupo de bloque, para el del sistema de archivos presente en el dispositivo.

**¿Cuál es el tamaño de bloque del file system?**

**¿Cuántos inodos en total contiene el file system?**  
**¿Cuántos archivos como máximo se pueden crear con el estado actual del file system?**

**¿Cuántos grupos de bloques existen?**

**EJEMPLO USANDO UNA PARTICION EXT4**

guillox182@buntu:~$ sudo dumpe2fs -h /dev/sda7

dumpe2fs 1.42.13 (17-May-2015)

Filesystem volume name: <none>

Last mounted on: /

Filesystem UUID: 9de003ef-22f5-4c84-8b75-4e6d38c1a0ce

Filesystem magic number: 0xEF53

Filesystem revision #: 1 (dynamic)

Filesystem features: has\_journal ext\_attr resize\_inode dir\_index filetype needs\_recovery extent flex\_bg sparse\_super large\_file huge\_file uninit\_bg dir\_nlink extra\_isize

Filesystem flags: signed\_directory\_hash

Default mount options: user\_xattr acl

Filesystem state: clean

Errors behavior: Continue

Filesystem OS type: Linux

Inode count: 2379216

Block count: 9511424

Reserved block count: 475571

Free blocks: 4942672

Free inodes: 1858728

First block: 0

Block size: 4096

Fragment size: 4096

Reserved GDT blocks: 1021

Blocks per group: 32768

Fragments per group: 32768

Inodes per group: 8176

Inode blocks per group: 511

Flex block group size: 16

Filesystem created: Thu Dec 3 13:37:54 2015

Last mount time: Fri Jun 24 13:20:28 2016

Last write time: Fri Jun 24 13:20:21 2016

Mount count: 154

Maximum mount count: -1

Last checked: Thu Dec 3 13:37:54 2015

Check interval: 0 (<none>)

Lifetime writes: 676 GB

Reserved blocks uid: 0 (user root)

Reserved blocks gid: 0 (group root)

First inode: 11

Inode size: 256

Required extra isize: 28

Desired extra isize: 28

Journal inode: 8

First orphan inode: 136805

Default directory hash: half\_md4

Directory Hash Seed: e3ef6cdc-0d6e-4fa2-a99e-8c7a67362cdb

Journal backup: inode blocks

Características del fichero de transacciones: journal\_incompat\_revoke

Tamaño del fichero de transacciones: 128M

Longitud del fichero de transacciones: 32768

Secuencia del fichero de transacciones: 0x000d2198

Inicio del fichero de transacciones: 11014

**¿Cómo haría para incrementar la cantidad de inodos de un file system?**

**9. ¿Qué es el file system procfs? ¿Y el sysfs?**

* ProcFS es un pseudo-filesystem montado comúnmente en el directorio /proc.
* Provee una interface a las estructuras de datos del kernel
* Presenta información sobre procesos y otra información del sistema en una estructura jerárquica de “files”
* No existe en disco, el kernel lo create en memoria (generalmente tamaño 0 de los “files”)
* Mayor´ıa de los “files” de solo lectura, aunque algunos pueden ser modificados (/proc/sys)
* echo 1 > /proc/sys/net/ipv4/ip forward (o con el comando sysctl)
* /proc/pid: dir. con información del proceso “pid”
* /proc/filesystems: lista los FS soportados por el kernel
* /proc/meminfo: información del uso de memoria f´ısica y swap

**SysFS**

* Con el paso del tiempo, /proc se convirtió en un verdadero desorden
* En Linux 2.6 se implementó un nuevo sistema de archivos virtual llamado “Sysfs”
* SysFS exporta información sobre los dispositivos de hardware y sus controladores desde el kernel hacia el espacio del usuario
* También permite la configuraci´on de par´ametros
* SysFS se monta en /sys

13. Los permisos por defecto de Linux al crear un archivo o directorio son 666 y 777 respectivamente.

Cree un nuevo archivo y analice sus permisos. ¿Es así? ¿Por qué sucede esto?

**Al crear un nuevo archivo con mi usuario no pasa esto, los permisos para archivos y directorios creados por mi usuario por defecto son 644 y 755 respectivamente, asumo que esto es por cuestiones de seguridad para que nadie salvo yo pueda escribir mis archivos.**

**14. ¿Qué es un link simbólico? ¿En qué se diferencia de un hard-link?**

**Link simbólico**: El fichero o directorio se encuentra en un único punto del disco y los enlaces son un puntero contra él. Cada enlace simbólico tiene su propio número de inodo lo que permite hacer enlaces simbólicos entre distintos sistemas de ficheros.

