Partitioner och filsystem 2

File systems

FAT

Unix-like

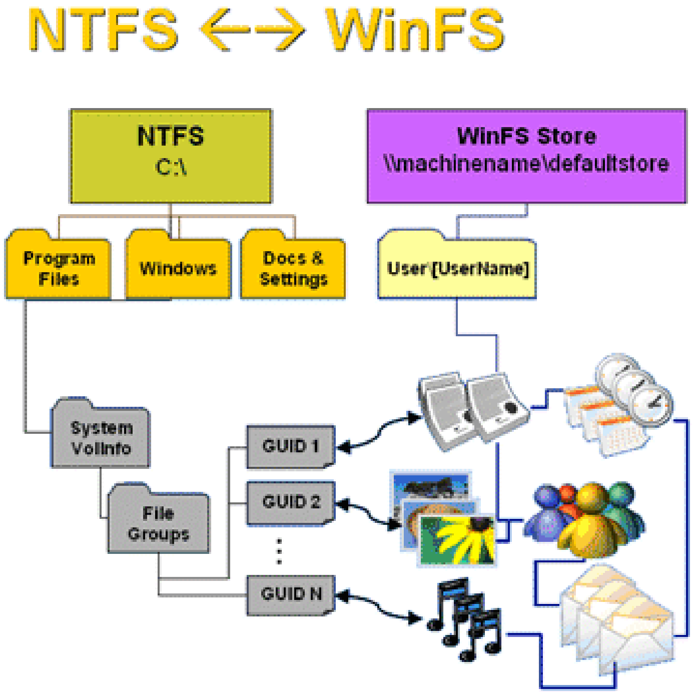
NTFS

Vad är ett filsystem?

- Datorer behöver en metod för att lagra och hämta data…
- Referensmodell för filsystem (Carrier)
  - **Filsystem** kategori
    - Layout och storleksinformation
  - **Innehålls** kategori
    - Kluster och block – data enheter
  - **Metadata** kategori
    - Tidsinformation, storlek, access kontroll
    - Adresser till allokerade data enheter
  - **Filnamn** kategori
    - Oftast ihop-kopplad med metadata
  - **Applikations** kategori
    - Quota
    - Journaler
- De modernaste påminner mycket om relations databaser

Windows

- NTFS (New Technology File System)
  - 6 versioner finns, de nyaste är v3.0 (Windows 2000) och v3.1 (XP, 2003, Vista, 2008, 7), kallas även 5.0, 5.1, 5.2, 6.0 och 6.1 (efter OS version)
  - Stöd för unicode, säkerhet, mm. - är mycket mer komplext än FAT!
  - http://en.wikipedia.org/wiki/Ntfs
- FAT 12/16/32, VFAT (långa filnamn i Win95)
  - Används fortfarande men är inte effektivt för större lagringskapaciteter (klusterstorleken)
  - Långsammare access än NTFS
- Windows Future Storage (WinFS) inställt projekt, enligt rykten var det en SQL-databas som ligger ovanpå ett NTFS filsystem
  - Läs mer på: http://www.ntfs.com/
  - Och: http://en.wikipedia.org/wiki/WinFS



FAT12, 16 och 32

- FAT12, finns på floppy diskar
  - Begränsad lagringskapacitet
  - Designat för MS-DOS 1.0
- FAT16, var designat för större diskar
  - Äldre OS använde detta
    - MS-DOS 3.0, Win95 OSR1, NT 3.5 och NT 4.0
  - Max diskstorlek 2 GB
- FAT32 kom när diskar större än 2GB kom
  - Vissa äldre och alla nya OS kan använda FAT32
    - Windows 98/Me/2000/XP/2003/Vista/7 och 2008
- Begränsningar med FAT32
  - Största formaterabara volymen är 32GB (större volymer kan dock användas, < 16 TiB)
  - Begränsade features vad gäller komprimering, kryptering, säkerhet och hastighet jämfört mot NTFS

- http://en.wikipedia.org/wiki/FAT_file_system

exFAT

- exFAT (Extended File Allocation Table, a.k.a. FAT64) is a proprietary file system suited especially for flash drives
- Introduced by Microsoft for embedded devices in Windows Embedded CE 6.0 and in their desktop operating system, starting with Windows Vista Service Pack 1
  - Support patches for XP and Linux is available
  - exFAT can be used where the NTFS file system is not a feasible solution, due to data structure overhead
- The advantages over previous File Allocation Table (FAT) file system versions include
  - Scalability to large disk sizes, up to 64 ZiB (Zebibyte)
  - Theoretical file size limit of 2^64 clusters,16 EiB (Exbibyte)
  - Support for **A**ccess **C**ontrol **L**ists (not supported in Windows Vista SP1)
  - Support for Transaction-Safe FAT File System (TFAT) (optionally WinCE activated function)
- The disadvantages compared to previous FAT versions include
  - Devices using exFAT are unable to use Windows Vista's ReadyBoost capability (Windows 7 supports the new exFAT filesystem with ReadyBoost)
  - Only one FAT and free space map (robustness?), TFAT have redundancy
  - Licensing status is unclear
  - At present limited or no support outside PC environment
- http://en.wikipedia.org/wiki/ExFAT

Tidsanalys av filer

- Är en viktig analys för att rekonstruera händelseförlopp
- Varje fil har följande attribut (MAC(E))
  - Sista modifieringen (Last modified time)
  - Sista åtkomsttiden (Last accessed time)
  - Skapande tid (Creation time)

E: Entry (in NTFS MFT) modified

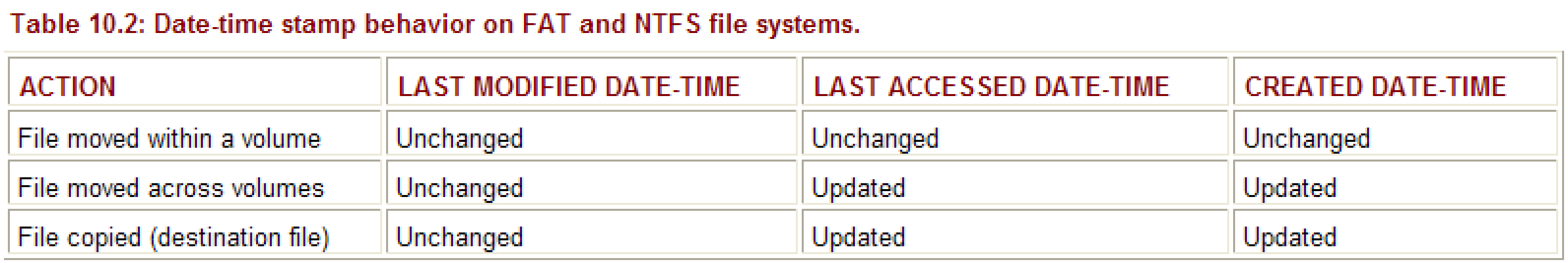


Table 10.2: Date-time stamp behavior on FAT and NTFS file systems.

ACTION	LAST MODIFIED DATE-TIME	LAST ACCESSED DATE-TIME	CREATED DATE-TIME
File moved within a volume	Unchanged	Unchanged	Unchanged
File moved across volumes	Unchanged	Updated	Updated
File copied (destination file)	Unchanged	Updated	Updated

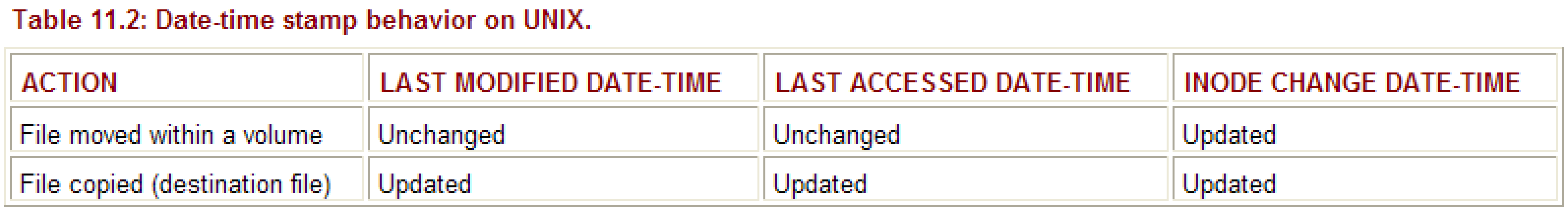


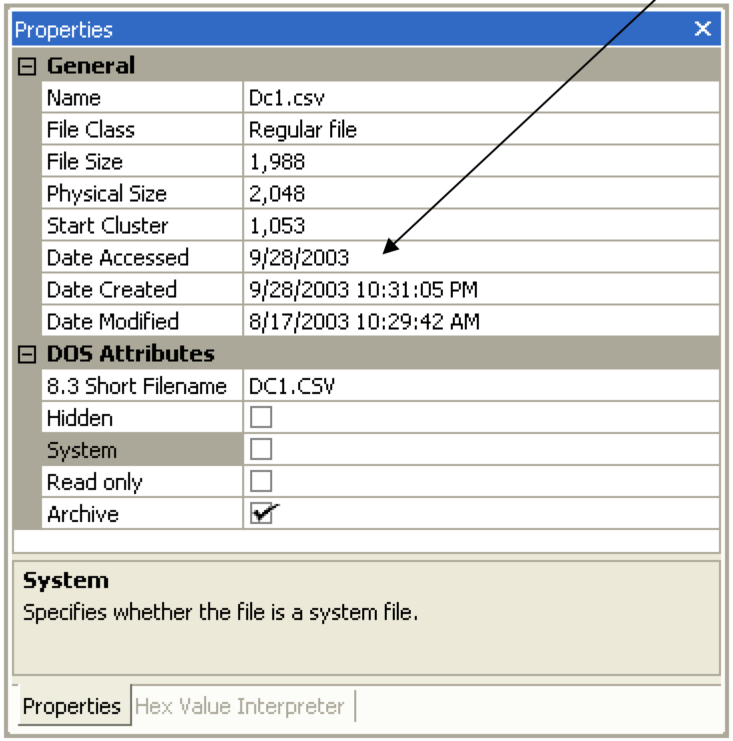
Table 11.2: Date-time stamp behavior on UNIX.

ACTION	LAST MODIFIED DATE-TIME	LAST ACCESSED DATE-TIME	INODE CHANGE DATE-TIME
File moved within a volume	Unchanged	Unchanged	Updated
File copied (destination file)	Updated	Updated	Updated

FAT/NTFS File Properties

Fat Properties

Note!



Properties ✕

⊟ **General**	
Name	Dc1.csv
File Class	Regular file
File Size	1,988
Physical Size	2,048
Start Cluster	1,053
Date Accessed	9/28/2003
Date Created	9/28/2003 10:31:05 PM
Date Modified	8/17/2003 10:29:42 AM
⊟ **DOS Attributes**	
8.3 Short Filename	DC1.CSV
Hidden	□
System	□
Read only	□
Archive	☑

System
Specifies whether the file is a system file.

MAC

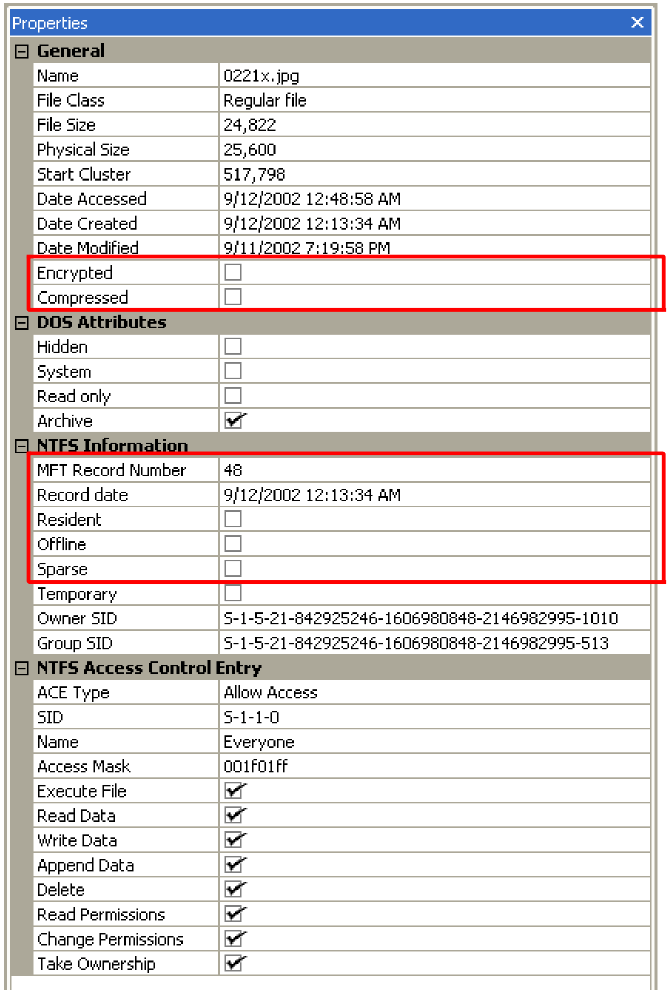
Modified Accessed Created

NTFS Properties

MAC

E

E: Entry modified. The time when the MFT entry itself was modified



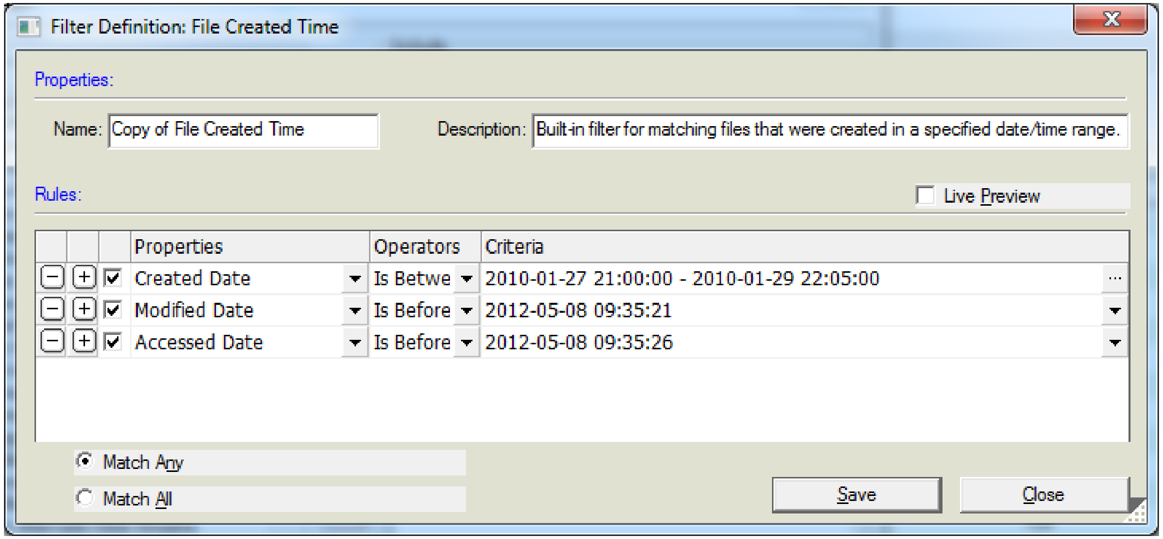
Properties ✕

⊟ **General**	
Name	0221x.jpg
File Class	Regular file
File Size	24,822
Physical Size	25,600
Start Cluster	517,798
Date Accessed	9/12/2002 12:48:58 AM
Date Created	9/12/2002 12:13:34 AM
Date Modified	9/11/2002 7:19:58 PM
Encrypted	□
Compressed	□
⊟ **DOS Attributes**	
Hidden	□
System	□
Read only	□
Archive	☑
⊟ **NTFS Information**	
MFT Record Number	48
Record date	9/12/2002 12:13:34 AM
Resident	□
Offline	□
Sparse	□
Temporary	□
Owner SID	S-1-5-21-842925246-1606980848-2146982995-1010
Group SID	S-1-5-21-842925246-1606980848-2146982995-513
⊟ **NTFS Access Control Entry**	
ACE Type	Allow Access
SID	S-1-1-0
Name	Everyone
Access Mask	001f01ff
Execute File	☑
Read Data	☑
Write Data	☑
Append Data	☑
Delete	☑
Read Permissions	☑
Change Permissions	☑
Take Ownership	☑

A (Date Accessed) in NTFS turned off from 2003/Vista!

Tidsanalys av filer live och på image

- Att undersöka filer på egen hand live är inte att rekommendera (tidskrävande)
- Verktyg för att analysera filer finns som tex. AFind (fungerar inte på Vista/7) som scannar igenom hela systemet enligt en viss konfiguration
- Kan visa var skadlig aktivitet pågår just nu i filsystemet
- Kom ihåg att falska indikationer kan finnas i form av
  - Bakgrundstjänster
  - Normal nätverksaktivitet
- De forensiska verktygen FTK, Encase mm. Har filter, addons (File Visualization) eller möjlighet att scripta



Filter Definition: File Created Time

Properties:

Name: Copy of File Created Time

Description: Built-in filter for matching files that were created in a specified date/time range.

Rules:

□ Live Preview

			Properties	Operators	Criteria
⊟	⊞	☑	Created Date	Is Betwe	2010-01-27 21:00:00 - 2010-01-29 22:05:00
⊟	⊞	☑	Modified Date	Is Before	2012-05-08 09:35:21
⊟	⊞	☑	Accessed Date	Is Before	2012-05-08 09:35:26

● Match Any

○ Match All

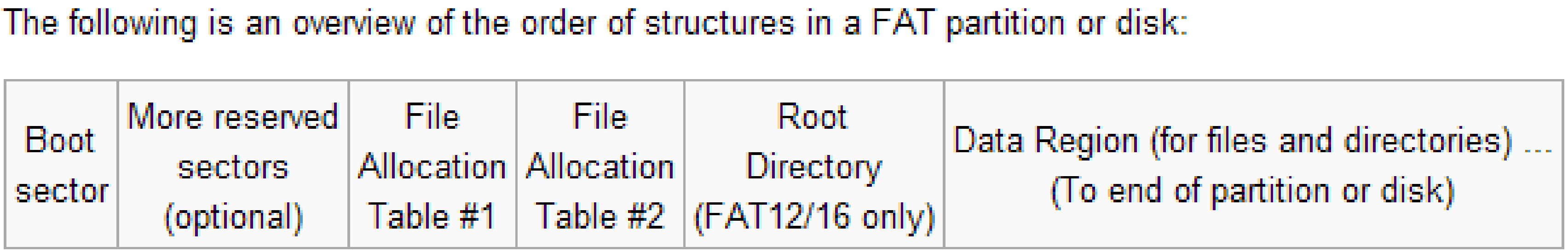
Fler filsystem

- Linux - native
  - ExtFS (Extended File System), ExtFS2, ExtFS3 (ext2 med journal), ExtFS4 (stödjer tex. extents = hela filen allokeras direkt)
  - UMSDOS (fixar Unix egenskaper i FAT)
- OS/2
  - HPFS (High Performance Filesystem)
- Macintosh/Apple OS X
  - MFS (Macintosh File System), HFS (Hierarchical File System), HFS+ (Mac OS Extended) and HFSX (Mac OS Extended with case sensitive file names), latest OS X have HFS read-only support
- UNiX och Solaris/OpenSolaris (HP, SUN etc.)
  - UFS (Unix File System), VxFS (Veritas File System), ZFS (Sun, Zettabyte File System – det extremaste? 128 bitars adressering…)
- IRIX (Silicon Graphics)
  - XFS

Ännu fler filsystem

- BSD/FreeBSD (Mac OS X är en BSD variant)
  - UFS/FFS (Fast File System), UFS2 and ZFS
- Andra
  - ReiserFS(3)/Reiser4 – Ett av de bättre filsystemen! Hanterar många filer extremt bra, http://en.wikipedia.org/wiki/Reiser4
  - IBM JFS 1/2 (Journaled File System) - AIX, OS/2, Linux
  - Btrfs (B-tree file system) bygger på design ideer från ReiserFS
- CD/DVD
  - UDF - Universal Disk Format (DVD-ROM filsystem)
  - ISO 9660 CD-ROM filesystem (ISO, Joliet, CDFS)
    - Joliet extensions medger unicode och långa filnamn
    - RockRidge medger länkar och långa filnamn
- Många fler finns… tex. inom embedded för flash minnen
  - http://www.forensics.nl/filesystems
  - http://en.wikipedia.org/wiki/Comparison_of_file_systems

FAT (File Allocation Table)



The following is an overview of the order of structures in a FAT partition or disk:

Boot sector	More reserved sectors (optional)	File Allocation Table #1	File Allocation Table #2	Root Directory (FAT12/16 only)	Data Region (for files and directories) ... (To end of partition or disk)

- Partition boot sector / Volume Boot Record (VBR)
  - BPB (Bios Parameter Block), pekare till OS boot loader kod
- FAT regionen
  - Två FAT tabeller, en för redundans, håller reda på vilka kluster som används och är lediga
- Root directory
  - En hierarkisk tabell som lagrar info om kataloger och filer
  - FAT 32 använder istället data regionen för detta (root directory kan ligga var som helst, dock oftast sekventiellt i början)
- Data regionen
  - Här lagras alla filer och katalogdata i kluster

FAT boot sector

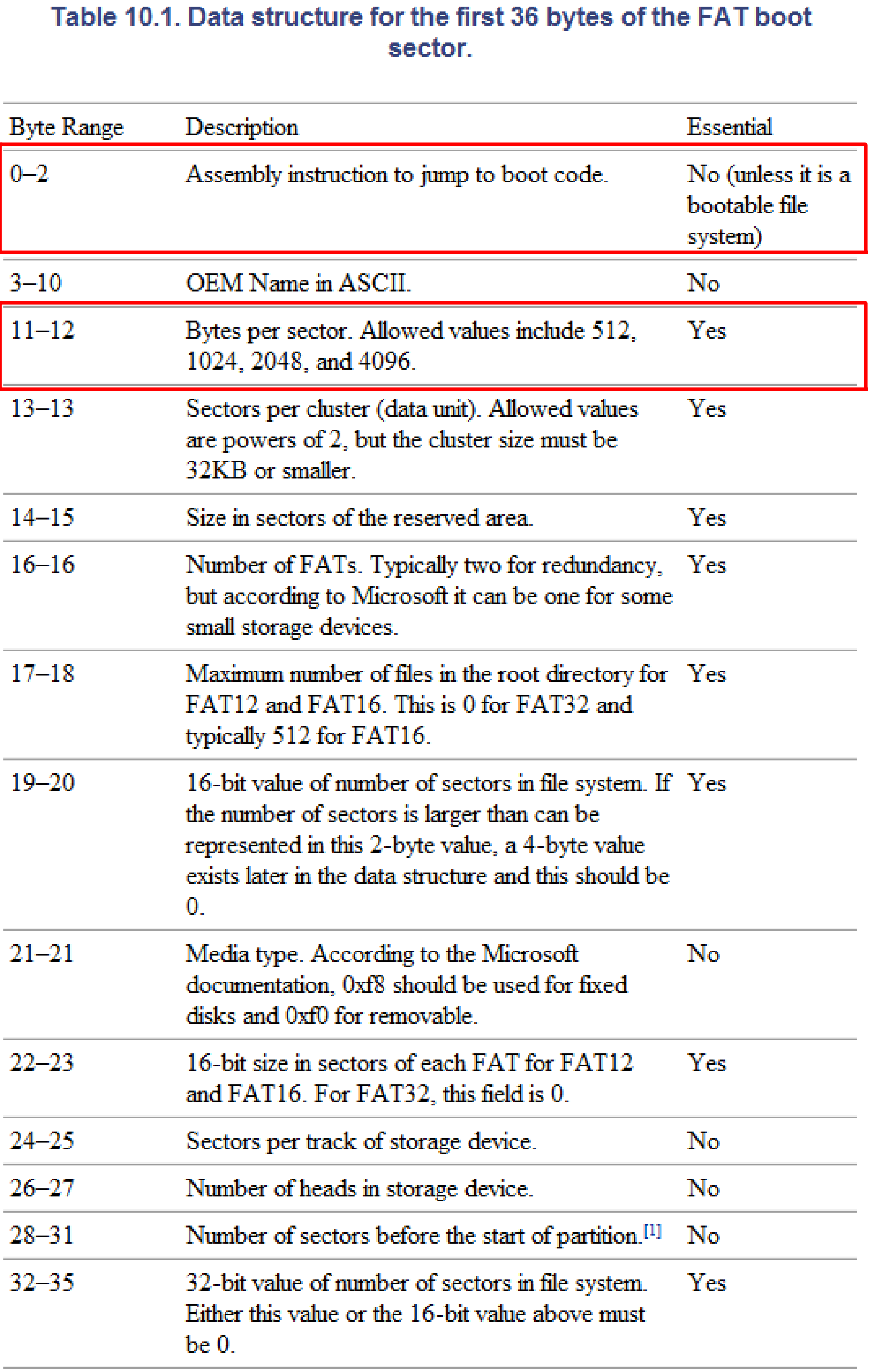


Table 10.1. Data structure for the first 36 bytes of the FAT boot sector.

