System forensic

Credit: SystemForensics - YouTube

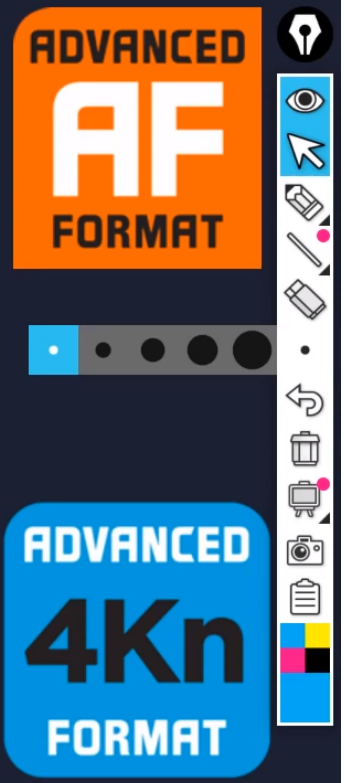
Noted by: Tsof (@tsof_relox) / X

```
Many thanks to @systemforensics for the great lecture series, myself and many people in the DF community learned a lot from you. With easy-to-understand content, logical presentation and immediately applicable examples. The only problem is the sound, which I don't even remember affecting my learning process.

If you use this note of mine and find it useful, I hope you can take a moment to subscribe and like @systemforensics products. After all he deserves it.
```

Disk layout - MBR - MPT - GPT

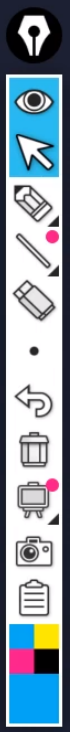
DISK LAYOUT – UNDERSTANDING SECTORS

- Each sector stores a fixed amount of user-accessible data, traditionally 512 bytes **for hard disk drives** (HDDs) Newer HDDs use 4096-byte (4 KiB) sectors, which are known as the Advanced Format (AF). The sector is the minimum storage unit of a hard drive.
- Many host computer hardware and software components assume the hard drive is configured around 512-byte sector boundaries.
- In order to maintain compatibility with legacy computing components, many hard disk drive suppliers support Advanced Format technologies on the recording media coupled with 512-byte conversion firmware, and referred to as Advanced Format 512e, or 512 emulation drives.



DISK LAYOUT – UNDERSTANDING PARTITIONS

- Every sector on a drive is addressed by LBA (Logical Block Address), starting from Sector 0
- Partitioning is logical dividing of the drive storage space (all addressable physical sectors) into several pieces (partitions)
- You can allocate only consecutive sectors into one partition
- The number of possible partitions on a drive will depend on the chosen Partitioning Style



DISK LAYOUT - PARTITION STYLES: MBR

- MBR – stands for Master Boot Record
- MBR style is common for the external USB drives (memory sticks, removable hard drives, SD-cards and Linux systems)
- MBR is located in Sector 0 and contains the Master Partition Table (MPT)
- MBR has a flag 0x55AA at the end of the sector
- Master Partition Table can accommodate up to 4 partition entries - Primary or Extended