**Los enlaces duros(hard-link)** lo que hacen es asociar dos o más ficheros compartiendo el mismo inodo. Esto hace que cada enlace duro es una copia exacta del resto de ficheros asociados, tanto de datos como de permisos, propietario, etc. Esto implica también que cuando se realicen cambios en uno de los enlaces o en el fichero este también se realizará en el resto de enlaces.

Los enlaces duros no pueden hacerse contra directorios y tampoco fuera del propio sistema de ficheros.

Los enlaces simbólicos se pueden hacer con ficheros y directorios mientras que los duros solo entre ficheros.

Los enlaces simbólicos se pueden hacer entre distintos sistemas de ficheros, los duros no.

Los enlaces duros comparten el número de inodo, los simbólicos no.

En los enlaces simbólicos si se borra el fichero o directorio original, la información se pierde, en los duros no.

Los enlaces duros son copias exactas del fichero mientras que los simbólicos son meros punteros o “accesos directos”.

16. ¿Para qué sirven los permisos especiales en Linux? Analizar el Sticky-bit, SUID y SGID

**Hay una serie de permisos especiales sobre el sistema de archivos de Linux que pueden resultarnos útiles para determinadas tareas o para organizar directorios colaborativos entre diferentes usuarios.**

**Sticky bit**

**El Sticky bit se utiliza para permitir que cualquiera pueda escribir y modificar sobre un archivo o directorio, pero que solo su propietario o root pueda eliminarlo. Un ejemplo de uso es el directorio /tmp, que debe tener permisos para ser utilizado por cualquier proceso, pero solo el dueño o root puede eliminar los archivos que crea.**

**Al directorio con el Sticky Bit aplicado se le agrega la t al final del descriptor de permisos.**

**SUID**

**El bit SUID activo en un archivo significa que el que lo ejecute va a tener los mismos permisos que el que creó el archivo. Esto puede llegar a ser muy util en algunas situaciones pero hay que utilizarlo con cuidado, dado que puede generar grandes problemas de seguridad.**

**Para que sea efectivo el archivo debe tener permisos de ejecución.**

**Flag s en los permisos de ejecución del usuario**

**SGID**

**El SGID es lo mismo que en el SUID, pero a nivel de grupo. Es decir, todo archivo que tenga activo el SGID, al ser ejecutado, tendrá los privilegios del grupo al que pertenece.**

**Esto es muy usado cuando queremos configurar un directorio colaborativo: si aplicamos este bit al directorio, cualquier archivo creado en dicho directorio, tendrá asignado el grupo al que pertenece el directorio.**

**Flag s en los permisos de ejecución del grupo**

17. ¿Cuáles son los permisos del archivo /etc/shadow? ¿Por qué puedo modificar mi password sino

soy usuario root?

**Los permisos para /etc/shadow son de lectura y escritura para root, de lectura para el grupo de root y nada para todos los demás.**

**Porque el archivo /usr/passwd tiene el flag suid activado**

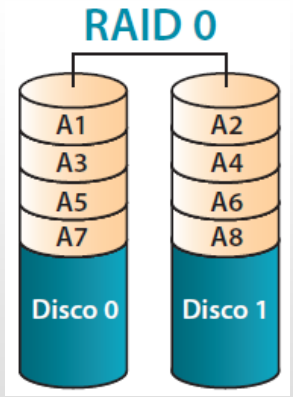
18. Crear un archivo en el directorio /tmp. Si abre otra consola y se loguea con un usuario distinto,

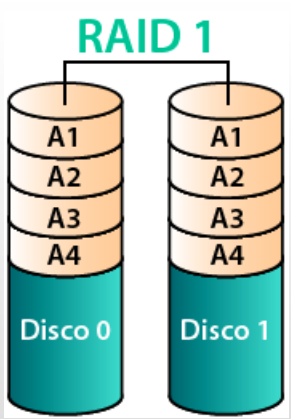
¿puede borrar ese archivo? ¿Por qué? (Hint.: ver permisos especiales en Linux)

**No lo hice porque ni ganas de crear usuario nuevo, pero cree el archivo y los permisos se ven normales. Al parecer no se puede borrar: el directorio /tmp tener permisos para ser utilizado por cualquier proceso, pero solo el dueño o root puede eliminar los archivos que crea.**

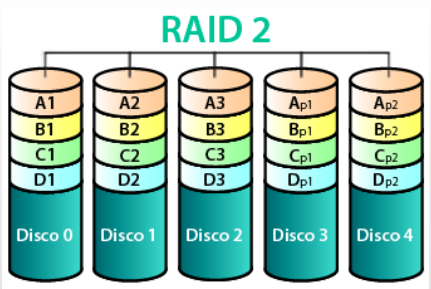
**RAID  
  
1. ¿Qué es un RAID? Explique las diferencias entre los distintos niveles de RAID**RAID - Redundant Array of Independent Disks (originalmente llamado Redundant Array of Inexpensive Disks).  
RAID es una técnica que permite usar múltiples discos en forma conjunta con el fin de construir un sistema de discos más rápido, más grande y más confiable.