Byte Range	Description	Essential
0–2	Assembly instruction to jump to boot code.	No (unless it is a bootable file system)
3–10	OEM Name in ASCII.	No
11–12	Bytes per sector. Allowed values include 512, 1024, 2048, and 4096.	Yes
13–13	Sectors per cluster (data unit). Allowed values are powers of 2, but the cluster size must be 32KB or smaller.	Yes
14–15	Size in sectors of the reserved area.	Yes
16–16	Number of FATs. Typically two for redundancy, but according to Microsoft it can be one for some small storage devices.	Yes
17–18	Maximum number of files in the root directory for FAT12 and FAT16. This is 0 for FAT32 and typically 512 for FAT16.	Yes
19–20	16-bit value of number of sectors in file system. If the number of sectors is larger than can be represented in this 2-byte value, a 4-byte value exists later in the data structure and this should be 0.	Yes
21–21	Media type. According to the Microsoft documentation, 0xf8 should be used for fixed disks and 0xf0 for removable.	No
22–23	16-bit size in sectors of each FAT for FAT12 and FAT16. For FAT32, this field is 0.	Yes
24–25	Sectors per track of storage device.	No
26–27	Number of heads in storage device.	No
28–31	Number of sectors before the start of partition.[1]	No
32–35	32-bit value of number of sectors in file system. Either this value or the 16-bit value above must be 0.	Yes

- The boot sector is located in the first sector of FAT file system and contains the bulk of the file system category of data.
- FAT12/16 and FAT32 have different versions of the boot sector, but they both have the same initial 36 bytes.
- The data structure for the first 36 bytes is given in Table 10.1, and the data structures for the remaining bytes are given in Tables 10.2 and 10.3.
- Boot sector/VBR = VBC + DPB (BPB)

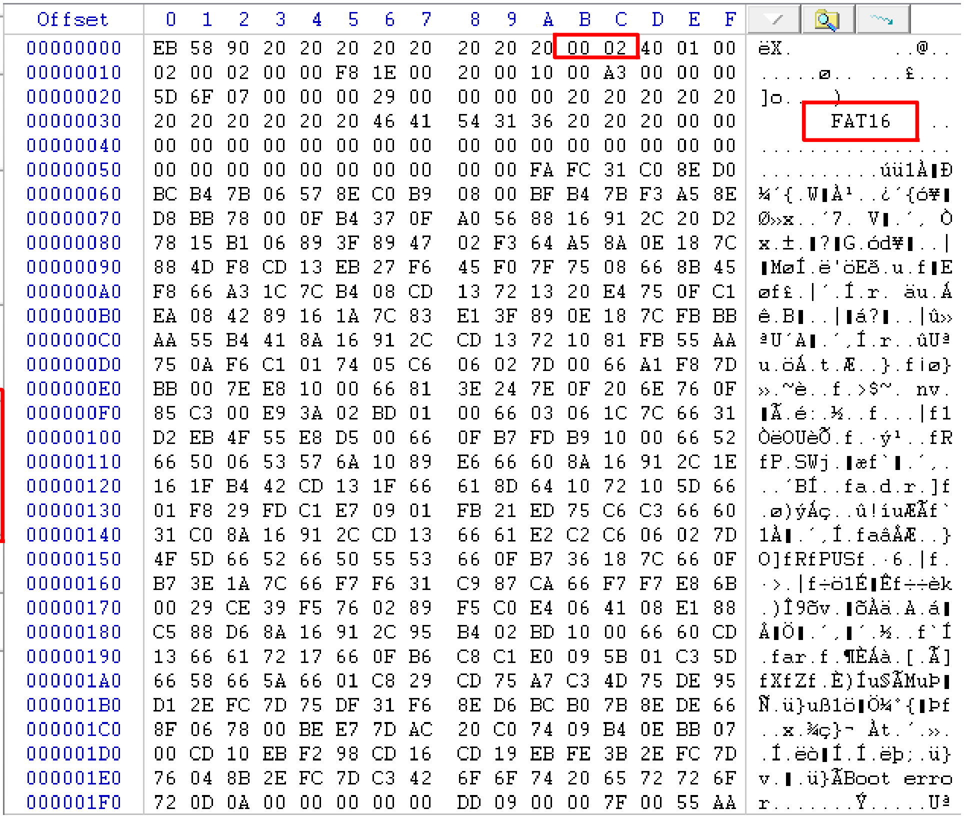
FAT12/16 boot sector

Table 10.2. Data structure for the remainder of the FAT12/16 boot sector.

Byte Range	Description	Essential
0–35	See Table 10.1.	Yes
36–36	BIOS INT13h drive number.	No
37–37	Not used.	No
38–38	Extended boot signature to identify if the next three values are valid. The signature is 0x29.	No
39–42	Volume serial number, which some versions of Windows will calculate based on the creation date and time.	No
43–53	Volume label in ASCII. The user chooses this value when creating the file system.	No
54–61	File system type label in ASCII. Standard values include "FAT," "FAT12," and "FAT16," but nothing is required.	No
62–509	Not used.	No
510–511	Signature value (0xAA55).	No

- Example VBR

May be empty if no OS



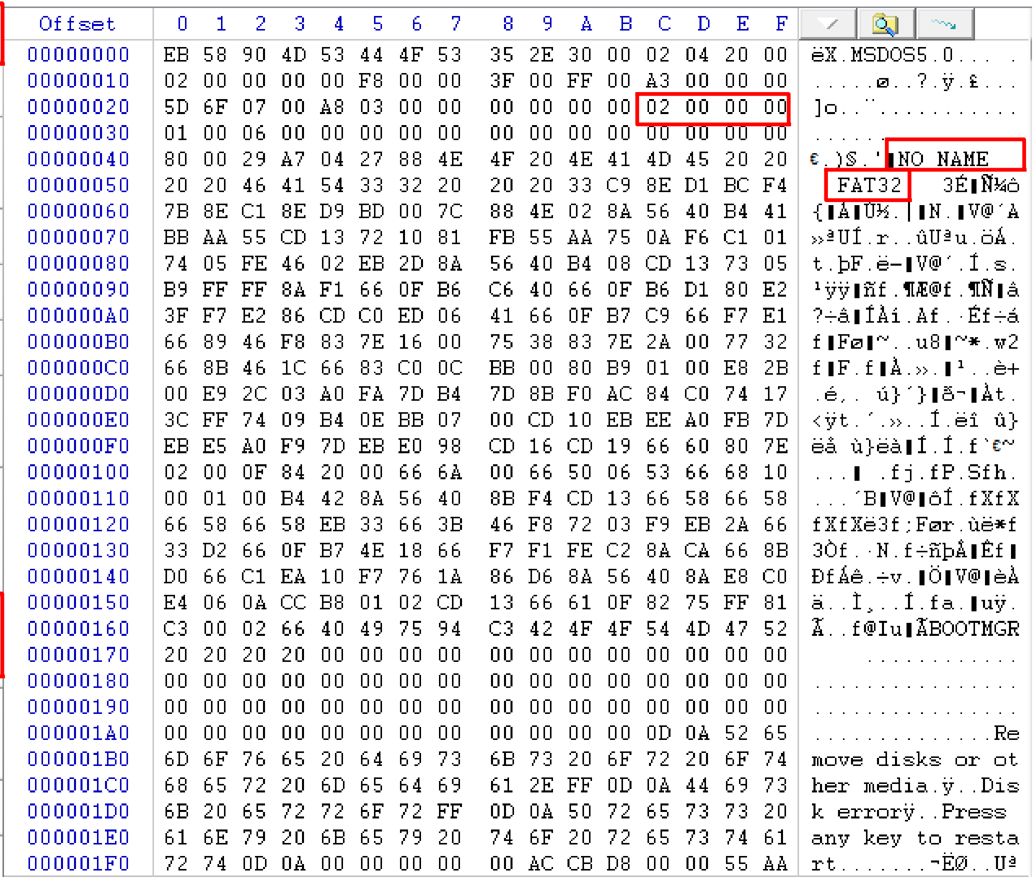
```
 Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00000000  EB 58 90 20 20 20 20 20  20 20 20 00 02 40 01 00  ëX.        ..@..
00000010  02 00 02 00 00 F8 1E 00  20 00 10 00 A3 00 00 00  .....ø.. ...£...
00000020  5D 6F 07 00 00 00 29 00  00 00 00 20 20 20 20 20  ]o.   )
00000030  20 20 20 20 20 20 46 41  54 31 36 20 20 20 00 00        FAT16   ..
00000040  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000050  00 00 00 00 00 00 00 00  00 00 FA FC 31 C0 8E D0  ..........úü1À▮Ð
00000060  BC B4 7B 06 57 8E C0 B9  08 00 BF B4 7B F3 A5 8E  ¼´{.W▮À¹..¿´{ó¥▮
00000070  D8 BB 78 00 0F B4 37 0F  A0 56 88 16 91 2C 20 D2  Ø»x..´7. V▮.´, Ò
00000080  78 15 B1 06 89 3F 89 47  02 F3 64 A5 8A 0E 18 7C  x.±.▮?▮G.ód¥▮..|
00000090  88 4D F8 CD 13 EB 27 F6  45 F0 7F 75 08 66 8B 45  ▮MøÍ.ë'öEð.u.f▮E
000000A0  F8 66 A3 1C 7C B4 08 CD  13 72 13 20 E4 75 0F C1  øf£.|´.Í.r. äu.Á
000000B0  EA 08 42 89 16 1A 7C 83  E1 3F 89 0E 18 7C FB BB  ê.B▮..|▮á?▮..|û»
000000C0  AA 55 B4 41 8A 16 91 2C  CD 13 72 10 81 FB 55 AA  ªU´A▮.´,Í.r..ûUª
000000D0  75 0A F6 C1 01 74 05 C6  06 02 7D 00 66 A1 F8 7D  u.öÁ.t.Æ..}.f¡ø}
000000E0  BB 00 7E E8 10 00 66 81  3E 24 7E 0F 20 6E 76 0F  ».~è..f.>$~. nv.
000000F0  85 C3 00 E9 3A 02 BD 01  00 66 03 06 1C 7C 66 31  ▮Ã.é:.½..f...|f1
00000100  D2 EB 4F 55 E8 D5 00 66  0F B7 FD B9 10 00 66 52  ÒëOUèÕ.f..·ý¹..fR
00000110  66 50 06 53 57 6A 10 89  E6 66 60 8A 16 91 2C 1E  fP.SWj.▮æf`▮.´,.
00000120  16 1F B4 42 CD 13 1F 66  61 8D 64 10 72 10 5D 66  ..´BÍ..fa.d.r.]f
00000130  01 F8 29 FD C1 E7 09 01  FB 21 ED 75 C6 C3 66 60  .ø)ýÁç..û!íuÆÃf`
00000140  31 C0 8A 16 91 2C CD 13  66 61 E2 C2 C6 06 02 7D  1À▮.´,Í.faâÂÆ..}
00000150  4F 5D 66 52 66 50 55 53  66 0F B7 36 18 7C 66 0F  O]fRfPUSf..·6.|f.
00000160  B7 3E 1A 7C 66 F7 F6 31  C9 87 CA 66 F7 F7 E8 6B  ·>.|f÷ö1É▮Êf÷÷èk
00000170  00 29 CE 39 F5 76 02 89  F5 C0 E4 06 41 08 E1 88  .)Î9õv.▮õÀä.A.á▮
00000180  C5 88 D6 8A 16 91 2C 95  B4 02 BD 10 00 66 60 CD  Å▮Ö▮.´,▮´.½..f`Í
00000190  13 66 61 72 17 66 0F B6  C8 C1 E0 09 5B 01 C3 5D  .far.f.¶ÈÁà.[.Ã]
000001A0  66 58 66 5A 66 01 C8 29  CD 75 A7 C3 4D 75 DE 95  fXfZf.È)ÍuSÃMuþ▮
000001B0  D1 2E FC 7D 75 DF 31 F6  8E D6 BC B0 7B 8E DE 66  Ñ.ü}uß1ö▮Ö¼°{▮Þf
000001C0  8F 06 78 00 BE E7 7D AC  20 C0 74 09 B4 0E BB 07  ..x.¾ç}¬ Àt.´.».
000001D0  00 CD 10 EB F2 98 CD 16  CD 19 EB FE 3B 2E FC 7D  .Í.ëò▮Í.Í.ëþ;.ü}
000001E0  76 04 8B 2E FC 7D C3 42  6F 6F 74 20 65 72 72 6F  v.▮.ü}ÃBoot erro
000001F0  72 0D 0A 00 00 00 00 00  DD 09 00 00 7F 00 55 AA  r.......Ý.....Uª
```

Table 10.3. Data structure for the remainder of the FAT32 boot sector.

Byte Range	Description	Essential
0–35	See Table 10.1.	Yes
36–39	32-bit size in sectors of one FAT.	Yes
40–41	Defines how multiple FAT structures are written to. If bit 7 is 1, only one of the FAT structures is active and its index is described in bits 0–3. Otherwise, all FAT structures are mirrors of each other.	Yes
42–43	The major and minor version number.	Yes
44–47	Cluster where root directory can be found.	Yes
48–49	Sector where FSINFO structure can be found.	No
50–51	Sector where backup copy of boot sector is located (default is 6).	No
52–63	Reserved.	No
64–64	BIOS INT13h drive number.	No
65–65	Not used.	No
66–66	Extended boot signature to identify if the next three values are valid. The signature is 0x29.	No
67–70	Volume serial number, which some versions of Windows will calculate based on the creation date and time.	No
71–81	Volume label in ASCII. The user chooses this value when creating the file system.	No
82–89	File system type label in ASCII. Standard values include "FAT32," but nothing is required.	No
90–509	Not used.	No
510–511	Signature value (0xAA55).	No

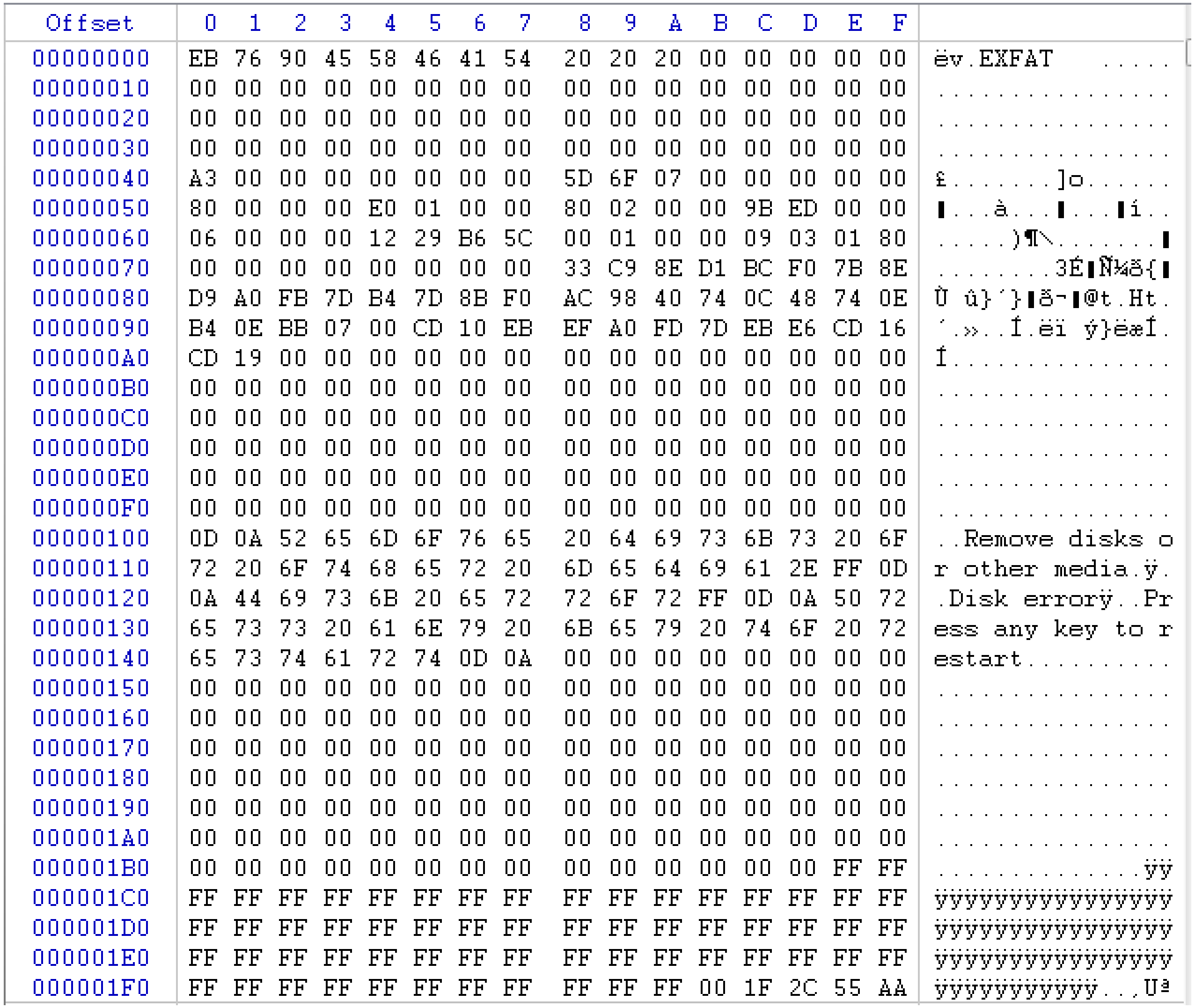
FAT32 boot sector

- Example VBR



```
 Offset    0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00000000  EB 58 90 4D 53 44 4F 53  35 2E 30 00 02 04 20 00   ëX.MSDOS5.0.... .
00000010  02 00 00 00 00 F8 00 00  3F 00 FF 00 A3 00 00 00   .....ø..?.ÿ.£...
00000020  5D 6F 07 00 A8 03 00 00  00 00 00 00 02 00 00 00   ]o..¨...........
00000030  01 00 06 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000040  80 00 29 A7 04 27 88 4E  4F 20 4E 41 4D 45 20 20   €.)§.'▮NO NAME  
00000050  20 20 46 41 54 33 32 20  20 20 33 C9 8E D1 BC F4     FAT32   3É▮Ñ¼ô
00000060  7B 8E C1 8E D9 BD 00 7C  88 4E 02 8A 56 40 B4 41   {▮Á▮Ù½.|▮N.▮V@´A
00000070  BB AA 55 CD 13 72 10 81  FB 55 AA 75 0A F6 C1 01   »ªUÍ.r..ûUªu.öÁ.
00000080  74 05 FE 46 02 EB 2D 8A  56 40 B4 08 CD 13 73 05   t.þF.ë-▮V@´.Í.s.
00000090  B9 FF FF 8A F1 66 0F B6  C6 40 66 0F B6 D1 80 E2   ¹ÿÿ▮ñf.¶Æ@f.¶Ñ▮â
000000A0  3F F7 E2 86 CD C0 ED 06  41 66 0F B7 C9 66 F7 E1   ?÷â▮ÍÀí.Af.·Éf÷á
000000B0  66 89 46 F8 83 7E 16 00  75 38 83 7E 2A 00 77 32   f▮Fø▮~..u8▮~*.w2
000000C0  66 8B 46 1C 66 83 C0 0C  BB 00 80 B9 01 00 E8 2B   f▮F.f▮À.».▮¹..è+
000000D0  00 E9 2C 03 A0 FA 7D B4  7D 8B F0 AC 84 C0 74 17   .é,. ú}´}▮ð¬▮Àt.
000000E0  3C FF 74 09 B4 0E BB 07  00 CD 10 EB EE A0 FB 7D   <ÿt.´.»..Í.ëî û}
000000F0  EB E5 A0 F9 7D EB E0 98  CD 16 CD 19 66 60 80 7E   ëå ù}ëà▮Í.Í.f`€~
00000100  02 00 0F 84 20 00 66 6A  00 66 50 06 53 66 68 10   ...▮ .fj.fP.Sfh.
00000110  00 01 00 B4 42 8A 56 40  8B F4 CD 13 66 58 66 58   ...´B▮V@▮ôÍ.fXfX
00000120  66 58 66 58 EB 33 66 3B  46 F8 72 03 F9 EB 2A 66   fXfXë3f;Før.ùë*f
00000130  33 D2 66 0F B7 4E 18 66  F7 F1 FE C2 8A CA 66 8B   3Òf.·N.f÷ñþÂ▮Êf▮
00000140  D0 66 C1 EA 10 F7 76 1A  86 D6 8A 56 40 8A E8 C0   ÐfÁê.÷v.▮Ö▮V@▮èÀ
00000150  E4 06 0A CC B8 01 02 CD  13 66 61 0F 82 75 FF 81   ä..Ì¸..Í.fa.▮uÿ.
00000160  C3 00 02 66 40 49 75 94  C3 42 4F 4F 54 4D 47 52   Ã..f@Iu▮ÃBOOTMGR
00000170  20 20 20 20 00 00 00 00  00 00 00 00 00 00 00 00       ............
00000180  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
00000190  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
000001A0  00 00 00 00 00 00 00 00  00 00 00 00 0D 0A 52 65   ..............Re
000001B0  6D 6F 76 65 20 64 69 73  6B 73 20 6F 72 20 6F 74   move disks or ot
000001C0  68 65 72 20 6D 65 64 69  61 2E FF 0D 0A 44 69 73   her media.ÿ..Dis
000001D0  6B 20 65 72 72 6F 72 FF  0D 0A 50 72 65 73 73 20   k errorÿ..Press 
000001E0  61 6E 79 20 6B 65 79 20  74 6F 20 72 65 73 74 61   any key to resta
000001F0  72 74 0D 0A 00 00 00 00  00 AC CB D8 00 00 55 AA   rt.......¬ËØ..Uª
```

