Sistemas Operativos 2

Unidad 4: Sistemas de archivos

Organización de archivos y directorios

René Ornelis
Primer semestre de 2025

Contenido

Índice de figuras

Organización de archivos y directorios

""

1 Introducción

Los sistemas de archivos son la forma en que se organiza la información persistente en los dispositivos de almacenamiento, en forma de archivos y directorios, sería demasiado complicado que el usuario final organice su información en forma de cilindros cabezas y sectores.

El sistema de archivos virtual es el API que el sistema operativo proporciona al usuario para que el mismo administre su información con *Independencia del medio de almacenamiento.*

> **Independencia del medio de almacenamiento:** acceder todos los archivos y directorios de la misma forma independientemente del medio de almacenamiento (disco u otro dispositivo).

Para cada sistema de archivo debemos comprender:

- La forma en que se estructura la información en el disco duro, si este se ve como una secuencia de bloques.
- Cómo se almacena y administran los archivos dentro de un directorio.
- La forma en que se estructura la información de un archivo en el sistema: atributos, seguridad y forma en que se controla el conjunto de bloques de datos que componen el archivo.
- Deducción de la fórmula del máximo tamaño de archivo.

2 FAT / FAT32

El sistema de archivos FAT (File Allocation Table) es uno de los sistemas de archivos más antiguos y simples, y ha sido utilizado en diversas versiones como FAT12, FAT16, FAT32 y exFAT. La estructura básica de un sistema de archivos FAT incluye los siguientes componentes principales (ver Figura 1):

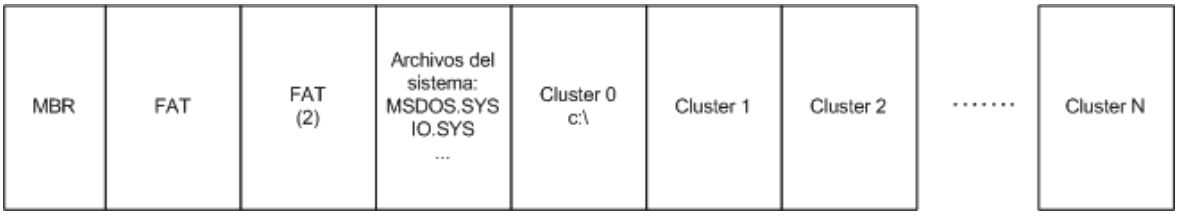

Figura 1: Organización por bloques de FAT

1. **Registro de Arranque (Boot Record)**: Es el primer sector del volumen (sector 0) y contiene información vital sobre el sistema de archivos, como el tamaño del sector, el tamaño de la unidad de asignación (cluster), el número de sectores en el volumen, el número de copias de la FAT, la ubicación de la FAT, y otros parámetros importantes.
2. **Tabla de Asignación de Archivos (FAT)**: Se encuentra después del Registro de Arranque y es una tabla que lleva el registro de la ubicación de los clusters de datos. Cada entrada en la FAT corresponde a un cluster y puede contener información sobre el estado del cluster (si está libre, reservado, o si es parte de un archivo y apunta al siguiente cluster del archivo).
3. **Copia de la FAT:** Luego de la FAT, se encuentra una copia de la FAT, la cual se mantiene como imagen y es utilizada en caso que se dañen los sectores del disco que contienen a la FAT.
4. **Archivos del sistema:** En las primeras versiones de FAT, se tenía el primer grupo de clústeres, luego de las tablas FAT, como un espacio especial para los archivos del sistema operativo DOS. Esto se eliminó en versiones posteriores y dichos archivos se integraron en el directorio raíz.
5. **Directorio Raíz (Root Directory)**: Inmediatamente después de las entradas de la FAT en se encuentra el primer cluster que corresponde al directorio raíz, el cual contiene las entradas de los archivos y subdirectorios del directorio raíz. Cada entrada incluye información como el nombre del archivo, la extensión, atributos del archivo (lectura/escritura, oculto, etc.), la fecha y hora de creación, el primer cluster de datos del archivo y el tamaño del archivo.
6. **Espacio de Datos (Data Area)**: Después del directorio raíz, se tiene .el espacio de datos para clusters que se utilizan para los archivos del sistema.

2.1 Estructura de la FAT

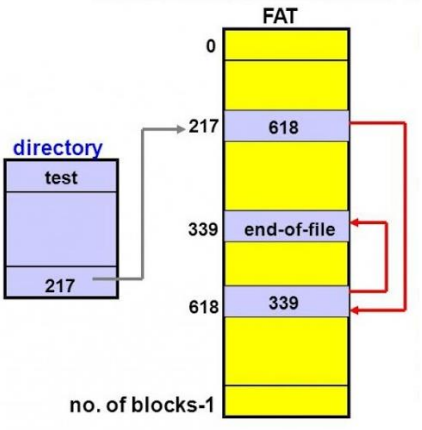

Figura 2: Estructura de la FAT

2.2 Directorios de archivos

Entre los campos almacenados en cada entrada del directorio están:

Nombre	8 caracteres
Extensión	3 caracteres
Atributos	1 byte, con los bits de oculto, sólo lectura, etc
Reservado	1 byte
Hora de creación	3 bytes
Fecha de creación	2 bytes (1980-2107)
Fecha de último acceso	2 bytes
Cluster inicial	Número de cluster

2.3 FAT12

La primera versión de FAT, conocida como FAT12, fue la primera implementación estandarizada del sistema de archivos FAT, introducida en 1980. Esta incluyó las siguientes características.:

- **Número de bits en los apuntadores a clústeres:** utiliza 12 bits para identificar los clústeres. Esto permite un máximo de 2^{12} (4096) posibles valores.
- **Número máximo de clusters :** Permite un máximo de 2^{12} (4096) clusters, y con un tamaño de cluster estándar de 512 bytes, el tamaño máximo del volumen es de 32 MB. FAT12 era más adecuado para discos más pequeños y sistemas de almacenamiento de menor capacidad.