**Level 0(Striping):** No es un nivel de RAID en absoluto. No existe redundancia. Array de discos con “striping” a nivel de bloque. Necesita de 2 o más discos para conformarse. La capacidad es la sumatoria de la capacidad de los discos participantes. Si falla un disco, los datos de todos los discos se vuelven inaccesibles. Sirve como “upper-bound” en cuanto a performance y capacidad. Disk Striping: proceso de dividir datos en bloques y esparcirlos en múltiples dispositivos de almacenamiento.

  
  
**Level 1 (Mirroring):**Asegura la redundancia mediante el mirroring (espejado) de datos. No hay striping de datos. Almacena datos duplicados en discos separados o independientes. Se requiere un mínimo de 2 discos. Trabaja con discos de a pares. Ineficiente por la escritura en espejos que provoca un desperdicio del 50 % de la capacidad total. Desperdicio del 50 % de la capacidad total.

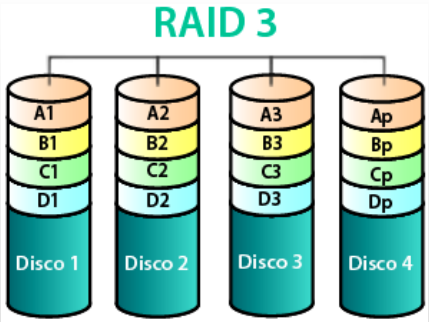


**Level 2 :**Striping a nivel de bit y paridad Hamming dedicada. Se requieren al menos 3 discos como mínimo. Buena protección de datos en caso de fallas de disco. Tasa de transferencia de datos puede llegar a ser muy elevada. Disco dedicados para almacenar información de verificación y corrección de errores. No muy aceptado por la industria.

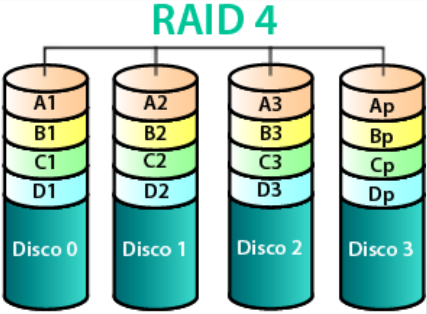
  
  
**Level 3:**

Striping a nivel de byte y paridad dedicada. Disco “dedicado” para almacenar la información de paridad. Datos se escriben en paralelo entre los disco del array. 3 discos como mínimo.

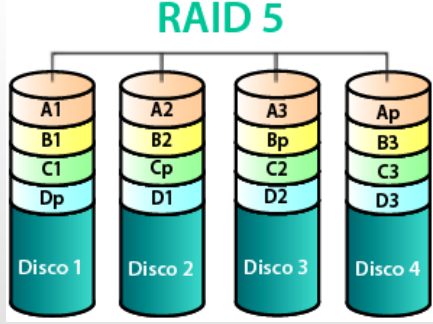
Alta tasa de transferencia en escritura y lectura. El disco de paridad puede convertirse en cuello de botella. No ofrece solución a fallo simultáneo de discos.

  
**Level 4:**

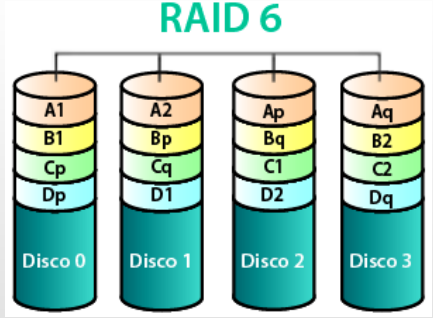
Striping a nivel de bloque y paridad dedicada. 3 disco requeridos como mı́nimo. Alta tasa de transferencia de datos. No muchas implementaciones disponibles.