exFAT boot sector



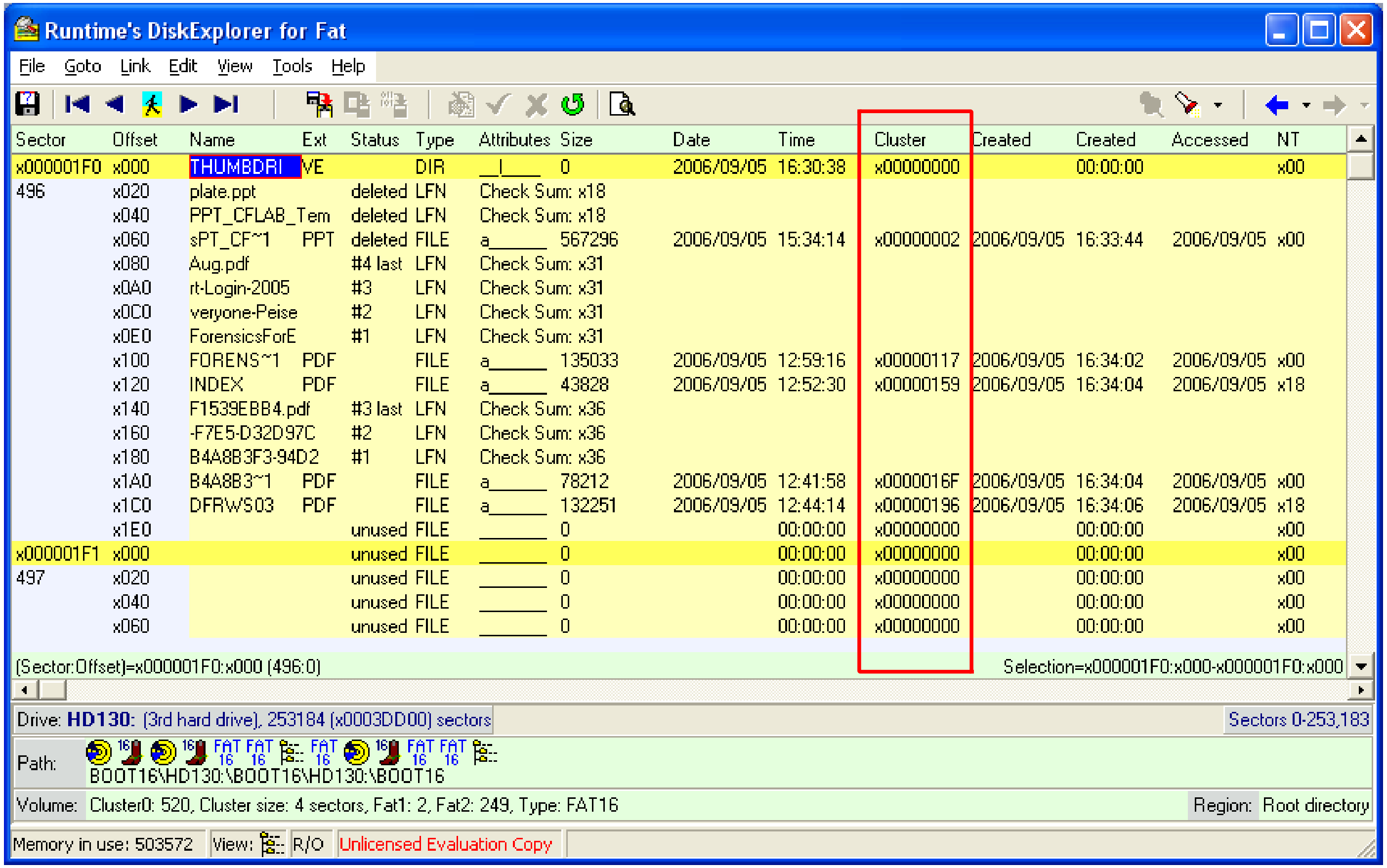
```
Offset     0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F

00000000  EB 76 90 45 58 46 41 54  20 20 20 00 00 00 00 00  ëv.EXFAT   .....
00000010  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000020  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000030  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000040  A3 00 00 00 00 00 00 00  5D 6F 07 00 00 00 00 00  £.......]o......
00000050  80 00 00 00 E0 01 00 00  80 02 00 00 9B ED 00 00  ▮...à...▮...▮í..
00000060  06 00 00 00 12 29 B6 5C  00 01 00 00 09 03 01 80  .....)¶\.......▮
00000070  00 00 00 00 00 00 00 00  33 C9 8E D1 BC F0 7B 8E  ........3É▮Ñ¼ð{▮
00000080  D9 A0 FB 7D B4 7D 8B F0  AC 98 40 74 0C 48 74 0E  Ù û}´}▮ð¬▮@t.Ht.
00000090  B4 0E BB 07 00 CD 10 EB  EF A0 FD 7D EB E6 CD 16  ´.»..Í.ëï ý}ëæÍ.
000000A0  CD 19 00 00 00 00 00 00  00 00 00 00 00 00 00 00  Í...............
000000B0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000C0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000D0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000E0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000000F0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000100  0D 0A 52 65 6D 6F 76 65  20 64 69 73 6B 73 20 6F  ..Remove disks o
00000110  72 20 6F 74 68 65 72 20  6D 65 64 69 61 2E FF 0D  r other media.ÿ.
00000120  0A 44 69 73 6B 20 65 72  72 6F 72 FF 0D 0A 50 72  .Disk errorÿ..Pr
00000130  65 73 73 20 61 6E 79 20  6B 65 79 20 74 6F 20 72  ess any key to r
00000140  65 73 74 61 72 74 0D 0A  00 00 00 00 00 00 00 00  estart..........
00000150  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000160  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000170  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000180  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
00000190  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001A0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
000001B0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 FF FF  ..............ÿÿ
000001C0  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000001D0  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000001E0  FF FF FF FF FF FF FF FF  FF FF FF FF FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
000001F0  FF FF FF FF FF FF FF FF  FF FF FF 00 1F 2C 55 AA  ÿÿÿÿÿÿÿÿÿÿÿ..,Uª
```

- Example VBR

Root directory i FAT

- Type - Dir or File Name - Short or Long (LFN)
- Extension
- Deleted?
- Size
- Starting Cluster
- Created Date/Time*
- Modified Date/Time*
- Access Date



Runtime's DiskExplorer for Fat
File Goto Link Edit View Tools Help

Sector	Offset	Name	Ext	Status	Type	Attributes	Size	Date	Time	Cluster	Created	Created	Accessed	NT
x000001F0	x000	THUMBDRI	VE		DIR	__|____	0	2006/09/05	16:30:38	x00000000		00:00:00		x00
496	x020	plate.ppt		deleted	LFN	Check Sum: x18								
	x040	PPT_CFLAB_Tem		deleted	LFN	Check Sum: x18								
	x060	sPT_CF~1	PPT	deleted	FILE	a______	567296	2006/09/05	15:34:14	x00000002	2006/09/05	16:33:44	2006/09/05	x00
	x080	Aug.pdf		#4 last	LFN	Check Sum: x31								
	x0A0	rt-Login-2005		#3	LFN	Check Sum: x31								
	x0C0	veryone-Peise		#2	LFN	Check Sum: x31								
	x0E0	ForensicsForE		#1	LFN	Check Sum: x31								
	x100	FORENS~1	PDF		FILE	a______	135033	2006/09/05	12:59:16	x00000117	2006/09/05	16:34:02	2006/09/05	x00
	x120	INDEX	PDF		FILE	a______	43828	2006/09/05	12:52:30	x00000159	2006/09/05	16:34:04	2006/09/05	x18
	x140	F1539EBB4.pdf		#3 last	LFN	Check Sum: x36								
	x160	-F7E5-D32D97C		#2	LFN	Check Sum: x36								
	x180	B4A8B3F3-94D2		#1	LFN	Check Sum: x36								
	x1A0	B4A8B3~1	PDF		FILE	a______	78212	2006/09/05	12:41:58	x0000016F	2006/09/05	16:34:04	2006/09/05	x00
	x1C0	DFRWS03	PDF		FILE	a______	132251	2006/09/05	12:44:14	x00000196	2006/09/05	16:34:06	2006/09/05	x18
	x1E0			unused	FILE	_______	0		00:00:00	x00000000		00:00:00		x00
x000001F1	x000			unused	FILE	_______	0		00:00:00	x00000000		00:00:00		x00
497	x020			unused	FILE	_______	0		00:00:00	x00000000		00:00:00		x00
	x040			unused	FILE	_______	0		00:00:00	x00000000		00:00:00		x00
	x060			unused	FILE	_______	0		00:00:00	x00000000		00:00:00		x00

(Sector:Offset)=x000001F0:x000 (496:0) Selection=x000001F0:x000-x000001F0:x000

Drive: **HD130:** (3rd hard drive), 253184 (x0003DD00) sectors Sectors 0-253,183

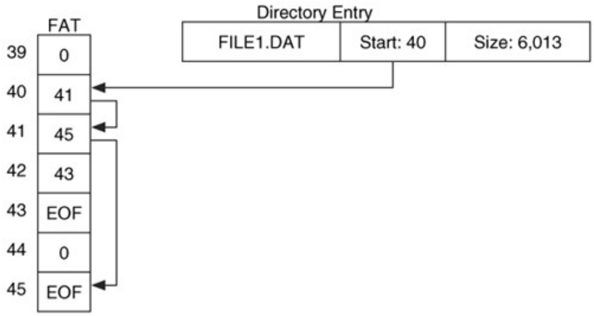
Path: BOOT16\HD130:\BOOT16\HD130:\BOOT16

Volume: Cluster0: 520, Cluster size: 4 sectors, Fat1: 2, Fat2: 249, Type: FAT16 Region: Root directory

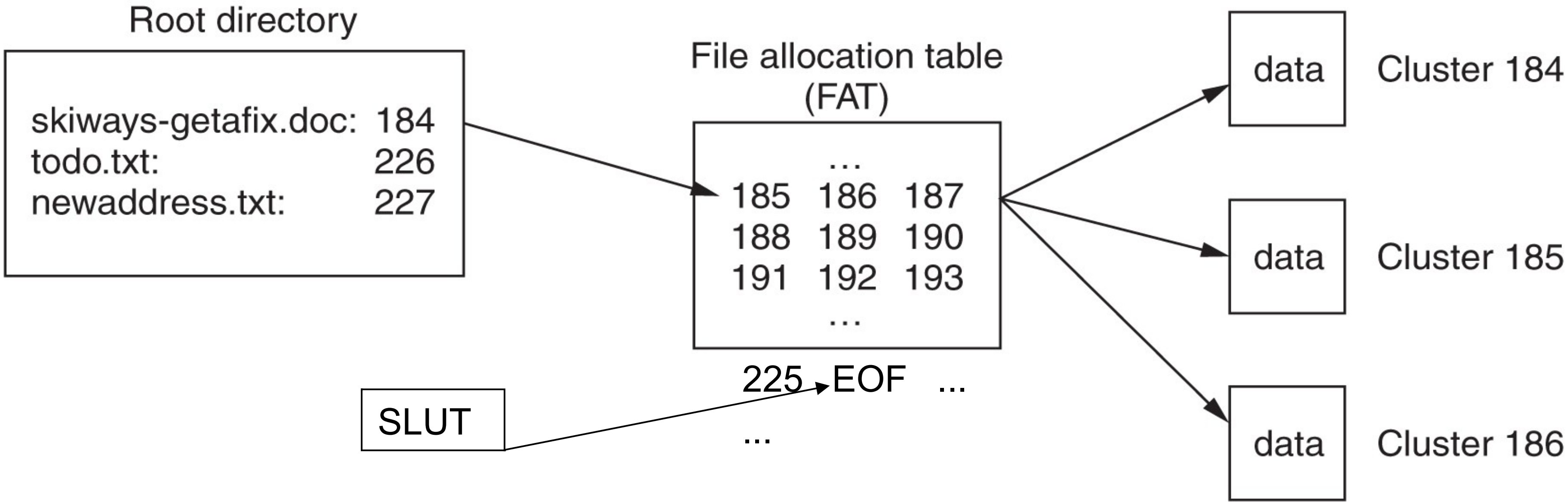
Memory in use: 503572 View: R/O Unlicensed Evaluation Copy

* = Time is stored as an even number

Läsa fil i FAT exempel



- Först letar man i root directory efter mappen (type attribut) filen ligger i, vilket pekar ut mappens lagring av filer
- Sedan läser man utpekade filens entry i root directory för att se i vilket kluster filen börjar på tex. 184
- Därefter går man till motsvarande post/entry i FAT (184) som med sin pekare indikerar att filen fortsätter i kluster 185
- I den posten finns en ny pekare till nästa kluster som innehåller filen. Man fortsätter på detta vis tills man stöter på EOF i post/entry 224, dvs. kluster 225, som markerar filslut



FAT delete

FAT File System Structures 1

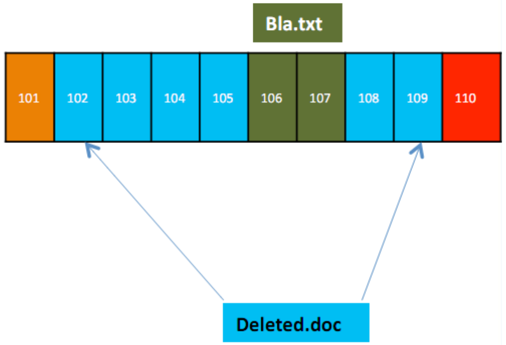
Root Directory Entries

File name	*Starting block*
Deleted.doc	102
Bla.txt	106
Archive.pst	110

FAT

Block Index	*Next Block*
101	Free
102	103
103	104
104	105
105	108
106	107
107	EOF
108	109
109	EOF
110	111

Media Data Block Area 1



FAT File System Structures 2

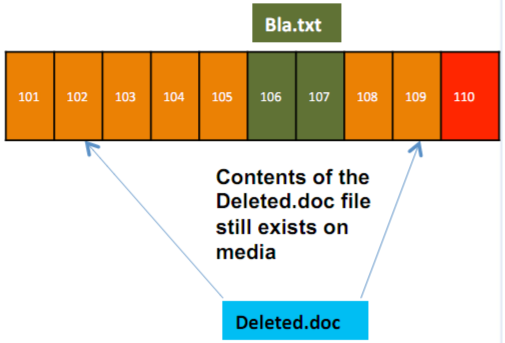
Root Directory Entries

File name	*Starting block*
_eleted.doc	102
Bla.txt	106
Archive.pst	110

FAT

Block Index	*Next Block*
101	Free
102	Free
103	Free
104	Free
105	Free
106	107
107	EOF
108	Free
109	Free
110	111

Media Data Block Area 2

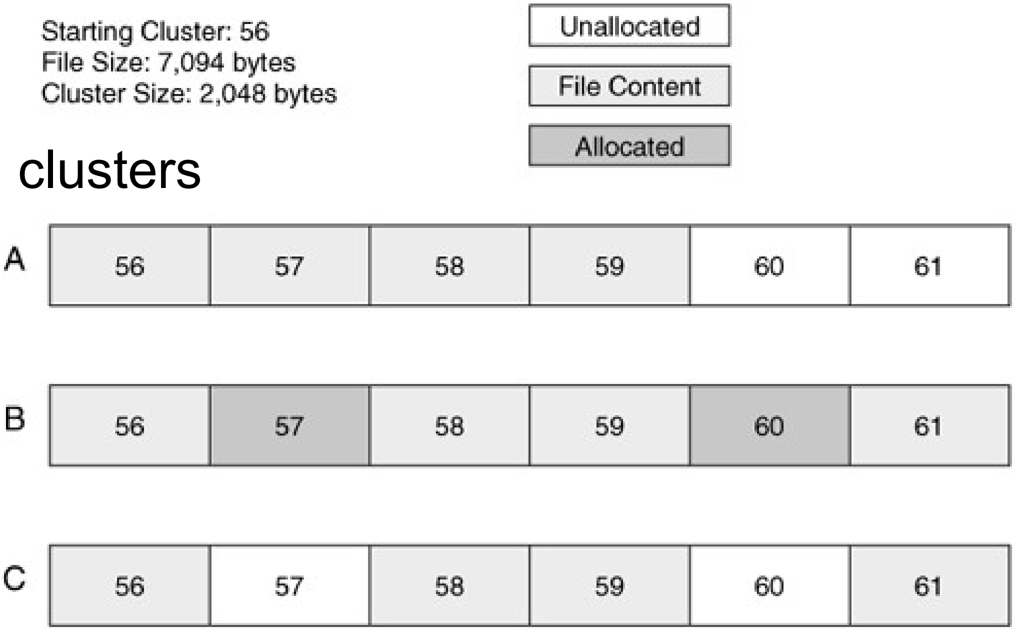


Contents of the Deleted.doc file still exists on media

Radera fil och recovery i FAT

- Raderad katalog eller fil sätter första byten i entryt till 0xe5 i root directory
  - FAT block/kluster pekarna sätts till 0/free för varje kluster, se sid 246 i Carrier
- File recovery
  - Två options
    1. Läsa blint
    2. Läsa NOT allocated clusters
  - Fil A: enkelt
  - Fil B: option 2 ok
  - Fil C: båda missar

Available cluster info



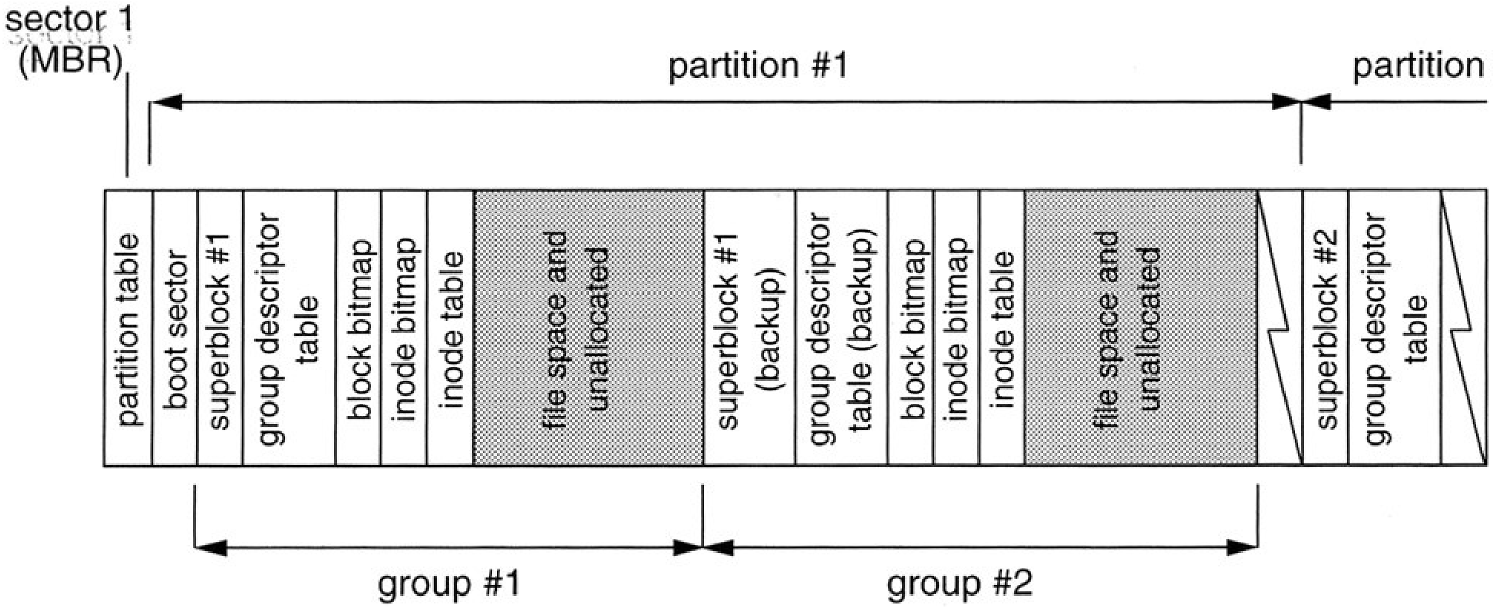
Starting Cluster: 56
File Size: 7,094 bytes
Cluster Size: 2,048 bytes

UNIX filsystem UFS, ext2… osv.

- Använder datastrukturer som kallas för index noder i en tabell för att representera filer, bibliotek och symboliska länkar
- Inode fälten (128 byte) är av ett fixt antal och lagrar metadata
- Varje fil och mapp har ett associerat entry i inode tabellen
- Inodens nr som hanterar filen/mappen kan visas med ls -i
- Inode nr 1 används vanligen för att lagra bad blocks
- Inode nr 2 används alltid för root directory
- Bra program för lågnivå diskundersökning
  - The Sleuth Kit – fsstat och istat kommandot
  - Linux Disk Editor – lde
  - Tune2fs – visar filsystem info mm. för ext2/ext3
- Vissa icke traditionella UNIX filsystem har en ganska olik uppbyggnad på låg nivå tex. ReiserFS
  - Kräver sina egna program/verktyg

UNIX filsystem UFS, ext2… osv.