DISK LAYOUT - PARTITION STYLES: MPT

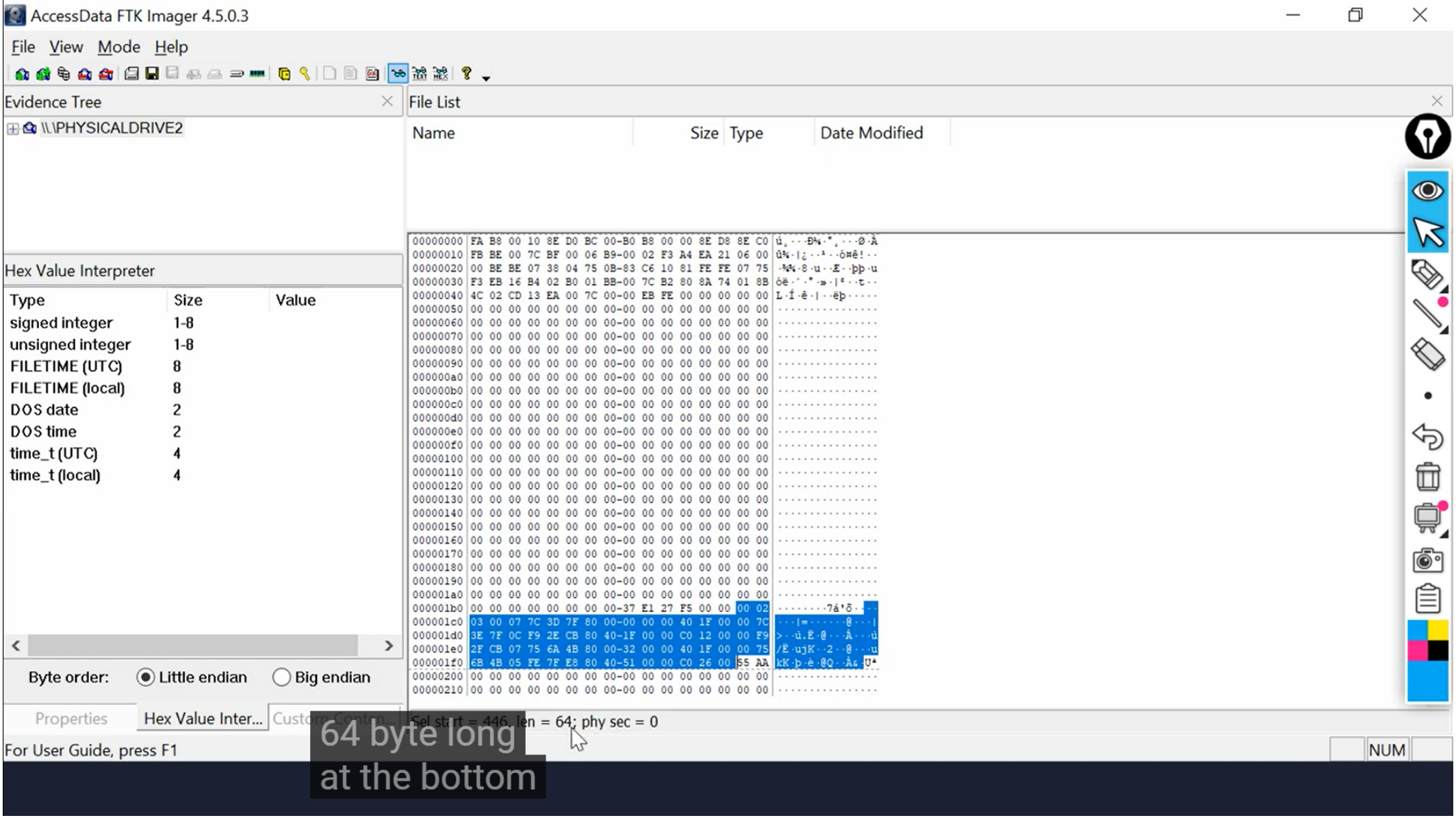
- MPT – Master Partition table, 64 bytes long
- Located at the end of the MBR sector, at the offset 446 bytes from the beginning of the sector
- Each partition entry is 16 bytes long: offsets within entry:

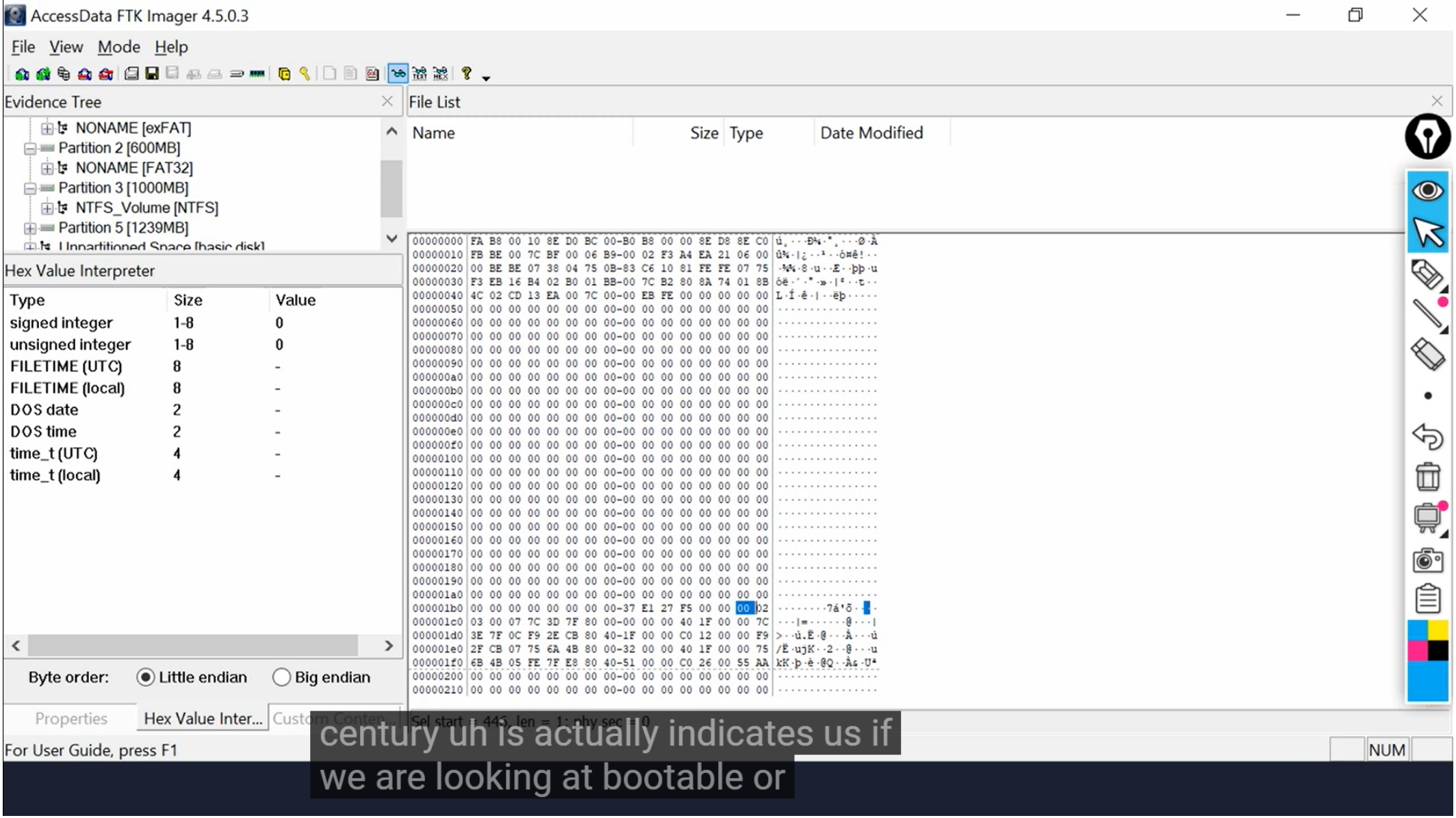
  offset 0 (1st byte) – boot indicator (0x80 – bootable partition, OS system partition)

  offset 4 (5th byte) – Partition type descriptor – indicates file system type on the partition