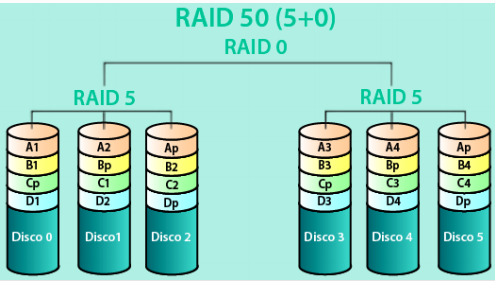


**Level 5:**Striping a nivel de bloque y paridad distribuida. Es una de las implementaciones más utilizadas. Distribuye la información de paridad entre todos los discos del array. Se necesitan 3 discos como mínimo. Alto rendimiento. No hay cuello de botella. No ofrece solución al fallo simultáneo de discos.

  
  
**Level 6:**

Striping a nivel de bloque y doble paridad distribuida. Recomendado cuando se tienen varios discos. Requiere 4 discos mínimo. Alta tolerancia a fallos (hasta dos discos).Operaciones de escrituras más lentas debido al cálculo de la doble paridad.



Además de estos niveles de RAID es posible utilizar niveles hibridos los cuales consisten en la combinación entre los ya nombrados.  
Como por ejemplo: **  
  
2. Utilizar el comando parted -l (debe ejecutarlo con sudo) para ver la tabla de particiones.**

**Conteste:**

**a) ¿Cómo llama Linux al dispositivo físico? ¿Cuál es su tamaño total?**

**b) ¿Cuántas particiones existen? ¿Qué tipo de file-system contiene cada una?**

**c) ¿Qué significa la bandera arranque?**

**3. Usando el comando parted crear una nueva partición de tipo extendida (debe seleccionar el dispositivo donde se van a generar las particiones):**

**1) sudo parted**

**2) (parted) print**

**3) (parted) mkpart**

**4) Tipo de partición: extendida**

**5) Inicio: #MB (igual al tamaño final de la partición existente más alta)**

**6) Fin: #MB (igual al tamaño máximo del dispositivo físico)**

**Dentro de esta partición extendida crear 3 nuevas particiones de 300MB cada una. Para esto utilizar nuevamente el comando mkpart, pero debe seleccionar logical como tipo de partición. Como tipo de sistema de archivos elija ext3.**

**Obs.: como las particiones lógicas se crean dentro de una partición extendida, los valores de inicio/fin de cada partición lógica deben estar dentro de los valores seleccionados en la partición extendida.**

**4. ¿Por qué es necesario crear una partición extendida? Si se usase GPT, ¿sería necesario este tipo de particiones?**En ciertos casos la cantidad de particiones primarias (4) es insuficiente al utilizar **MBR**. Entonces se crea una partición extendida la cual puede contener varias particiones lógicas. El disco quedaría con 3 primarias y una extendida (No puede haber más extendidas). Dentro de esta partición extendida puede haber un gran número de particiones lógicas. Y de este modo romper la limitación de solo 4 primarias.  
En **GPT** no son necesarias las particiones lógicas ya que se pueden realizar 128 particiones por defecto. Otra ventaja de gpt es que soporta discos mucho más grandes qué MBR. Además cuenta con un backup de la tabla al final del disco y CRC para proteger el contenido de toda la GPT (Mayor seguridad)

**5. Mirar nuevamente la tabla de particiones para ver las nuevas particiones y salga del entorno parted.**

**6. Utilizar el comando mdadm para crear un RAID 5 utilizando las 3 particiones lógicas que se generaron en el punto anterior** (fdisk -l para ver el nombre de las particiones que generaron):

mdadm --create --verbose /dev/md0 --level=5 --raid-devices=3 /dev/sda5  
/dev/sda6 /dev/sda7

(Obs.: md0 es el nombre que le dará al nuevo RAID)

**7. ¿Qué significan los valores sda5, sda6 y sda7?**

Hace referencia a 3 particiones lógicas dentro de lo que sería una 4° partición extendida. sda hace referencia a un primer disco, un segundo disco sería sdb.  
[Explicación más extendida y fácil.](http://www.enmimaquinafunciona.com/pregunta/4682/diferencias-entredevsda-ydevsda1)

**8. Ejecutar la siguiente consulta y contestar:**

**mdadm --detail /dev/md0**

**a) ¿Cuál es el tamaño total del RAID?**

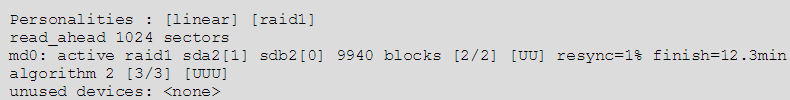
**b) ¿Cuál es el tamaño total utilizable para almacenar datos?**