- Delar upp partitionen i ett antal block grupper för redundans, ca: 128MB (32k*4k) per grupp för file space
- Superblocket (1 kB) innehåller viktig filsysteminfo som block size, ant. block, block per grupp, last mounted mm.
  - Sparse superblock, group desc.
- Group descriptor håller reda på grupperna och var saker finns (bitmaps, inode table)
- Block/inode - bitmappar hanterar allokeringsstatus



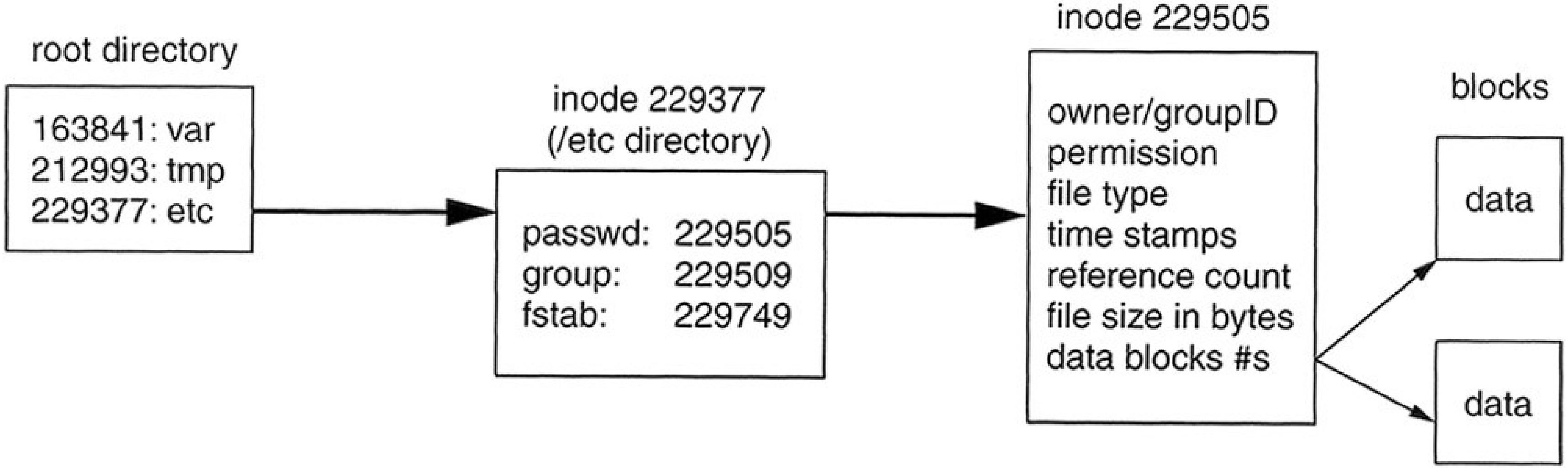
tune2fs kommandot

- Ger info om inoders index/fält- och block size
- Antalet inodes och blocks per grupp
- Mm. mm.

```
linuxbox:~# tune2fs -l /dev/hda2
tune2fs 1.40-WIP (14-Nov-2006)
Filesystem volume name:   <none>
Last mounted on:          <not available>
Filesystem UUID:          70be05e9-3e15-4456-be27-4153e420d320
Filesystem magic number:  0xEF53
Filesystem revision #:    1 (dynamic)
Filesystem features:      filetype sparse_super
Default mount options:    (none)
Filesystem state:         not clean
Errors behavior:          Continue
Filesystem OS type:       Linux
Inode count:              14469312
Block count:              14460508
Reserved block count:     723025
Free blocks:              9935321
Free inodes:              14340692
First block:              0
Block size:               4096
Fragment size:            4096
Blocks per group:         32768
Fragments per group:      32768
Inodes per group:         32736
Inode blocks per group:   1023
Last mount time:          Sun Mar 28 21:06:36 2010
Last write time:          Mon Apr 26 21:31:38 2010
Mount count:              1
Maximum mount count:      37
Last checked:             Sun Mar 28 21:02:21 2010
Check interval:           15552000 (6 months)
Next check after:         Fri Sep 24 21:02:21 2010
Reserved blocks uid:      0 (user root)
Reserved blocks gid:      0 (group root)
First inode:              11
Inode size:               128
```

UNIX filsystem, slå upp fil

- När systemet skall visa en viss fil tex. /etc/passwd går man först till superblocket för att hitta inod 2 (root directory)
- Man letar sedan upp mappen ”etc” i blocket som inodens info lagras i
- När ”etc” hittats går man till den inode som ”etc” pekar på och letar efter ”passwd” i dess info data block
- När ”passwd” hittats går man till ”passwd” inodens info och de data block ”passwd” inoden refererar till och kan slutligen läsa in själva fildatat



Inode 2 (root directory) -> block 5

```
linuxbox:~# lde -i 2 /dev/hda2
Device "/dev/hda2" is mounted, be careful
User requested autodetect filesystem. Checking device . . .
Found ext2fs on device.
Warning: First block (0) != Normal first block (1)
---------------------------------------------------------------------------
INODE: 2      (0x00000002)
drwxr-xr-x       root     root         4096 Sun Dec 24 01:10:00 2006
TYPE:                 directory
LINKS:                21
MODEFLAGS.MODE:        004.0755
SIZE:                 4096
BLOCK COUNT:             8
UID:                 00000 (root)
GID:                 00000 (root)
ACCESS TIME:           Tue Apr 28 10:14:37 2009
CREATION TIME:          Sun Dec 24 01:10:00 2006
MODIFICATION TIME:     Sun Dec 24 01:10:00 2006
DELETION TIME:          Thu Jan  1 01:00:00 1970
DIRECT BLOCKS:          0x00000005


INDIRECT BLOCK:
DOUBLE INDIRECT BLOCK:
TRIPLE INDIRECT BLOCK:
```

Linux Disk Editor – lde

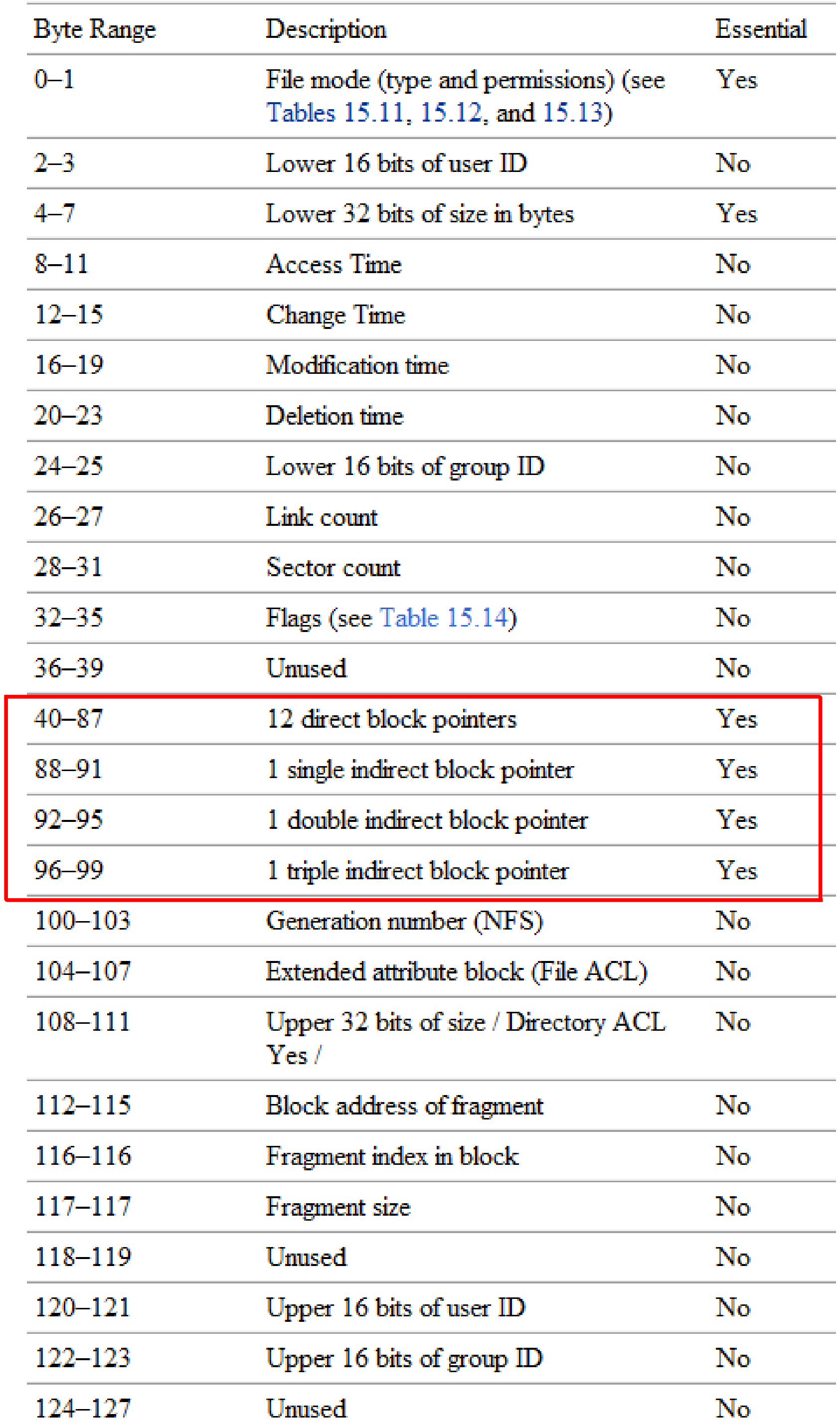
block 5   4096 stort

```
linuxbox:~# lde -b 5 /dev/hda2
0x00005000  02 00 00 00 0C 00 01 02 : 2E 00 00 00 02 00 00 00  ................
0x00005010  0C 00 02 02 2E 2E 00 00 : 0B 00 00 00 14 00 0A 02  ................
0x00005020  6C 6F 73 74 2B 66 6F 75 : 6E 64 00 00 0C 00 00 00  lost+found......
0x00005030  0C 00 03 02 65 74 63 00 : 23 05 00 00 0C 00 04 02  ....etc.#.......
0x00005040  72 6F 6F 74 59 16 00 00 : 0C 00 03 02 74 6D 70 00  rootY.......tmp.
0x00005050  79 16 00 00 0C 00 04 02 : 62 6F 6F 74 8C 16 00 00  y.......boot....
0x00005060  10 00 07 07 76 6D 6C 69 : 6E 75 7A 00 8D 16 00 00  ....vmlinuz.....
0x00005070  0C 00 03 02 6C 69 62 00 : 29 19 00 00 0C 00 03 02  ....lib.).......
0x00005080  75 73 72 00 0F EE 00 00 : 0C 00 04 02 73 62 69 6E  usr.........sbin
0x00005090  B3 EE 00 00 0C 00 03 02 : 76 61 72 00 5F 16 01 00  ........var._...
0x000050A0  0C 00 03 02 62 69 6E 00 : B5 16 01 00 0C 00 03 02  ....bin.........
0x000050B0  64 65 76 00 A9 2A 01 00 : 0C 00 04 02 68 6F 6D 65  dev..*......home
0x000050C0  AC 41 01 00 0C 00 03 02 : 6D 6E 74 00 AE 41 01 00  .A......mnt..A..
0x000050D0  0C 00 04 02 70 72 6F 63 : AF 41 01 00 0C 00 03 02  ....proc.A......
0x000050E0  6F 70 74 00 B0 41 01 00 : 10 00 06 02 66 6C 6F 70  opt..A......flop
0x000050F0  70 79 00 00 B1 41 01 00 : 10 00 05 02 63 64 72 6F  py...A......cdro
0x00005100  6D 00 00 00 B2 41 01 00 : 10 00 06 02 69 6E 69 74  m....A......init
0x00005110  72 64 00 00 B3 41 01 00 : EC 0E 03 02 73 79 73 00  rd...A......sys.
0x00005120  00 00 00 00 E0 0E 05 02 : 2E 72 6F 6F 74 00 00 00  .........root...
0x00005130  00 00 00 00 00 00 00 00 : 00 00 00 00 00 00 00 00  ................
```

ls -i   kommandot visar filens inod i inode table

Inode structure

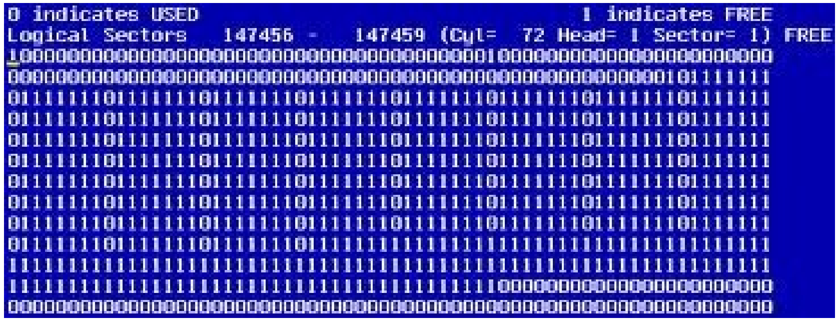
```
linuxbox:~# lde -i 1636804 /dev/hda2
Device "/dev/hda2" is mounted, be careful
User requested autodetect filesystem. Checking device . . .
Found ext2fs on device.
Warning: First block (0) != Normal first block (1)
----------------------------------------------------------------------
INODE: 1636804 (0x0018F9C4)
-rwxr--r--      hjo      hjo      17923572 Mon Apr 13 11:45:58 2009
TYPE:                     regular file
LINKS:                    1
MODEFLAGS.MODE:        010.0744
SIZE:                    17923572
BLOCK COUNT:             35056          = 512 byte block
UID:                     01000 (hjo)
GID:                     01000 (hjo)
ACCESS TIME:          Mon Apr 13 11:45:58 2009
CREATION TIME:         Mon Apr 13 11:46:04 2009
MODIFICATION TIME:     Mon Apr 13 11:45:58 2009
DELETION TIME:         Thu Jan  1 01:00:00 1970
DIRECT BLOCKS:        0x0019500A 0x0019500B 0x0019500C
0x0019500D 0x0019500E 0x0019500F 0x00195010 0x00195011
0x00195012 0x00195013 0x00195014 0x00195015
INDIRECT BLOCK:        0x00195016
DOUBLE INDIRECT BLOCK: 0x00196B05
TRIPLE INDIRECT BLOCK:
```



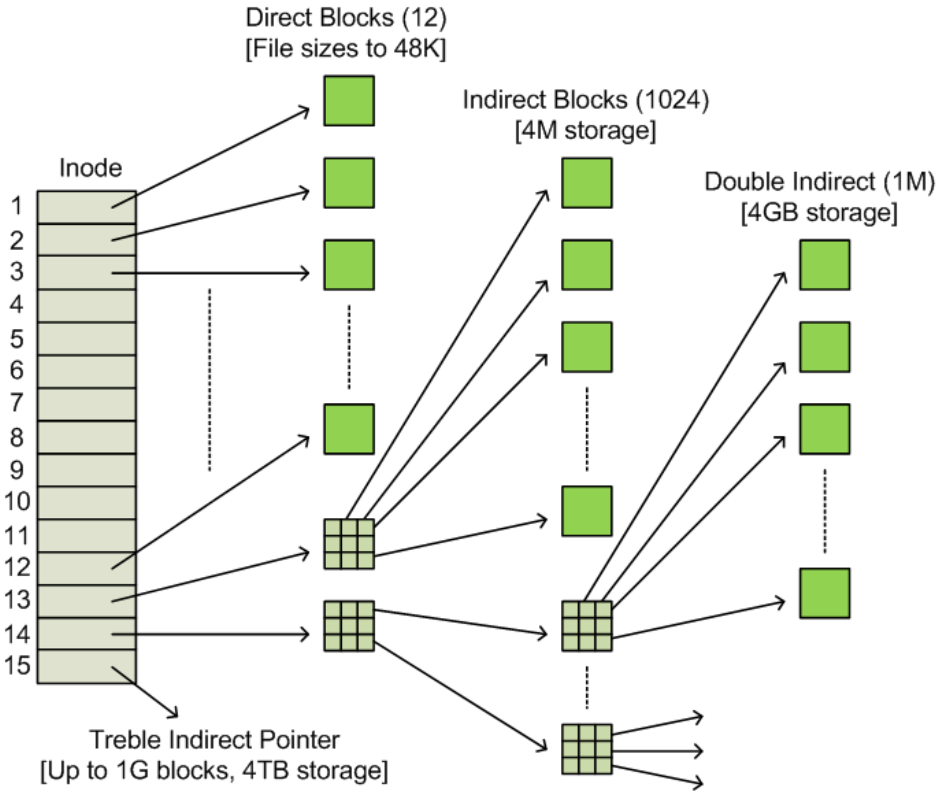
Byte Range	Description	Essential
0–1	File mode (type and permissions) (see Tables 15.11, 15.12, and 15.13)	Yes
2–3	Lower 16 bits of user ID	No
4–7	Lower 32 bits of size in bytes	Yes
8–11	Access Time	No
12–15	Change Time	No
16–19	Modification time	No
20–23	Deletion time	No
24–25	Lower 16 bits of group ID	No
26–27	Link count	No
28–31	Sector count	No
32–35	Flags (see Table 15.14)	No
36–39	Unused	No
40–87	12 direct block pointers	Yes
88–91	1 single indirect block pointer	Yes
92–95	1 double indirect block pointer	Yes
96–99	1 triple indirect block pointer	Yes
100–103	Generation number (NFS)	No
104–107	Extended attribute block (File ACL)	No
108–111	Upper 32 bits of size / Directory ACL Yes /	No
112–115	Block address of fragment	No
116–116	Fragment index in block	No
117–117	Fragment size	No
118–119	Unused	No
120–121	Upper 16 bits of user ID	No
122–123	Upper 16 bits of group ID	No
124–127	Unused	No

UNIX filsystem forts.

Block/inode - bitmap



```
0 indicates USED                                   1 indicates FREE
Logical Sectors   147456 -   147459 (Cyl=  72 Head= 1 Sector= 1) FREE
1000000000000000000000000000000000000000100000000000000000000000
0000000000000000000000000000000000000000000000000000000101111111
0111111101111111011111110111111101111111011111110111111101111111
0111111101111111011111110111111101111111011111110111111101111111
0111111101111111011111110111111101111111011111110111111101111111
0111111101111111011111110111111101111111011111110111111101111111
0111111101111111011111110111111101111111011111110111111101111111
0111111101111111011111110111111101111111011111110111111101111111
0111111101111111011111110111111101111111011111110111111101111111
0111111101111111011111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111111111111111111111111111
1111111111111111111111111111111111111111100000000000000000000000
0000000000000000000000000000000000000000000000000000000000000000
```



- Inoden har pekare till block där data lagras
  - Vid stora filer lagrar dessa istället pekare till nya block, upp till 3 ggr.
- Det finns speciella filtyper som inte lagrar data
  - Pekare till hårdvara, symbolisk länk etc. allt är filer i UNIX!
- Raderad fil fungerar lite olika i ext2 och ext3 filsystem
  - ext2fs markerar inoder med block pekare som lediga i block bitmaps och markerar ”info inoden” som ”deleted” i inode bitmap - men låter block pekarna stå kvar i inoden
  - ext3fs nollställer även block pekarna i inoder med block pekare
- Det finns inga verktyg för att hantera journalen i journalbaserade filsystem ännu som tex. ext3?

Ext3 delete

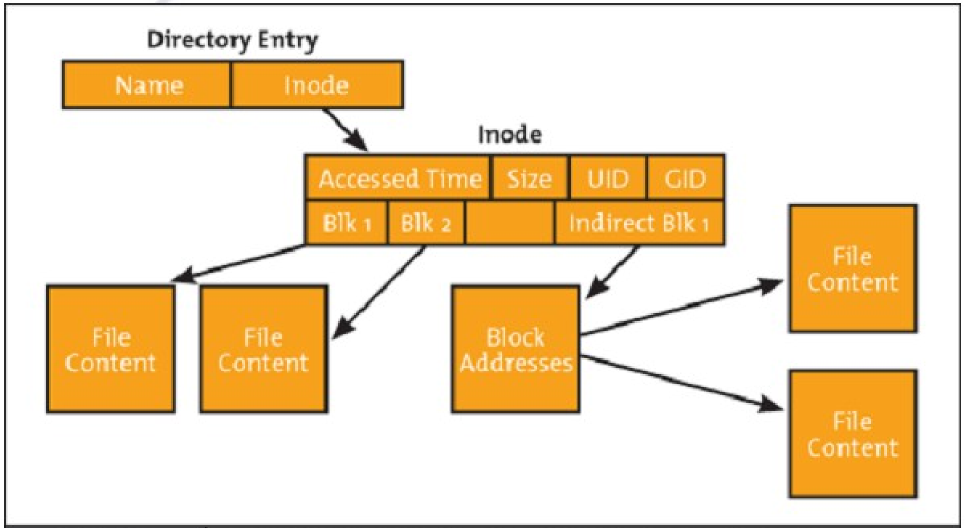


FIGURE 1 RELATIONSHIP BETWEEN THE DIRECTORY ENTRY, AN INODE, AND BLOCKS OF AN ALLOCATED FILE

Before deletion

HOWTO recover deleted files on an ext3 file system

http://www.xs4all.nl/~carlo17/howto/undelete_ext3.html

After deletion

Block pointers are zeroed out in the inode

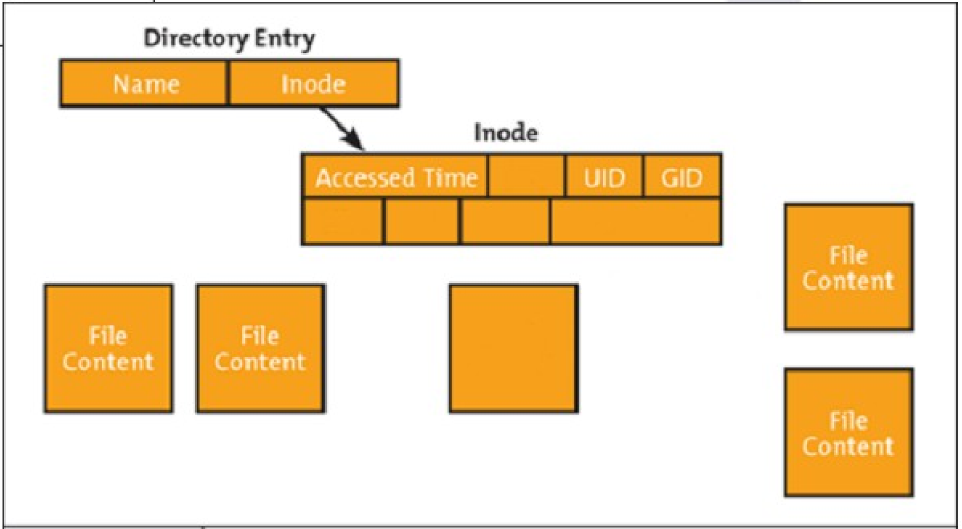
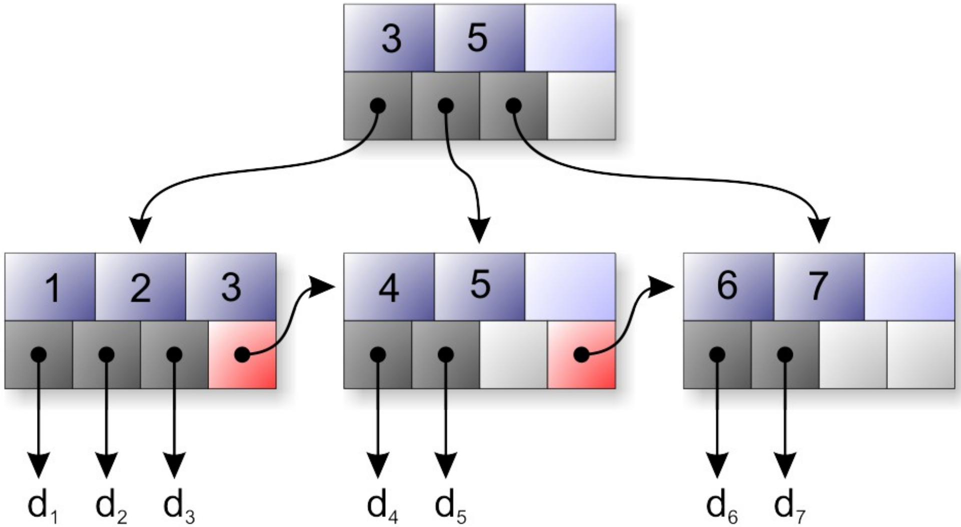


FIGURE 2 RELATIONSHIP BETWEEN THE DIRECTORY ENTRY, AN INODE, AND BLOCKS OF AN UNALLOCATED EXT3 FILE. THE LINKS BETWEEN THE INODE AND BLOCKS HAS BEEN CLEARED.

