

# NTFS (Windows NT)

## Principales características que se deseaba que tuviera

### **Soporte para discos grandes**

FAT usa entradas de 16 bits para referenciar clusters (puede referenciar hasta  $2^{16} = 65536$  clusters, 65Mb con clusters de 1Kb). Para poder usar discos mas grandes la unica forma es aumentar el tamaño de cluster, aumentando el desperdicio por fragmentación. NTFS usa direcciones de 64 bits para referenciar clusters, pudiendo entonces utilizar discos de una capacidad muy superior  $2^{64} = 18.446.744.073.709.551.616$  clusters.

### **Seguridad a nivel de archivo**

NTFS usa dos tipos de listas para implementar seguridad y auditoría

Discretionary access control lists (DACLS) : indican quien tiene permiso a hacer que esobre cada archivo

System access control lists (SACLs): que eventos quedaran registrados en el log de acciones sobre un archivo

**Soporte Unicode** (código similar al ASCII que usa 2 bytes para cada carácter. A diferencia del ASCII, por usar 2 bytes, puede representar todos los símbolos de todos los alfabetos)

Esto permite que los nombres de archivos y directorios no estén limitados al alfabeto inglés.

### **Características de autorecuperación** (a diferencia de FAT o HPFS)

Mantiene un log de transacciones del file system. Si el sistema falla, es posible restaurar un estado consistente del file system a partir del log minimizando la pérdida de datos.

## Manejo de disco:

Si al formatear no se especifica tamaño de cluster, se toma un default que depende del tamaño de la partición, según la siguiente tabla.

| Tamaño de partición o disco | Tamaño de Cluster |
|-----------------------------|-------------------|
| $\leq 512$ Mb               | 512 bytes         |
| 513Mb-1024Mb (1GB)          | 1Kb               |
| 1025Mb-2048Mb(2GB)          | 2Kb               |
| $\geq 2049$ Mb              | 4Kb               |

Los metadatos de NTFS se guardan como archivos dentro del file system. Al inicializar una partición NTFS se crean 11 archivos que se usan para administración

| Archivo   | Registro en la MFT | Descripción  |
|-----------|--------------------|--|
| \$MFT     | 0                  | Master File Table  |
| \$MFTMIRR | 1                  | Copia de los primeros 16 registros de la MFT   |
| \$LOGFILE | 2                  | Archivo de log transaccional   |
| \$VOLUME  | 3                  | Numero de serie del volumen, hora de creación  |
| \$ATTRDEF | 4                  | Definiciones de atributos  |
| .         | 5                  | Directorio raiz del disco  |
| \$BITMAP  | 6                  | Mapa de clusters de la unidad (in-use vs. free)  |
| \$BOOT    | 7                  | Boot record de la unidad   |
| \$BADCLUS | 8                  | Lista de clusters dañados  |
| \$QUOTA   | 9                  | Contiene a partir de NT 5.0, información sobre el espacio máximo de disco que puede usar cada usuario. |
| \$UPCASE  | 10                 | Mapea los caracteres en minúscula a sus equivalentes en mayúscula                                      |

NT no muestra estos archivos pero pueden verse usando `dir /ah <nombre_arch>`

### ***\$MFT Master File Table***

La MFT es analoga a la FAT en el sentido que contiene un mapa de todos los archivos y directorios del disco, incluyendo los archivos de metadatos de NTFS.

La tabla está dividida en registros (generalmente de 1Kb pero en discos grandes pueden ser mayores) y la descripción de un archivo o directorio puede ocupar uno o mas registros. En estos registros se guardan las opciones de seguridad del archivo, y sus atributos como si es sólo lectura, oculto, de sistema, etc. Y su ubicación en el disco. El hecho de que para cada archivo se

pueda ocupar mas de un registro y de que la MFT sea a su vez un archivo y por ende pueda crecer o achicarse permite que se pueda ocupar con metadatos sólo el espacio necesario. Una desventaja de esto es que, como no se asigna de antemano todo el espacio que la MFT va a usar, al ir creciendo y achicándose, la MFT se fragmenta, lo cual produce una baja en la velocidad de acceso a la MFT, alargando los tiempos de acceso al file system. La solución propuesta para esto es destinar una cantidad de clusters después de la MFT (Zona MFT) que queden sin asignar a ningún archivo ni directorio en tanto se cuente con otros clusters vacíos en disco. Cuando el disco se va llenando, se empiezan a usar esos clusters y aumenta la posibilidad de que la MFT se fragmente, por eso el acceso a disco se hace mas lento. NTFS no permite que las herramientas de defragmentación defragmenten la MFT.

Internamente NTFS identifica los directorios y archivos por el número de registro de la MFT donde se encuentra su registro inicial. Los archivos de metadatos tiene un número de registro base predeterminado (Aclarar que de aca se deduce que los registros para un determinado archivo no necesitan estar contiguos).  
LA MFT se almacena cerca del inicio del disco.