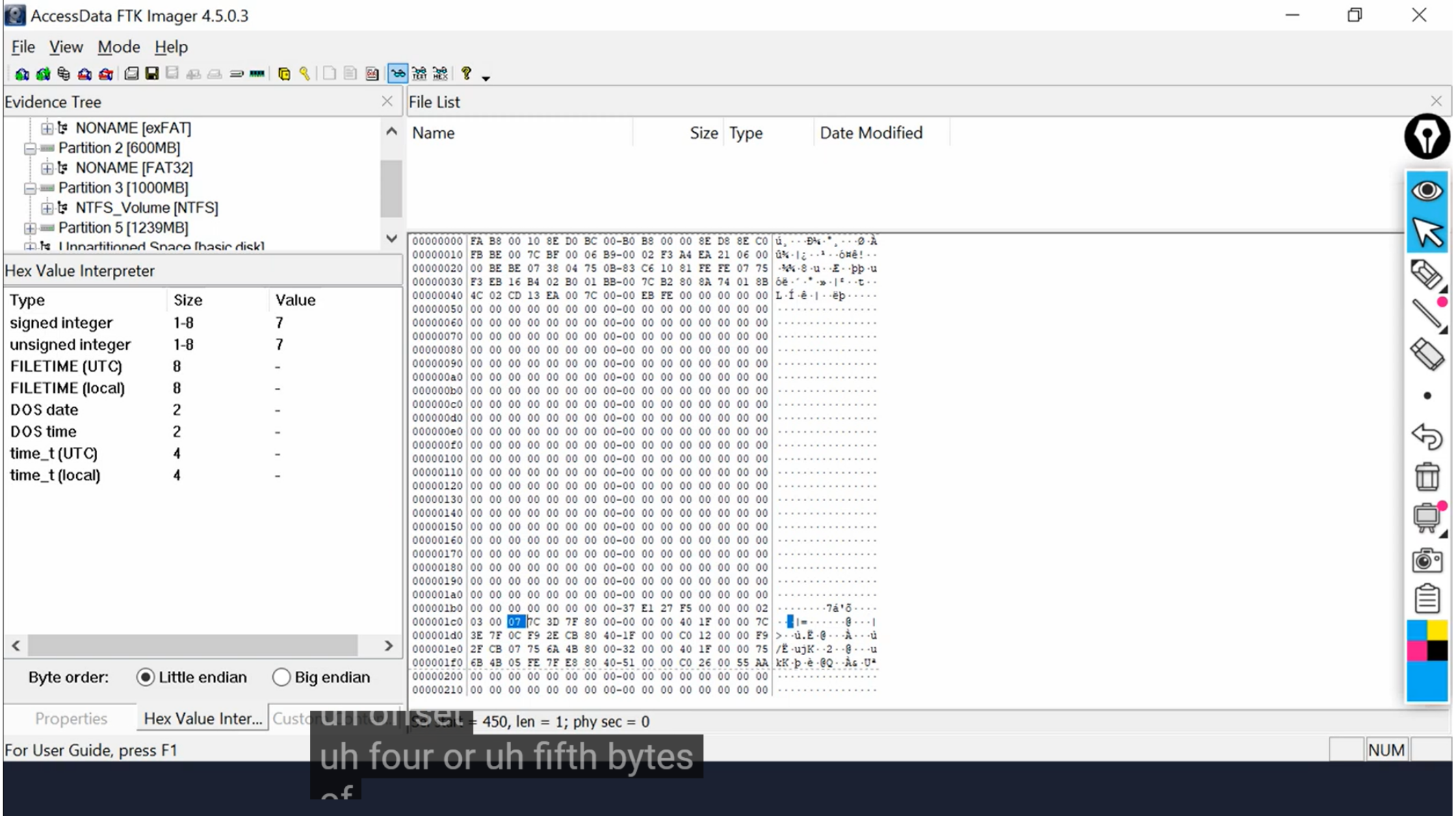
  offset 8-11 (9th to 12th bytes) – Starting Sector of the partition

  offset 12-15 (13th to 16th bytes) – Partition Size (in sectors)