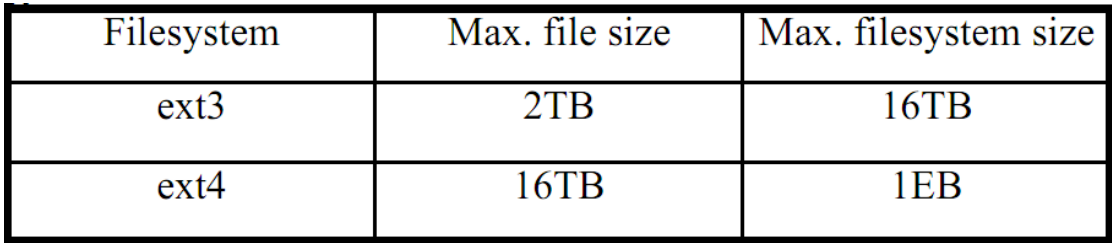
B-tree/B+ tree (not a binary tree)

- Representerar sorterad data som medger effektiv insättning, hämtning och borttagning av poster, samt indexering av metadata i filsystem och databaser
  - http://en.wikipedia.org/wiki/B%2B_tree
  - http://en.wikipedia.org/wiki/B-tree
- Används av NTFS, HFS, ReiserFS, XFS, JFS2, btrfs, ext4 mm.
- Ett enkelt B+ träd som länkar nycklarna 1-7 till datavärdena d1-d7
  - Den länkade listan (rött) medger snabb in-order traversering



Ext4 file system

- Ext4 is the successor of ext3 which is developed to solve performance issues and scalability bottleneck on ext3 and also provide backward compatibility with ext3
- Ext4 features
  - **Bigger file/filesystem size support (assuming 4 kB blocks):** because the block pointers are 48 bits instead of 32 bits

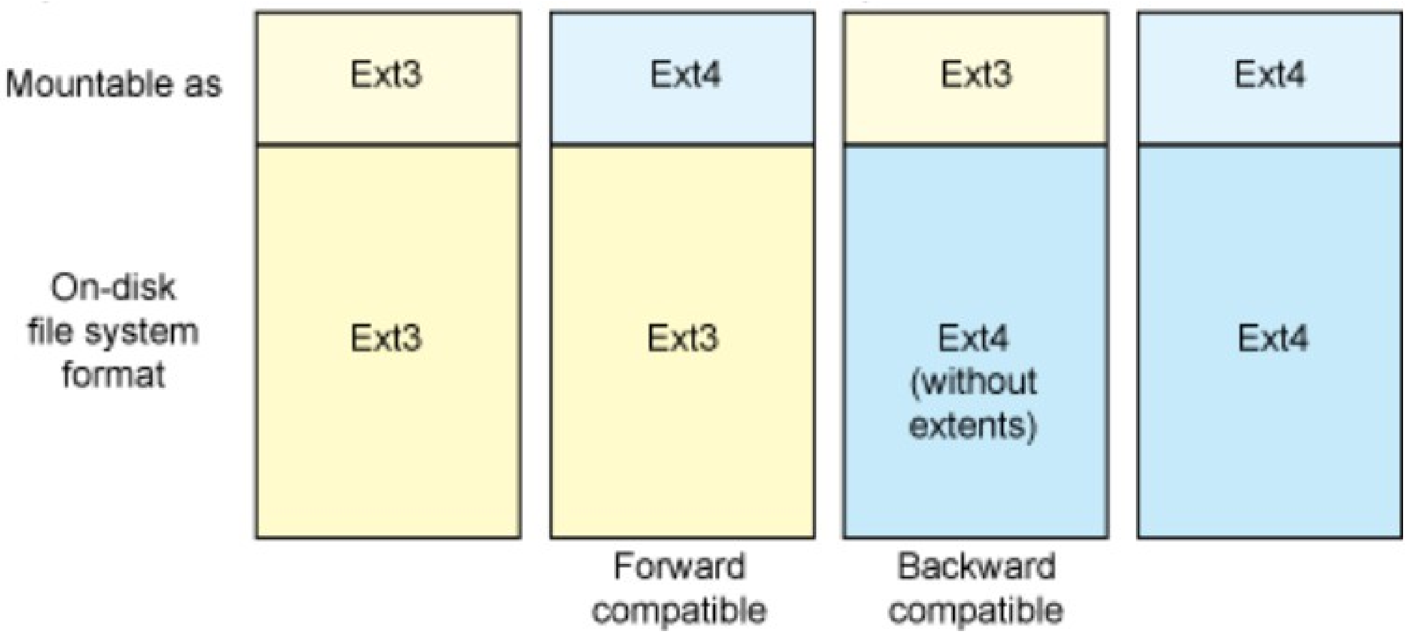


Filesystem	Max. file size	Max. filesystem size
ext3	2TB	16TB
ext4	16TB	1EB

  - **I/O performance improvement:** delayed allocation, multi block allocator extent map and persistent preallocation
  - **Fast fsck:** flex_bg and uninit_bg (file system feature flags)
  - **Reliability:** journal checksumming
  - **Maintenance:** online defrag
  - **Misc:** backward compatibility with ext2/ext3, nanosec timestamps, subdir scalability, etc.

Ext4 file system - compability

- When you want to migrate an ext3 file system to ext4, you can do so gradually
  - This means that old files that you have not moved can remain in the older ext3 format, while new files (or older files that have been copied) will occupy the new ext4 data structures
  - In this way, you can migrate an ext3 file system online to an ext4 file system



Mountable as	Ext3	Ext4	Ext3	Ext4
On-disk file system format	Ext3	Ext3	Ext4 (without extents)	Ext4
		Forward compatible	Backward compatible	

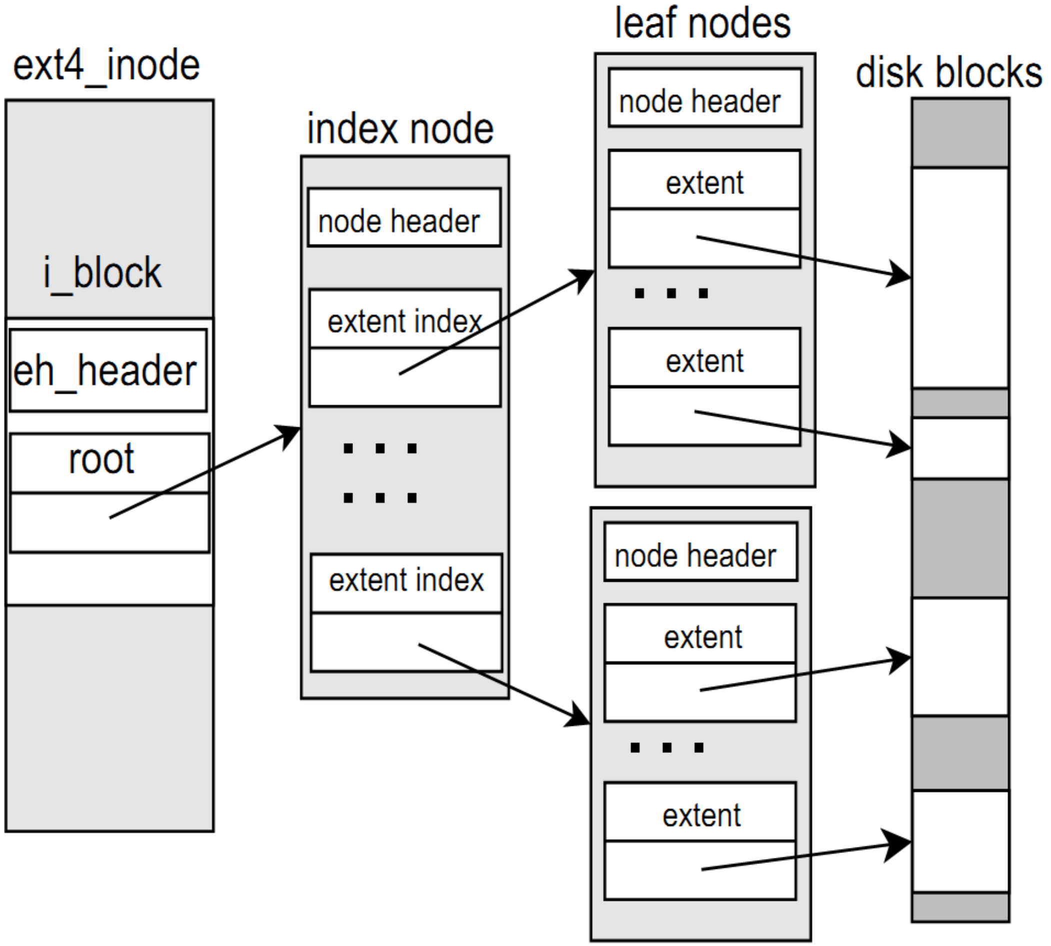
Ext4 file system – Extents 1

- One of the primary disadvantages of ext3 was its method of allocation. Files were allocated using a bit map of free space, which was not very fast nor very scalable.
- Ext3's format is very efficient for small files but horribly inefficient for large files. Ext4 replaces ext3's mechanism with extents to improve allocation and support a more efficient storage structure.
- An extent is simply a way to represent a contiguous sequence of blocks. In doing this, metadata shrinks, because instead of maintaining information about where a block is stored, the extent maintains information about where a long list of contiguous blocks is stored (thus reducing the overall metadata storage).
- Extents in ext4 adopt a layered approach to efficiently represent small files as well as extent trees to efficiently represent large files. For example, a single ext4 inode has sufficient space to reference four extents (where each extent represents a set of contiguous blocks).
- For large files (including those that are fragmented), an inode can reference an index node, each of which can reference a leaf node (referencing multiple extents). This **constant depth extent tree** provides a rich representation scheme for large, potentially sparse files. The nodes also include self-checking mechanisms to further protect against file system corruption.

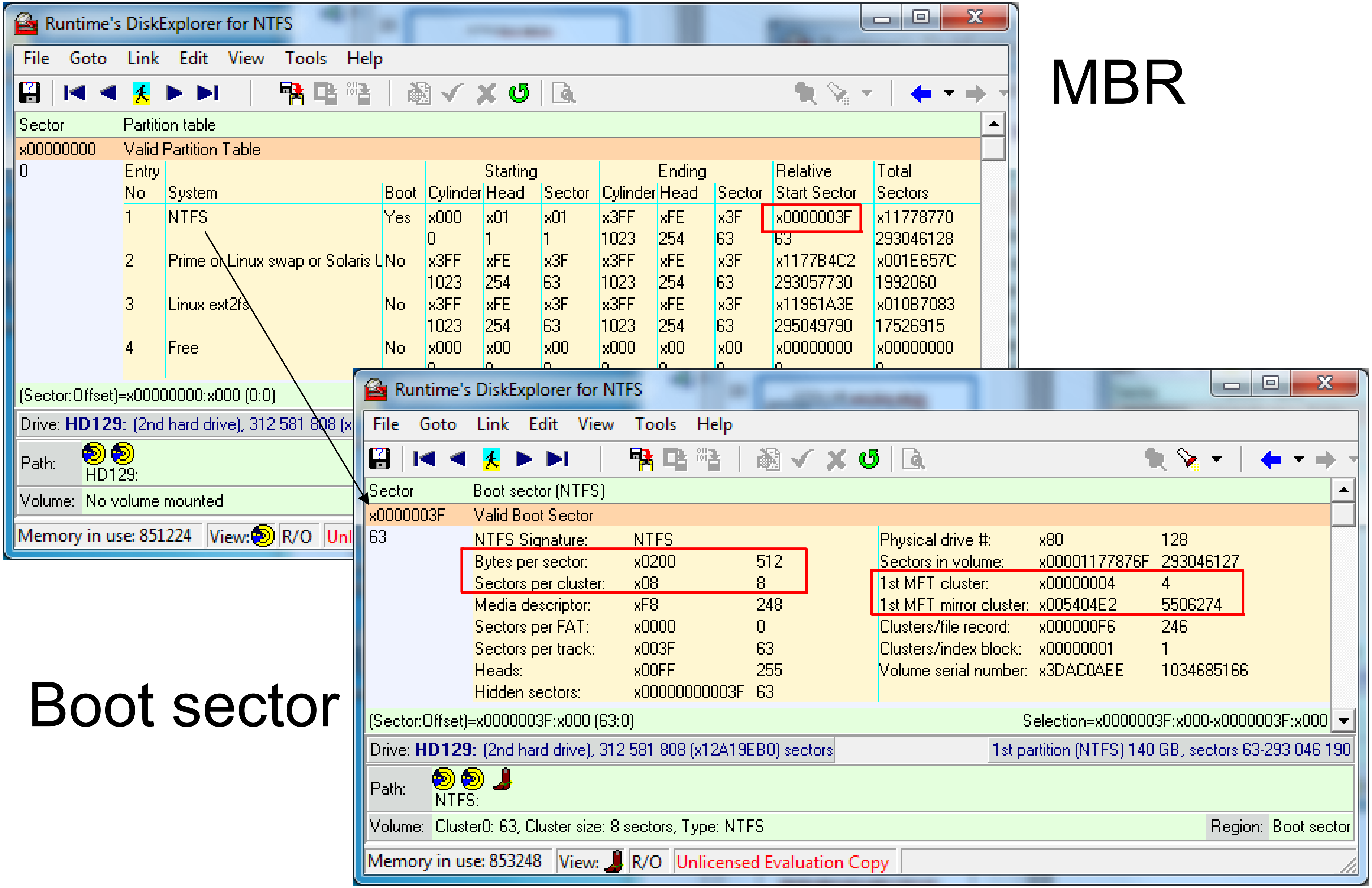
Read more: http://www.kernel.org/doc/ols/2007/ols2007v2-pages-21-34.pdf

Ext4 file system – Extents 2

- Ext4 supports two block maps. Extent map is more efficient and can handle large file in comparison with the old indirect block map



Disk MBR and NTFS Boot sector (VBR/PBR)



MBR

Runtime's DiskExplorer for NTFS

Sector	Partition table										
x00000000	Valid Partition Table										
0	Entry			Starting			Ending			Relative	Total
	No	System	Boot	Cylinder	Head	Sector	Cylinder	Head	Sector	Start Sector	Sectors
	1	NTFS	Yes	x000	x01	x01	x3FF	xFE	x3F	x0000003F	x11778770
				0	1	1	1023	254	63	63	293046128
	2	Prime or Linux swap or Solaris L	No	x3FF	xFE	x3F	x3FF	xFE	x3F	x1177B4C2	x001E657C
				1023	254	63	1023	254	63	293057730	1992060
	3	Linux ext2fs	No	x3FF	xFE	x3F	x3FF	xFE	x3F	x11961A3E	x010B7083
				1023	254	63	1023	254	63	295049790	17526915
	4	Free	No	x000	x00	x00	x000	x00	x00	x00000000	x00000000
				0	0	0	0	0	0	0	0

(Sector:Offset)=x00000000:x000 (0:0)

Drive: **HD129:** (2nd hard drive), 312 581 808 (x

Path: HD129:

Volume: No volume mounted

Memory in use: 851224 View: R/O Unl

Runtime's DiskExplorer for NTFS

Sector	Boot sector (NTFS)					
x0000003F	Valid Boot Sector					
63	NTFS Signature:	NTFS		Physical drive #:	x80	128
	Bytes per sector:	x0200	512	Sectors in volume:	x00001177876F	293046127
	Sectors per cluster:	x08	8	1st MFT cluster:	x00000004	4
	Media descriptor:	xF8	248	1st MFT mirror cluster:	x005404E2	5506274
	Sectors per FAT:	x0000	0	Clusters/file record:	x000000F6	246
	Sectors per track:	x003F	63	Clusters/index block:	x00000001	1
	Heads:	x00FF	255	Volume serial number:	x3DAC0AEE	1034685166
	Hidden sectors:	x000000000003F	63			

(Sector:Offset)=x0000003F:x000 (63:0) Selection=x0000003F:x000-x0000003F:x000

Drive: **HD129:** (2nd hard drive), 312 581 808 (x12A19EB0) sectors 1st partition (NTFS) 140 GB, sectors 63-293 046 190

Path: NTFS:

Volume: Cluster0: 63, Cluster size: 8 sectors, Type: NTFS Region: Boot sector

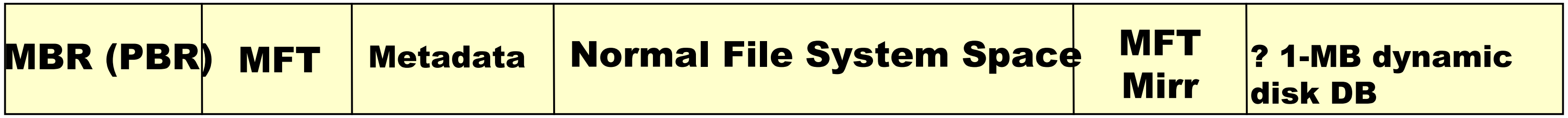
Memory in use: 853248 View: R/O Unlicensed Evaluation Copy

Boot sector

NTFS (ej dokumenterat av MS)

http://technet2.microsoft.com/WindowsServer/en/library/8cc5891d-bf8e-4164-862d-dac5418c59481033.mspx

- NTFS har ingen speciell layout förutom i MBR/PBR
  - Alla administrativa metadata är vanliga filer som är synliga och kan finnas varsomhelst i volymen
- NTFS ökar komplexiteten i en forensisk analys
  - Eftersom filer (kluster) över tiden allokeras och deallokeras (skapas, raderas, ändrar storlek)
  - NTFS återanvänder gamla MFT entryn innan nya skapas
  - Gör det svårt att binda en viss fil till vissa kluster
- Om metadata blir korrupt eller skadat är det svårt att återskapa filerna
  - Specialiserade verktyg är att föredra



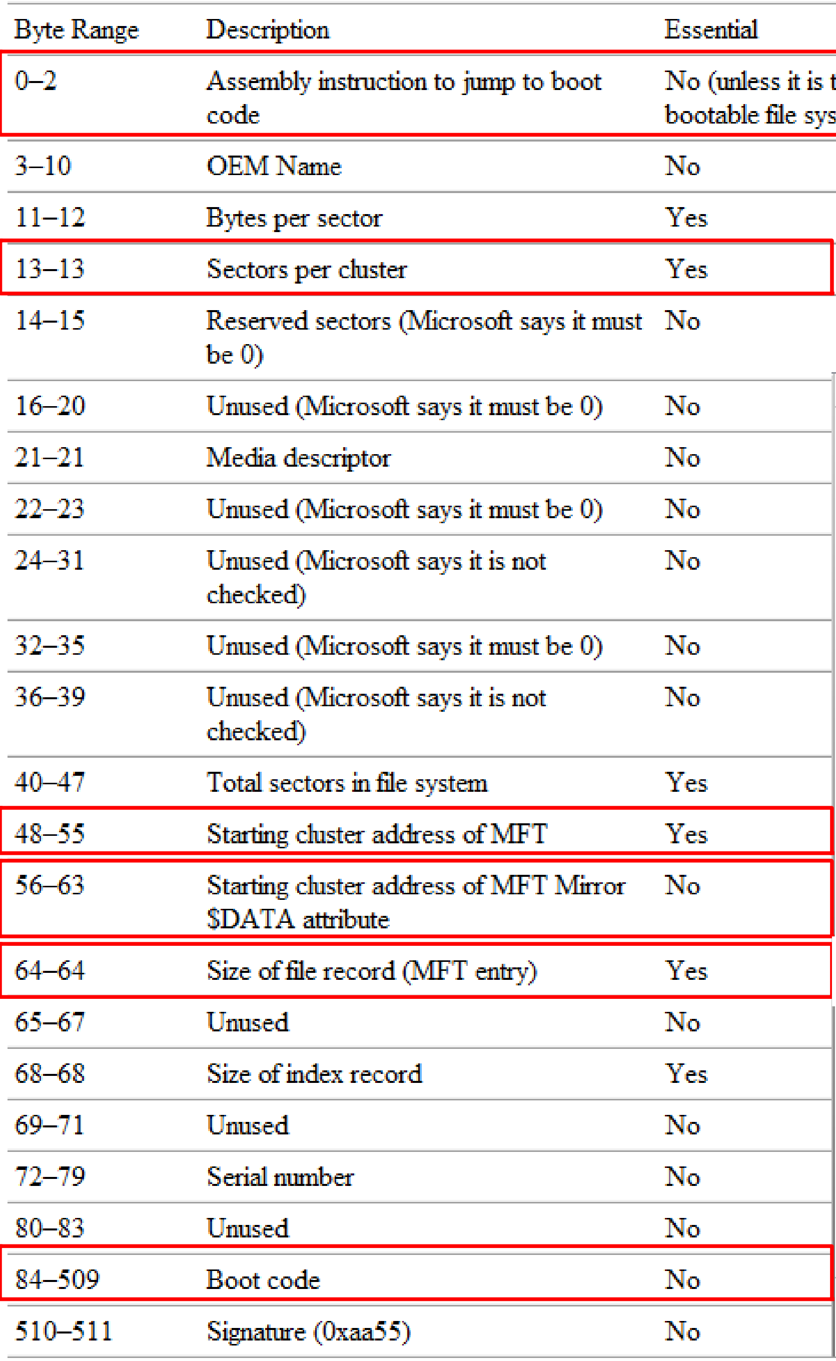
MBR (PBR)	**MFT**	**Metadata**	**Normal File System Space**	**MFT Mirr**	**? 1-MB dynamic disk DB**

