NTFS Dosya Sistemi

Dosya Sistem Analizi Dersi

Yrd. Doç. Dr. Erhan AKBAL

Giriş

- New Technology File System - Yeni Teknoloji Dosya Sistemi (NTFS), Microsoft tarafından tasarlanmıştır ve Microsoft Windows NT, Windows 2000, Windows XP, Windows 7,8,10 ve Windows Server için varsayılan dosya sistemidir.
- FAT mobil ve küçük depolama aygıtlarında halen mevcuttur, ancak NTFS Windows incelemeleri için en yaygın dosya sistemidir.
- NTFS, çok özellikli ve çok ölçeklenebilir olduğu için FAT'den çok daha karmaşık bir dosya sistemidir.

Giriş

- NTFS, güvenilirlik, güvenlik ve büyük depolama aygıtları için tasarlanmıştır.
- Ölçeklenebilirlik, belirli içerikli veri yapılarını saran genel veri yapılarının kullanılmasıyla sağlanmaktadır.
- Ölçeklenebilir bir tasarım yapısı vardır, çünkü dahili veri yapısı, yeni istekler dosya sistemi üzerine yerleştirildiğinde zaman içinde değişebilir.
- Bir NTFS dosya sistemindeki verilerin her baytının bir dosyaya ayrılmış olmasıdır.

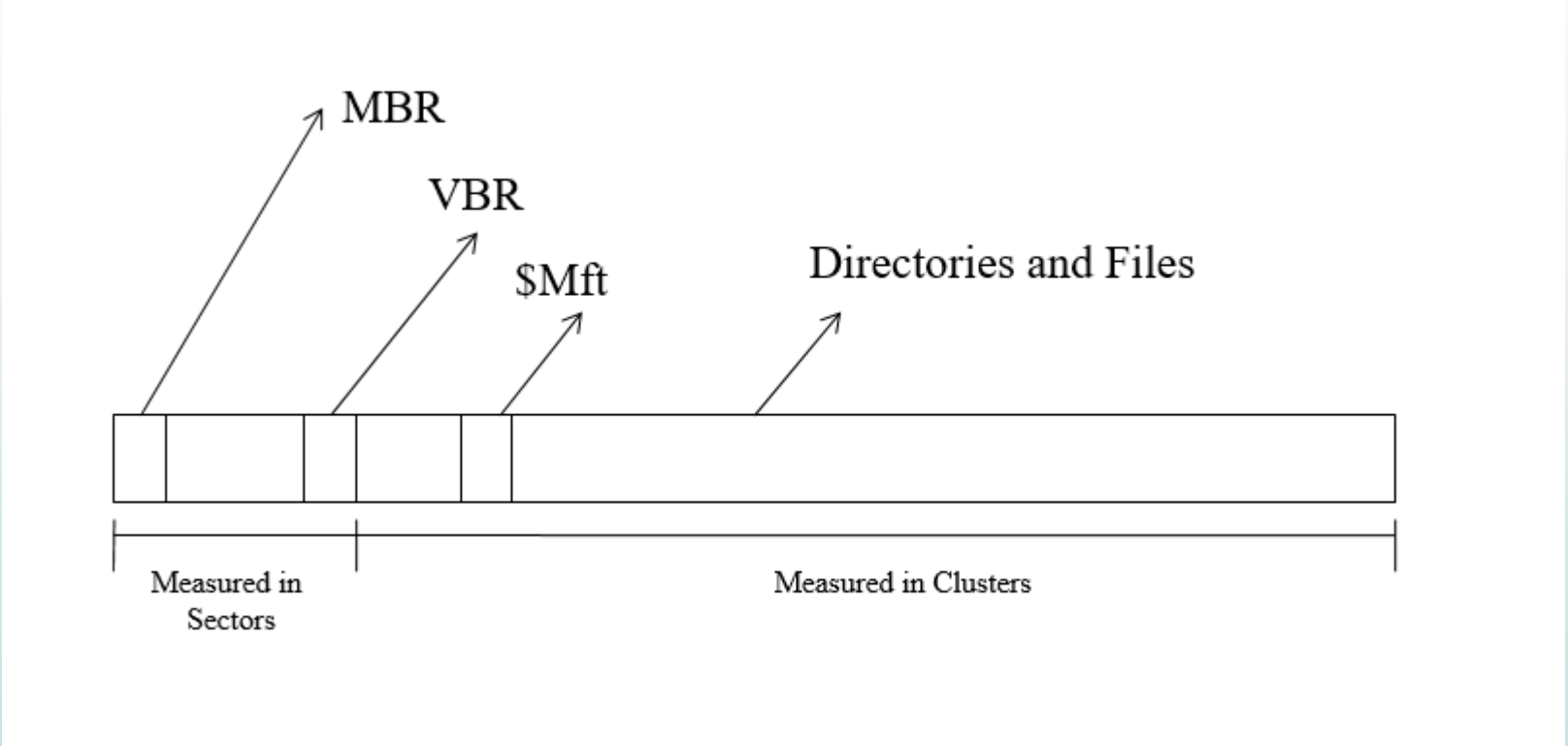
NFTS Ailesi

- NTFS'nin beş sürümünü yayımlanmıştır:
- V1.0: 1993'te Windows NT 3.1 ile çıktı. V1.0, v1.1 ve sonraki sürümlerle uyumlu değildir:
- V1.1: 1995 yılında Windows NT 3.51 ile birlikte yayınlandı. Akışları ve erişim denetim listelerini içeren sıkıştırılmış dosyaları destekliyor.
- V1.2: 1996'da Windows NT 4.0 ile birlikte yayımlanmıştır. Güvenlik tanımlayıcılarını destekler. OS çıktıktan sonra yaygın olarak NTFS 4.0 olarak adlandırılır.
- V3.0: Windows 2000 ile çıktı. Disk kotalarını, Şifreleme Dosya Sistemi, seyrek dosyalar, yeniden ayrıştırma noktaları, güncelleme sıra numarası (USN) günlüğü, $ Uzantı klasörünü ve dosyalarını destekler. Genellikle işletim sistemi sürümü bittikten sonra NTFS 5.0 olarak adlandırılır.
- V3.1: 2001'de (ve daha sonra Windows Vista ve Windows 7'de de kullanılan) Windows XP ile birlikte yayınlandı. Master File Table (MFT) girişlerini yedek MFT kayıt numarasıyla genişletti (hasarlı MFT dosyalarını kurtarmak için kullanışlıdır). OS yayımlandıktan sonra yaygın olarak NTFS 5.1 adı verilir

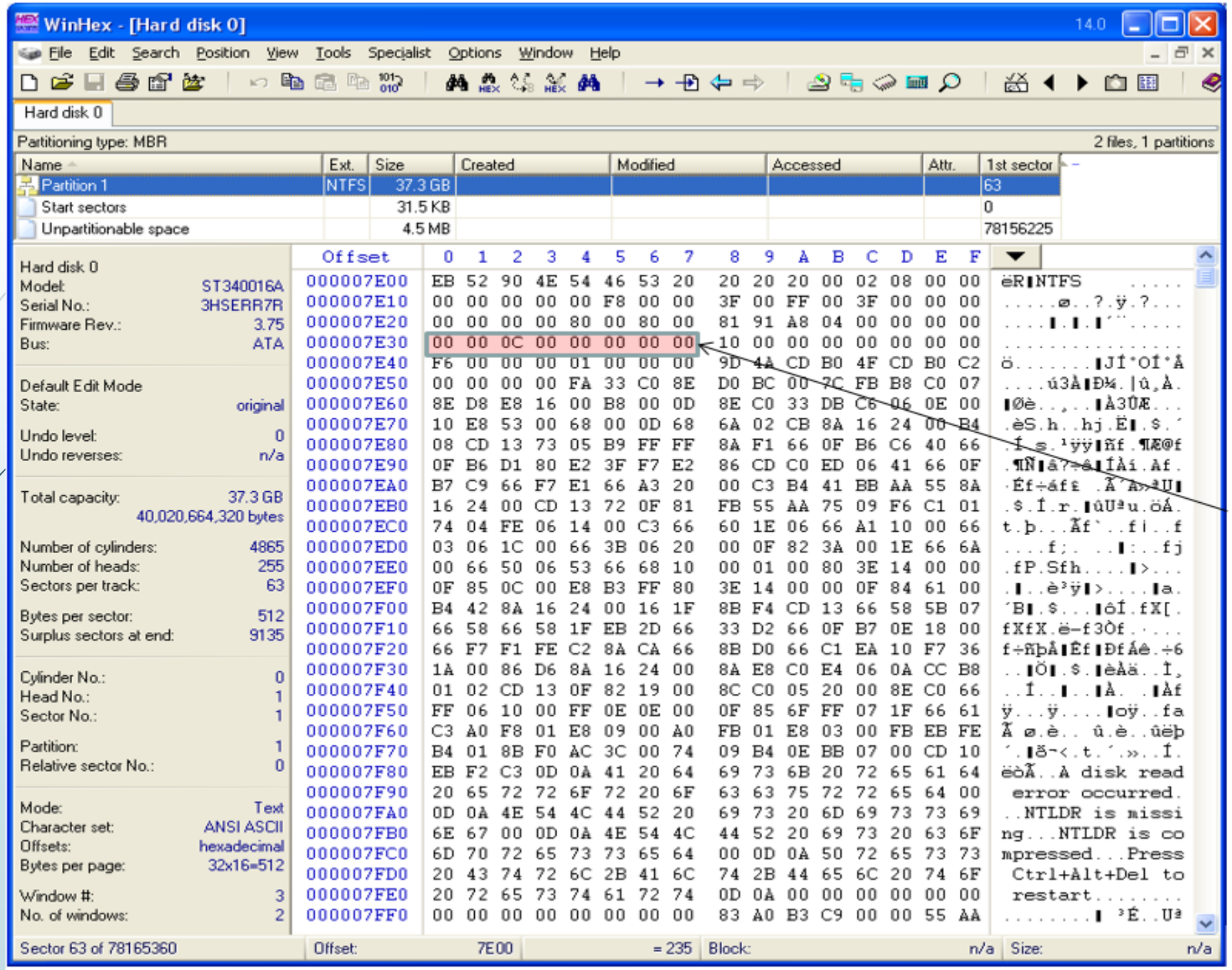
NTFS Yapısı

- NTFS de her şey bir dosyadır
  - Klasörler, Dosyalar
  - Önyükleme (Bootstrap) verileri
  - Dosya tahsis Bitmapleri
  - Metadata
- Master File Table NTFS'in kalbidir
- MFT başlangıcı Volume Boot Record içindedir
- VBR MFT'deki $Boot kaydındadır.

NTFS Partition Yapısı



VBR



Location of $MFT

Little Endian

0x0C0000 * 8 + 0x3F = Sector count of $MFT

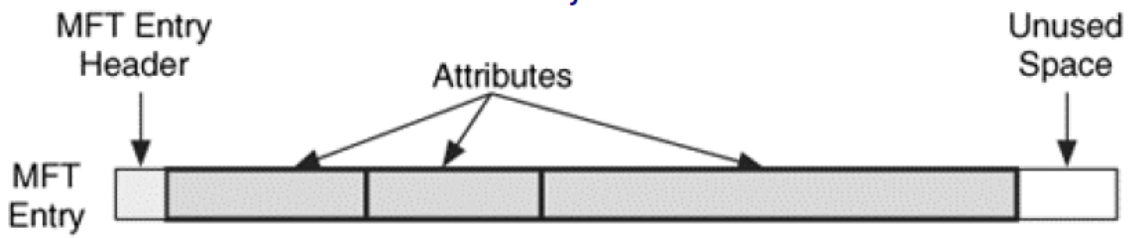
NTFS'deki VBR yapısı

Byte Offset	Field Length	Sample Value	Field Name
0x00	3		Jump to boot code
0s03	8	NTFS	OEM Name
0x0B	2	0x0200	Bytes Per Sector
0x0D	1	0x08	Sectors Per Cluster
0x0E	2	0x0000	Reserved Sectors
0x10	3	0x000000	*always 0*
0x13	2	0x0000	*not used by NTFS*
0x15	1	0xF8	Media Descriptor
0x16	2	0x0000	*always 0*
0x18	2	0x3F00	Sectors Per Track
0x1A	2	0xFF00	Number Of Heads
0x1C	4	0x3F000000	Hidden Sectors
0x20	4	0x00000000	*not used by NTFS*
0x24	4	0x80008000	*not used by NTFS*
0x28	8	0x4AF57F0000000000	Total Sectors
0x30	8	0x0000000000040000	Logical Cluster Number for the file $MFT
0x38	8	0x54FF070000000000	Logical Cluster Number for the file $MFTMirr
0x40	4	0xF6000000	Clusters Per File Record Segment
0x44	4	0x01000000	Clusters Per Index Block
0x48	8	0x14A51B74C91B741C	Volume Serial Number
0x50	4	0x00000000	Checksum
0x54	426		Bootstrap program code
0xFE	2	0x55AA	Signature bytes

MFT (Master File Table) Konsepti

- Tüm dosya ve dizinler hakkında bilgi içerdiğinden, Ana Dosya Tablosu (MFT) NTFS'in kalbidir.
- Her dosyanın ve dizininin tabloda en az bir girişi vardır ve girdilerin kendileri çok basittir.
- Her kayıt 1 KB boyutundadır, ancak yalnızca ilk 42 baytın tanımlanmış bir amacı vardır. Geri kalan baytlar, çok özel bir amaca sahip küçük veri yapıları olan öz nitelikleri (attributtları) depolar.
- Örneğin, bir öznitelik dosyanın adını depolamak için kullanılır ve başka biri dosya içeriğini depolamak için kullanılır.
- Şekilde, bazı üstbilgi bilgileri ve üç nitelik bulunduğu bir MFT girişinin temel düzenini göstermektedir.