### Registros de la MFT

|        |                      |               |
|--------|----------------------|---------------|
| Header | Headers de Atributos | Espacio libre |
|--------|----------------------|---------------|

#### Header

Nro de secuencia usado para verificación de integridad  
Un puntero al primer atributo en el registro  
Un puntero al primer byte de espacio libre en el registro  
Si no es el registro base del archivo en la MFT, tiene el número del registro base.

#### Atributos

Hay 14 tipos de atributo.

|                        |   |
|------------------------|---|
| \$VOLUME_VERSION       | Versión del disco   |
| \$VOLUME_NAME          | Nombre del disco  |
| \$VOLUME_INFORMATION   | versión de NTFS   |
| \$FILE_NAME            | Nombre del directorio o archivo                                       |
| \$STANDARD_INFORMATION | Fecha y hora de creación, modificación, oculto, sistema, sólo lectura |
| \$SECURITY_DESCRIPTOR  | Información de seguridad  |
| \$DATA                 | Datos del archivo   |
| \$INDEX_ROOT           | Contenido del directorio  |
| \$INDEX_ALLOCATION     | Contenido del directorio  |
| \$BITMAP               | Bitmap del contenido del directorio                                   |
| \$ATTRIBUTE_LIST       | Headers de atributos no residentes.                                   |
| \$SYMBOLIC_LINK        | No usado. El resource kit tiene una herramienta para crear links      |
| \$EA_INFORMATION       | Atributos de compatibilidad OS/2                                      |
| \$EA                   | Atributos de compatibilidad OS/2                                      |

En disco, un atributo se guarda en dos partes, el header del atributo y los datos del atributo.

#### Header de atributo:

Tipo de atributo  
Nombre de atributo  
Flags  
Ubicación de los datos del atributo \*

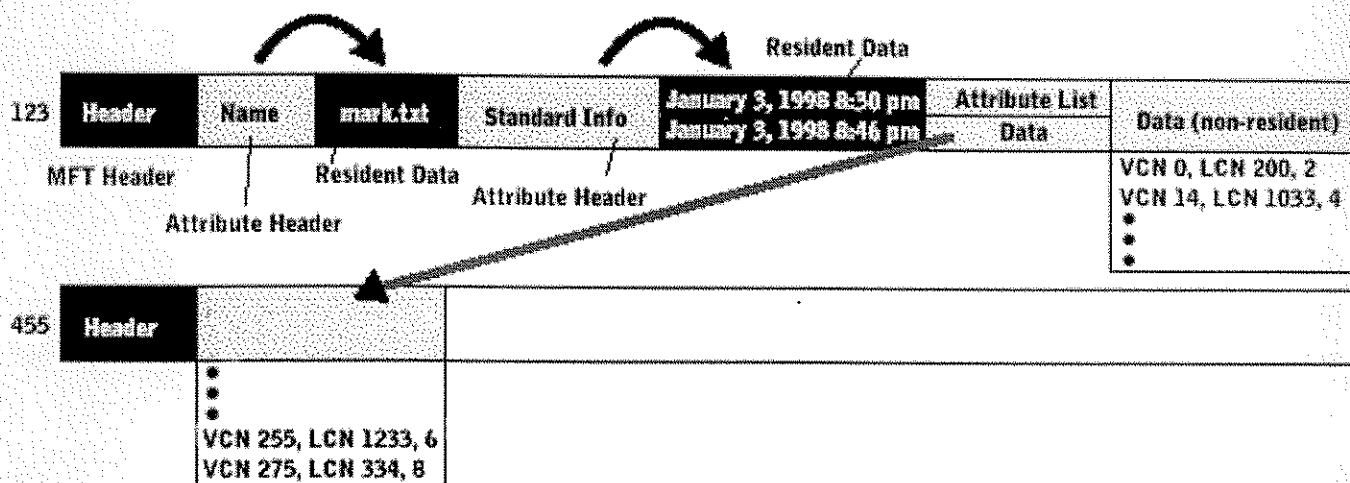
\*Los datos del atributo se guardan siempre que sea posible en registros de la MFT, en estos casos se los llama *residentes*. Si no es posible guardarlos en la MFT (porque el espacio en el registro no es suficiente), se asignan clusters en el disco para guardarlos y en este caso se lo llama atributo *No residente* (los datos del archivo son un ejemplo de esto). El nombre de archivo, los atributos de información standard y los atributos de seguridad son siempre *residentes*. Si todos los atributos que deben ser residentes no entran en un solo registro de la MFT, se asignan mas registros y se guarda en el primero una lista de los atributos y su ubicación en los otros registros.

Para identificar la ubicación de los atributos no residentes se guardan en la MFT tres valores por cada fragmento del archivo

VCN (*virtual cluster number*): Número que el primer cluster del fragmento ocupa en el archivo.  
LCN (*logical cluster number*): Número de cluster real del disco en el cuál se encuentra el inicio del fragmento del archivo  
Cantidad de clusters contiguos en disco para el archivo a partir del LCN.

Ejemplo: Archivo Mark.txt.