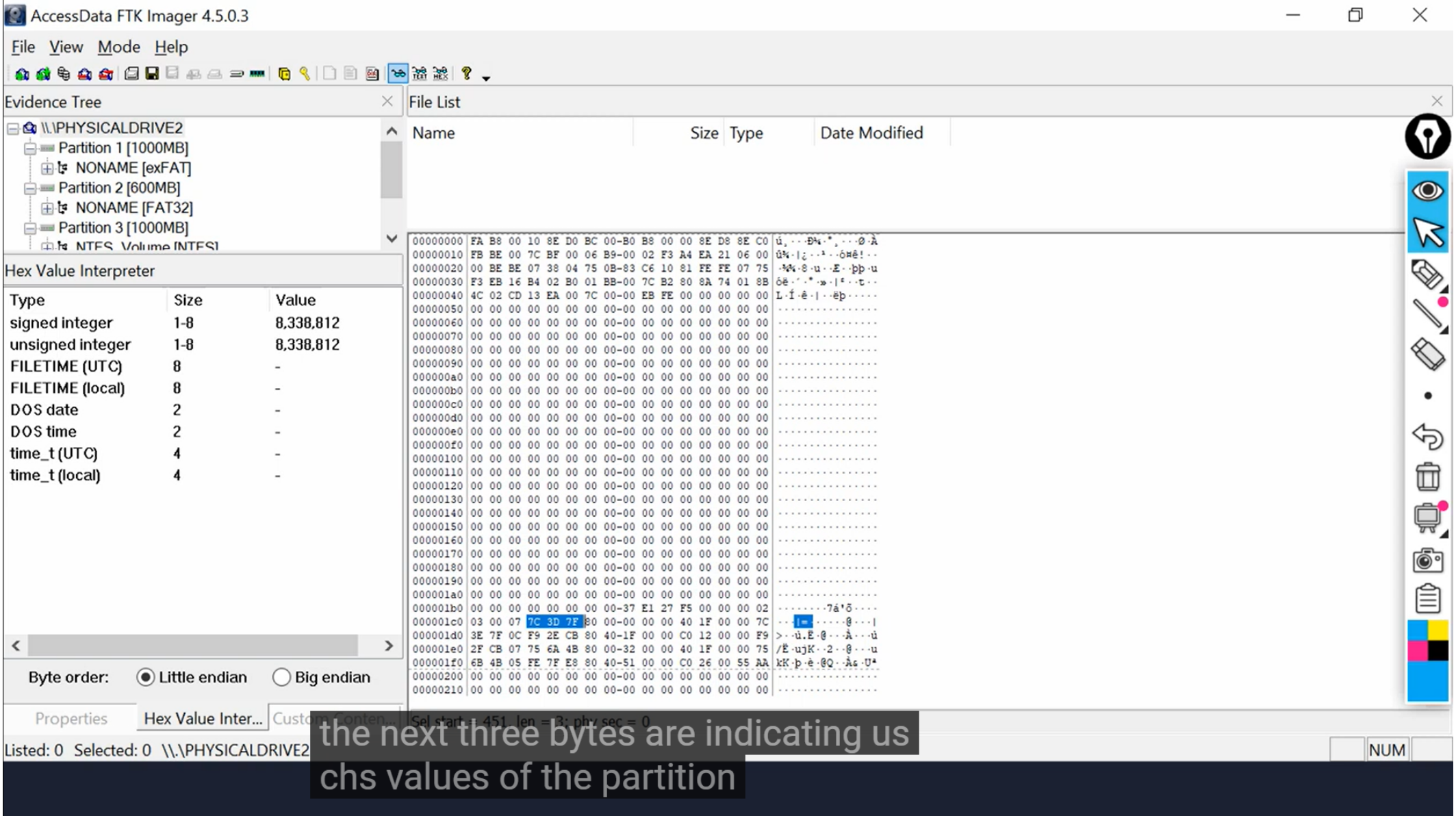

AccessData FTK Imager 4.5.0.3
File View Mode Help
Evidence Tree
\\.\PHYSICALDRIVE2
File List
Name
Size
Type
Date Modified
Hex Value Interpreter
Type
Size
Value
signed integer 1-8
unsigned integer 1-8
FILETIME (UTC) 8
FILETIME (local) 8
DOS date 2
DOS time 2
time_t (UTC) 4
time_t (local) 4
Byte order: Little endian Big endian
Properties
Hex Value Inter...
Sel start = 446, len = 64; phy sec = 0
For User Guide, press F1
NUM
64 byte long
at the bottom