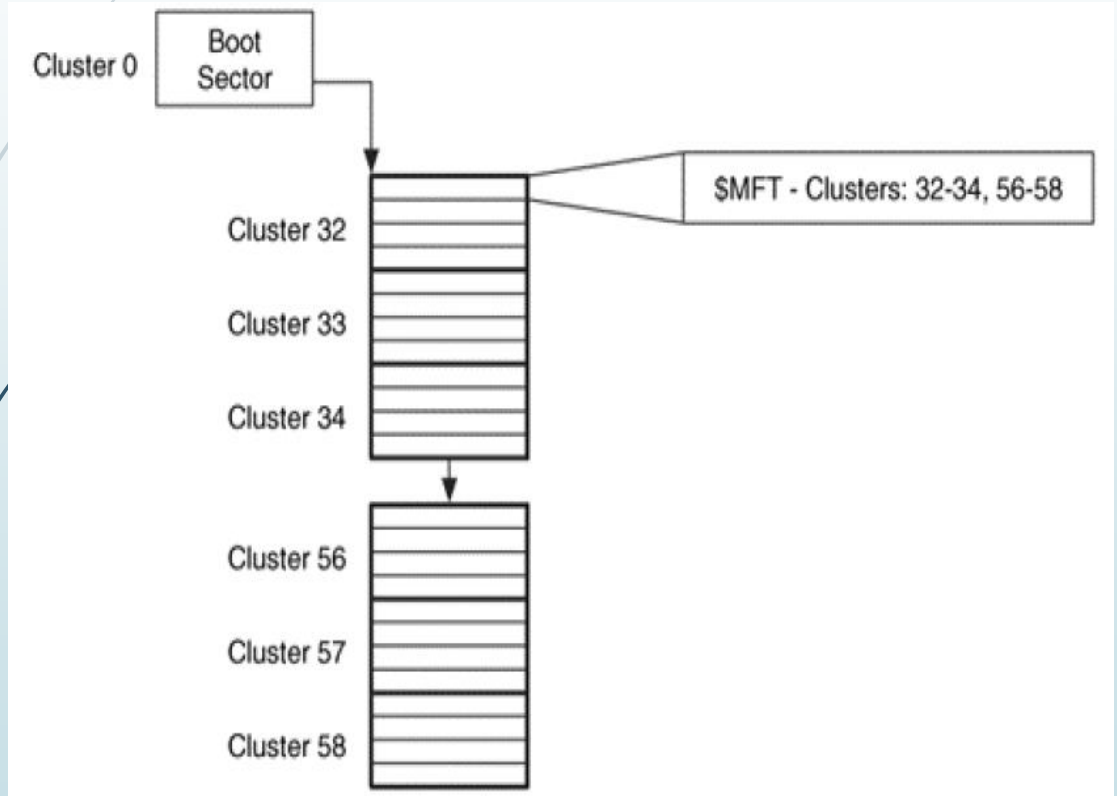
Figure 11.1. An MFT entry has a small header, and the rest of it is used to store different types of attributes. This entry has three attributes.



MFT Yapısı

- Microsoft, tablodaki her bir entriyi bir dosya kaydı olarak çağırır.
- Her bir girdiye, tablonun 0'dan başlayarak bulunduğu yere dayalı bir adres verilir.
- Tüm girdilerin boyutu 1,024 bayttır, ancak tam boyut önyükleme sektöründe tanımlanır.
- MFT bir dosyadır.
- MFT'in kendisi için bir girişi olmasıdır. MFT nin ismi $MFT'dir.
- İlk 16 Kayıt metadata dosyaları için rezerve edilmiştir.
- Tablodaki ilk girdi, $ MFT olarak adlandırılır ve MFT'nin disk üzerindeki konumunu tanımlar.
- Bu nedenle, MFT'nin düzenini ve boyutunu belirlemek için onu işlemeniz gerekir.
- MFT'nin başlangıç yeri, dosya sisteminin her zaman ilk bölümünde bulunan önyükleme sektöründe verilir.
- Önyükleme sektörü, MFT'nin parçalanmış olduğunu ve 32'den 34 ve 56'dan 58'e kümelere gittiğini gösteren ilk MFT girişini bulmak için kullanılır.
- FAT gibi, NTFS de gruplardan oluşan Ardışık sektör kümeleri kullanır..

MFT Yapısı

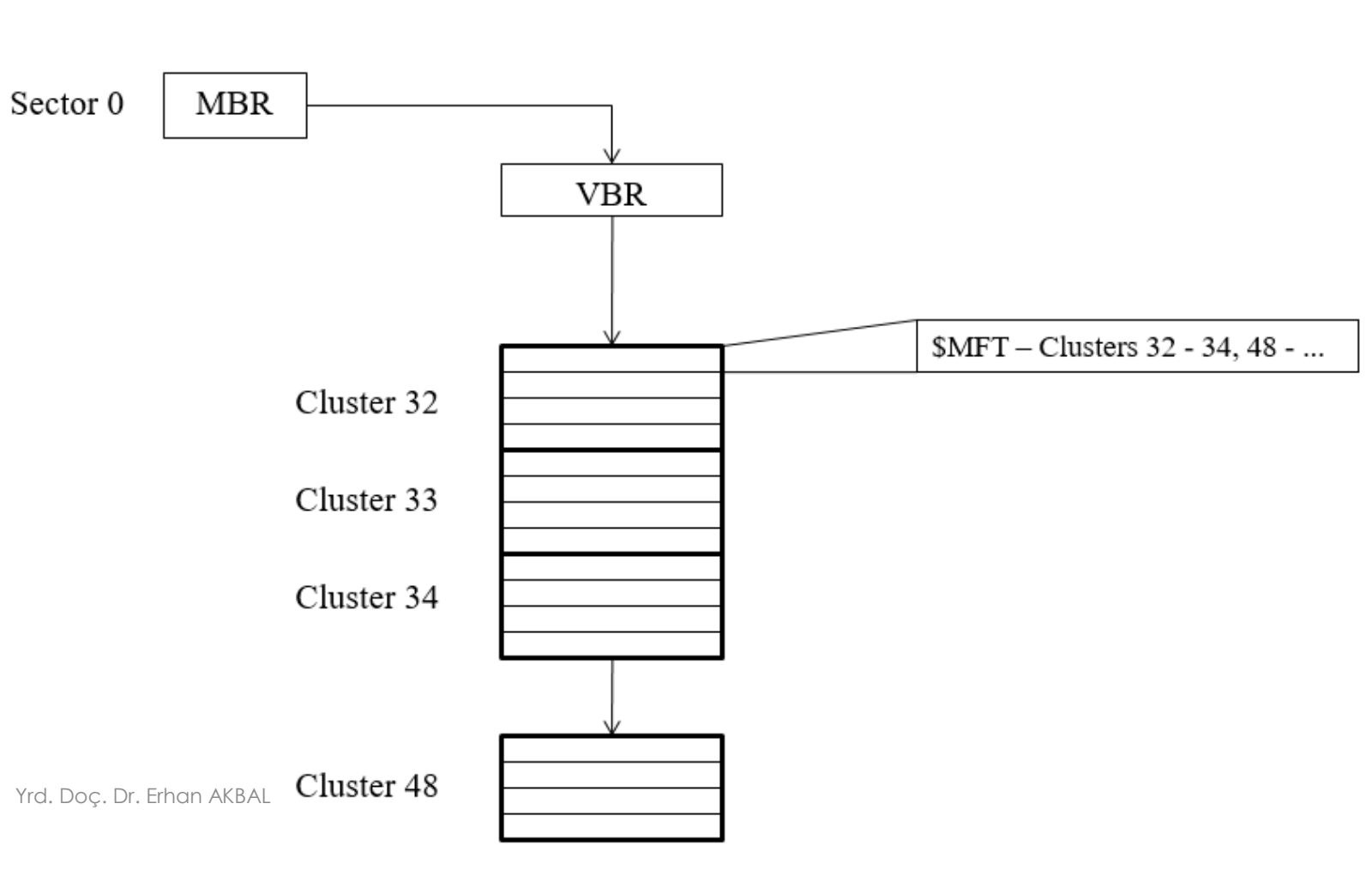


- NTFS uygulamasında MFT, mümkün olduğunca küçük başlar ve daha fazla girdi gerektiğinde genişler.
- Teorik olarak, bir işletim sistemi, dosya sistemi oluşturulduğunda sabit sayıda girdi oluşturabilir ancak Microsoft'un uygulamadaki dinamik yapısı, volüm genişlemesinden daha fazla alan eklendiğinde dosya sistemini daha kolay hale getirir.
- Microsoft, MFT girişleri oluşturulduktan sonra silmez.

MFT

- MFT bir dizi dosya kaydıdır.
- Her kayıt 1024 bayt
- MFT'deki ilk kayıt, MFT'nin kendisidir
- MFT'nin adı $ MFT'dir
- MFT'deki ilk 16 kayıt metadata dosyaları için ayrılmıştır

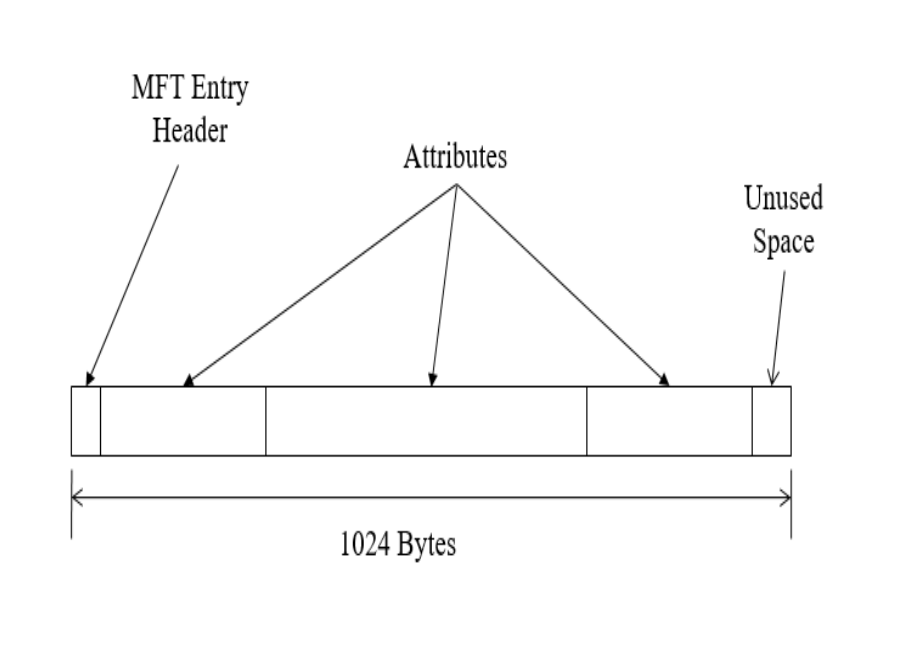
MFT


Sector 0
MBR
VBR
$MFT – Clusters 32 - 34, 48 - ...
Cluster 32
Cluster 33
Cluster 34
Cluster 48

MFT Kayıtları

- İçeriğinde aşağıdaki bilgiler bulunur
- Kayıt (entry) başlığı
- Attributes (Öznitelikler)
  - Öznitelik başlığı
  - Öznitetlik verileri
- Öznitelikler serbest formdadır
  - Sabit liste öznitelikler

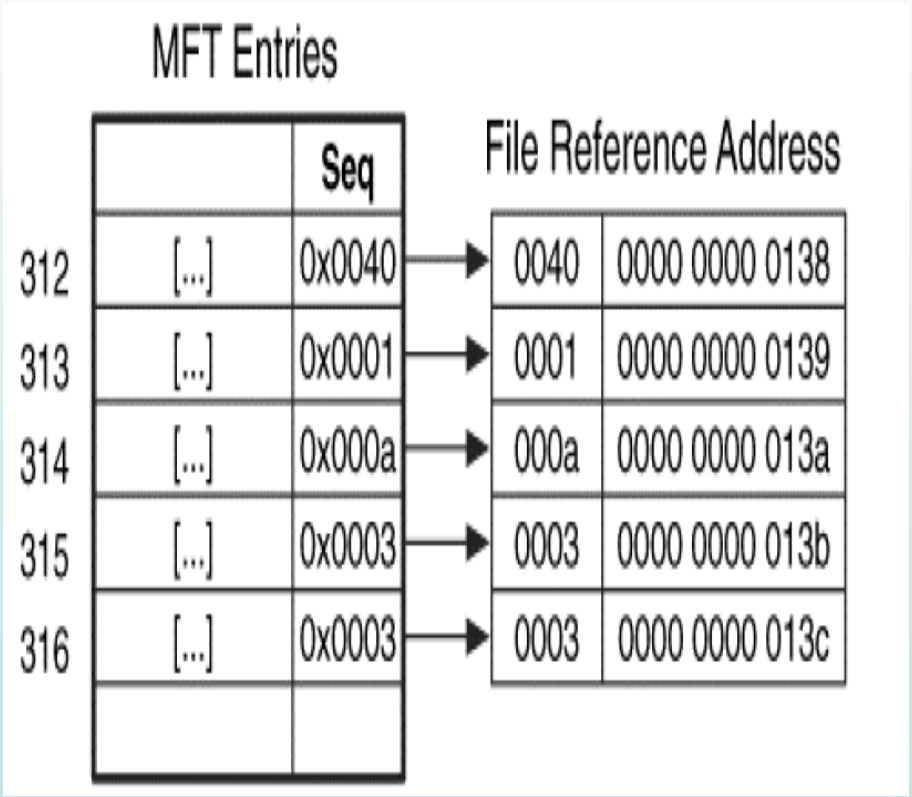
MFT Entry Yapısı



- Her MFT girişi, giriş ayrıldığında artan 16 bitlik bir sıra numarası da vardır.
- Örneğin, MFT girişi 313'ü bir sıra numarası 1 olarak düşünün. Giriş 313'ü tahsis eden dosya silinir ve giriş yeni bir dosyaya yeniden atanır.
- Giriş yeniden ayrıldığında, yeni sıra 2 numarası alır.

MFT Entry Yapısı

- Sıra numarası, dosya sistemi ne zaman bozuk bir durumda olduğunu belirlemeyi kolaylaştırdığı için NTFS, MFT girişlerine atıf yapmak için **dosya referans adresini** kullanır.
- Örneğin, bir dosya için çeşitli veri yapıları ayrılırken sistem bazı noktalarda çakılırsa, sıra numarası, önceki dosya tarafından kullanıldığından veya yeni dosyanın bir parçası olduğu için bir veri yapısının bir MFT girdi adresi içerdiğini belirleyebilir
- Silinen içeriği kurtarmaya çalışırken de kullanabiliriz. Örneğin, içinde bir dosya referans numarası olan ayrılmamış bir veri yapısına sahipsek, MFT girişinde yeniden tahsis edilip edilmediğini belirleyebiliriz.