2.4 FAT16

- Aumenta las direcciones de los clusters a 16 bits
- Sigue limitado a 32Mb, pero permite disminuir el tamaño de los clusters

2.5 vFAT

Es una extensión del sistema de archivos FAT16, introducido con Windows 95 para mejorar la gestión de archivos y permitir nombres de archivo largos. VFAT agrega soporte para nombres de archivos largos (más de 8 caracteres) a la estructura FAT16.

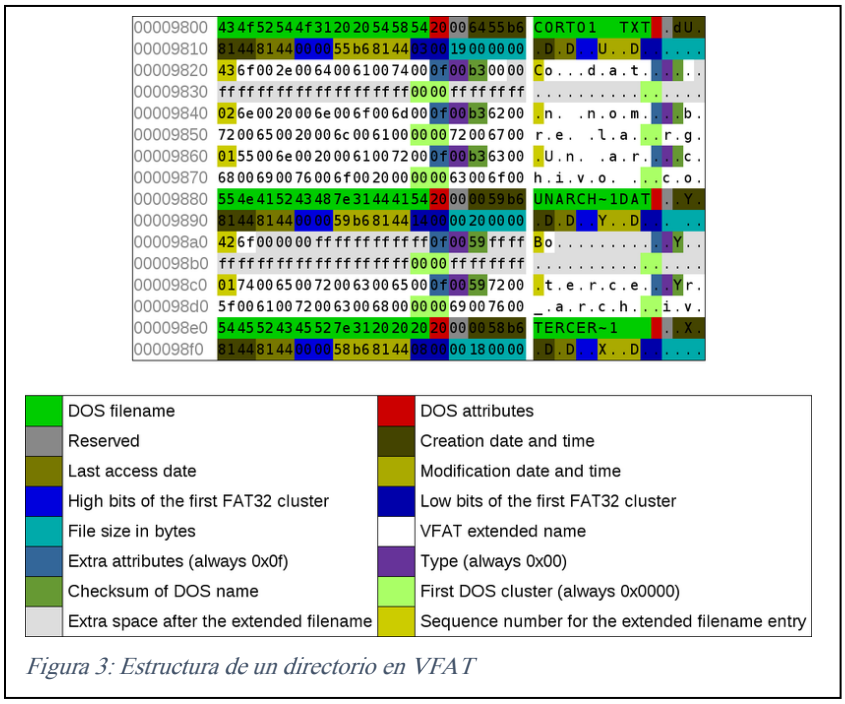

Figura 3: Estructura de un directorio en VFAT

- Nombres largos de archivos
  - Entradas adicionales para almacenar más de 8.3
  - Una entrada, la primera, en 8.3, posiblemente ajusta con secuencia (NOMBRE~1.TXT)
  - Se utilizan otras entradas del directorio y cada, se escribe como nombre todos los campos de la entrada, excepto:
    - El primer caracter del nombre: número de secuencia (interpretado como borrado)
    - atributos: Volume Label, System, Hidden, and Read Only
    - Cluster inicial: 0
  - Se agrega un CRC, para detectar si programas viejos corrompieron la entrada

2.6 FAT32

- Utiliza apuntadores a clústeres de 32 bits.

2.7 Máximo tamaño de archivo

Tam = max (2^{bits}, Número de clusters) * tamaño_cluster

Donde bits = 12, 16 o 32

3 EXT2

El sistema de archivos **ext2fs** (Second Extended File System) es uno de los primeros sistemas de archivos desarrollados para Linux. Fue diseñado para superar las limitaciones del sistema ext (Extended File System) original de Minix y proporcionar mayor estabilidad y

eficiencia. Aunque ha sido reemplazado en la mayoría de los casos por ext3, ext4 y otros sistemas más avanzados, ext2fs sigue siendo relevante en ciertos entornos, especialmente para dispositivos con memoria flash, debido a su simplicidad y falta de journaling (lo que reduce el desgaste en estas unidades).

3.1 Estructura básica

- Organiza los datos en bloques, grupos de bloques, inodos y superbloques.
- Utiliza una tabla de inodos para rastrear archivos y directorios.