MBR = Master Boot Record
MFT = Master File Table
PBR = Partition Boot Record (VBR)

Generell layout av ett NTFS filsystem

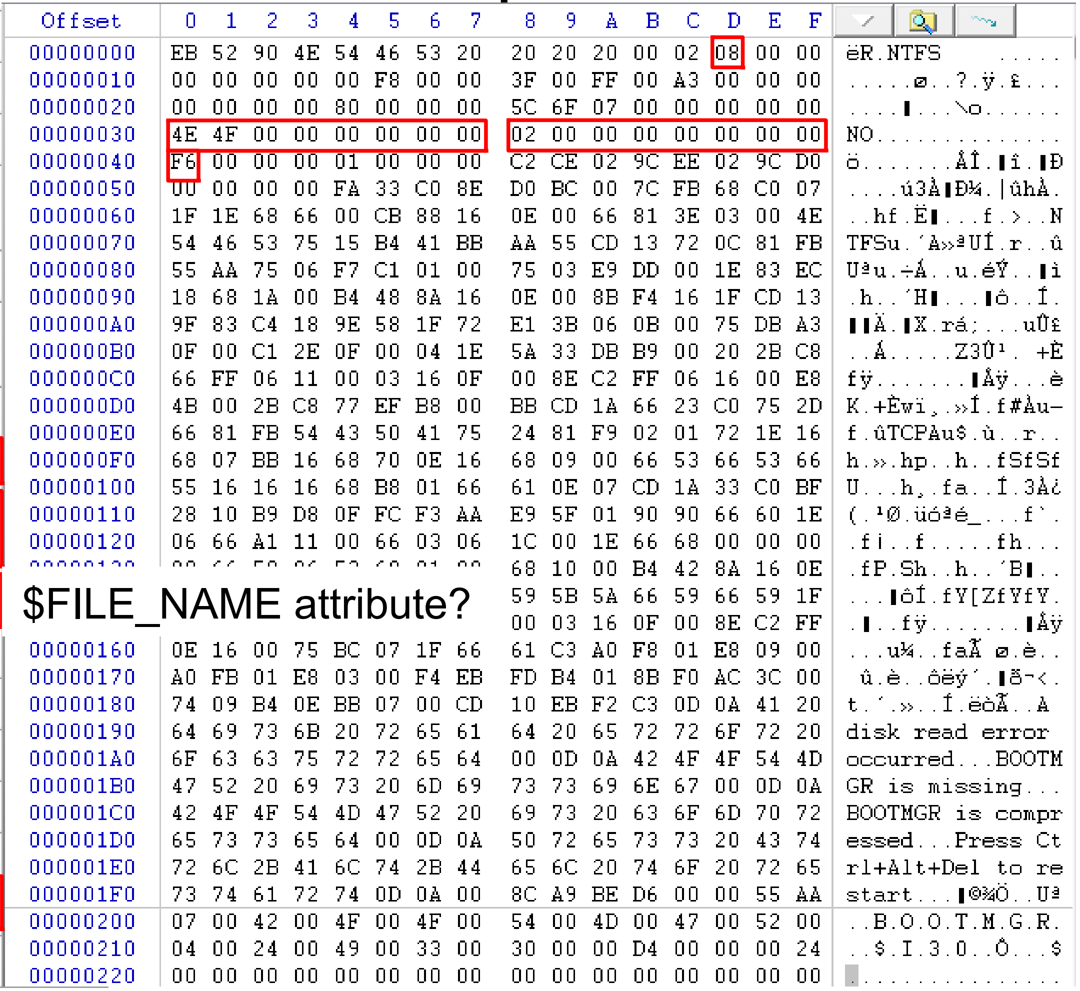
NTFS boot sector

Table 13.18. Data structure for the boot sector.



Byte Range	Description	Essential
0–2	Assembly instruction to jump to boot code	No (unless it is the bootable file system)
3–10	OEM Name	No
11–12	Bytes per sector	Yes
13–13	Sectors per cluster	Yes
14–15	Reserved sectors (Microsoft says it must be 0)	No
16–20	Unused (Microsoft says it must be 0)	No
21–21	Media descriptor	No
22–23	Unused (Microsoft says it must be 0)	No
24–31	Unused (Microsoft says it is not checked)	No
32–35	Unused (Microsoft says it must be 0)	No
36–39	Unused (Microsoft says it is not checked)	No
40–47	Total sectors in file system	Yes
48–55	Starting cluster address of MFT	Yes
56–63	Starting cluster address of MFT Mirror $DATA attribute	No
64–64	Size of file record (MFT entry)	Yes
65–67	Unused	No
68–68	Size of index record	Yes
69–71	Unused	No
72–79	Serial number	No
80–83	Unused	No
84–509	Boot code	No
510–511	Signature (0xaa55)	No

- Example VBR

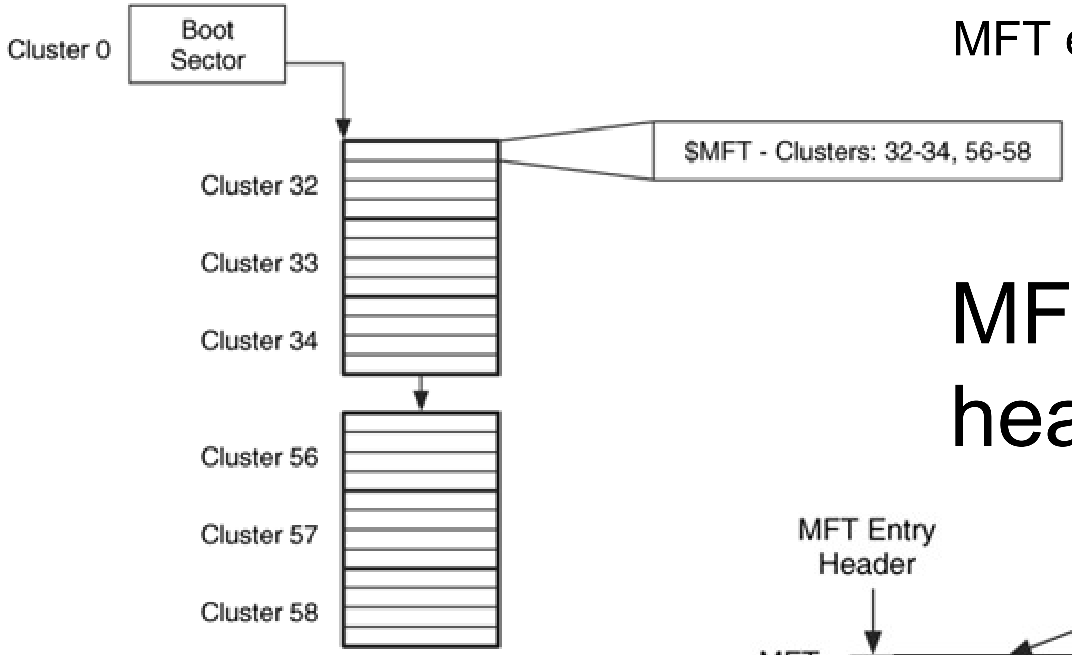


```
Offset     0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F
00000000  EB 52 90 4E 54 46 53 20  20 20 20 00 02 08 00 00  ëR.NTFS     .....
00000010  00 00 00 00 00 F8 00 00  3F 00 FF 00 A3 00 00 00  .....ø..?.ÿ.£...
00000020  00 00 00 00 80 00 00 00  5C 6F 07 00 00 00 00 00  ....▮...\o......
00000030  4E 4F 00 00 00 00 00 00  02 00 00 00 00 00 00 00  NO..............
00000040  F6 00 00 00 01 00 00 00  C2 CE 02 9C EE 02 9C D0  ö.......ÂÎ.▮î.▮Ð
00000050  00 00 00 00 FA 33 C0 8E  D0 BC 00 7C FB 68 C0 07  ....ú3À▮Ð¼.|ûhÀ.
00000060  1F 1E 68 66 00 CB 88 16  0E 00 66 81 3E 03 00 4E  ..hf.Ë▮...f.>..N
00000070  54 46 53 75 15 B4 41 BB  AA 55 CD 13 72 0C 81 FB  TFSu.´A»ªUÍ.r..û
00000080  55 AA 75 06 F7 C1 01 00  75 03 E9 DD 00 1E 83 EC  Uªu.÷Á..u.éÝ..▮ì
00000090  18 68 1A 00 B4 48 8A 16  0E 00 8B F4 16 1F CD 13  .h..´H▮...▮ô..Í.
000000A0  9F 83 C4 18 9E 58 1F 72  E1 3B 06 0B 00 75 DB A3  ▮▮Ä.▮X.rá;...uÛ£
000000B0  0F 00 C1 2E 0F 00 04 1E  5A 33 DB B9 00 20 2B C8  ..Á.....Z3Û¹. +È
000000C0  66 FF 06 11 00 03 16 0F  00 8E C2 FF 06 16 00 E8  fÿ.......▮Âÿ...è
000000D0  4B 00 2B C8 77 EF B8 00  BB CD 1A 66 23 C0 75 2D  K.+Èwï¸.»Í.f#Àu-
000000E0  66 81 FB 54 43 50 41 75  24 81 F9 02 01 72 1E 16  f.ûTCPAu$.ù..r..
000000F0  68 07 BB 16 68 70 0E 16  68 09 00 66 53 66 53 66  h.».hp..h..fSfSf
00000100  55 16 16 16 68 B8 01 66  61 0E 07 CD 1A 33 C0 BF  U...h¸.fa..Í.3À¿
00000110  28 10 B9 D8 0F FC F3 AA  E9 5F 01 90 90 66 60 1E  (.¹Ø.üóªé_...f`.
00000120  06 66 A1 11 00 66 03 06  1C 00 1E 66 68 00 00 00  .f¡..f.....fh...
                                   68 10 00 B4 42 8A 16 0E  .fP.Sh..h..´B▮..
                                   59 5B 5A 66 59 66 59 1F  ...▮ôÍ.fY[ZfYfY.
                                   00 03 16 0F 00 8E C2 FF  .▮..fÿ.......▮Âÿ
00000160  0E 16 00 75 BC 07 1F 66  61 C3 A0 F8 01 E8 09 00  ...u¼..faà ø.è..
00000170  A0 FB 01 E8 03 00 F4 EB  FD B4 01 8B F0 AC 3C 00   û.è..ôëý´.▮ð¬<.
00000180  74 09 B4 0E BB 07 00 CD  10 EB F2 C3 0D 0A 41 20  t.´.»..Í.ëòÃ..A 
00000190  64 69 73 6B 20 72 65 61  64 20 65 72 72 6F 72 20  disk read error 
000001A0  6F 63 63 75 72 72 65 64  00 0D 0A 42 4F 4F 54 4D  occurred...BOOTM
000001B0  47 52 20 69 73 20 6D 69  73 73 69 6E 67 00 0D 0A  GR is missing...
000001C0  42 4F 4F 54 4D 47 52 20  69 73 20 63 6F 6D 70 72  BOOTMGR is compr
000001D0  65 73 73 65 64 00 0D 0A  50 72 65 73 73 20 43 74  essed...Press Ct
000001E0  72 6C 2B 41 6C 74 2B 44  65 6C 20 74 6F 20 72 65  rl+Alt+Del to re
000001F0  73 74 61 72 74 0D 0A 00  8C A9 BE D6 00 00 55 AA  start...▮©¾Ö..Uª
00000200  07 00 42 00 4F 00 4F 00  54 00 4D 00 47 00 52 00  ..B.O.O.T.M.G.R.
00000210  04 00 24 00 49 00 33 00  30 00 00 D4 00 00 00 24  ..$.I.3.0..Ô...$
00000220  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  ................
```

$FILE_NAME attribute?

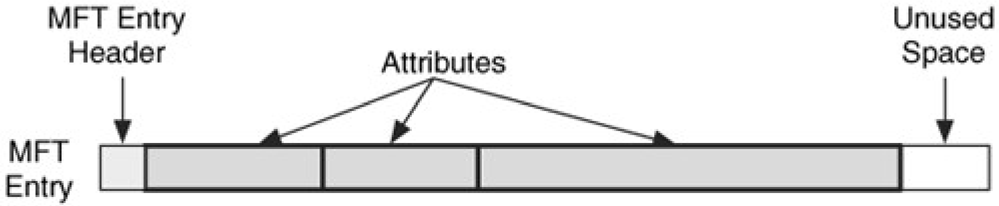
$MFT och MFT entry

- Var MFT börjar pekas ut i boot sectorn
- MFT och dess layout



MFT entry 0, MFT beskriver sig själv!

MFT entry med header och attribut

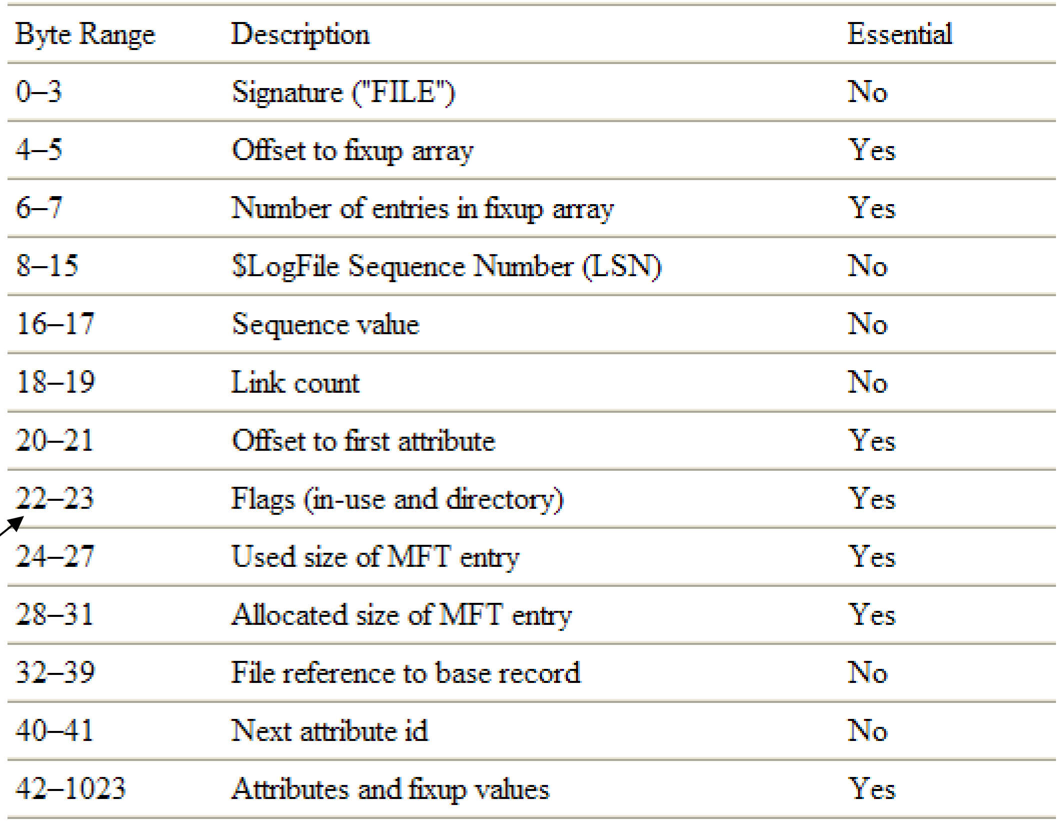


MFT entry innehåll

Varje MFT entry är 1kB

- De första 42 byten i ett MFT entry används till 12 fixerade fält
- De överblivande 982 byten som initialt är tomt kan användas till att lagra vadsomhelst så länge det är mindre
- Det första fältet innehåller en signatur
  - FILE
  - BAAD (trasig)
- Flags fältet
  - Används filen?
  - Mapp eller fil?

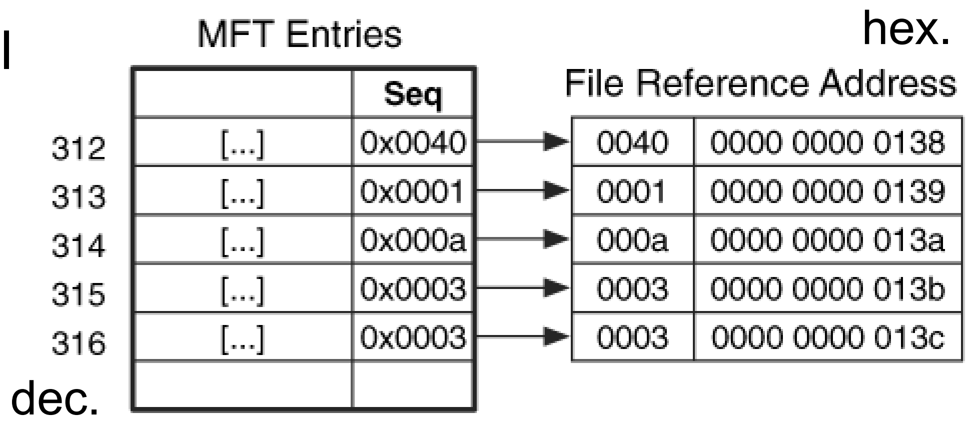
Table 13.1. Data structure for a basic MFT entry.



Byte Range	Description	Essential
0–3	Signature ("FILE")	No
4–5	Offset to fixup array	Yes
6–7	Number of entries in fixup array	Yes
8–15	$LogFile Sequence Number (LSN)	No
16–17	Sequence value	No
18–19	Link count	No
20–21	Offset to first attribute	Yes
22–23	Flags (in-use and directory)	Yes
24–27	Used size of MFT entry	Yes
28–31	Allocated size of MFT entry	Yes
32–39	File reference to base record	No
40–41	Next attribute id	No
42–1023	Attributes and fixup values	Yes

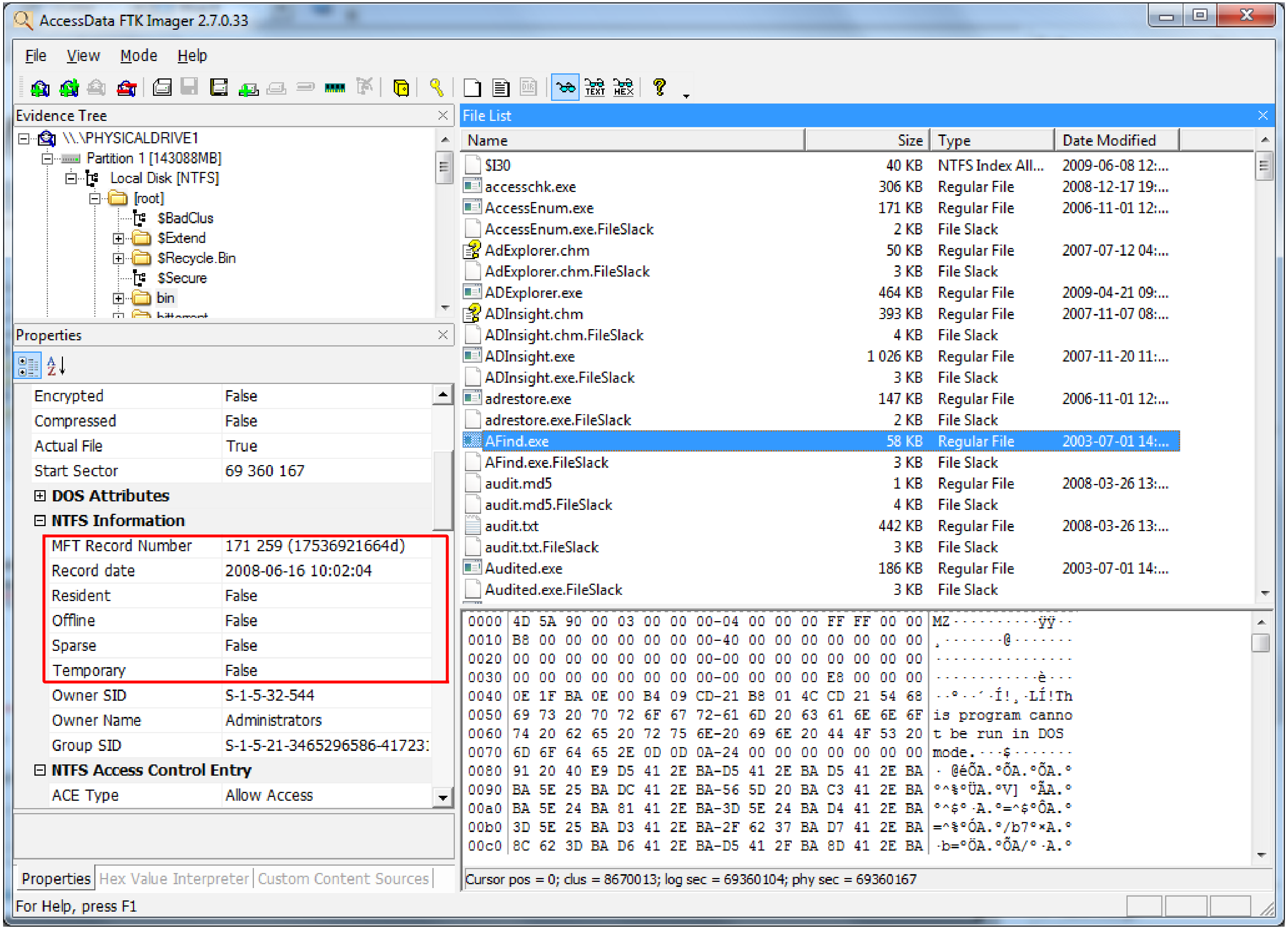
MFT innehåll/adresser

- MFT entrys raderas inte efter att ha skapats
- Om en fil inte kan få rum med sina attribut i ett entry kan filen använda multipla entryn
    - Om detta inträffar så kallas första entryt ”base file record” eller ”base file MFT”
    - Alla underliggande MFT entries har en referens till bas entryt i sitt fixerade fält (32-39)
- Varje MFT entry adresseras sekventiellt med 48 bitar
- Maximala MFT adressen ändras i takt med att MFT växer
- Varje MFT entry har även ett 16-bit sekvensnummer som inkrementeras varje gång entryt allokeras, fält (16-17)
- MFT entryt (index) och sekvensnumret kombineras till en 64-bitars filreferens adress (se bild)
- Sekvensnumret kan användas till
    - Detektera korrupt FS state
    - Del av ny fil
    - Återskapa raderad data



		Seq		File Reference Address
312	[...]	0x0040	0040	0000 0000 0138
313	[...]	0x0001	0001	0000 0000 0139
314	[...]	0x000a	000a	0000 0000 013a
315	[...]	0x0003	0003	0000 0000 013b
316	[...]	0x0003	0003	0000 0000 013c