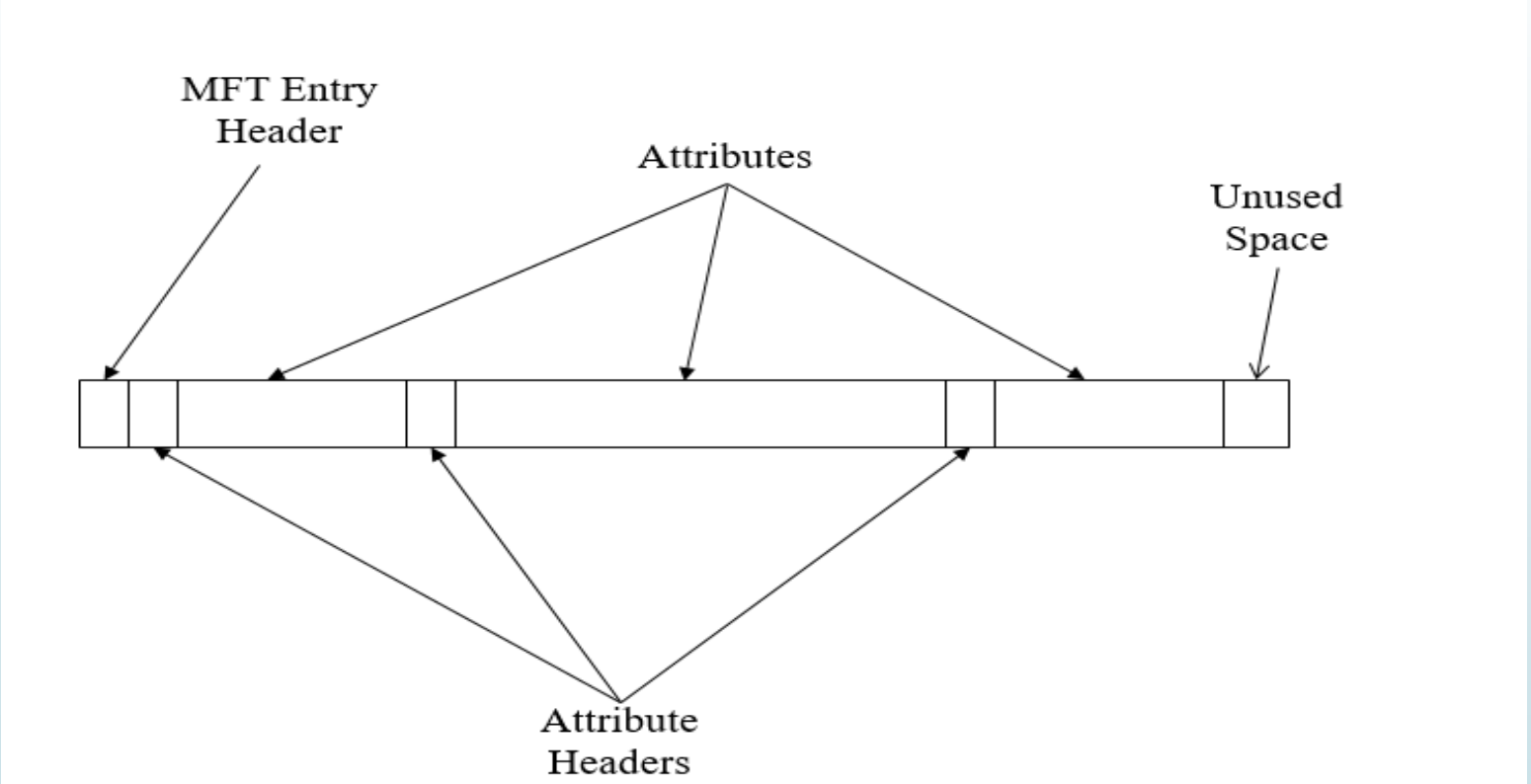
MFT Kayıtlarının Başlığı (Header)

0x0	0 – 3	Signature (“FILE”) if good otherwise (“BAAD”)	No
0x4	4 – 5	Offset to fixup array	Yes
0x6	6 – 7	Number of entries in fixup array	Yes
0x8	8 – 15	$LogFile LSN	No
0x10	16 – 17	Sequence value	No
0x12	18 – 19	Link Count	No
0x14	20 – 21	Offset to first attribute	Yes
0x16	22 – 23	Flags (in-use and directory)	Yes
0x18	24 – 27	Used size of MFT entry	Yes
0x1A	28 – 31	Allocated size of MFT entry	Yes
0x20	32 – 39	File reference to base record	No
0x28	40 – 41	Next attribute ID	No
0x2A	42 – 1023	Attributes and fixup areas	Yes

Dosya Sistemi Metadata Dosyaları

- Volümdeki her bayt bir dosyaya ayrıldığından, dosya sisteminin yönetim verilerini depolayan dosyalar olmalıdır. Microsoft bu dosyaları meta veri dosyaları olarak adlandırır.
- Microsoft, dosya sistemi meta verileri dosyaları için ilk 16 MFT entryi tutar.
- Kullanılmayan ayrılmış entriyler tahsis edilmiş bir durumda olup yalnızca temel ve genel bilgilere sahiptir.
- Genellikle çoğu kullanıcıdan gizlenmiş olsa da, her dosya sistemi meta veri dosyası kök dizinde listelenir.
- Her dosya sistemi meta veri dosyasının adı "$" ile başlar ve ilk harf büyük harf kullanılır.

MFT Öznitelik Görünümü



Standart NTFS Metadata Dosyaları

Entry	File Name	Description
0	$MFT	The entry for the MFT itself.
1	$MFTMirr	Contains a backup of the first entries in the MFT. See the "File System Category" section in Chapter 12.
2	$LogFile	Contains the journal that records the metadata transactions. See the "Application Category" section in Chapter 12.
3	$Volume	Contains the volume information such as the label, identifier, and version. See the "File System Category" section in Chapter 12.
4	$AttrDef	Contains the attribute information, such as the identifier values, name, and sizes. See the "File System Category" section in Chapter 12.
5	.	Contains the root directory of the file system. See the "File Name Category" section in Chapter 12.
6	$Bitmap	Contains the allocation status of each cluster in the file system. See the "Content Category" section in Chapter 12.
7	$Boot	Contains the boot sector and boot code for the file system. See the "File System Category" section in Chapter 12.
8	$BadClus	Contains the clusters that have bad sectors. See the "Content Category" section in Chapter 12.
9	$Secure	Contains information about the security and access control for the files (Windows 2000 and XP version only). See the "Metadata Category" section in Chapter 12.
10	$Upcase	Contains the uppercase version of every Unicode character.
11	$Extend	A directory that contains files for optional extensions. Microsoft does not typically place the files in this directory into the reserved MFT entries.

$MFT

- Giriş 0
- Ana Dosya Tablosu
- Her dosya için bir girdi içerir
- MFT'ye ilk giriş
- $ BITMAP özniteliği var
- $ DATA özniteliği, MFT tarafından kullanılan kümeleri içerir
- Ayrıca $ STANDARD_INFORMATION ve $ FILE_NAME özelliklere de sahiptir.

$MFTMirr

- Giriş 1
- MFT için yedekleme
- MFT'de ikinci giriş (giriş # 1)
  - Yerleşik olmayan bir özelliği var
- MFT'de birkaç girdi içerir
  - $ MFT, $ MFTMirr, $ LogFile, $ Hacim
- Dosya sisteminin ortasında
  - $ DATA attributte tarafından tahsis edilir
- $ MFT ile ilgili sorunlarda lazımdır
  - Dosya sisteminin orta kısmını bulur
  - "FILE" İmzaları aranır

$LogFile

- Giriş 2
- NTFS journal-günlük olarak kullanılır
- Standart niteliklere sahiptir
- Günlük verileri, $ DATA'da saklanır.

$Volume

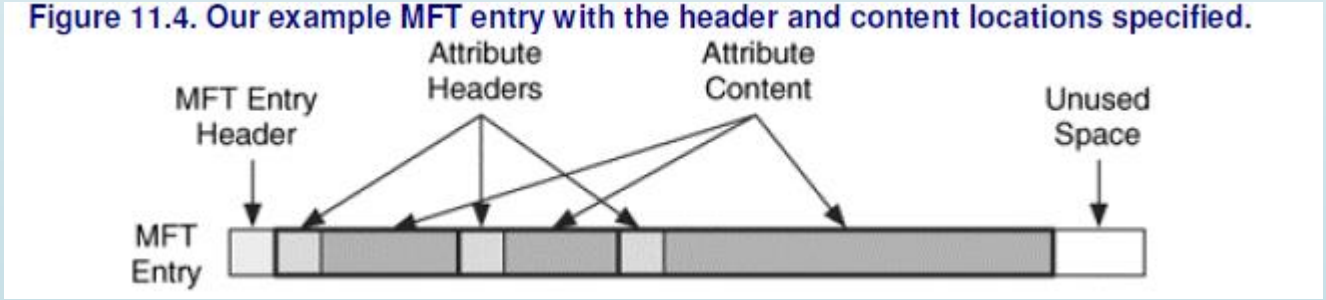
- MFT giriş numarası 3
- Volüm etiketi ve sürüm bilgisi içerir
- 2 önemli özelliği var
  - $VOLUME NAME
  - $ VOLUME_INFORMATION
- $ STD_INFO, FILE_NAME, OBJECT_ID öznitelikleri var
- $ DATA'da 0 bayt vardır

MFT Entry Attribute Konsepti

- Bir MFT girişi, iç yapısı basittir ve çoğu, belirli bir veri türünü depolayan veri yapıları olan öznitelikleri depolamak için kullanılır.
- Birçok özellik türü vardır ve her biri kendi iç yapısına sahiptir.
- Örneğin, bir dosya adı, tarih ve saati ile içerikleri için nitelikler vardır.
- Çoğu dosya sistemi, dosya içeriğini okumak ve yazmak için arar, ancak NTFS, dosya içeriğinde bulunan özelikleri okumak ve yazmak için arar.
- *Bir MFT girişini, başlangıçta boş olan büyük bir kutu olarak düşünün. Özellikler, büyük kutunun içindeki küçük kutulara benzer ve küçük kutular nesneyi en verimli biçimde depolayan herhangi bir şekil olabilir.*
- *Örneğin, şapka kısa yuvarlak bir kutuda saklanabilir ve poster uzun bir yuvarlak kutuda saklanabilir.*

MFT Entry Attribute Konsepti

- Her özellik türü farklı bir veri türü saklarken, tüm özniteliklerin iki bölümü vardır: başlık ve içerik. Şekilde, dört başlık ve içerik çiftine sahip bir MFT girişi göstermektedir.
- Başlık genel ve tüm niteliklere standarttır.
- İçerik, nitelik türüne özeldir ve herhangi bir boyut olabilir. Kutularımızın analojisini düşünürsek, her küçük kutunun dış tarafında aynı temel bilgiler olur, ancak her kutunun şekli farklı olabilir.



Attribute Başlıkları

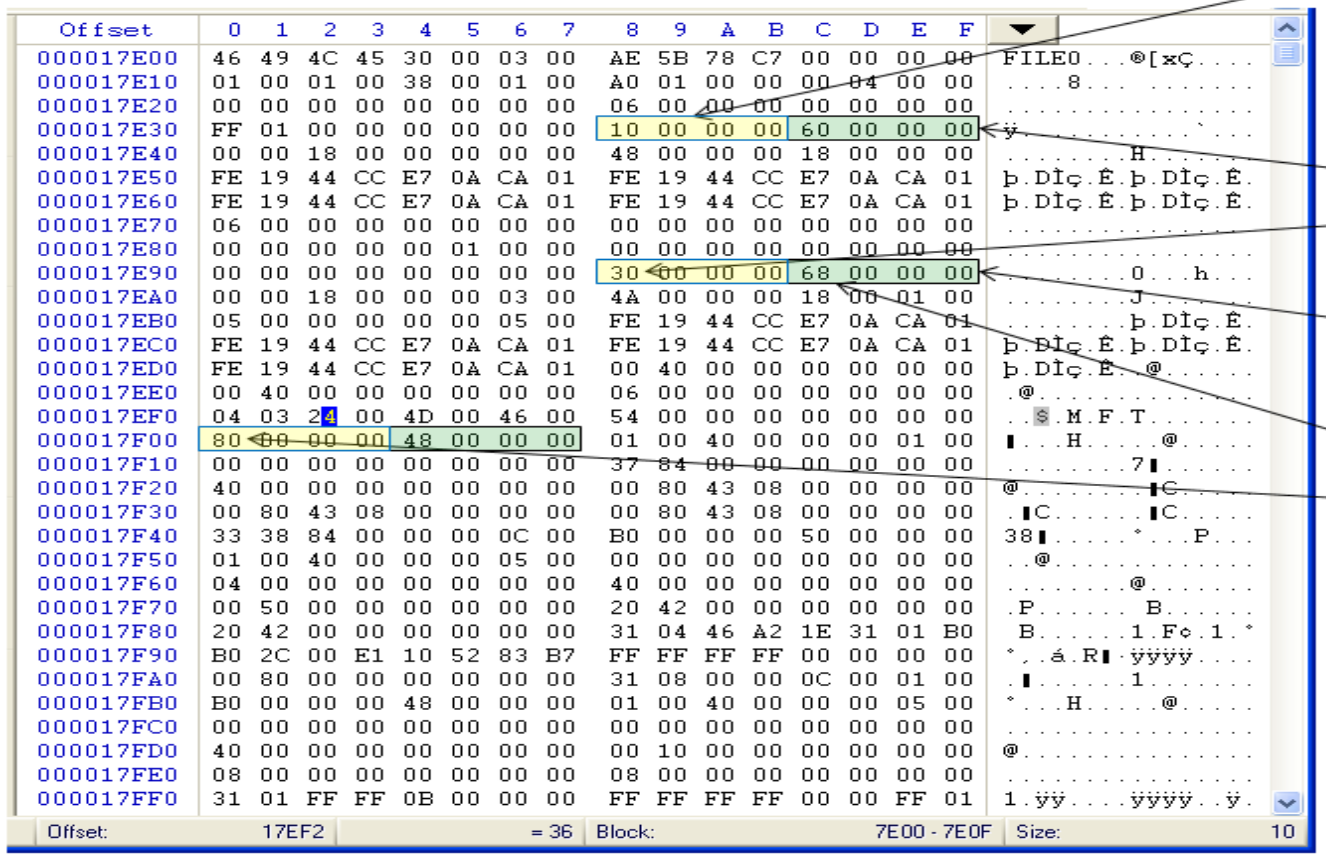
- Öznitelik başlığı, öznitelik türünü, boyutunu ve adını tanımlar.
- Ayrıca, değerin sıkıştırılıp şifrelenmediğini belirlemek için bayrakları vardır.
- Özellik türü, veri türüne dayalı sayısal bir tanımlayıcıdır.
- Bir MFT girişi, aynı türden birden fazla öznitelik içerebilir. Bazı özelliklere bir ad atanabilir ve özellik başlığında UTF-16 Unicode olarak saklanır.
- Bir öznitelikte, kendisine atanan ve bu MFT girişine özgü bir tanımlayıcı değeri vardır.
- Bir girdi aynı türden birden çok özniteliğe sahipse, bu tanımlayıcı aralarında ayrım yapmak için kullanılabilir