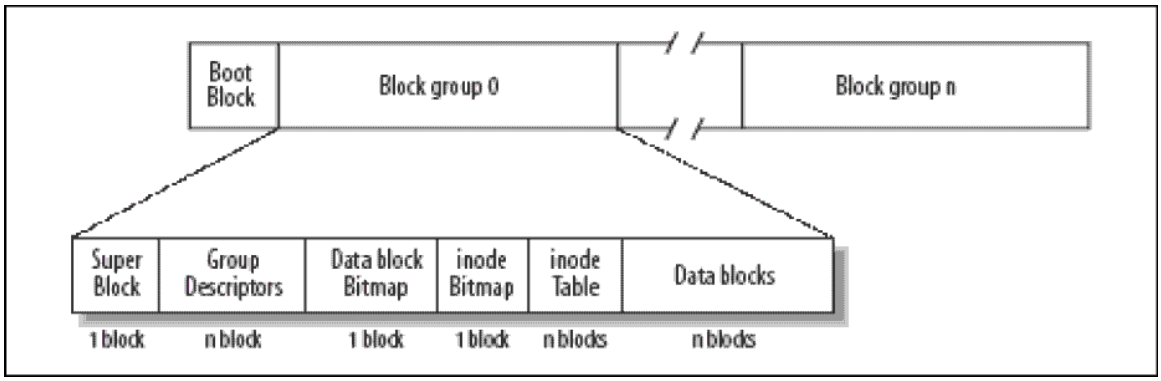

Figura 4: Estructura por bloques de Ext2

- Divide el disco en **grupos de bloques**, lo que mejora la localización de datos y reduce la fragmentación.
- Usa una política de asignación que minimiza la fragmentación al colocar los datos de un archivo cerca de sus metadatos.
- Soporte para permisos de lectura, escritura y ejecución a nivel de usuario, grupo y otros.
- Es compatible con enlaces simbólicos y enlaces duros.
- Es ampliamente compatible con herramientas y sistemas Linux más antiguos y puede ser montado en otros sistemas operativos con el software adecuado.

3.2 Estructura del superbloque

s_inodes_count	_u32	Número total de inodos del sistema de ficheros.
s_blocks_count	_u32	Número total de bloques del sistema de ficheros.
s_r_blocks_count	_u32	Número de bloques reservados para el administrador.
s_free_blocks_count	_u32	Número total de bloques libres.
s_free_inodes_count	_u32	Número total de inodos libres.
s_first_data_block	_u32	Primer bloque de datos.
s_log_block_size	_u32	Utilizado para calcular el tamaño del bloque lógico.
s_log_frag_size	_u32	Utilizado para calcular el tamaño del fragmento.
s_blocks_per_group	_u32	Número de bloques incluidos en un grupo.

s_frags_per_group	_u32	Número de fragmentos incluidos en un grupo.
s_inodes_per_group	_u32	Número de inodos contenidos en un grupo.
s_mtime	_u32	Determina cuándo se realizó el último montaje de este sistema de ficheros.
s_wtime	_u32	Determina cuándo se realizó la última escritura del superbloque.
s_mnt_count	_u16	Número de veces que se ha montado el sistema de ficheros en modo lectura-escritura sin haberse realizado una comprobación del mismo.
s_max_mnt_count	_u16	Máximo valor que puede tomar s_mnt_count. Llegado a este tope se obliga a la comprobación del sistema de ficheros.
s_magic	**_u16**	**Número "mágico" que identifica al sistema de ficheros (para Ext2 vale 0xEF53).**
s_state	_u16	Estado del sistema de ficheros.
s_errors	_u16	Comportamiento ante errores.
s_minor_rev_level	_u16	Nivel de revisión menor.
s_lastcheck	_u32	Hora del último chequeo.
s_checkinterval	_u32	Tiempo máximo entre chequeos.
s_creator_os	_u32	Sistema operativo creador del sistema de ficheros.
s_rev_level	_u32	Nivel de revisión.
s_dev_resuid	_u16	uid por defecto para bloques reservados.
s_dev_resgid	_u16	gid por defecto para bloques reservados.
s_first_ino	_u32	Primer inodo no reservado.
s_inode_size	_u16	Tamaño de la estructura de inodos.
s_block_group_nr	_u16	Número de grupo de bloques de este superbloque.
s_feature_compat	_u32	Conjunto de características compat.
s_feature_incompat	_u32	Conjunto de características incompat.
s_feature_ro_compat	_u32	Características compat. sólo lectura.
s_uuid[16]	_u8	uuid de 128 bits para volumen.
s_volume_name[16]	char	Nombre del volumen.
s_last_mounted[64]	char	Directorio del último montaje.
s_algorithm_usage_bitmap	_u32	Para compresión.
s_prealloc_blocks	_u8	Número de bloques a intentar preasignar.
s_prealloc_dir_blocks	_u8	Número a preasignar para directorios.
s_padding1	_u16	No documentada.
s_reserved[204]	_u32	Relleno hasta el final del bloque.

3.3 Descriptores de grupo

bg_block_bitmap	_u32	Puntero al bloque que contiene el mapa de bits de los bloques de datos del grupo.

bg_inode_bitmap	_u32	Puntero al bloque de datos que contiene el mapa de bits de los inodos del grupo.
bg_inode_table	_u32	Puntero al primer bloque de la tabla de inodos.
bg_free_block_count	_u16	Número de bloques de datos libres en el grupo.
bg_free_inode_count	_u16	Número de inodos libres en el grupo.
bg_used_dirs_count	_u16	Número de inodos asociados a directorios en el grupo.
bg_pad	_u16	No documentada.
bg_reserved[3]	_u32	Supuestamente reservados para uso futuro.