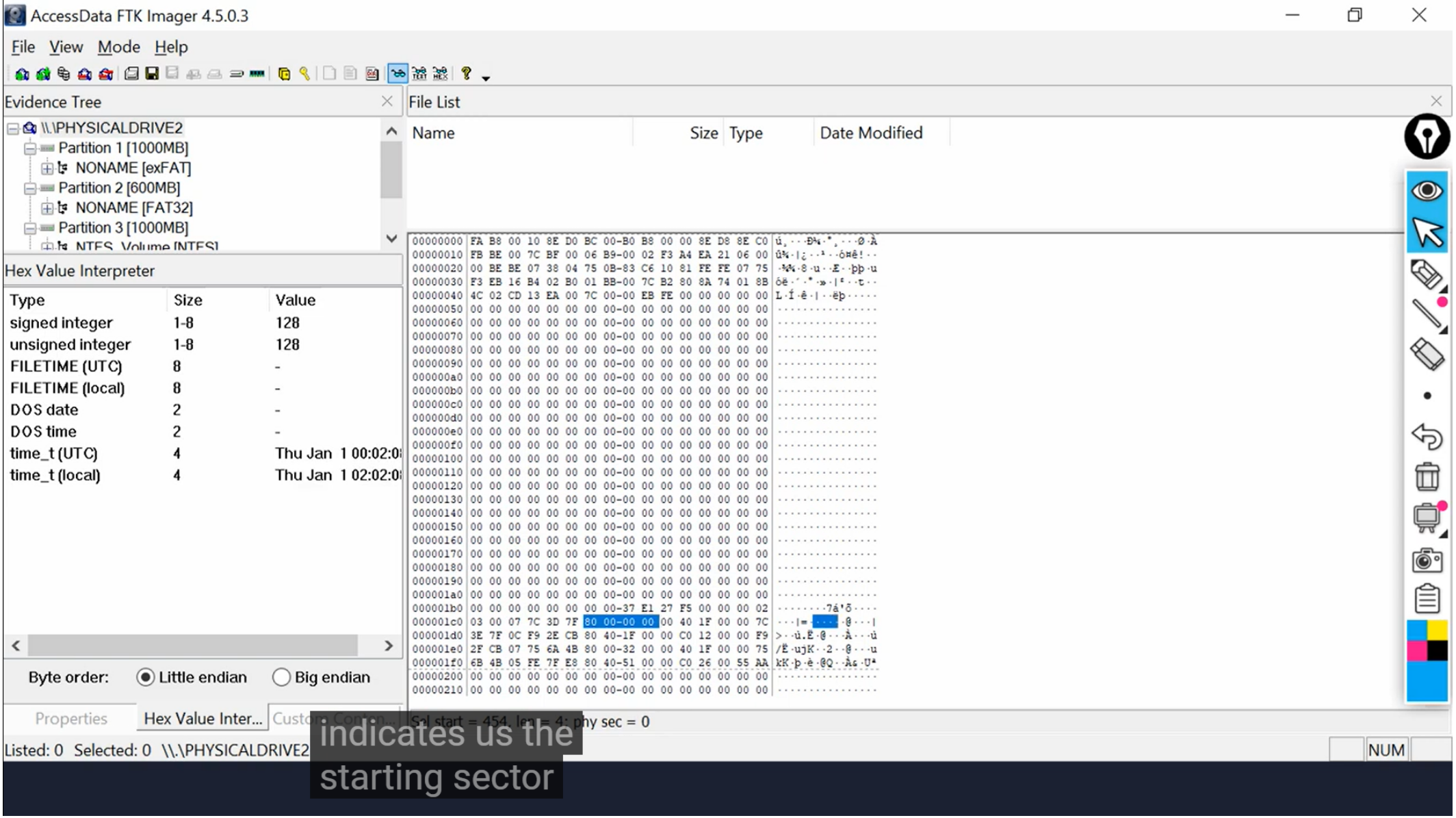
offset 0 (1st byte): non-bootable partition (00)