Attribute (Öznitelik) Başlığı Veri Yapısı

0x0	0 – 3	Attribute type identifier	Yes
0x4	4 – 7	Lenght of attribute	Yes
0x8	8 – 8	Non-resident flag	Yes
0x9	9 – 9	Length of name	Yes
0xA	10 – 11	Offset to name	Yes
0xC	12 – 13	Flags	Yes
0xE	14 – 15	Attribute identifier	Yes

Attribute Başlığı

Attribute Header



Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
000017E00	46	49	4C	45	30	00	03	00	AE	5B	78	C7	00	00	00	00
000017E10	01	00	01	00	38	00	01	00	A0	01	00	00	00	04	00	00
000017E20	00	00	00	00	00	00	00	00	06	00	00	00	00	00	00	00
000017E30	FF	01	00	00	00	00	00	00	10	00	00	00	60	00	00	00
000017E40	00	00	18	00	00	00	00	00	48	00	00	00	18	00	00	00
000017E50	FE	19	44	CC	E7	0A	CA	01	FE	19	44	CC	E7	0A	CA	01
000017E60	FE	19	44	CC	E7	0A	CA	01	FE	19	44	CC	E7	0A	CA	01
000017E70	06	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000017E80	00	00	00	00	00	01	00	00	00	00	00	00	00	00	00	00
000017E90	00	00	00	00	00	00	00	00	30	00	00	00	68	00	00	00
000017EA0	00	00	18	00	00	00	03	00	4A	00	00	00	18	00	01	00
000017EB0	05	00	00	00	00	00	05	00	FE	19	44	CC	E7	0A	CA	01
000017EC0	FE	19	44	CC	E7	0A	CA	01	FE	19	44	CC	E7	0A	CA	01
000017ED0	FE	19	44	CC	E7	0A	CA	01	00	40	00	00	00	00	00	00
000017EE0	00	40	00	00	00	00	00	00	06	00	00	00	00	00	00	00
000017EF0	04	03	24	00	4D	00	46	00	54	00	00	00	00	00	00	00
000017F00	80	00	00	00	48	00	00	00	01	00	40	00	00	00	01	00
000017F10	00	00	00	00	00	00	00	00	37	84	00	00	00	00	00	00
000017F20	40	00	00	00	00	00	00	00	00	80	43	08	00	00	00	00
000017F30	00	80	43	08	00	00	00	00	00	80	43	08	00	00	00	00
000017F40	33	38	84	00	00	00	0C	00	B0	00	00	00	50	00	00	00
000017F50	01	00	40	00	00	00	05	00	00	00	00	00	00	00	00	00
000017F60	04	00	00	00	00	00	00	00	40	00	00	00	00	00	00	00
000017F70	00	50	00	00	00	00	00	00	20	42	00	00	00	00	00	00
000017F80	20	42	00	00	00	00	00	00	31	04	46	A2	1E	31	01	B0
000017F90	B0	2C	00	E1	10	52	83	B7	FF	FF	FF	FF	00	00	00	00
000017FA0	00	80	00	00	00	00	00	00	31	08	00	00	0C	00	01	00
000017FB0	B0	00	00	00	48	00	00	00	01	00	40	00	00	00	05	00
000017FC0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
000017FD0	40	00	00	00	00	00	00	00	00	10	00	00	00	00	00	00
000017FE0	08	00	00	00	00	00	00	00	08	00	00	00	00	00	00	00
000017FF0	31	01	FF	FF	0B	00	00	00	FF	FF	FF	FF	00	00	FF	01

Offset: 17EF2 = 36 Block: 7E00 - 7E0F Size: 10

Beginning of the first attribute.
Type = 0x10

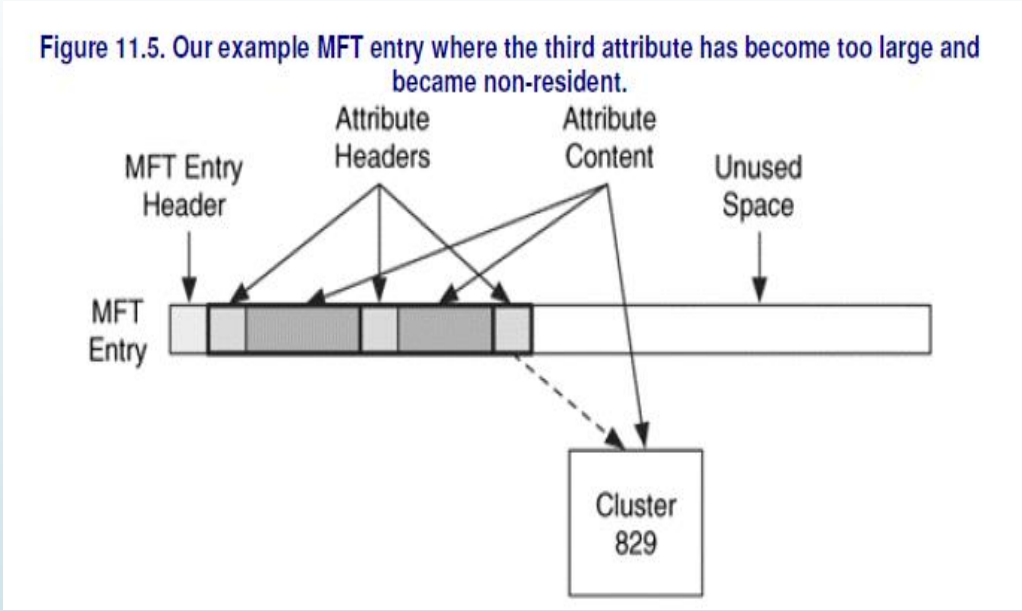
Length of the attribute
= 0x60
Offset to next attribute

Beginning of the next attribute.
Type = 0x30

Length of this attribute
= 0x68
Offset to next attribute

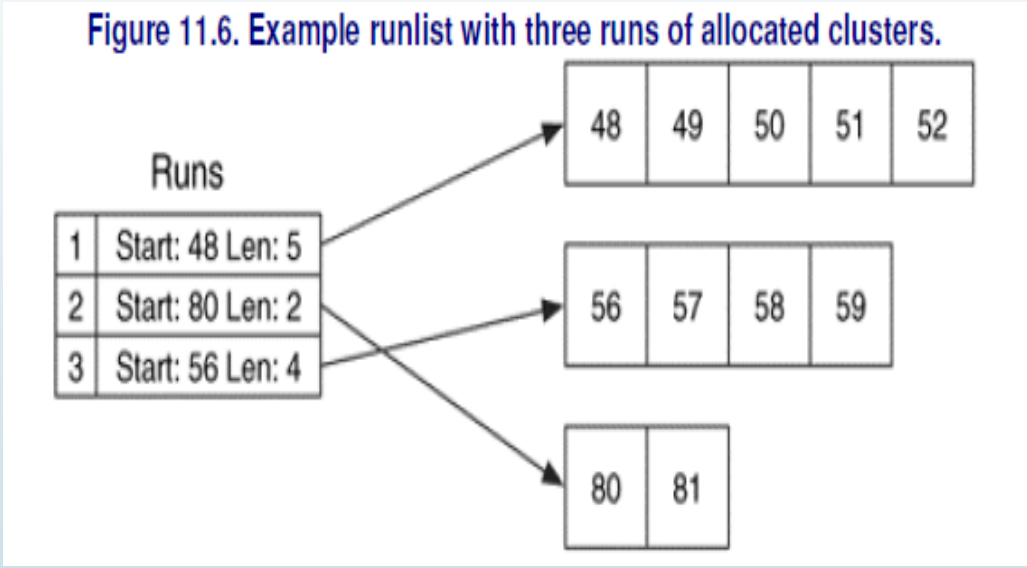
Attribute İçerikleri

- Öznitelik içeriği, herhangi bir biçimde ve herhangi bir boyuta sahip olabilir.
- Örneğin, bir dosyanın içeriğini depolamak için özelliklerden biri kullanılır; bu nedenle, birkaç MB veya GB boyutunda olabilir. Bu miktarda veriyi yalnızca 1,024 bayt olan bir MFT girişinde depolamak pratik değildir.
- Bu sorunu çözmek için NTFS, öznitelik içeriğinin depolanabileceği iki konum sağlar.
- Yerleşik bir özellik, içeriğini öznitelik başlığına sahip MFT girişinde saklar. Bu yalnızca küçük özellikler için geçerlidir. Yerleşik olmayan bir özellik, içeriğini dosya sistemindeki harici bir kümede depolar.
- Özniteliğin başlığı, özelliğin yerleşik veya yerleşik olmayan olup olmadığını tanımlar. Bir özellik varsa, içerik hemen başlığı takip edecektir. Özellik geçersiz ise, üstbilgi küme adreslerini verecektir.
- Şekil 11.5'te, daha önce gördüğümüz MFT girişi örneğini görüyoruz, ancak şimdi üçüncü özniteliği MFT'ye sığmayacak kadar büyük ve küme 829'u tahsis etmiştir.



Attribute İçerikleri

- Yerleşik olmayan nitelikler, ardışık kümeler olan **cluster runs** larda saklanır ve run, başlangıç küme adresini ve çalışma uzunluğunu kullanarak belgelenir.
- Örneğin, bir öznitelik 48, 49, 50, 51 ve 52 kümelerini tahsis etmişse, küme 48'de 5 uzunluğunda başlayan bir run vardır.
- Öznitelik ayrıca 80 ve 81 kümeleri ayrılmışsa, ikinci bir run Küme 80'de 2 uzunluğu ile başlar.
- Üçüncü küme 56 kümesinde başlayabilir ve 4 uzunluğa sahip olabilir.



NTFS Adresleme Yapısı

- NTFS adresler için farklı terimler kullanır.
- Mantıksal Küme Numarası (LCN), mantıksal dosya sistemi adresi ile aynıdır ve Sanal Küme Numarası (VCN) mantıksal bir dosya adresi ile aynıdır.
- NTFS, yerleşik olmayan öznitelik çalıştırmalarını tanımlamak için VCN'den LCN'ye eşlemeler kullanır.
- Önceki örneğe dönersek, bu özniteliğin çalışması, 0-4 arası bir adresin 48'den 52'ye LCN adreslerine, VCN adreslerinin 5'den 6'ya LCN adreslerine 80'den 81'e, VCN adreslerinin 7'den 10'a LCN adreslerine 56 eşleme yaptığını gösterir.

Standart Attribute Tipleri

- Her öznitelik türü için bir sayı tanımlanmıştır ve Microsoft bu girdiyi kullanarak bir girdideki öznitelikleri sıralar. Standart niteliklerin kendilerine atanmış bir varsayılan tür değeri vardır.
- Bir sayıya ek olarak, her özellik türü bir ada sahiptir ve tüm büyük harflere sahiptir ve "$" ile başlar.

Type Identifier	Name	Description
16	$STANDARD_INFORMATION	General information, such as flags; the last accessed, written, and created times; and the owner and security ID.
32	$ATTRIBUTE_LIST	List where other attributes for file can be found.
48	$FILE_NAME	File name, in Unicode, and the last accessed, written, and created times.
64	$VOLUME_VERSION	Volume information. Exists only in version 1.2 (Windows NT).
64	$OBJECT_ID	A 16-byte unique identifier for the file or directory. Exists only in versions 3.0+ and after (Windows 2000+).
80	$SECURITY_ DESCRIPTOR	The access control and security properties of the file.
96	$VOLUME_NAME	Volume name.
112	$VOLUME_ INFORMATION	File system version and other flags.
128	$DATA	File contents.
144	$INDEX_ROOT	Root node of an index tree.
160	$INDEX_ALLOCATION	Nodes of an index tree rooted in $INDEX_ROOT attribute.
176	$BITMAP	A bitmap for the $MFT file and for indexes.
192	$SYMBOLIC_LINK	Soft link information. Exists only in version 1.2 (Windows NT).
192	$REPARSE_POINT	Contains data about a reparse point, which is used as a soft link in version 3.0+ (Windows 2000+).
208	$EA_INFORMATION	Used for backward compatibility with OS/2 applications (HPFS).
224	$EA	Used for backward compatibility with OS/2 applications (HPFS).
256	$LOGGED_UTILITY_STREAM	Contains keys and information about encrypted attributes in version 3.0+ (Windows 2000+).

Standart Attribute Tipleri

- Hemen hemen her ayrılmış MFT girişi bir **$FILE_NAME** ve bir **$STANDARD_INFORMATION** türü özniteliğine sahiptir.
- **$FILE_NAME** özniteliği, dosya adı, boyut ve zamansal bilgileri içerir.
- **$STANDARD_INFORMATION** özniteliği geçici, sahiplik ve güvenlik bilgilerini içerir.
- Her dosya, dosya içeriğini içeren bir **$DATA** özniteliğine sahiptir. İçerik yaklaşık olarak 700 bayttan büyükse, yerleşik olmayacak ve harici kümelerde saklanacaktır. Bir dosyada birden fazla **$DATA** özniteliği varsa, ek özniteliklere bazen alternatif veri akışları (ADS) adı verilir.
- Bir dosya oluşturulduğunda oluşturulan varsayılan **$DATA** özniteliğinin kendisiyle ilişkili bir adı yoktur, ancak ek $DATA özniteliklerinin bir tane olması gerekir.

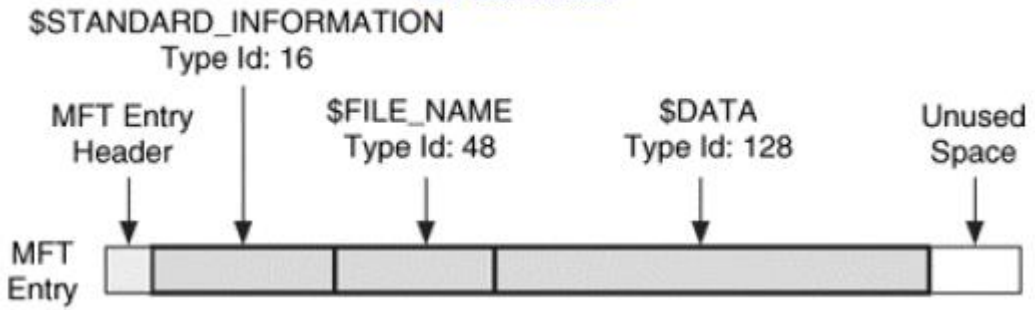
Standart Attribute Tipleri

- Her dizinin içinde bulunan dosyalar ve alt dizinler hakkında bilgi içeren bir **$INDEX_ROOT** özniteliği vardır.
- **$INDEX_ROOT** Dizin büyükse, bilgileri depolamak için **$INDEX_ALLOCATION ve $BITMAP** nitelikleri de kullanılır.
- İşleri kafa karıştırmak için, bir dizinin özniteliğine ek olarak bir $DATA özniteliğine sahip olması mümkündür. Başka bir deyişle, bir dizin hem dosya içeriğini hem de dosya ve alt dizinlerinin bir listesini saklayabilir.
- **$DATA** özniteliği, bir uygulamanın veya kullanıcının orada saklamak istediği içeriği saklayabilir. Bir dizinin **$INDEX_ROOT** ve **$INDEX_ALLOCATION** niteliklerinin genelde **"$I30"** adı vardır.