3.4 I-node

i_mode	_u16	Determina el tipo de fichero (ordinario, directorio, tubería, etc.) y derechos de acceso en modo lectura, escritura o ejecución para el propietario, grupo y resto de usuarios.
i_uid	_u16	Identificador del usuario propietario del fichero.
i_size	_u32	Tamaño del fichero en bytes.
i_atime	_u32	Última vez en que el fichero fue accedido.
i_ctime	_u32	Última vez en que se modificó el contenido de su inodo.
i_mtime	_u32	Última vez en que se modificó el contenido del fichero.
i_dtime	_u32	Hora de borrado.
i_gid	_u16	Identificador del grupo al que pertenece el propietario del fichero.
i_links_count	_u16	Número de enlaces duros a este fichero.
i_blocks	_u32	Número de bloques que comprende el fichero correspondiente al inodo.
i_flags	_u32	Contiene información miscelánea, por ejemplo si los datos del fichero están comprimidos o si cuando se borre el fichero queremos un borrado seguro de su contenido sustituyendo la información de los bloques físicos por datos aleatorios.
osd1	96 bits	Dependiente del sistema operativo 1.
i_block[EXT2_N_BLOCKS]	**_u32**	**Matriz de punteros a los bloques de datos.**
i_version	_u32	Versión del archivo (para NFS).
i_file_acl	_u32	ACL del archivo.
i_dir_acl	_u32	ACL del directorio.
i_faddr	_u32	Dirección del fragmento.
osd2	224 bits	Dependiente del sistema operativo 2.

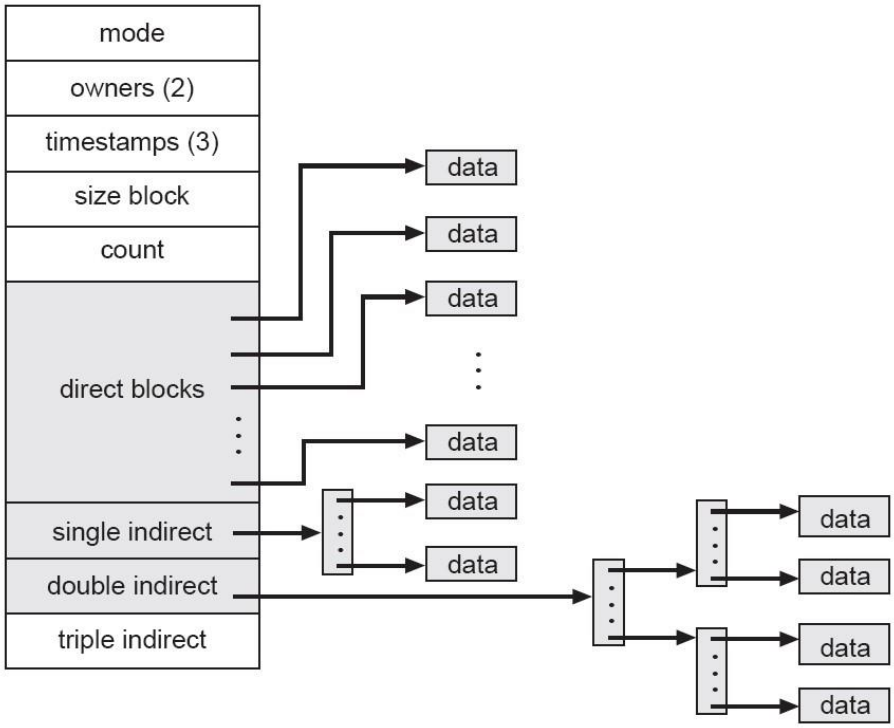

Figura 5: Estructura de inodes

3.5 Máximo tamaño de archivo

$$tam = directos * bloque + \frac{bloque}{aptr} bloque + \left(\frac{bloque}{aptr}\right)^2 bloque + \left(\frac{bloque}{aptr}\right)^3 bloque$$

$$tam = directos * bloque + \frac{bloque^2}{aptr} + \frac{bloque^3}{aptr^2} + \frac{bloque^4}{aptr^3}$$

3.6 Ventajas de ext2fs

- Bajo consumo de CPU y memoria, ideal para sistemas con recursos limitados.
- Mayor espacio utilizable, ya que no se reserva espacio para journaling.

3.7 Desventajas de ext2fs

- Mayor riesgo de corrupción de datos en caso de apagones o fallos inesperados.
- Rendimiento inferior en comparación con sistemas modernos que utilizan journaling.

3.8 Usos Actuales

Aunque ext2fs ha sido reemplazado por sistemas como ext3 o ext4 en la mayoría de los casos, todavía es útil para:

- Dispositivos con memoria flash o sistemas inmersos.
- Recuperación de datos y entornos donde se necesita un sistema de archivos simple y sin journaling.

Ext2fs marcó un avance importante en el desarrollo de sistemas de archivos para Linux y sentó las bases para sus sucesores más modernos, como ext3 y ext4.

4 EXT3

Las principales diferencias entre **ext3** y **ext2** radican en la incorporación de nuevas características en ext3 que mejoran la seguridad y el rendimiento, especialmente frente a fallos inesperados. A continuación, se detallan las diferencias clave:

1. Journaling (Registro de Cambios)

ext3 Introduce **journaling**, una característica que registra los cambios pendientes en un área especial del disco antes de aplicarlos. Esto permite una recuperación rápida del sistema de archivos tras un apagado inesperado o un fallo, reduciendo la probabilidad de corrupción. ext2 no tiene journaling, lo que lo hace más rápido en algunas operaciones, pero también más susceptible a la corrupción en caso de fallos del sistema.

El journal permite superar la complejidad de una operación atómica en el sistema de archivos ya que todas las operaciones se realizan en un área contigua de disco y las aplicaciones tienen confirmación que la información ha sido almacenada en disco. Esta área se ve lógicamente como una tabla de entradas, donde se tiene la siguiente información:

1. Banderas de operación: indica si la operación está pendiente de procesar, en proceso o procesada
2. Archivo (i-node): identificación del i-node sobre el cual se está haciendo la operación.
3. Tipo de operación: update, write, append
4. Posición relativa dentro del archivo donde se debe realizar la operación
5. Data: datos que se deben operar. Este campo es de tamaño variable, ya que puede involucrar varios bloques.