Creado el 3 de enero de 1998 a las 8:30  
Modificado 3 de enero de 1998 a las 8:46



### Directorios:

Son archivos con atributos de tipo \$INDEX\_. El registro en la MFT de un directorio contiene una lista ordenada con los nombres e información standard de cada uno de los archivos que contiene de forma que para mostrar el contenido de un directorio sólo hay que acceder al registro en la MFT del directorio y no a los registros de los archivos. Si todos estos datos no enran en un solo registro de la MFT, NTFS NO utiliza mas registros sino un mecanismo de buffers.

Recuperación de fallas mediante el uso de logs:

Cada partición tiene un archivo de log con registros de dos tipos: REDO y UNDO. Los registros de tipo UNDO se usan para acciones como por ejemplo agregar datos a un archivo. Si hay una falla y la operación no se completó correctamente, hay que deshacer todas las actualizaciones que se hicieron. Los registros de tipo REDO se usan para la eliminación de archivos por ejemplo.

Las actualizaciones se hacen en memoria y se va actualizando el log. Cada 5 segundos se produce un checkpoint en el cual se registran los cambios de todas las estructuras en disco y si se completa esto con éxito, se eliminan los registros correspondientes del log evitando que se llene.

### Control de Acceso

Se utiliza la MFT, por medio de un atributo de tipo \$SECURITY\_DESCRIPTOR que tiene la siguiente estructura:

#### Security Descriptor

|                 |
|-----------------|
| SID's del owner |
| SID del grupo   |
| DACL            |
| SACL            |

ACE : Son las entradas de las ACL

|  |
|--|
| SID  |
| Mask   |
| Tipo de ACE [Acceso no permitido   Acceso permitido   Auditoría] |
| Bitfield de herencia   |

#### Principales valores de Mask

GENERIC\_EXECUTE  
GENERIC\_READ  
GENERIC\_WRITE

DELETE  
WRITE\_DAC  
WRITE\_OWNER

**Trustee:** Es un usuario o grupo.

**Discretionary Access Control List:** Es una lista de ACEs que contiene entradas que indican para un usuario dado, a que tipo de acceso tiene permiso o a que tipo de acceso NO tiene permiso.

**System Access Control List:** Es una lista de ACEs que contiene entradas que indican para un usuario dado, que tipos de intento de acceso se deben incluir en el log de eventos de seguridad.

**Access token:** es la forma en que el thread que quiere acceder a un objeto, le indica al file system qué accesos necesita, el SID del owner y del grupo del thread.

Como funciona:

```
Leer el access token del proceso
Si el objeto no tiene DACL entonces
    Permitir acceso
Sino
    Leer el primer ACE de la DACL
    Mientras (queden entradas en la DACL) y (no se encuentre un ACE que niegue alguno de los accesos solicitados)
        y (no se encuentren ACE's que permitan todos los accesos solicitados)
            Leer el siguiente ACE;
    Si se encontraron todos los ACE's para permitir todos los accesos solicitados
        Permitir acceso
    Sino{
        Si el objeto tiene SACL entonces
            Mientras (queden entradas en la SACL) {
                Si el SID leído es igual al del access token y alguno de los tipos de acceso coincide
                    Agregar evento al log de seguridad;
                Leer el siguiente ACE;
            }
        }
    }
```

### **Como leer las ACL:**

**GetExplicitEntriesFromAcl:** devuelve los ACEs en una ACL para un trustee determinado. Esto es útil si se desea copiar una ACL.

**GetEffectiveRightsFromAcl:** dado un usuario, devuelve todos los accesos permitidos y negados al usuario, ya sea por medio de su propio id o del/los grupos a los que pertenece.

### **Como modificar las ACL:**

**GetSecurityInfo:** obtiene el handle a la DACL o SACL a partir del Security Descriptor del objeto

**BuildExplicitAccessWithName:** crea un ACE con las opciones indicadas

**SetEntriesInAcl:** permite agregar ACEs o crear una lista nueva

**SetSecurityInfo:** adjunta una nueva ACL a un Security Descriptor

En las funciones para leer y modificar las ACL, se utiliza las siguientes estructuras de datos:

```
typedef struct _EXPLICIT_ACCESS {
    DWORD grfAccessPermissions;
    ACCESS_MODE grfAccessMode;
    DWORD grfInheritance;
    TRUSTEE Trustee;
} EXPLICIT_ACCESS;
```

```
typedef struct _ACL {
```

```
BYTE AclRevision;  
BYTE Sbz1;  
WORD AclSize;  
WORD AceCount;  
WORD Sbz2;  
} ACL;
```

### ***\$MFTMIRR***

Contiene una copia de los primeros 16 registros de la MFT (o sea que contiene información sobre la ubicación de los archivos de metadatos) y se almacena por la mitad del disco. Hay un boot record de NTFS con ubicación fija en los primeros 512 bytes del disco que contiene información sobre la ubicación exacta de la MFT y \$MFTMIRR. Si se produce un error al leer la MFT, se usa \$MFTMIRR.

Nota: La función NtFsControlFile de la biblioteca NTDLL.DLL permite enviar comandos directamente al file system y, por ejemplo, mover clusters de lugar.