Örnek Yapı

- Üç standart dosya özniteliklerine sahiptir. Bu örnekte, tüm özellikler yerleşiktir.

Figure 11.7. Our example MFT entry where the type names and identifiers have been added to the attributes.



MFT

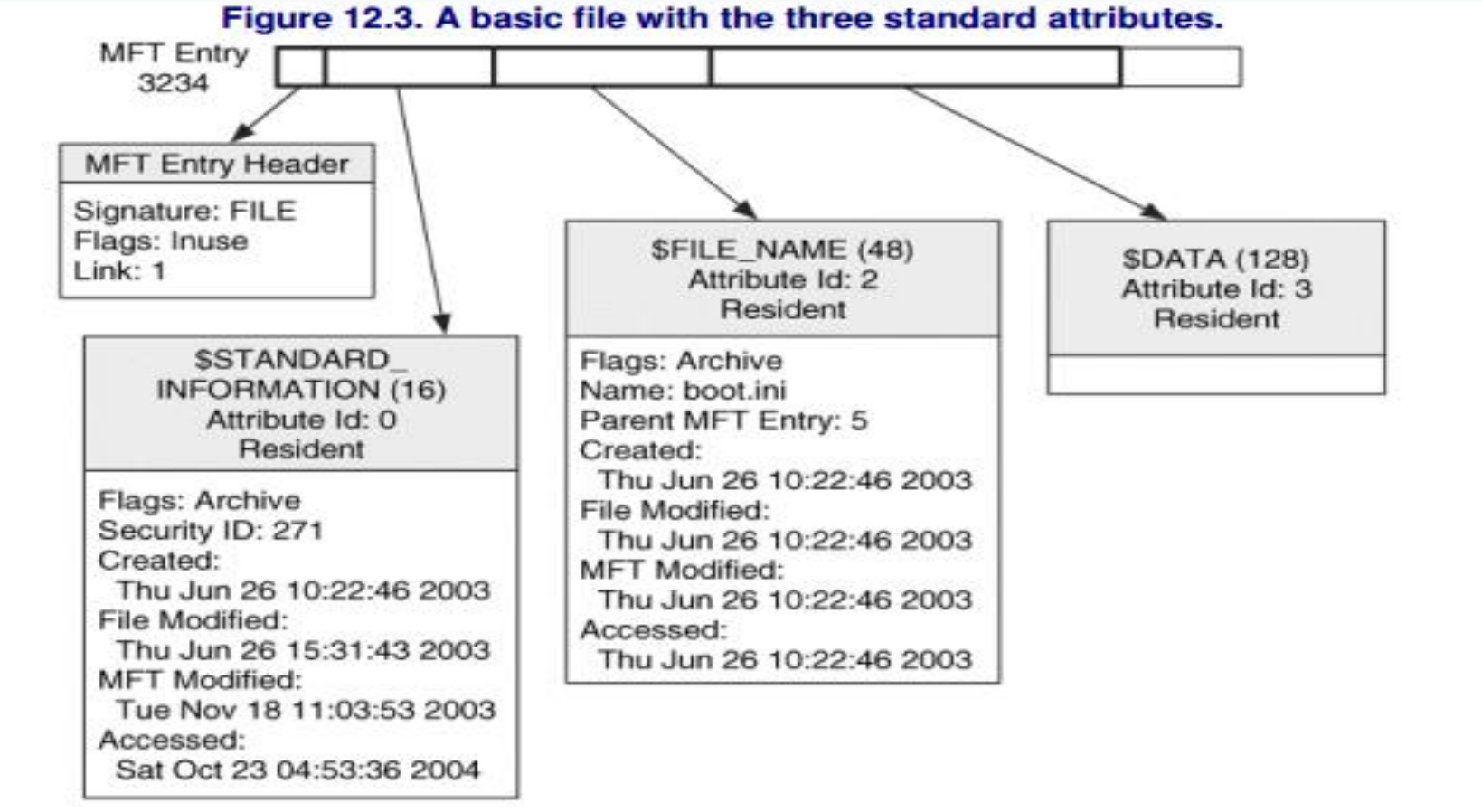


Figure 12.3. A basic file with the three standard attributes.
MFT Entry
3234
MFT Entry Header
Signature: FILE
Flags: Inuse
Link: 1
$STANDARD_
INFORMATION (16)
Attribute Id: 0
Resident
Flags: Archive
Security ID: 271
Created:
Thu Jun 26 10:22:46 2003
File Modified:
Thu Jun 26 15:31:43 2003
MFT Modified:
Tue Nov 18 11:03:53 2003
Accessed:
Sat Oct 23 04:53:36 2004
$FILE_NAME (48)
Attribute Id: 2
Resident
Flags: Archive
Name: boot.ini
Parent MFT Entry: 5
Created:
Thu Jun 26 10:22:46 2003
File Modified:
Thu Jun 26 10:22:46 2003
MFT Modified:
Thu Jun 26 10:22:46 2003
Accessed:
Thu Jun 26 10:22:46 2003
$DATA (128)
Attribute Id: 3
Resident

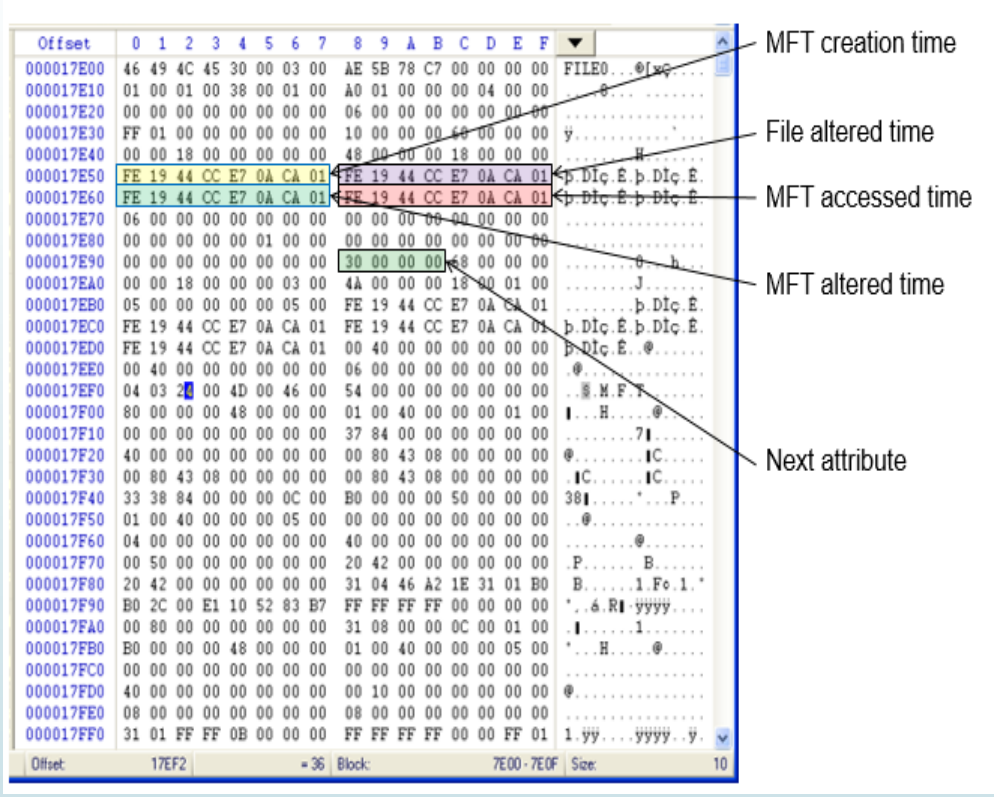
$STANDARD_INFORMATION Attribute

- $STANDARD_INFORMATION nitelik tüm dosyaları ve dizinleri için var ve çekirdek meta verileri içerir.
- Saat ve tarih damgaları kümesi ve ayrıca sahibi, güvenlik ve kota bilgileri burada bulunur.
- Bu özelliğin varsayılan türü kimliği 16 olduğunu ve Windows 2000 ve XP sürümlerinde 72 byte ve Windows NT sürüm 48 bayt statik boyutu vardır.
- Microsoft, bir MFT girişinde özelliklerini sıralar.
- Bu özellikte dört tarih ve zaman damgaları vardır.
- NTFS tarih ve saat damgaları size on sekizinci yüzyılda kullanılan bir bilgisayarı araştırmak için gerekirse yararlıdır
- 1 Ocak değeri, 1970 UTC 116.444.736.000.000.000 dir.

- **Oluşturulma Zamanı:** Dosyanın oluşturulduğu zaman.
- **Modifiye Zaman:** $ VERİLERİN içeriği veya $ ENDEKSİ son düzenleme niteliklerini saati.
- **MFT Modifiye Zamanı:** Dosyanın metadata verilerinin son değiştirildiği zaman.
- **Erişim Zamanı:** dosyanın içeriğine son kez erişilen zaman.

$STANDARD_INFORMATION Attribute

0x0	0 – 7	Creation time
0x8	8 – 15	File altered time
0x10	16 – 23	MFT altered time - not shown in file properties
0x18	24 – 31	File accessed time
0x20	32 – 35	Flags
0x26	36 -39	Maximum number of versions
0x2A	40 – 43	Version number
0x2C	44 – 47	Class ID
0x30	48 -51	Owner ID
0x34	52 – 55	Security ID
0x38	56 – 63	Quota charged
0x40	64 – 71	Update Sequence Number(USN)



$FILE_NAME Attribute

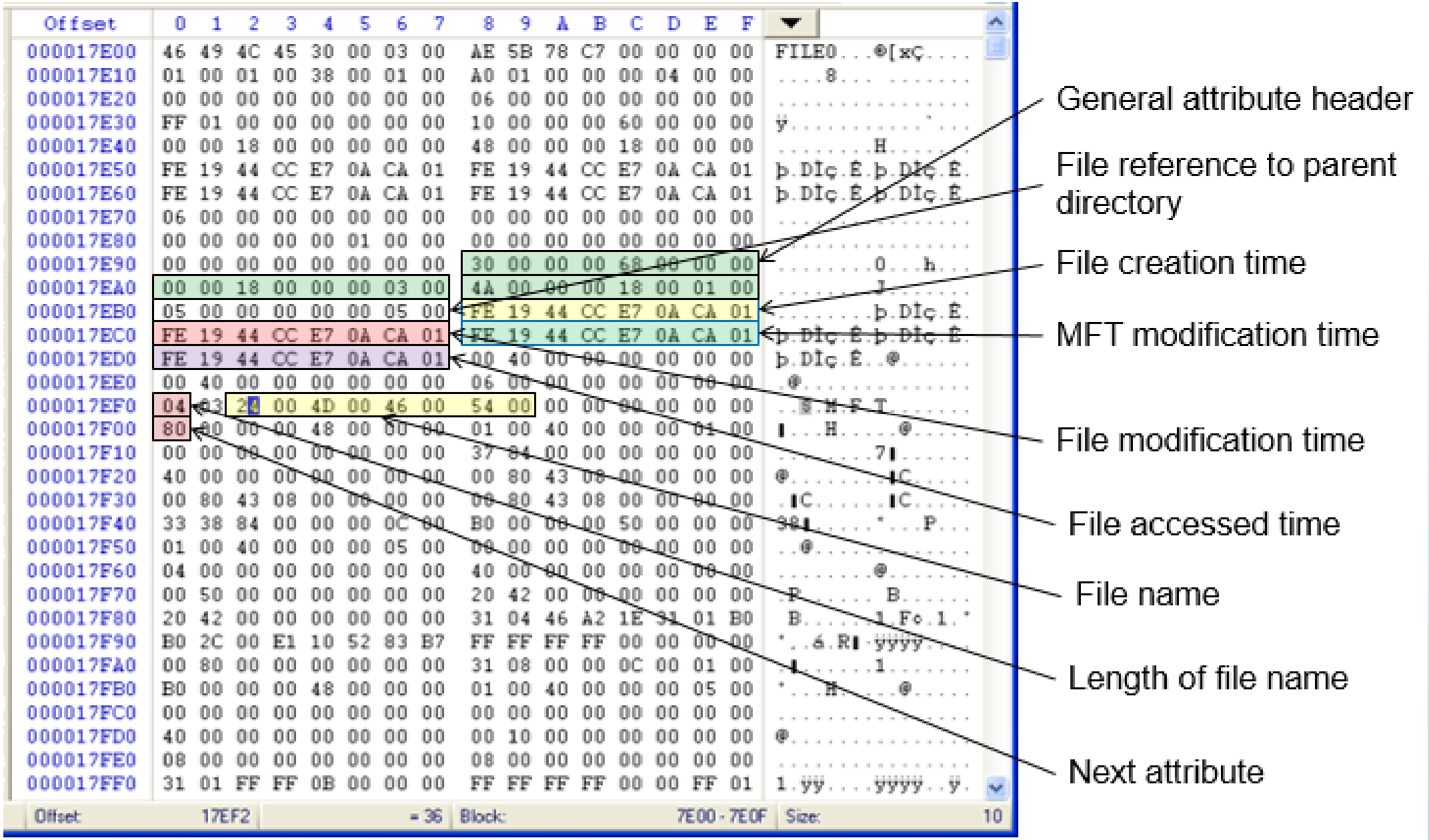
- Her dosya ve dizin için MFT girişinde en az bir $ FILE_NAME niteliği vardır.
- Özellik, dosya adının uzunluğuna bağlıdır 48 bit türü bir tanımlayıcı ve değişken bir uzunluğa sahiptir.
- $ FILE_NAME nitelik UTF-16 Unicode olarak kodlanmış dosyanın adını içerir.