El journal trabaja de la siguiente forma:

1. Todas la operaciones de escritura se hacen al journal y se le confirma a la aplicación el éxito de la operación.
2. Periódicamente, o cuando el sistema está desocupado o cuando se llena el journal, se inicia la aplicación en el file system de las operaciones pendientes.
3. Al iniciar el sistema si la partición no fue cerrada correctamente, también se revisa las transacciones pendientes en el journal.

2. Tiempo de Recuperación

En **ext3 g**racias al journaling, la verificación del sistema de archivos después de un fallo es mucho más rápida, ya que solo necesita revisar las transacciones pendientes en el journal. **ext2** requiere una verificación completa del sistema de archivos con herramientas como fsck, lo que puede llevar mucho tiempo en discos grandes.

3. Compatibilidad

ext3 es totalmente compatible con ext2. Un sistema ext3 puede ser montado como ext2 si se desactiva el journaling. Además, es posible convertir un sistema ext2 a ext3 sin reformatearlo.

4. Rendimiento

ext3 puede ser un poco más lento que ext2 debido al overhead del journaling, pero en muchos casos la diferencia no es significativa (debido a la poca fragmentación de disco) y se ve compensada por la seguridad adicional. **ext2 e**s más rápido en operaciones de escritura intensiva, ya que no necesita realizar operaciones de journal.

5. Modos de journaling en ext3

Ext3 permite diferentes modos de journaling, lo que le da flexibilidad dependiendo de las necesidades:

- **Writeback**: Solo registra la metadata, sin garantizar que los datos y la metadata estén sincronizados.
- **Ordered**: Garantiza que los datos se escriban antes de actualizar la metadata (el modo más utilizado).
- **Journal**: Registra tanto los datos como la metadata, proporcionando la máxima seguridad, pero es más lento.

6. Uso de Recursos

ext3 consume más recursos (CPU y espacio de disco) debido al journaling.

7. Estabilidad

ext3: Es más adecuado para entornos críticos donde la estabilidad frente a fallos es esencial.

Resumen

Característica	ext2	ext3
Journaling	No	Sí
Tiempo de recuperación	Lento	Rápido
Compatibilidad	No incluye journaling	Compatible con ext2
Rendimiento	Más rápido sin journaling	Algo más lento, pero seguro
Usos comunes	Sistemas embebidos, flash	Servidores, sistemas críticos

Ext3 fue un avance importante respecto a ext2, al combinar compatibilidad y seguridad mejorada, lo que lo hizo ideal para sistemas donde la confiabilidad es crucial. Sin embargo, hoy ha sido mayormente reemplazado por **ext4**, que ofrece aún más mejoras.

5 Ext4

Además de las características de **ext3**, **ext4** implementa las siguientes características:

- **Extensión de espacio (Extents)**: Una de las principales mejoras en EXT4 respecto a EXT3 es el uso de "extensiones". En lugar de asignar bloques individuales para almacenar los datos, EXT4 usa un concepto denominado "extents", que es un rango de bloques consecutivos. Esto reduce la fragmentación y mejora el rendimiento, especialmente en archivos grandes.
- **Revisión de los tiempos de fecha**: EXT4 tiene una mejor precisión para las fechas de acceso, modificación y cambio (timestamps), con la posibilidad de almacenar fechas a nivel de nanosegundos, lo cual es más preciso que en sistemas anteriores como EXT3.
- **Pre-asignación de espacio (Pre-allocation)**: EXT4 puede realizar la pre-asignación de bloques, lo que es útil para aplicaciones que crean grandes archivos de manera continua. Esto mejora el rendimiento al evitar la fragmentación.
- **Verificación de la integridad**: EXT4 es más robusto en cuanto a la verificación de la integridad de los datos. La estructura de metadatos del sistema de archivos está diseñada para prevenir corrupciones a través de sumas de comprobación.

Ventajas:

- Mejor rendimiento en comparación con EXT3, especialmente en operaciones con archivos grandes.
- Mayor fiabilidad y recuperación rápida en caso de fallos debido al journaling.
- Menor fragmentación gracias a los extents y la asignación más eficiente de bloques.
- Soporte para grandes tamaños de archivos y sistemas de archivos.

Desventajas:

- La configuración y las características avanzadas pueden hacer que sea más complejo de administrar en ciertos entornos.
- Aunque más rápido que EXT3, no es tan rápido como otros sistemas de archivos más especializados en rendimiento (por ejemplo, XFS).

6 NTFS

El **NTFS** (New Technology File System) es un sistema de archivos desarrollado por Microsoft, que se utiliza en sus sistemas operativos Windows desde Windows NT. Es el sucesor de sistemas de archivos más antiguos como FAT16 y FAT32, y es ampliamente utilizado en sistemas Windows debido a sus características avanzadas en comparación con esos sistemas más antiguos. Las características más importante de NTFS son:

1. **Estructura del Sistema de Archivos NTFS**

2. **Master File Table**:
   Al igual que en otros sistemas de archivos, NTFS comienza con un área de datos denominada **MFT (Master File Table)**, que contiene información crítica sobre la estructura del sistema de archivos. La MFT es el componente central de NTFS y puede considerarse el "corazón" del sistema de archivos. La MFT contiene registros para cada archivo y directorio en el sistema. Cada archivo o directorio tiene una entrada en la MFT que contiene:
   - El nombre del archivo.
   - Información sobre los permisos y atributos de seguridad.
   - El tamaño del archivo.
   - El puntero o dirección donde se almacenan los datos del archivo.