MFT entry/record # kopplad till fil


AccessData FTK Imager 2.7.0.33
File View Mode Help
Evidence Tree
File List
Properties
Properties | Hex Value Interpreter | Custom Content Sources
For Help, press F1

```
\\.\PHYSICALDRIVE1
  Partition 1 [143088MB]
    Local Disk [NTFS]
      [root]
        $BadClus
        $Extend
        $Recycle.Bin
        $Secure
        bin
        bittorrent
```

Encrypted	False
Compressed	False
Actual File	True
Start Sector	69 360 167
DOS Attributes	
NTFS Information	
MFT Record Number	171 259 (17536921664d)
Record date	2008-06-16 10:02:04
Resident	False
Offline	False
Sparse	False
Temporary	False
Owner SID	S-1-5-32-544
Owner Name	Administrators
Group SID	S-1-5-21-3465296586-41723
NTFS Access Control Entry	
ACE Type	Allow Access

Name	Size	Type	Date Modified
$I30	40 KB	NTFS Index All...	2009-06-08 12:...
accesschk.exe	306 KB	Regular File	2008-12-17 19:...
AccessEnum.exe	171 KB	Regular File	2006-11-01 12:...
AccessEnum.exe.FileSlack	2 KB	File Slack	
AdExplorer.chm	50 KB	Regular File	2007-07-12 04:...
AdExplorer.chm.FileSlack	3 KB	File Slack	
ADExplorer.exe	464 KB	Regular File	2009-04-21 09:...
ADInsight.chm	393 KB	Regular File	2007-11-07 08:...
ADInsight.chm.FileSlack	4 KB	File Slack	
ADInsight.exe	1 026 KB	Regular File	2007-11-20 11:...
ADInsight.exe.FileSlack	3 KB	File Slack	
adrestore.exe	147 KB	Regular File	2006-11-01 12:...
adrestore.exe.FileSlack	2 KB	File Slack	
AFind.exe	58 KB	Regular File	2003-07-01 14:...
AFind.exe.FileSlack	3 KB	File Slack	
audit.md5	1 KB	Regular File	2008-03-26 13:...
audit.md5.FileSlack	4 KB	File Slack	
audit.txt	442 KB	Regular File	2008-03-26 13:...
audit.txt.FileSlack	3 KB	File Slack	
Audited.exe	186 KB	Regular File	2003-07-01 14:...
Audited.exe.FileSlack	3 KB	File Slack	

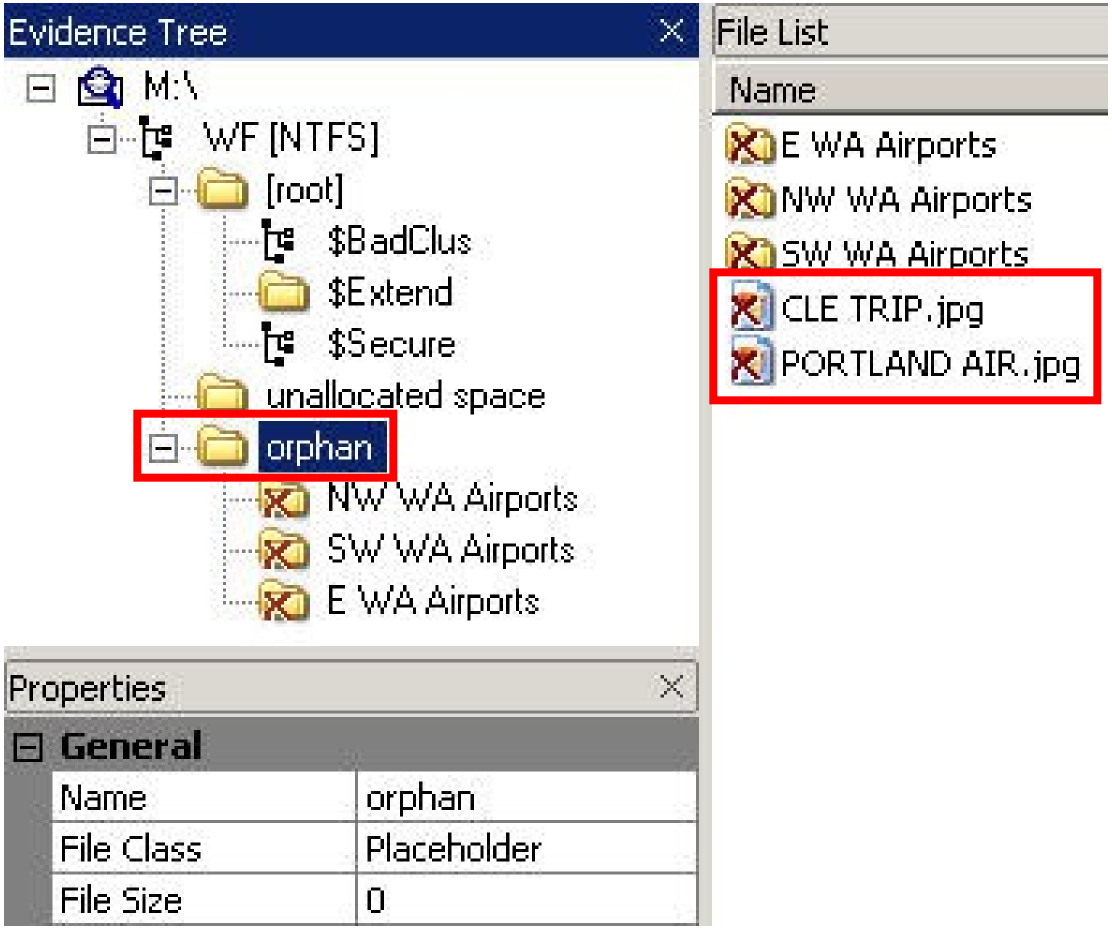
```
0000 4D 5A 90 00 03 00 00 00-04 00 00 00 FF FF 00 00 MZ··········ÿÿ··
0010 B8 00 00 00 00 00 00 00-40 00 00 00 00 00 00 00 ¸·······@·······
0020 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
0030 00 00 00 00 00 00 00 00-00 00 00 00 E8 00 00 00 ············è···
0040 0E 1F BA 0E 00 B4 09 CD-21 B8 01 4C CD 21 54 68 ··º··´·Í!¸·LÍ!Th
0050 69 73 20 70 72 6F 67 72-61 6D 20 63 61 6E 6E 6F is program canno
0060 74 20 62 65 20 72 75 6E-20 69 6E 20 44 4F 53 20 t be run in DOS 
0070 6D 6F 64 65 2E 0D 0D 0A-24 00 00 00 00 00 00 00 mode.···$·······
0080 91 20 40 E9 D5 41 2E BA-D5 41 2E BA D5 41 2E BA · @éÕA.ºÕA.ºÕA.º
0090 BA 5E 25 BA DC 41 2E BA-56 5D 20 BA C3 41 2E BA º^%ºÜA.ºV] ºÃA.º
00a0 BA 5E 24 BA 81 41 2E BA-3D 5E 24 BA D4 41 2E BA º^$º·A.º=^$ºÔA.º
00b0 3D 5E 25 BA D3 41 2E BA-2F 62 37 BA D7 41 2E BA =^%ºÓA.º/b7º×A.º
00c0 8C 62 3D BA D6 41 2E BA-D5 41 2F BA 8D 41 2E BA ·b=ºÖA.ºÕA/º·A.º
```

Cursor pos = 0; clus = 8670013; log sec = 69360104; phy sec = 69360167

File system metadata files

- Eftersom allt på volymen är allokerat till filer så måste metadata lagras i filer
- MS reserverar de 16 (24) första entryna i MFT till att lagra administrativ filsystem metadata
- Första MFT entryt är en beskrivning av MFT (sig själv)
- Dessa entryn börjar alltid med $ och stor bokstav
- Finns i root katalogen men döljs
- Testa tex. med FTK Imager (eller WinHex)!
  - Add Evidence Item -> Physical Drive
  - Markera [root]
- Orphan files
  - Pekar på parent (base) record i MFT som numera hanterar andra filer (dvs. orphan files är underliggande MFT entries utan förälder)
  - [server]\forensics\docs\AccessData\White Papers
    - wp.NT_Orphan_Files.en_us.pdf

The [Orphan] Folder



Evidence Tree

- M:\
  - WF [NTFS]
    - [root]
      - $BadClus
      - $Extend
      - $Secure
    - unallocated space
    - orphan
      - NW WA Airports
      - SW WA Airports
      - E WA Airports

File List

Name
E WA Airports
NW WA Airports
SW WA Airports
CLE TRIP.jpg
PORTLAND AIR.jpg

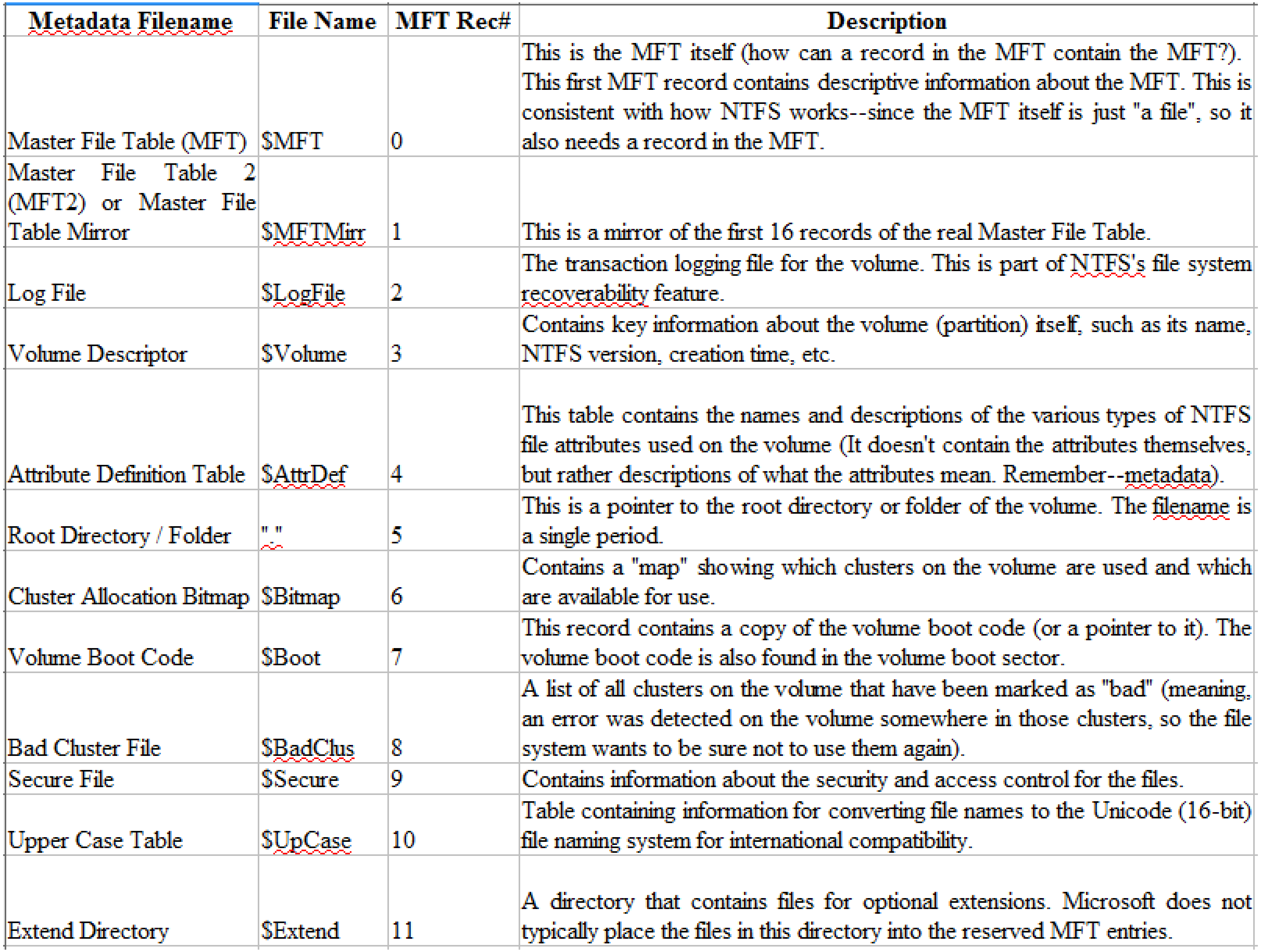
Properties

General

Name	orphan
File Class	Placeholder
File Size	0

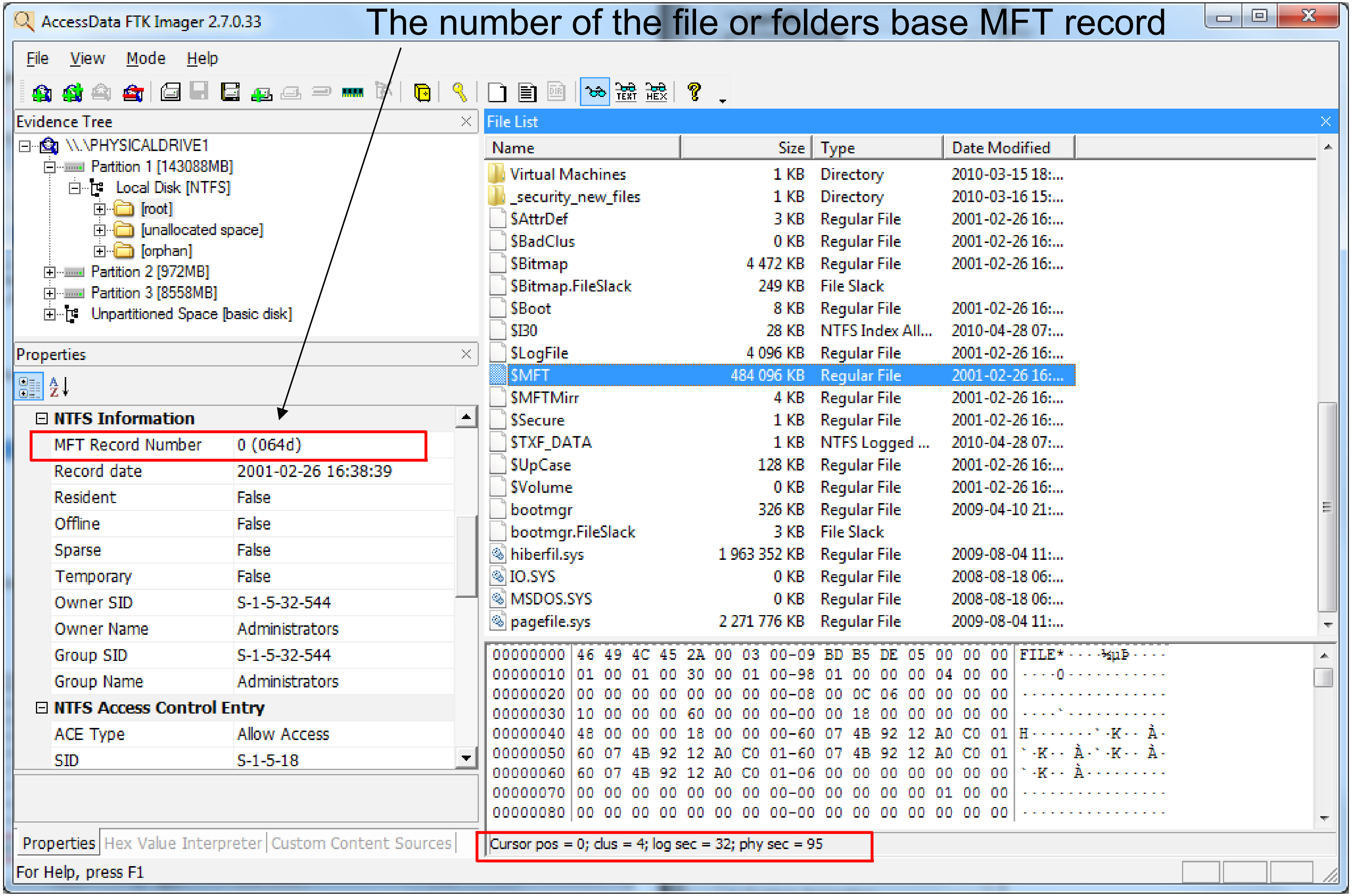
NOTE: The Orphan folder is created by FTK Imager and FTK to display recovered, orphaned files and folders

File system metadata files



Metadata Filename	File Name	MFT Rec#	Description
Master File Table (MFT)	$MFT	0	This is the MFT itself (how can a record in the MFT contain the MFT?). This first MFT record contains descriptive information about the MFT. This is consistent with how NTFS works--since the MFT itself is just "a file", so it also needs a record in the MFT.
Master File Table 2 (MFT2) or Master File Table Mirror	$MFTMirr	1	This is a mirror of the first 16 records of the real Master File Table.
Log File	$LogFile	2	The transaction logging file for the volume. This is part of NTFS's file system recoverability feature.
Volume Descriptor	$Volume	3	Contains key information about the volume (partition) itself, such as its name, NTFS version, creation time, etc.
Attribute Definition Table	$AttrDef	4	This table contains the names and descriptions of the various types of NTFS file attributes used on the volume (It doesn't contain the attributes themselves, but rather descriptions of what the attributes mean. Remember--metadata).
Root Directory / Folder	"."	5	This is a pointer to the root directory or folder of the volume. The filename is a single period.
Cluster Allocation Bitmap	$Bitmap	6	Contains a "map" showing which clusters on the volume are used and which are available for use.
Volume Boot Code	$Boot	7	This record contains a copy of the volume boot code (or a pointer to it). The volume boot code is also found in the volume boot sector.
Bad Cluster File	$BadClus	8	A list of all clusters on the volume that have been marked as "bad" (meaning, an error was detected on the volume somewhere in those clusters, so the file system wants to be sure not to use them again).
Secure File	$Secure	9	Contains information about the security and access control for the files.
Upper Case Table	$UpCase	10	Table containing information for converting file names to the Unicode (16-bit) file naming system for international compatibility.
Extend Directory	$Extend	11	A directory that contains files for optional extensions. Microsoft does not typically place the files in this directory into the reserved MFT entries.

FTK Imager och NTFS admin data



The number of the file or folders base MFT record

```
\\.\PHYSICALDRIVE1
  Partition 1 [143088MB]
    Local Disk [NTFS]
      [root]
      [unallocated space]
      [orphan]
  Partition 2 [972MB]
  Partition 3 [8558MB]
  Unpartitioned Space [basic disk]
```

NTFS Information

MFT Record Number	0 (064d)
Record date	2001-02-26 16:38:39
Resident	False
Offline	False
Sparse	False
Temporary	False
Owner SID	S-1-5-32-544
Owner Name	Administrators
Group SID	S-1-5-32-544
Group Name	Administrators

NTFS Access Control Entry

ACE Type	Allow Access
SID	S-1-5-18

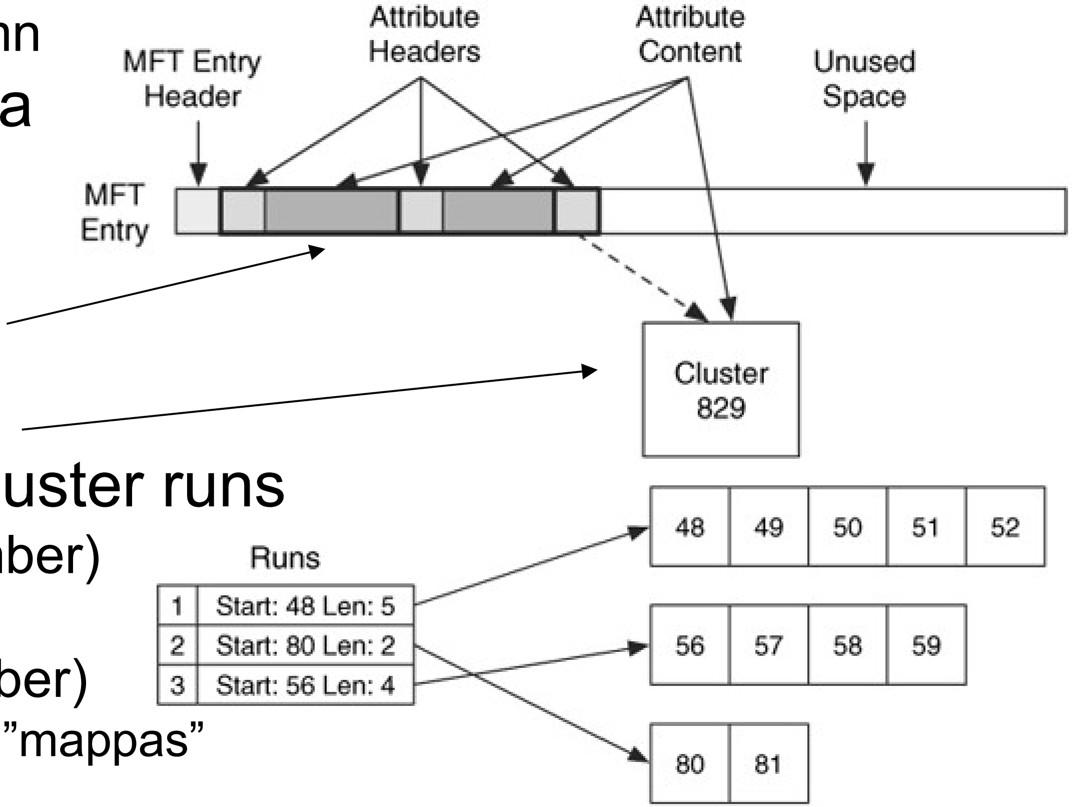
Name	Size	Type	Date Modified
Virtual Machines	1 KB	Directory	2010-03-15 18:...
_security_new_files	1 KB	Directory	2010-03-16 15:...
$AttrDef	3 KB	Regular File	2001-02-26 16:...
$BadClus	0 KB	Regular File	2001-02-26 16:...
$Bitmap	4 472 KB	Regular File	2001-02-26 16:...
$Bitmap.FileSlack	249 KB	File Slack	
$Boot	8 KB	Regular File	2001-02-26 16:...
$I30	28 KB	NTFS Index All...	2010-04-28 07:...
$LogFile	4 096 KB	Regular File	2001-02-26 16:...
$MFT	484 096 KB	Regular File	2001-02-26 16:...
$MFTMirr	4 KB	Regular File	2001-02-26 16:...
$Secure	1 KB	Regular File	2001-02-26 16:...
$TXF_DATA	1 KB	NTFS Logged ...	2010-04-28 07:...
$UpCase	128 KB	Regular File	2001-02-26 16:...
$Volume	0 KB	Regular File	2001-02-26 16:...
bootmgr	326 KB	Regular File	2009-04-10 21:...
bootmgr.FileSlack	3 KB	File Slack	
hiberfil.sys	1 963 352 KB	Regular File	2009-08-04 11:...
IO.SYS	0 KB	Regular File	2008-08-18 06:...
MSDOS.SYS	0 KB	Regular File	2008-08-18 06:...
pagefile.sys	2 271 776 KB	Regular File	2009-08-04 11:...

```
00000000  46 49 4C 45 2A 00 03 00-09 BD B5 DE 05 00 00 00  FILE*····½µÞ····
00000010  01 00 01 00 30 00 01 00-98 01 00 00 00 04 00 00  ····0···········
00000020  00 00 00 00 00 00 00 00-08 00 0C 06 00 00 00 00  ················
00000030  10 00 00 00 60 00 00 00-00 00 18 00 00 00 00 00  ····`···········
00000040  48 00 00 00 18 00 00 00-60 07 4B 92 12 A0 C0 01  H·······`·K·· À·
00000050  60 07 4B 92 12 A0 C0 01-60 07 4B 92 12 A0 C0 01  `·K·· À·`·K·· À·
00000060  60 07 4B 92 12 A0 C0 01-06 00 00 00 00 00 00 00  `·K·· À·········
00000070  00 00 00 00 00 00 00 00-00 00 00 00 01 00 00 00  ················
00000080  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
```