$FILE_NAME Attribute

- Tür Tanımlayıcı - 48 (0x30)
- Dosyanın adını saklar
- Ana Dizin
- Dizin dizini
- Standart dosyalar veya dizinler için $ FILE_NAME ikinci niteliğidir ve ikamet eder
- Bir dosya birden fazla MFT girişi gerektiriyorsa $ ATTRIBUTE_LIST ikinci sıradadır

$FILE_NAME Attribute

0x0	0 – 7	File reference of a parent directory
0x8	8 – 15	File Creation time
0x10	16 -23	File modification time
0x18	24 – 31	MFT modification time - not shown in file properties
0x20	32 – 39	File access time
0x28	40 – 47	Allocated size of file
0x30	48 – 55	Real size of file
0x38	56 – 59	Flags (same as $STANDARD_INFORMATION flags)
0x3C	60 – 63	Reparse value
0x40	64 – 64	Lengthe of name
0x41	65 – 65	Namespace
0x42	66+	Name


Offset 0 1 2 3 4 5 6 7 8 9 A B C D E F
000017E00 46 49 4C 45 30 00 03 00 AE 5B 78 C7 00 00 00 00 FILE0...®[xÇ....
000017E10 01 00 01 00 38 00 01 00 A0 01 00 00 00 04 00 00 ....8... ....
000017E20 00 00 00 00 00 00 00 00 06 00 00 00 00 00 00 00 ....
000017E30 FF 01 00 00 00 00 00 00 10 00 00 00 60 00 00 00 ÿ.........`...
000017E40 00 00 18 00 00 00 00 00 48 00 00 00 18 00 00 00 ....H.......
000017E50 FE 19 44 CC E7 0A CA 01 FE 19 44 CC E7 0A CA 01 þ.DÌç.Ê.þ.DÌç.Ê.
000017E60 FE 19 44 CC E7 0A CA 01 FE 19 44 CC E7 0A CA 01 þ.DÌç.Ê.þ.DÌç.Ê.
000017E70 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
000017E80 00 00 00 00 00 01 00 00 00 00 00 00 00 00 00 00 ....
000017E90 00 00 00 00 00 00 00 00 30 00 00 00 68 00 00 00 ....0...h...
000017EA0 00 00 18 00 00 00 03 00 4A 00 00 00 18 00 01 00 ....J.......
000017EB0 05 00 00 00 00 00 05 00 FE 19 44 CC E7 0A CA 01 ....þ.DÌç.Ê.
000017EC0 FE 19 44 CC E7 0A CA 01 FE 19 44 CC E7 0A CA 01 þ.DÌç.Ê.þ.DÌç.Ê.
000017ED0 FE 19 44 CC E7 0A CA 01 00 40 00 00 00 00 00 00 þ.DÌç.Ê..@......
000017EE0 00 40 00 00 00 00 00 00 06 00 00 00 00 00 00 00 .@..............
000017EF0 04 03 24 00 4D 00 46 00 54 00 00 00 00 00 00 00 ..$.M.F.T.......
000017F00 80 00 00 00 48 00 00 00 01 00 40 00 00 00 01 00 ....H.....@.....
000017F10 00 00 00 00 00 00 00 00 37 84 00 00 00 00 00 00 ....7.......
000017F20 40 00 00 00 00 00 00 00 00 80 43 08 00 00 00 00 @.........C.....
000017F30 00 80 43 08 00 00 00 00 00 80 43 08 00 00 00 00 ..C.......C.....
000017F40 33 38 84 00 00 00 0C 00 B0 00 00 00 50 00 00 00 38.....°...P...
000017F50 01 00 40 00 00 00 05 00 00 00 00 00 00 00 00 00 ..@.............
000017F60 04 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00 ....@.......
000017F70 00 50 00 00 00 00 00 00 20 42 00 00 00 00 00 00 .P...... B......
000017F80 20 42 00 00 00 00 00 00 31 04 46 A2 1E 91 01 B0 B.....1.F¢.1.°
000017F90 B0 2C 00 E1 10 52 83 B7 FF FF FF FF 00 00 00 00 °.,.á.R·ÿÿÿÿ....
000017FA0 00 80 00 00 00 00 00 00 31 08 00 00 0C 00 01 00 ....1.......
000017FB0 B0 00 00 00 48 00 00 00 01 00 40 00 00 00 05 00 °...H.....@.....
000017FC0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 ....
000017FD0 40 00 00 00 00 00 00 00 00 10 00 00 00 00 00 00 @...............
000017FE0 08 00 00 00 00 00 00 00 08 00 00 00 00 00 00 00 ....
000017FF0 31 01 FF FF 0B 00 00 00 FF FF FF FF 00 00 FF 01 1.ÿÿ....ÿÿÿÿ..ÿ.
Offset: 17EF2 = 36 Block: 7E00 - 7E0F Size: 10
General attribute header
File reference to parent directory
File creation time
MFT modification time
File modification time
File accessed time
File name
Length of file name
Next attribute

$FILE_NAME - Namespace

1. Posix: '/' ve NULL haricinde tüm Unicode karakterleri büyük / küçük harfe duyarlı
2. Win32: '/', '\', ':', '<', '>' ve '?' Haricinde tüm Unicode karakterler büyük / küçük harfe duyarlı
3. DOS: Büyük / küçük harf duyarlı değil, büyük harf ve hiçbir özel karakter yok
4. Win32 ve DOS: Özgün ad zaten DOS ad alanına sığdığında ve iki ismin gerekli olmadığı durumlarda kullanılır

$DATA - Attribute

- Tür Kimliği - 128 (0x80)
- Genel öznitelik başlık alanlarına sahiptir
- İlk $ DATA özniteliğinin bir adı yok
- Alternatif Veri Akımları için ek $ DATA öznitelikleri kullanılabilir ve her biri bir ada sahip olmalıdır.
  C: \> "Merhaba dünya"> echo "file.txt: stuff
- İçeriğin 700 bayttan büyük boyuta çıkması halinde yerleşik olmayacak
- Dizinler $DATA niteliklerine sahip olabilir

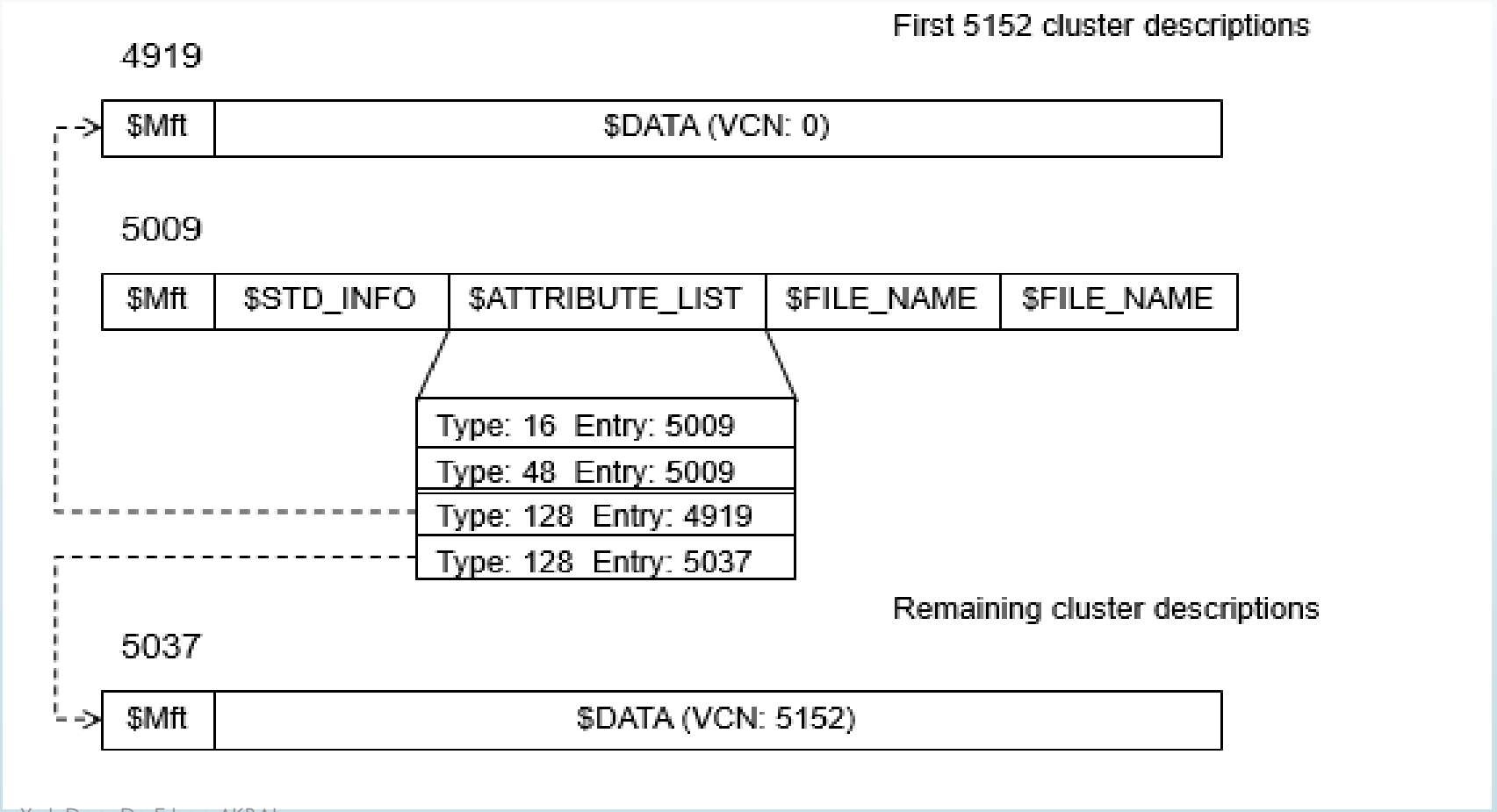
$ATTRIBUTE_LIST - Attribute

- Tür Kimliği - 32 (0x20)
- Bir MFT'ye sığabildiğinden daha fazla özellik varsa kullanılır
- Diğer özniteliklerin nerede bulunabileceğinin bir listesini içerir
- Listedeki her girişin, her öznitelikte ortak olan standart alanlara ek olarak 7 alan vardır

$ATTRIBUTE_LIST Yapısı

0x0	0 – 3	Attribute type
0x4	4- 5	Length of this entry
0x6	6 – 6	Length of name of this attribute
0x7	7 – 7	Offset to name (relative to start of this entry)
0x8	8 – 15	Starting VCN in attribute
0x10	16 – 23	File reference where attribute is located
0x18	24 – 24	Attribute ID

Örnek Yapı


First 5152 cluster descriptions
4919
$Mft
$DATA (VCN: 0)
5009
$Mft
$STD_INFO
$ATTRIBUTE_LIST
$FILE_NAME
$FILE_NAME
Type: 16 Entry: 5009
Type: 48 Entry: 5009
Type: 128 Entry: 4919
Type: 128 Entry: 5037
Remaining cluster descriptions
5037
$Mft
$DATA (VCN: 5152)

$OBJECT_ID

- Tür Kimliği - 64 (0x40)
- Dosyanın 128 bit Global Object Identifier
- Dosya adı yerine kullanılır
- Dosya adı değişikliği ile sabit kalır
- $Volume metadata dosyasının bir $OBJECT_ID özniteliği var

$OBJECT_ID Yapısı

0x0	0 – 15	Object ID
0x10	16 – 31	Birth volume ID
0x20	32 – 47	Birth object ID
0x40	48 – 63	Birth Domain ID

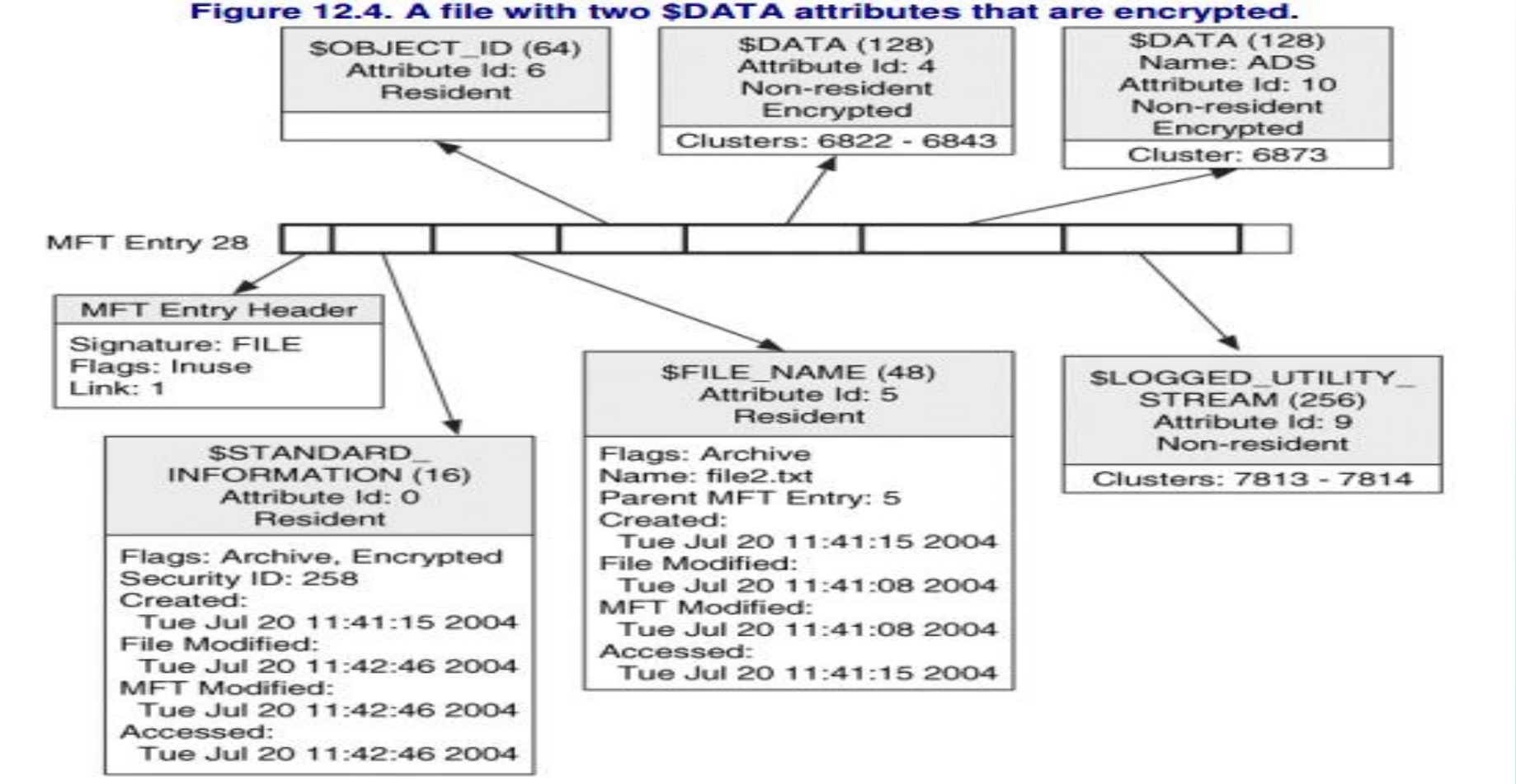
$REPARSE_POINT

- Tür Kimliği - 192 (0xC0)
- Yeniden gözden geçirme noktaları olan dosyalar için kullanılır
  - Sembolik linkler
  - Bağlantı noktaları
  - Volümler için mount noktaları
- Çoğu özellik alanı, uygulamaya özeldir.