**9. Analizar el contenido del siguiente comando:**

**cat /proc/mdstat**

Obs.: puede suceder que al reiniciar la VM el RAID se vea como de solo lectura y con el número  
127. Para solucionar esto deben ejecutar los comandos *mdadm –stop /dev/md127* para parar  
el RAID y *mdadm –assemble –scan* para volverlo a generar como md0 y de lectura/escritura.  
Esto se debe hacer cada vez que se inicia la VM. Si quiere que quede en forma persistente  
a través de los reboots debe guardar la configuración en el archivo mdadm.conf, *mdadm –  
assemble –scan >> /etc/mdadm/mdadm.conf* y luego *update-initramfs -u* (esto último puede  
tardar un poco de tiempo)

Este archivo contiene la información actual sobre la configuración de discos múltiples de RAID. Si su sistema no contiene dicha configuración, el archivo /proc/mdstat será parecido a:

  
Este archivo se mantiene en el mismo estado que el mostrado arriba a menos que un software RAID o dispositivo md esté presente. En ese caso, visualice /proc/mdstat para ver el estado actual de los dispositivos RAID md*X*.  
El archivo /proc/mdstat a continuación, muestra un sistema con su md0 configurado como un dispositivo RAID 1, mientras está resincronizando los discos:  


**10. Ahora se va a probar la funcionalidad del RAID 5. Para esto completar los siguientes pasos:**

**a) Crear un file system de tipo ext3 en el RAID 5 recién generado**

**mkfs.ext3 /dev/md0**

**b) Montar la partición con el file system generado en el directorio /mnt/rd5**

**c) Crear un directorio con dos archivos**

**d ) Quitar una de las particiones del RAID. Para esto ponemos uno de los componentes en falla: mdadm --fail /dev/md0 /dev/sda7**

**e) Observar el estado del RAID y contestar**

**1) ¿Cuál es el estado del RAID? ¿Cuántos dispositivos activos existen?**

**2) ¿Qué sucedería si se ejecuta el comando anterior sobre una de las particiones**

**restantes?**

**f ) Quitar del RAID el componente puesto en falla en el paso anterior**

**mdadm -r /dev/md0 /dev/sda7**

**g ) Observar nuevamente el estado del RAID y contestar:**

**1) ¿Se puede acceder al directorio /mnt/rd5? ¿Están los archivos creados anteriormente?**

**2) ¿Qué hubiese sucedido si teníamos otra partición como “hot-spare”?**

hot-spare es un disco de reserva es un [disco duro](https://es.wikipedia.org/wiki/Disco_duro) que se puede utilizar en los sistemas [RAID](https://es.wikipedia.org/wiki/RAID)  
tanto [hardware](https://es.wikipedia.org/wiki/Hardware) como [software](https://es.wikipedia.org/wiki/Software). Se trata de un dispositivo físicamente instalado en el sistema RAID que se mantiene inactivo hasta que uno de los discos activos falla. En ese momento el disco de reserva comienza a funcionar reemplazando el disco erróneo reconstruyendo el [conjunto de discos](https://es.wikipedia.org/w/index.php?title=Conjunto_de_discos&action=edit&redlink=1). Este proceso reduce el tiempo medio de recuperación ([MTBF](https://es.wikipedia.org/wiki/MTBF)) del sistema aunque no elimina por completo el fallo. Los subsecuentes fallos que ocurran antes de que se reconstruya el sistema RAID pueden representar una pérdida de datos. La reconstrucción del sistema puede llevar varias horas, sobre todo en sistemas muy cargados. La rápida sustitución de los discos erróneos es de vital importancia puesto que los discos de un conjunto RAID suelen tener la misma carga de uso.[RAID 6](https://es.wikipedia.org/wiki/RAID#RAID_6) sin discos de reserva usa el mismo número de discos que un [RAID 5](https://es.wikipedia.org/wiki/RAID#RAID_5) con un disco de reserva y protege los datos contra dos fallos simultáneos, pero requiere un controlador RAID más sofisticado. Además, un disco de reserva se puede utilizar simultáneamente por múltiples conjuntos RAID.

**13. Usando el comando parted generar una nueva partición, /dev/sda8**[Todo sobre Parted](http://rm-rf.es/gestion-de-particiones-con-parted-en-linux/)  
  
  
  
**LVM (Logical Volumen Management)**

**1. ¿Qué es LVM? ¿Qué ventajas presenta sobre el particionado tradicional de Linux?**

Logical Volume Management (LVM) provee un método más flexible que los convencionales esquemas de particionamiento para alocar espacio en los dispositivos de almacenamiento masivo. Permite una administración “on-line” de los discos de almacenamiento agregando una capa adicional al subsistema de entrada del kernel

**2. ¿Cómo funcionan los “snapshots” en LVM?**

Snapshots: backup on-line de las particiones