Cursor pos = 0; clus = 4; log sec = 32; phy sec = 95

MFT attribut concept

- Ett MFT entry har liten intern struktur – allt är egentligen attribut i NTFS
  - Tid, rättigheter, filnamn, fil-innehåll…
- Alla attribut lagrar två slags data
  - Header (generisk) och innehåll (specifik), se bild
  - Header = typ, storlek, namn
- Attributets innehåll kan ha vilket format som helst och storlek
  - Resident (lagring i MFT)
  - Non-resident (lagring i externt kluster)
- Non-resident kallas för cluster runs
  - LCN (Logical Cluster Number)
    - Filsystem adress
  - VCN (Virtual Cluster Number)
    - Filadress, kluster 0 – 10 ”mappas”

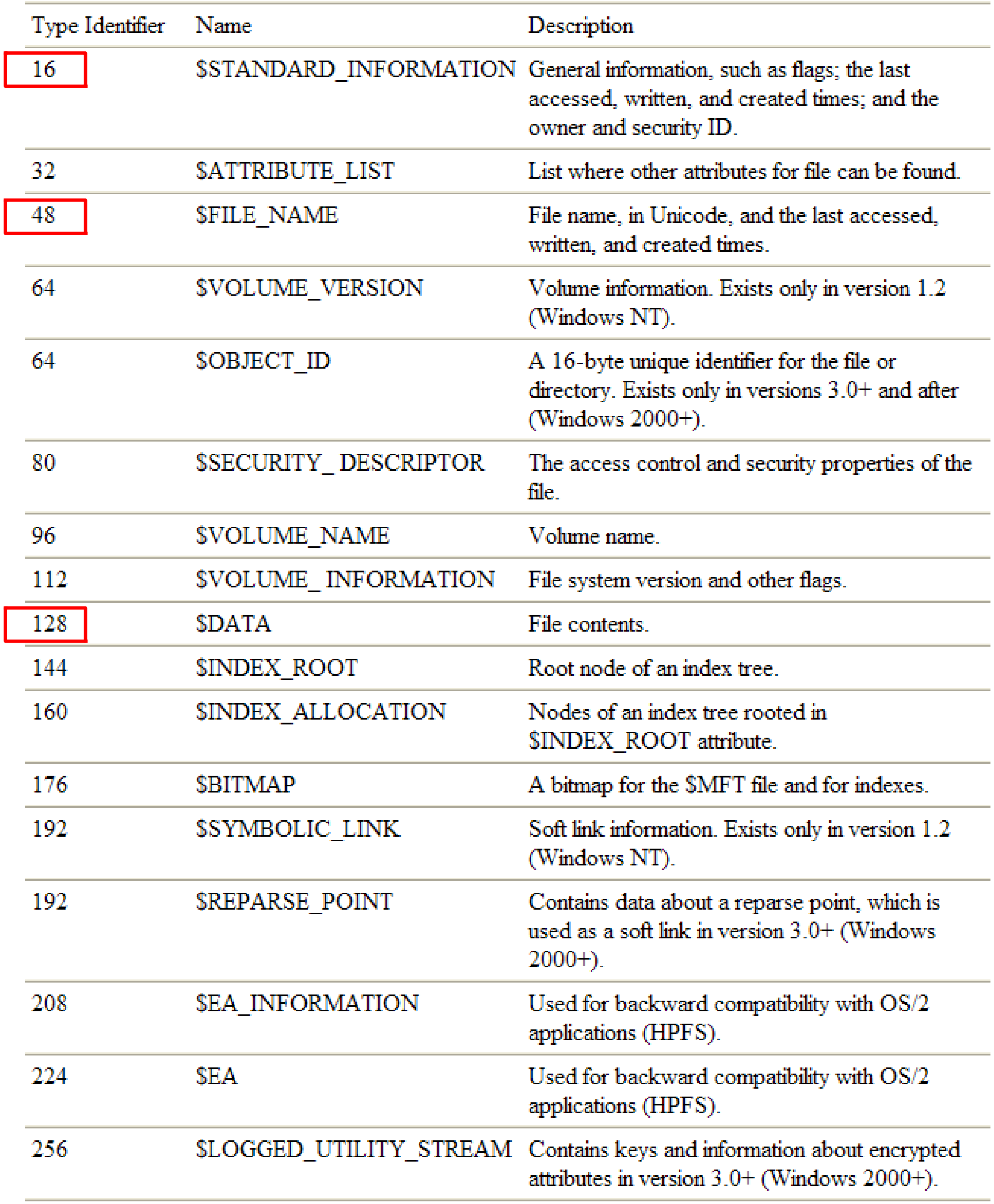


1	Start: 48 Len: 5
2	Start: 80 Len: 2
3	Start: 56 Len: 4

MFT standard attribut

- Ett nummer är definierat för varje typ av attribut
- Några ges i tabellen till höger
- Nästan alla entryn har ett 16 och 48 type id attribut
- Varje fil som har $Data > 700 byte innebär att den blir non-resident
- Filer med flera $Data indikerar ADS:er

Table 11.2. List of default MFT entry attribute types.

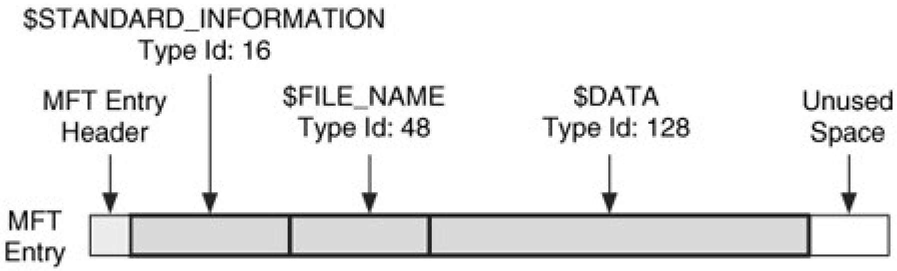


Type Identifier	Name	Description
16	$STANDARD_INFORMATION	General information, such as flags; the last accessed, written, and created times; and the owner and security ID.
32	$ATTRIBUTE_LIST	List where other attributes for file can be found.
48	$FILE_NAME	File name, in Unicode, and the last accessed, 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+).

NTFS MFT attribut

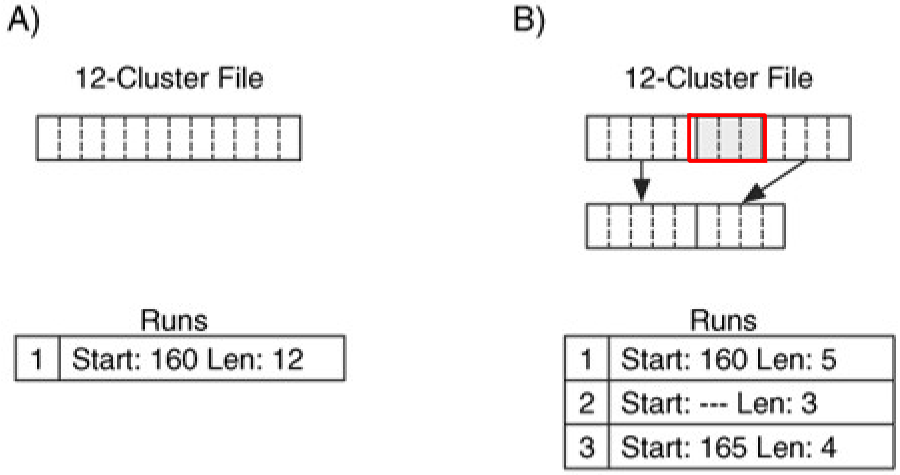
- Default ”$Data” attributet som skapas när en fil skapas har inget namn associerat till sig (main stream)
  - Däremot måste nya $Data attribut som läggs till ha det
- Varje MFT entry som är en mapp har ett $INDEX_ROOT attribut som innehåller information om alla filer och undermappar som finns i mappen
- Om mappen är stor så används även $INDEX_ALLOCATION och $BITMAP attributen för att lagra info
  - Ett mapp entry kan dessutom även ha $Data attribut
  - Ett mapp entry kan alltså lagra både filinnehåll, en lista med filer och submappar
- $INDEX_ROOT och $INDEX_ALLOCATION attributen för en mapp har namnet $I30

Entry med standard attribut



Speciella attribut

- Kap 12 i Carrier tar upp fördjupad analys av attribut
- Kap 13 i Carrier tar upp innehåll i attribut headers
- En fil kan ha upp till 65536 attribut! (2^16 type id)
  - Base MFT entry och non base MFT entrys för att få plats
  - Attribut headers måste alltid befinna sig i ett MFT entry
- Sparse attributes
  - Attribut med kluster som innehåller endast nollor skrivs inte till disken
  - Reducerar filens storlek genom att spara non-resident $DATA attribut som sparse
  - Typiskt innehåller sparse attributet i ”cluster run” bara storlek



A) Normal Layout

1	Start: 160 Len: 12

B) Sparse Layout

1	Start: 160 Len: 5
2	Start: --- Len: 3
3	Start: 165 Len: 4

Speciella attribut
Komprimering och kryptering

- Endast $DATA attributet bör komprimeras, måste vara non-resident
- Attributflaggan i $STANDARD_INFORMATION och $FILE_NAME indikerar om filen är komprimerad
  - Delas upp i compression units
    - Okomprimerade runs
    - Sparse runs
    - Komprimerade runs
- Endast $DATA attributet tillåts att krypteras (ej headern)
  - En $LOGGED_UTILITY_STREAM skapas för filen/katalogen som innehåller krypteringsnycklarna
  - En flagga sätts i $STANDARD_INFORMATION attributet och i varje attributs header vars attributinnehåll krypteras

$STANDARD_INFORMATION och $FILE_NAME samt flaggor

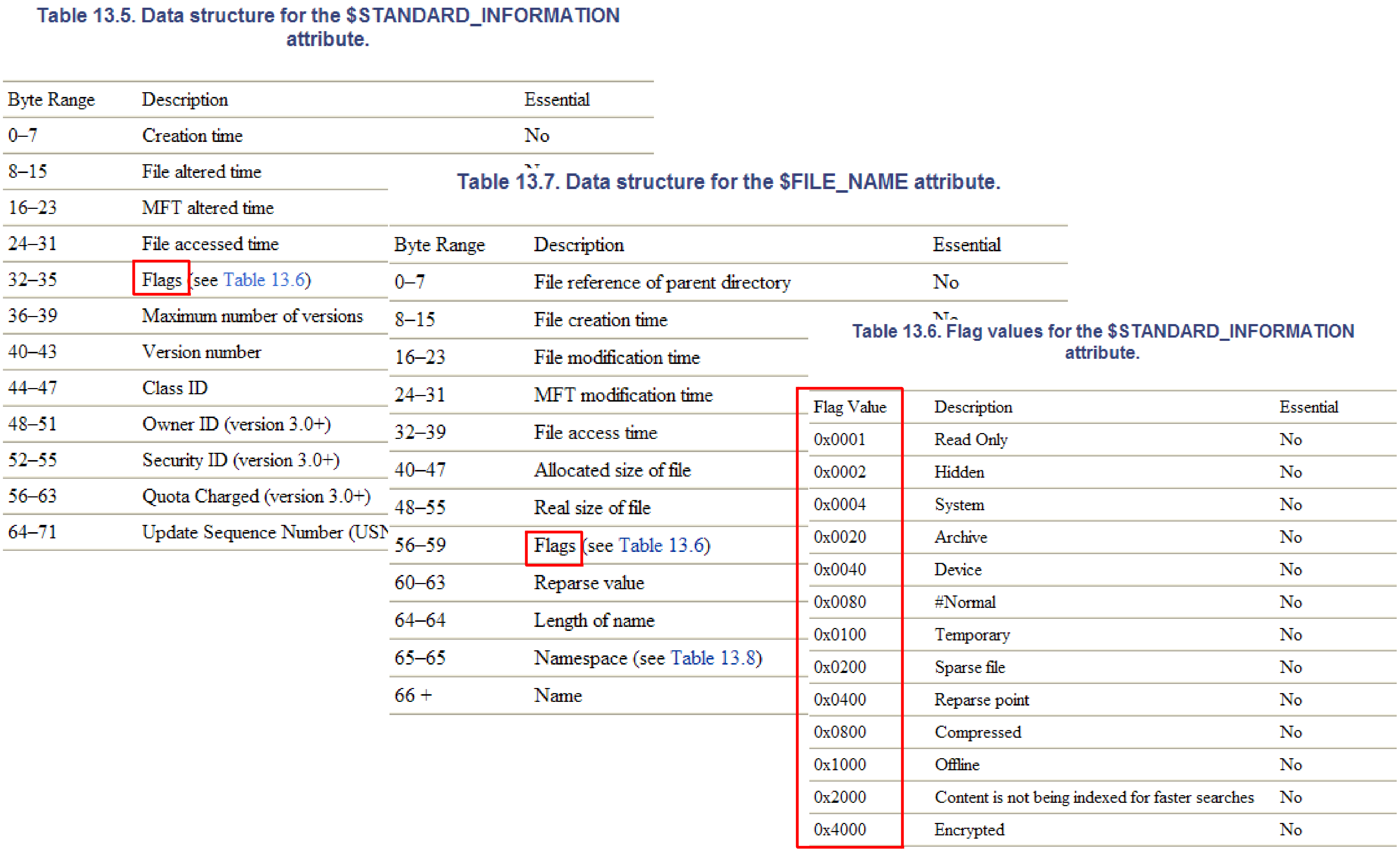


Table 13.5. Data structure for the $STANDARD_INFORMATION attribute.

Byte Range	Description	Essential
0–7	Creation time	No
8–15	File altered time	
16–23	MFT altered time	
24–31	File accessed time	
32–35	Flags (see Table 13.6)	
36–39	Maximum number of versions	
40–43	Version number	
44–47	Class ID	
48–51	Owner ID (version 3.0+)	
52–55	Security ID (version 3.0+)	
56–63	Quota Charged (version 3.0+)	
64–71	Update Sequence Number (USN	

Table 13.7. Data structure for the $FILE_NAME attribute.

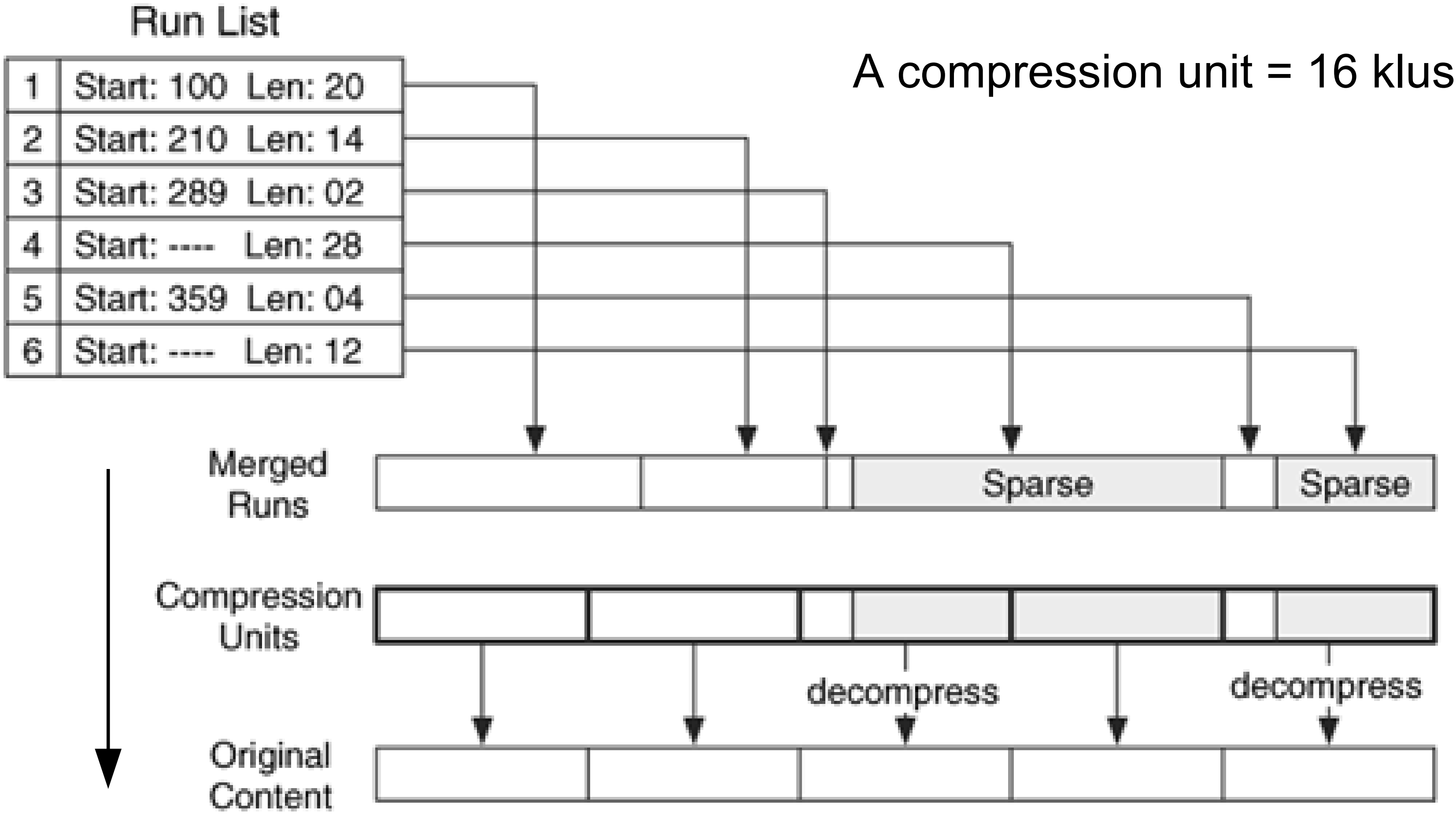
Byte Range	Description	Essential
0–7	File reference of parent directory	No
8–15	File creation time	
16–23	File modification time	
24–31	MFT modification time	
32–39	File access time	
40–47	Allocated size of file	
48–55	Real size of file	
56–59	Flags (see Table 13.6)	
60–63	Reparse value	
64–64	Length of name	
65–65	Namespace (see Table 13.8)	
66 +	Name	

Table 13.6. Flag values for the $STANDARD_INFORMATION attribute.

Flag Value	Description	Essential
0x0001	Read Only	No
0x0002	Hidden	No
0x0004	System	No
0x0020	Archive	No
0x0040	Device	No
0x0080	#Normal	No
0x0100	Temporary	No
0x0200	Sparse file	No
0x0400	Reparse point	No
0x0800	Compressed	No
0x1000	Offline	No
0x2000	Content is not being indexed for faster searches	No
0x4000	Encrypted	No

Komprimerat attribut med fragmenterade runs på ojämna units

A compression unit = 16 kluster



Run List

1	Start: 100 Len: 20
2	Start: 210 Len: 14
3	Start: 289 Len: 02
4	Start: ---- Len: 28
5	Start: 359 Len: 04
6	Start: ---- Len: 12

AnalyzeMFT with Python

- analyzeMFT.py is designed to fully parse the MFT file from an NTFS filesystem and present the results as accurately as possible in a format that allows further analysis with other tools

  http://www.integriography.com/

- At present, it parses the attributes from a $MFT file to produce the following output (only description and the first entry):
  - python analyzeMFT.py -f $MFT -o outfile.csv

```
"Record Number","Good","Active","Record type","Parent Folder","Record Sequence","Filename #1","Std Info Creation date",
"Std Info Modification date","Std Info Access date","Std Info Entry date","FN Info Creation date","FN Info Modification
date","FN Info Access date","FN Info Entry date","Object ID","Birth Volume ID","Birth Object ID","Birth Domain ID",
"Filename #2","FN Info Creation date","FN Info Modify date","FN Info Access date","FN Info Entry date","Filename #3",
"FN Info Creation date","FN Info Modify date","FN Info Access date","FN Info Entry date","Filename #4",
"FN Info Creation date","FN Info Modify date","FN Info Access date","FN Info Entry date","Standard Information",
"Attribute List","Filename","Object ID","Volume Name","Volume Info","Data","Index Root","Index Allocation","Bitmap",
"Reparse Point","EA Information","EA","Property Set","Logged Utility Stream"

"0","Good","Active","File","5 - 5","1","$MFT","2007/07/31 19:16:13.734373","2007/07/31 19:16:13.734373",
"2007/07/31 19:16:13.734373","2007/07/31 19:16:13.734373","2007/07/31 19:16:13.734373","2007/07/31 19:16:13.734373",
"2007/07/31 19:16:13.734373","2007/07/31 19:16:13.734373","","","","","","","","","","","","","","","","","","","",
"True","False","False","False","False","False","True","False","False","True","False","False","False","False","False"
```

ReFS (Resilient File System)

- Improved reliability for on-disk structures
  - ReFS uses B+ trees for all on-disk structures
  - The maximum file size is 16 Exbibytes (everything is 64-bit) and maximum volume size is 1 Yobibyte
  - Metadata and file data are organized into tables similar to relational database
  - File names and file paths are each limited to a 32 KB Unicode text string
- Built-in resiliency
  - ReFS employs an allocation-on-write update strategy for metadata
  - All ReFS metadata has built-in 64-bit checksums
  - No need to periodically run error-checking tools such as CHKDSK when using ReFS
- Compatibility with existing APIs and technologies
  - ReFS does not require new system APIs and most file system filters continue to work with ReFS volumes
  - ReFS supports many existing Windows and NTFS features as encryption, ACLs, symbolic links etc.
- Some NTFS features are not supported in ReFS
  - ADS will disappear, EFS file level compression, sparse files, ...
  - Will not work with earlier Windows than 8 - only 64-bit support, **no booting**, ...

More info: http://en.wikipedia.org/wiki/Windows_Server_2012#ReFS