Diğer Öznitelikler

- **80(0x50) $SECURITY_DESCRIPTOR**
  - Dosyanın Erişim Kontrolü ve Güvenlik Özellikleri için
- **96(0x60) $VOLUME_VERSION**
  - Volume ismi
- **112(0x70) $VOLUME_INFORMATION**
  - Dosya Sistem versiyonu ve diğer flaglar
- **144(0x90) $INDEX_ROOT**
  - İndex ağacının kök düğümü
- **160(0xA0) $INDEX_ALLOCATION**
  - $ INDEX_ROOT özniteliğinde kök dizin ağacının düğümleri
- **176(0xB0) $BITMAP**
  - $ MFT dosyası ve dizinler için bir bitmap
- **192(0xC0) $SYMBOLIC_LINK**
  - Windows NT sürüm 1.2 ve daha küçük sistemler için bağlantı bilgisi.
- **208(0xD0) $EA_INFORAMTION**
  - Sürüm 1.2 uygulamaları ile geriye dönük uyumluluk için kullanılır (HPFS)
- **224(0xE0) $EA**
  - Sürüm 1.2 uygulamaları ile geriye dönük uyumluluk için kullanılır (HPFS)
- **256(0xF0) $LOGGED_UTILTIY_STREAM**
  - Anahtarlar ve şifrelenmiş öznitelikler hakkında 3.0+ sürümünde bilgi içerir

2 $DATA Özellikli bir dosya



İndeksler

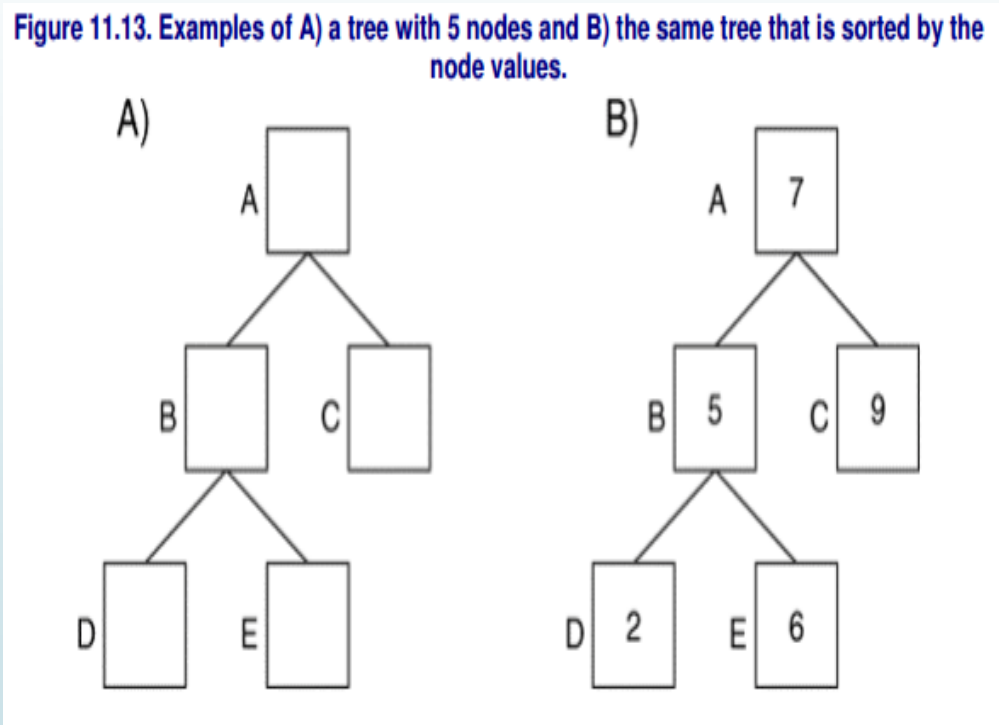
- NTFS, birçok durumda indeks veri yapılarını kullanır.
- NTFS'deki bir indeks, sıralanmış bir siparişte depolanan özniteliklerin toplamıdır.
- İndeksin en yaygın kullanımı bir dizinde bulunur, çünkü dizinler bir $FILE_NAME öznitelikleri içerir.
- NTFS'nin (Windows 2000 ile birlikte gelen) sürüm 3.0'dan önce bir indekste yalnızca $FILE_NAME özniteliği vardı ancak şimdi indeksleri birkaç kullanımı var ve bunlar farklı öznitelikler içeriyor.
- Örneğin, güvenlik bilgileri, kota bilgisi olduğu gibi bir indekste saklanır.

İndeks Öznitelikleri ve Veri Yapıları

- $INDEX_ROOT özniteliğinde Ağacın Kökü saklanır.
- $INDEX_ALLOCATION özniteliğinde geri kalan düğümler saklanır.
- $BITMAP özniteliği tahsis durumlarını yönetmek için kullanılır.

B-Trees

- Bir NTFS dizini, öznitelikleri bir ağaca, özellikle bir B-Tree lerde sıralar.
- Ağaç, bir kafa düğümü olduğu ve diğer düğümlere dallandığı şekilde birbirine bağlı düğüm adı verilen veri yapıları grubudur.
- Şekili düşünelim; üstte düğüm A, düğüm B ve C'ye bağlanır. Düğüm B, düğüm D ve E'ye bağlar. Bir üst düğüm, diğer düğümlere bağlanan ve bir alt düğümdür.. Örneğin, A A'nın çocukları olan B ve C'ye ait bir üst düğümdür. Yaprak düğüm, hiçbir bağlantı içermeyen bir düğümdür. C, D ve E düğümleri yapraklardır. Gösterilen örnek ikili bir ağaçtır çünkü düğüm başına maksimum iki çocuk vardır.

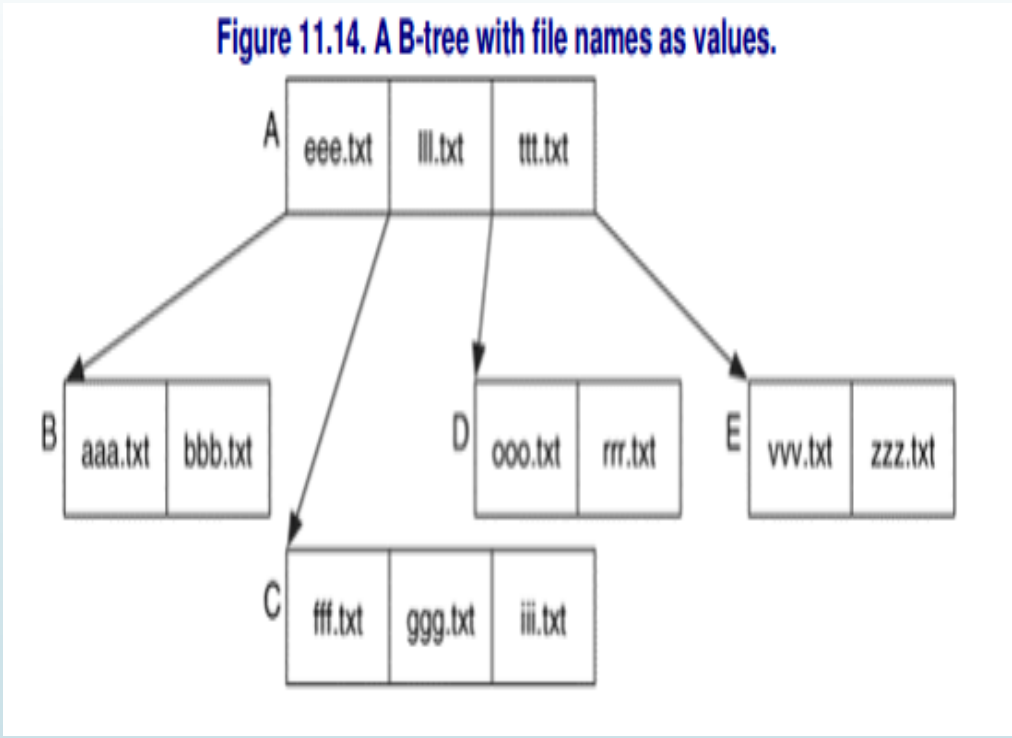


B-Trees

- Ağaçlar kullanışlıdır, çünkü bunlar kolayca veri sıralamak ve bulmak için kullanılabilir.
- Bir değeri aramak istiyorsak onu kök düğümle karşılaştırın. Kök düğüm daha büyükse, soldaki çocuğa gideriz. Kök düğüm daha küçükse sağdaki çocuğa gideriz.
- Örneğin, 6 değerini bulmak istersek, onu kök değer olan 7 ile karşılaştırırız. Düğüm daha büyüktür, bu nedenle soldaki çocuğa gidip değerini 5 ile karşılaştıralım. Bu düğüm Daha küçük, yani sağ çocuğuna gidiyoruz ve değeri 6 ile kıyaslıyoruz.
- Değerimizi sadece üç karşılaştırma ile bulduk. Değerler bir listede bulunuyorsa, değeri 9 yerine sadece iki karşılaştırmada bulabilirdik.

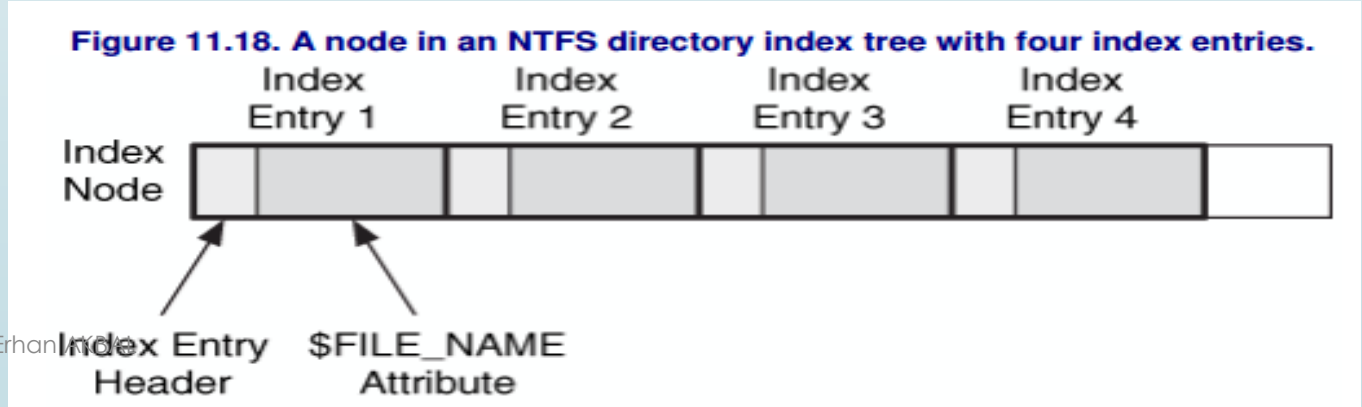
B-Trees

- NTFS, gördüğümüz ikili ağaca benzeyen B-ağaçları kullanır, ancak düğüm başına ikiden fazla çocuk olabilir.
- Genellikle, bir düğümün sahip olduğu çocuk sayısı, her düğümde kaç değer depolayabileceğine dayanır.
- Örneğin, ikili ağaçta her düğümde bir değer depoladık ve iki çocuğu oldu. Her düğümde beş değer depolayabilirsek altı çocuğa sahibiz.
- Şekilde Düğüm A üç değer ve dört çocuk içerir. Ggg.txt dosyasını arıyor olsaydık, kök düğümdeki değerlere bakar ve ismin alfabetik olarak eee.txt ve lll.txt arasında olduğunu belirleriz. Dolayısıyla, düğüm C'yi işler ve değerlerine bakarız. Bu düğümde dosya adını buluyoruz.



NTFS İndeks Atrributları

- İndeks oluşturmak için bunların NTFS'de nasıl uygulandığını tanımlamamız gerekiyor.
- Ağacın her girişi, değerleri her düğümde depolamak için bir **index entries (indeks girişi)** adı verilen bir veri yapısı kullanır.
- Çok sayıda dizin girişi vardır, ancak hepsinin aynı standart başlık alanları vardır.
- Örneğin, bir klasör indeks girişi birkaç başlık değeri ve bir $ FILE_NAME özniteliği içerir.
- İndeks girişi girdileri ağacın düğümleri halinde organize edilir ve bir listede saklanır. Listenin sonunu işaretlemek için boş bir giriş kullanılır. Şekilde, dört adet $ FILE_NAME dizin girdisine sahip bir dizin indeksindeki bir düğüm örneğini göstermektedir.