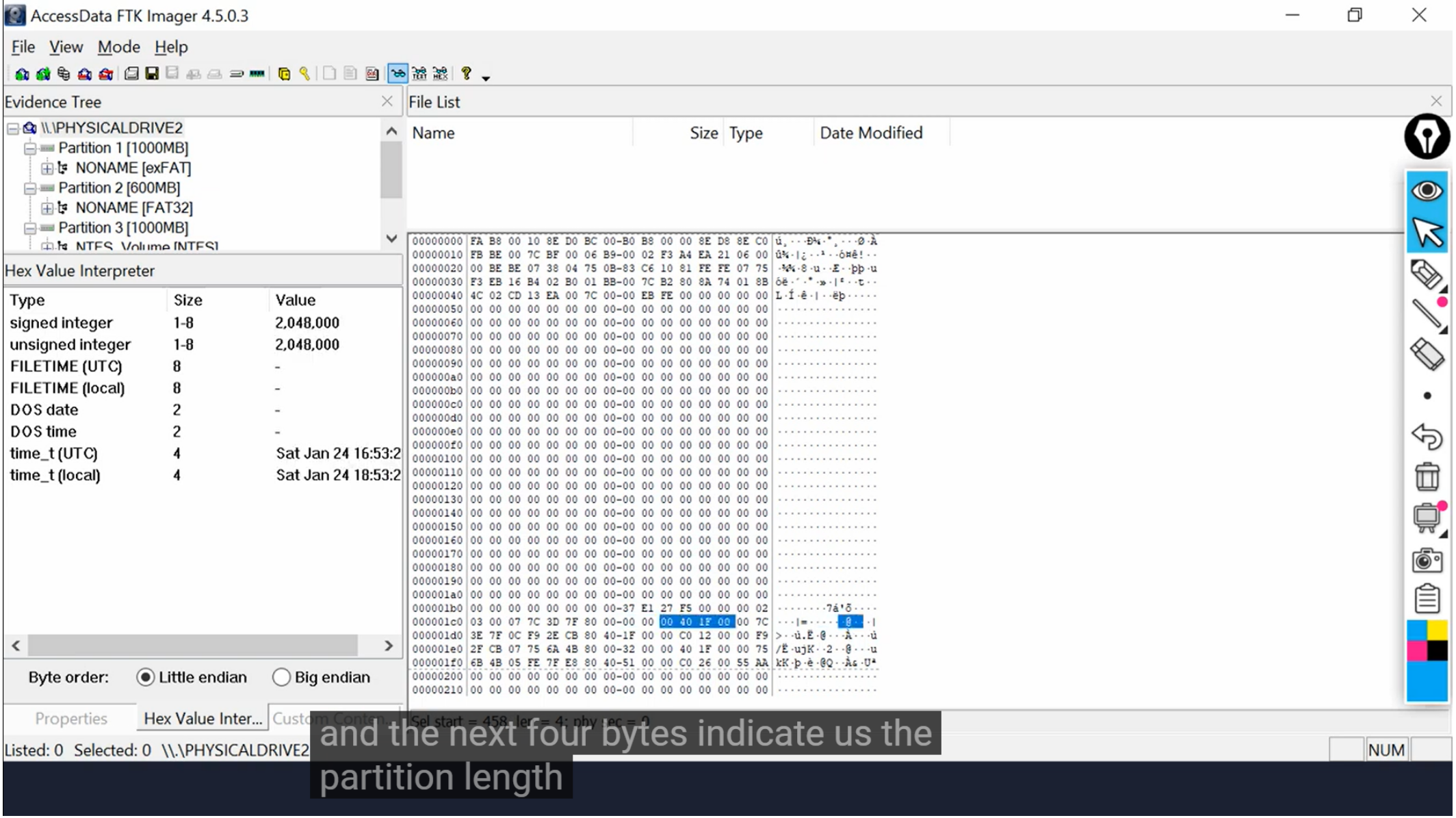
offset 4 (5th byte): flag file system type (exfat (07), ext32 (0C), ntfs (07),extended primary partition (05)...)



offset 5-7 (6th - 8th byte): chs value of the partition start



offset 8-11 (9th-12th byte): starting sector of partition



offset 12-15 (13th-16th byte): partition length in sector (1 sector = 512bytes)

DISK LAYOUT - PARTITION STYLES: MPT

- Partition Type Descriptors represent the partition's file system:
- Examples:
- 0x07 – NTFS
- 0x0C – FAT32
- 0x83 – Linux native
- 0x82 – Linux swap
- 0x05 – Extended Primary Partition
- **More info at: https://en.wikipedia.org/wiki/Partition_type**

uh

DISK LAYOUT - PARTITION STYLES: MBR

- EXTENDED PRIMARY PARTITION – used to overcome 4 partitions MBR limitation
- Type signature in MPT partition record – 0x05 (DoS extended Partition)