   En NTFS, la MFT es una base de datos que contiene tanto los metadatos como los datos de los archivos pequeños. Los archivos y directorios que no son demasiado grandes pueden almacenarse directamente en la MFT, lo que mejora la eficiencia.

3. **Registros de archivo**:
   Cada archivo en NTFS tiene un registro de archivo asociado en la MFT. Los registros de archivo contienen la metadata (información como el nombre, permisos, y fechas) así como los punteros a los bloques de datos que almacenan el contenido real del archivo.

4. **Cluster (Bloque de Datos)**:
   NTFS utiliza **clusters** (o unidades de asignación) para almacenar los datos del archivo. Un cluster es un conjunto de sectores en el disco y es la unidad más pequeña de almacenamiento de datos. NTFS agrupa los datos en clusters de tamaño variable, generalmente entre 512 bytes y 64 KB, dependiendo de la configuración del sistema de archivos y el tamaño de la partición.

5. **Bitmaps de espacio libre**:
   NTFS mantiene un bitmap que permite realizar un seguimiento de los bloques de datos que están ocupados y libres. Esto permite una asignación eficiente de espacio y facilita la recuperación de espacio libre cuando se eliminan archivos.

6. **Árbol B+ (B+ Tree)**:
   Para optimizar la búsqueda de archivos y directorios, NTFS utiliza estructuras de datos llamadas **Árboles B+**. Los árboles B+ son estructuras de búsqueda que permiten acceder rápidamente a los archivos y directorios dentro de una partición NTFS. La organización en árboles B+ facilita la indexación y mejora el rendimiento de las operaciones de lectura y escritura.

7. **Características avanzadas de NTFS**

1. **Journaling (Registro de transacciones)**:
   NTFS soporta **journaling**, lo que significa que mantiene un registro de las transacciones realizadas en el sistema de archivos antes de que se efectúen. Si el sistema se apaga inesperadamente o se produce un fallo, NTFS puede usar el journal para restaurar el sistema de archivos a un estado consistente. Esto proporciona una mayor fiabilidad y reduce el riesgo de corrupción de datos.

2. **Soporte de archivos grandes**:
   NTFS permite el almacenamiento de archivos muy grandes, con un tamaño máximo de archivo que puede superar los 16 exabytes, mucho más allá de los límites de FAT32.
3. **Seguridad y control de acceso**:
   NTFS ofrece un control de acceso robusto utilizando **ACLs (listas de control de acceso)**. Cada archivo y directorio puede tener configurados permisos detallados que determinan quién puede leer, escribir, ejecutar o modificar el archivo. Esto se basa en el modelo de seguridad de Windows.
4. **Compresión de archivos**:
   NTFS permite la **compresión de archivos** a nivel del sistema de archivos, lo que significa que los archivos pueden ser comprimidos automáticamente para ahorrar espacio. Los archivos comprimidos se descomprimen automáticamente cuando se accede a ellos.
5. **Encriptación de archivos (EFS)**:
   NTFS incluye soporte para **EFS (Encrypting File System)**, lo que permite a los usuarios encriptar archivos y carpetas, protegiendo la información sensible. Los archivos encriptados solo pueden ser leídos por los usuarios que tienen la clave de encriptación adecuada.
6. **Quotas de disco**:
   NTFS permite la implementación de **cuotas de disco**, lo que permite a los administradores establecer límites sobre cuánto espacio de almacenamiento puede utilizar cada usuario en un volumen NTFS.
7. **Reparse Points**:
   NTFS admite **puntos de reanálisis** (reparse points), que son estructuras de metadatos que permiten a NTFS asociar archivos o directorios con otros datos o dispositivos. Un ejemplo común de reparse point es el **symlink** (enlace simbólico), que apunta a otro archivo o directorio en el sistema de archivos.
8. **Hard Links**:
   NTFS soporta **hard links** (enlaces duros), lo que permite que varios archivos tengan el mismo contenido físico en el disco pero diferentes nombres. Los hard links son útiles para crear referencias a archivos sin duplicar los datos.
9. **Acceso optimizado para grandes volúmenes y servidores**:
   NTFS está diseñado para escalar y ser eficiente en entornos con grandes volúmenes de datos, como servidores. Es capaz de manejar grandes cantidades de archivos y grandes volúmenes de almacenamiento, y permite optimizar el rendimiento en escenarios empresariales.

6.1 Ventajas y Desventajas de NTFS

Ventajas:

- **Seguridad avanzada** mediante ACLs (listas de control de acceso) y EFS (encriptación de archivos).
- **Mayor fiabilidad** gracias al journaling y a las características de recuperación ante fallos.

- Soporte para **archivos grandes** y **particiones grandes** (hasta 16 exabytes).
- Funcionalidades como **compresión de archivos**, **hard links** y **puntos de reanálisis** que mejoran la flexibilidad y eficiencia.
- **Control de cuotas de disco** para gestionar el uso de espacio en sistemas compartidos.

Desventajas:

- **Compatibilidad limitada**: NTFS no es nativamente compatible con muchos sistemas operativos fuera de Windows, aunque se pueden usar herramientas de terceros para leer y escribir en NTFS en Linux y macOS.
- **Complejidad**: La administración de NTFS puede ser más compleja debido a sus características avanzadas como cuotas de disco, encriptación y journaling.