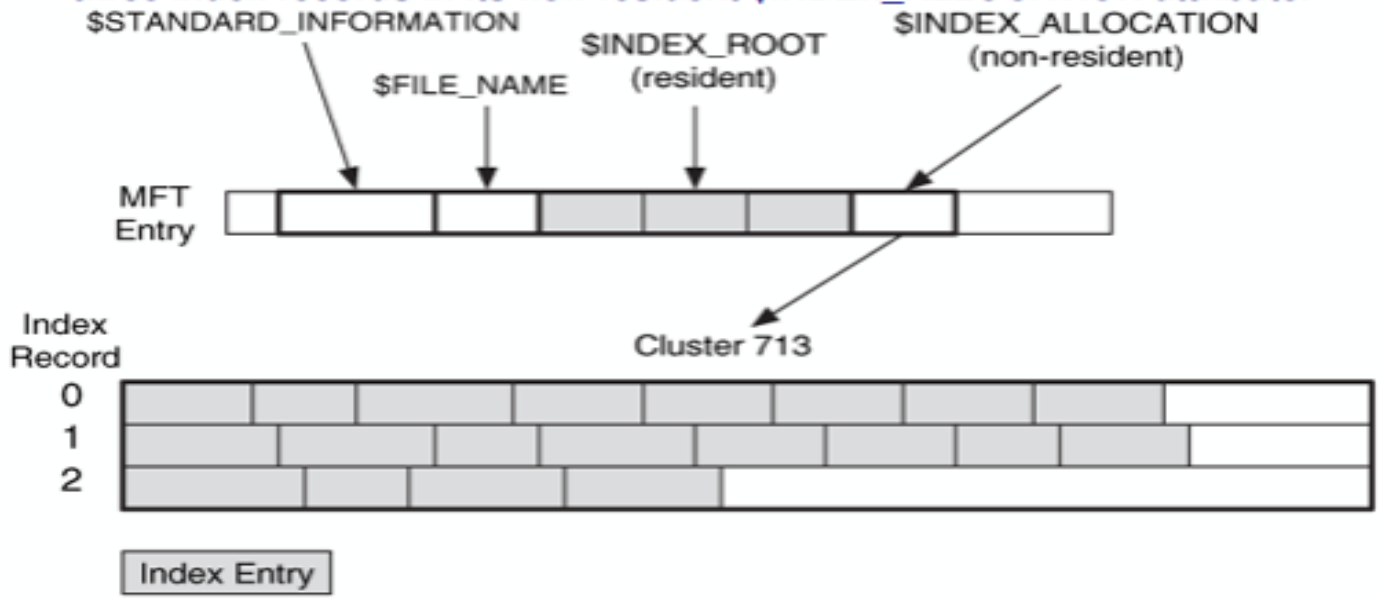
NTFS İndeks Atrributları

- Dizin düğümleri iki tür MFT giriş niteliğinde depolanabilir.
- $ INDEX_ROOT özniteliği her zaman bulunur ve az sayıda indeks girişi içeren yalnızca bir düğüm depolayabilir.
- $ INDEX_ROOT özniteliği daima indeks ağacının köküdür.
- Daha büyük indeksler, ihtiyaç duyulmayan sayıda düğüm içerebilen, yerleşik olmayan bir $ INDEX_ALLOCATION özniteliği tahsis eder.
- Bu öznitelik içeriği, bir veya daha fazla indeks girişi içeren büyük bir arabellektir.
- Bir indeks kaydı, genellikle 4,096 bayt statik bir boyuta sahiptir ve indeks girdilerinin bir listesini içerir.

NTFS İndeks Atrributları

- Her indeks kaydına 0'dan başlayan bir adres verilir. Şekilde üç indeks girdisine sahip bir $INDEX_ROOT özniteliğine ve küme 713'ü ayrılmış olan bir yerinde olmayan $ INDEX_ALLOCATION özniteliğine sahibiz ve üç indeks kaydı kullanıyor.

Figure 11.19. This directory has three index entries in its resident $INDEX_ROOT attribute and three index records in its non-resident $INDEX_ALLOCATION attribute.



NTFS İndeks Atrributları

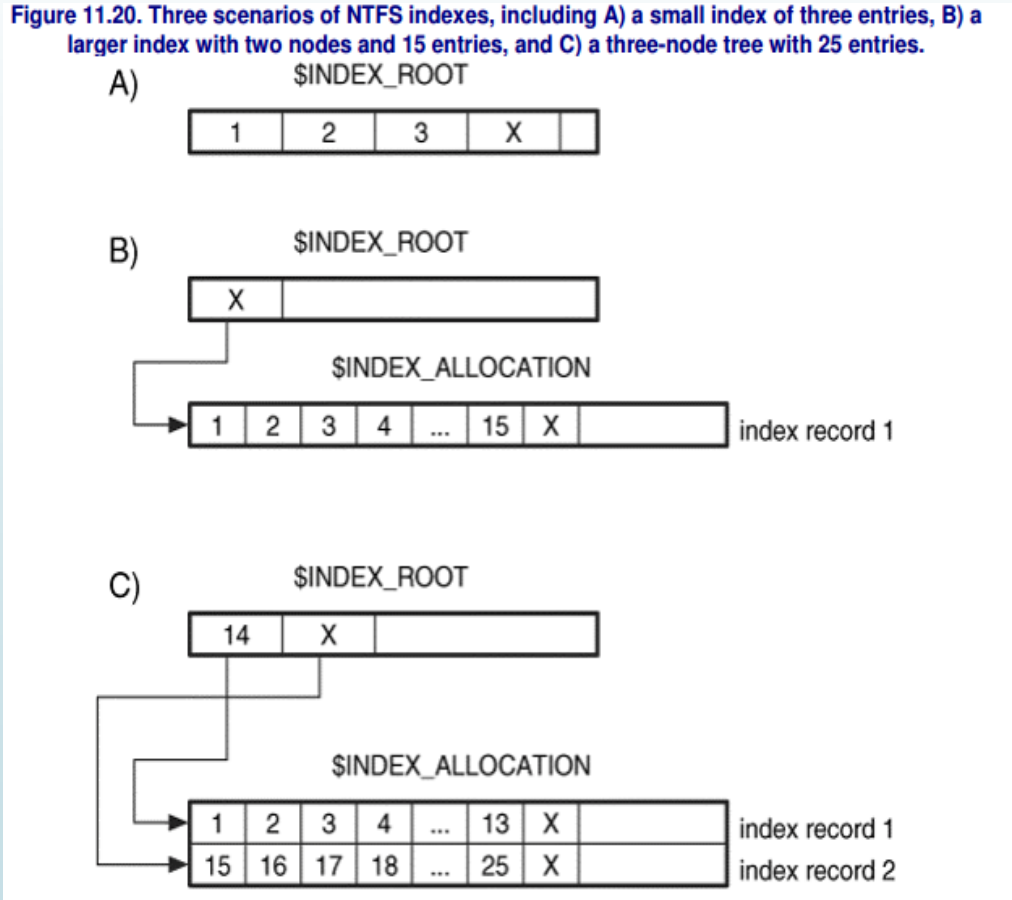
- Bir $ INDEX_ALLOCATION özniteliği indeks kayıtları için kullanılmayan alanı atayabilir.
- $ BITMAP özniteliği, dizin kayıtlarının ayırma durumunu yönetmek için kullanılır.
- Ağaç için yeni bir düğüm ayrılması gerekiyorsa, kullanılabilir bir indeks kaydı bulmak için $ BITMAP kullanılır; Aksi halde daha fazla alan eklenir.
- Her indekse bir ad verilir ve indeks için $INDEX_ROOT, $ INDEX_ALLOCATION ve $ BITMAP niteliklerine, nitelik headerında aynı ad verilir.

NTFS İndeks Atrributları

- Her indeks girişi, herhangi bir çocuk düğümü olup olmadığını gösteren bir bayrağı vardır.
- Çocuk düğümleri varsa, indeks kayıt adresleri indeks girdisinde verilir.
- Bir düğümdeki indeks girdileri isteğe göre sıralanmıştır ve aradığınız değer indeks girdisinden küçükse ve indeks girişi bir çocuk varsa, çocuğuna bakarsınız.
- $ INDEX_ROOT'a uyan üç girişi olan bir indeksi düşünün. Bu durumda, yalnızca bir $ INDEX_ROOT ayrılır ve üç dizin girişi veri yapısı içerir ve boş giriş, listenin sonundadır.

Örnek İndeks

- Şimdi, bir $ INDEX_ROOT'a uymayan, ancak bir dizin kaydı olan bir $ INDEX_ALLOCATION özniteliğine sığmayan 15 girişli bir dizin düşünün. Şekil 11.20 (B) 'de görülebilir.
- İndeks kaydındaki indeks girdilerini doldurduktan sonra yeni bir düzey eklemeliyiz ve üç düğümlü bir ağaç oluşturuyoruz. Bu senaryo Şekil 11.20'de (C) gösterilmektedir.
- Bu, kök düğümde ve iki çocuk düğümünde bir değer taşır. Her alt düğüm aynı $ INDEX_ALLOCATION özniteliğinde ayrı bir indeks kaydında bulunur ve $ INDEX_ROOT düğümündeki girdilerin işaret ettiği durumdur.



İndeks Veri Yapıları

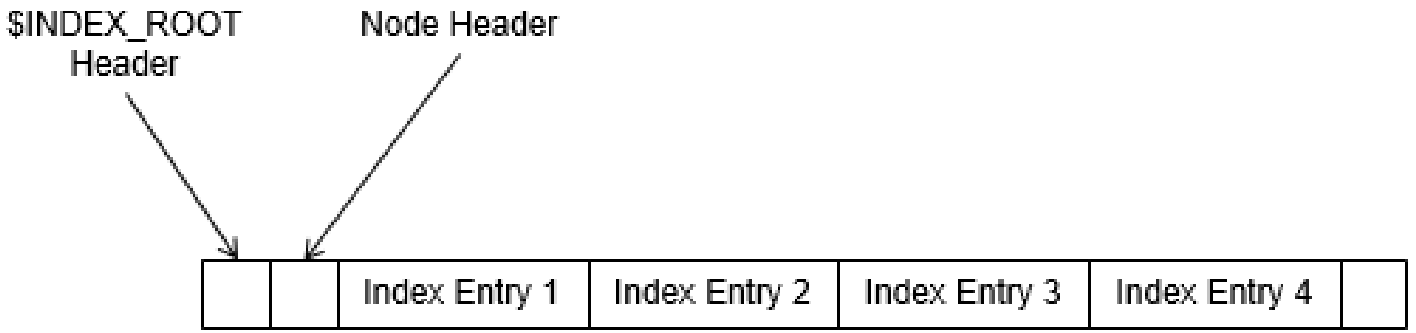
- İndeksler için özellikler ve veri yapıları
- İndeks
  - Sıralı bir ağaçtaki yapı
- Ağaç
  - Bir veya daha fazla düğüm
- Düğüm
  - Bir veya daha fazla indeks girişi
- Ağacın kökleri $ INDEX_ROOT Özelliklerindedir
- Düğümlerin geri kalanı $ INDEX_ALLOCATION özniteliğinde
- $ BITMAP özniteliği, tahsisat durumunu yönetmek için kullanılır

$INDEX_ROOT - Attribute

- Tür Kimliği - 144 (0x90)
- Daima bulunur
- Sadece küçük bir dizin girişi listesini saklayabilir
- 16 baytlık üstbilgi
- Düğüm başlığı
- Dizin girdilerinin listesi

$INDEX_ROOT - Yapısı

0x0	0 – 3	Type of attribute in index (0 if entry does not use an attribute)
0x4	4 – 7	Collation sorting rule
0x8	8 – 11	Size of each index record in bytes
0xC	12 – 12	Size in clusters
0xD	13 – 15	Unused
0x10	16+	Node header

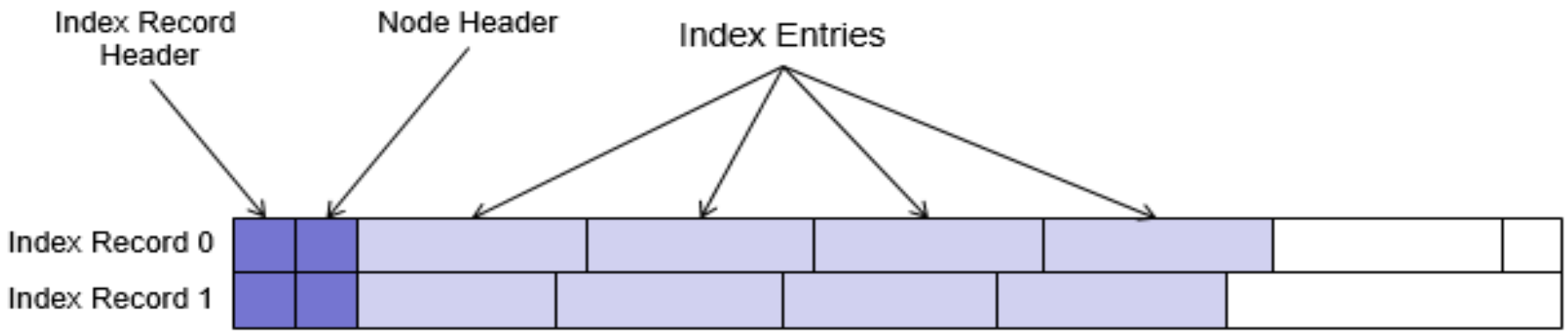


$INDEX_ALLOCATION - Özniteliği

- Tür Kimliği - 160 (0xA0)
- Büyük dizinler, yerleşik olmayan bir $ INDEX_ALLOCATION özniteliğine ihtiyaç duyar
- İndeks kayıtlarıyla doldurulur
- Dizin kaydı, $ INDEX_ROOT öznitelik başlığında tanımlanan statik bir boyutu vardır
- Dizin kaydı, sıralanmış ağacın bir düğüm içeriyor
- Tipik boyut 4096 bayt

$INDEX_ALLOCATION - Index Record Header

0x0	0 – 3	Signature value ("INDX")
0x4	4 – 5	Offset to fixup array
0x6	6 – 7	Number of entries in fixup array
0x8	8 – 15	$LogFile Sequence Number (LSN)
0x10	16 – 23	VCN of this record in the full index stream
0x18	24+	Node header



$I30 Files

- $ INDEX ROOT ve $ INDEX_ALLOCATION öznitelikleri bir klasör için genellikle $ I30 dosyalarını refere eder.

Index Düğüm Başlığı

0x0	0 – 3	Offset to start of index entry list Relative to start of node header
0x4	4 – 7	Offset to end of used portion of index entry list Relative to start of node header
0x8	8 – 11	Offset to end of allocated index entry list buffer Relative to start of node header
0xC	12 – 15	Flags - 0x01 is set when there are children nodes

Index Entry Genel

0x0	0 – 7	Undefined
0x8	8 – 9	Length of this entity
0xA	10 – 11	Length of content
0xC	12 – 15	Flags
0x10	16+	Content

Last 8 bytes of entry VCN of child node in $INDEX_ALLOCATION

Flags

0x01	Child node exists
0x02	Last entry in list

Index Entry Klasör

0x0	0 – 7	MFT file reference for file name
0x8	8 – 9	Length of this entity
0xA	10 – 11	Length of $FILE_NAME attribute
0xC	12 – 15	Flags
0x10	16+	$FILE_NAME attribute

Last 8 bytes of entry VCN of child node in $INDEX_ALLOCATION
Provided flag && 0x01 = 0x01

Flags

0x01	Child node exists
0x02	Last entry in list

$BITMAP Attribute

- $ INDEX_ALLOCATION özniteliğinde hangi dizin kayıtlarının kullanıldığını izler
- Dosyalar silindiğinde dizin kayıtları kullanılmaz hale gelir