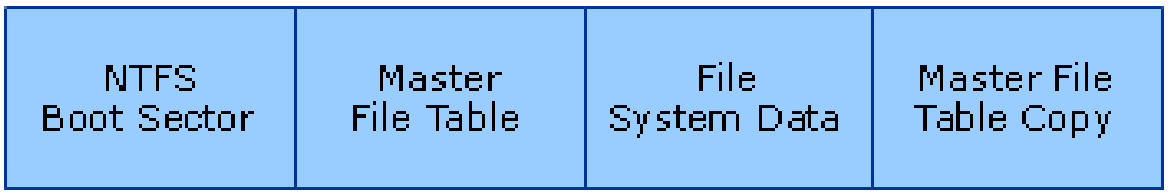

Figura 6: Estructura de bloques de NTFS

6.2 Master File Table

- Master File Table
  - Descriptor por archivo 4K
  - 1.5 K de datos
  - Atributos adicionales
  - Listado de bloques
  - Descriptores adicionales
  - Accesos (ACL)
- Descriptores especiales
  - Copia de la MFT
  - Archivo de errores

- Manejo de journal

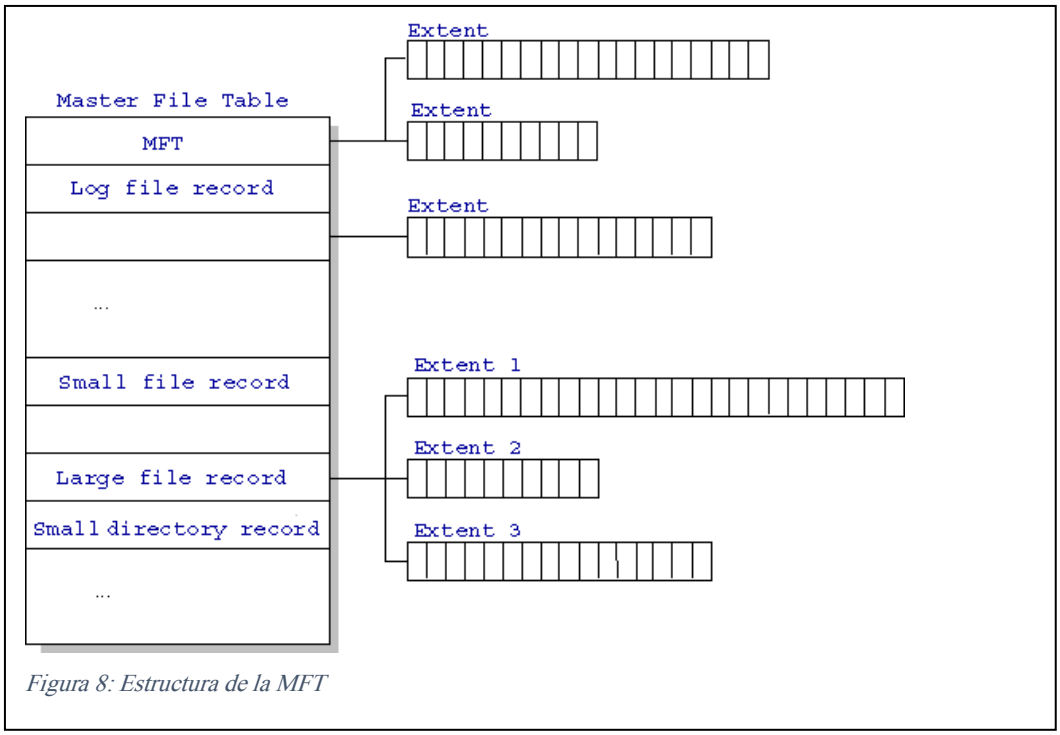

Figura 8: Estructura de la MFT

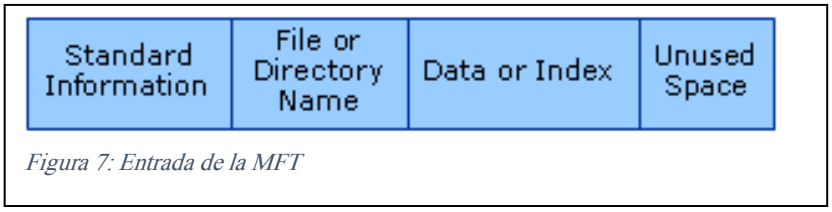

Figura 7: Entrada de la MFT

7 Sistemas de archivos sobre dispositivos flash

El tiempo de vida de un dispositivo flash está determinado por varios factores clave que afectan la durabilidad y la fiabilidad de las celdas de memoria flash. El factor principal es el límite de **ciclos de escritura/lectura**. Las celdas de memoria flash tienen una cantidad finita de ciclos de escritura y borrado antes de que comiencen a degradarse. Cada vez que se escribe en una celda (programación), se desgasta un poco. A medida que se realizan más operaciones de escritura y borrado en una celda específica, su vida útil se reduce gradualmente.

La **nivelación de desgaste** (wear leveling) en dispositivos de almacenamiento flash es una técnica diseñada para distribuir de manera uniforme las operaciones de escritura y borrado en todas las celdas de memoria flash disponibles. Esta técnica es crucial para prolongar al vida útil del dispositivo.

Aquí están los puntos clave de la nivelación de desgaste:

- **Desgaste desigual**: Debido a las limitaciones de las celdas de memoria flash, las celdas que experimentan escrituras frecuentes pueden degradarse más rápido que las que no. Esto podría llevar a una falla prematura del dispositivo si no se gestiona adecuadamente.
- **Técnica de nivelación**: La nivelación de desgaste es un proceso implementado en el controlador del dispositivo de almacenamiento flash. Su objetivo es redistribuir las operaciones de escritura y borrado a lo largo de todas las celdas de memoria disponibles, evitando así que unas pocas celdas se desgasten más rápido que otras.
- **Algoritmos utilizados**: Los controladores de almacenamiento flash utilizan algoritmos sofisticados para realizar la nivelación de desgaste de manera eficiente. Estos algoritmos consideran varios factores como el uso actual de las celdas, la historia de las operaciones de escritura y borrado, y la capacidad disponible.
- **Beneficios**: La nivelación de desgaste ayuda a prolongar la vida útil efectiva del dispositivo de almacenamiento flash al asegurar que todas las celdas de memoria sean utilizadas de manera equitativa. Esto también mejora la fiabilidad y el rendimiento a largo plazo del dispositivo.

Existen varios algoritmos principales utilizados para la nivelación de desgaste en dispositivos de almacenamiento flash. Estos algoritmos están diseñados para distribuir de manera eficiente las operaciones de escritura y borrado a lo largo de todas las celdas de memoria flash disponibles, con el objetivo de prolongar la vida útil del dispositivo y mejorar su rendimiento. Aquí te presento algunos de los más comunes:

1. **Nivelación de desgaste dinámica (*Dynamic Wear Leveling*):**

Este algoritmo monitorea continuamente el estado de las celdas de memoria y redistribuye las operaciones de escritura y borrado según sea necesario. Es adaptable y puede ajustar la distribución en tiempo real según los patrones de uso y el estado de desgaste de las celdas.

2. Nivelación de desgaste estática (*Static Wear Leveling*):

A diferencia de la dinámica, este método redistribuye las operaciones de escritura y borrado de manera predeterminada en intervalos regulares o cuando ciertos umbrales de desgaste son alcanzados. Es menos flexible pero más simple de implementar.

3. Nivelación de desgaste global (*Global Wear Leveling*)

Este algoritmo intenta nivelar el desgaste a través de todas las celdas del dispositivo de almacenamiento flash en una escala más amplia, considerando toda la capacidad del dispositivo. Es eficaz para dispositivos con alta variabilidad en el uso de celdas.

4. **Nivelación de desgaste local (*Local Wear Leveling*)**

A diferencia del global, este algoritmo trata de equilibrar el desgaste entre celdas de memoria en regiones más pequeñas o específicas del dispositivo. Es útil cuando ciertas áreas del dispositivo están más activamente utilizadas que otras.

5. Nivelación de desgaste estática con detección de hot-spots

Este enfoque combina la estática con la detección de áreas de alta actividad (hot-spots). Identifica y trata de manera proactiva las áreas del dispositivo que experimentan un uso intensivo para minimizar el desgaste desigual.

6. Algoritmos basados en mapeo de bloques (*Block Mapping Algorithms*):

Estos algoritmos asignan de manera inteligente las operaciones de escritura y borrado a bloques de memoria flash específicos, considerando el historial de uso y la distribución de desgaste.

7. Algoritmos adaptativos y heurísticos:

Algoritmos más avanzados que pueden combinar diferentes técnicas mencionadas anteriormente, adaptándose dinámicamente a los patrones de uso y a las características específicas del dispositivo de almacenamiento flash.

Cada uno de estos algoritmos tiene sus ventajas y desventajas dependiendo del tipo de dispositivo flash y de las aplicaciones para las que se utilice. La elección del algoritmo adecuado puede marcar la diferencia en la durabilidad, el rendimiento y la confiabilidad a largo plazo del dispositivo de almacenamiento flash.

8 Otros sistemas de archivos

Los sistemas de archivos más utilizados actualmente varían según el entorno y el tipo de dispositivo. Aquí están los más relevantes por categoría:

1. Sistemas operativos Linux

- **ext4 (Fourth Extended File System)**: Es el más común en distribuciones Linux, conocido por su estabilidad, rendimiento y soporte para discos grandes.
- **XFS**: Optado en servidores por su alto rendimiento en operaciones de entrada/salida y su capacidad para manejar grandes volúmenes de datos.
- **Btrfs (B-Tree File System)**: Se está popularizando gracias a características avanzadas como snapshots, compresión y capacidades de reparación automática.

2. macOS

- **APFS (Apple File System)**: Introducido con macOS High Sierra, está optimizado para unidades SSD y ofrece encriptación avanzada, snapshots y mejor administración de espacio.
- **HFS+ (Hierarchical File System Plus)**: Usado anteriormente en macOS antes del reemplazo por APFS.

3. Sistemas de servidores y redes

- **ZFS (Zettabyte File System)**: Conocido por su confiabilidad y características avanzadas, como snapshots, detección y corrección de errores, y administración de almacenamiento combinada.

4. Sistemas especializados

- **ReFS (Resilient File System)**: Sistema de archivos de Microsoft diseñado para alta disponibilidad y resistencia a fallos en servidores.
- **ISO 9660**: Utilizado en discos ópticos como CD y DVD.

Cada sistema de archivos tiene ventajas específicas dependiendo del uso. Por ejemplo, NTFS es ideal para entornos Windows con características de seguridad, mientras que ext4 es la opción predeterminada para la mayoría de las distribuciones de Linux debido a su balance entre rendimiento y estabilidad.