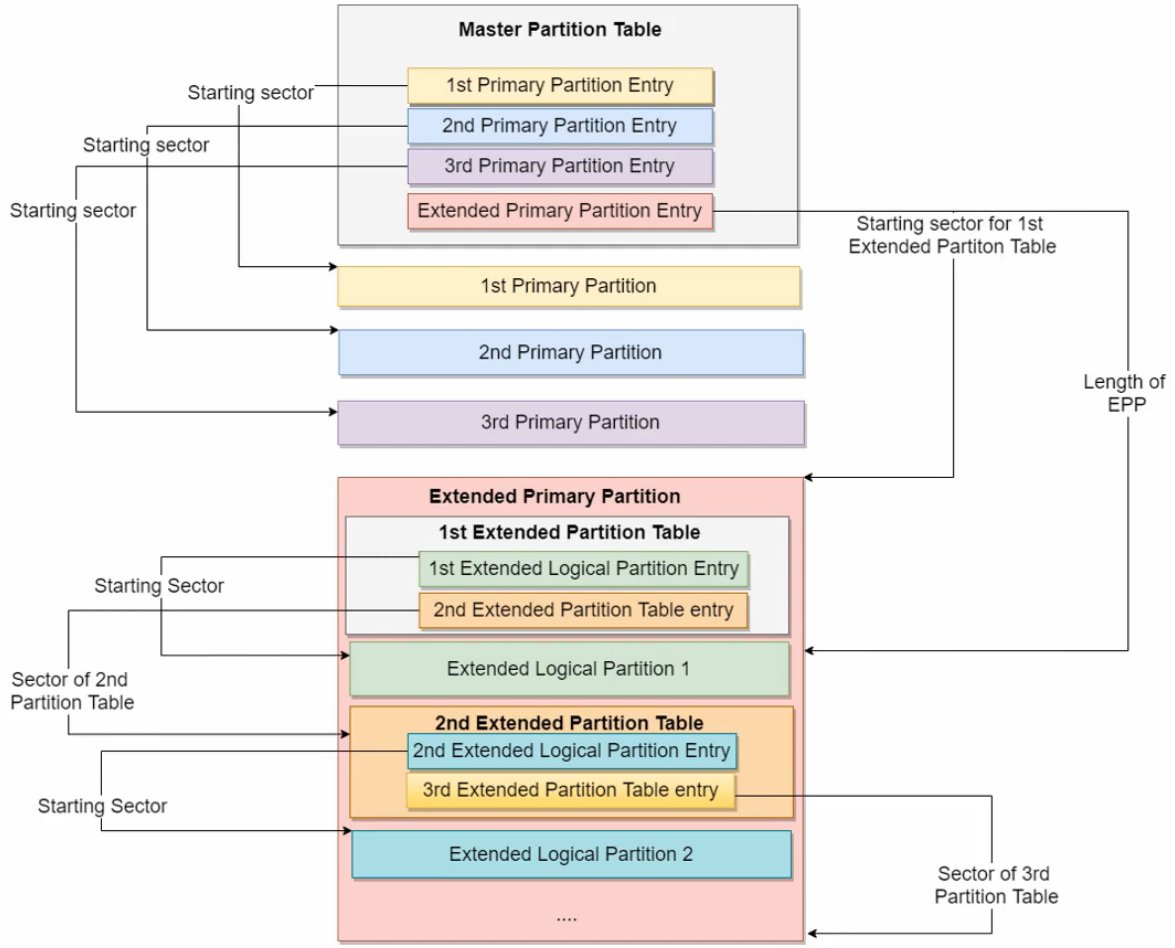
  Sometimes can also be 0x0F or 0x0C (FAT32 extended)

  EXTENDED PRIMARY partition entry – inside Master Boot Rercord

  EXTENDED LOGICAL partition entry – outside Master Boot Record

Photos - Extended_Primary_Partition.png

See all photos | Add to | Edit & Create | Share



okay let's examine this

Type here to search

57°F Cloudy

ENG ET 11:11 AM 9/6/2021

DISK LAYOUT - PARTITION STYLES: GPT

Also named GUID partitioning style – as partitions now have Global Unique Identifiers assigned to them

Compatible with UEFI (Unified Extensible Firmware Interface) – a lot of boot data strored on NAND flash or hard drive (boot-time diagnostics, utilities for backup restore or scanning

Default partitioning style in Windows (system drive or large external and internal disks)

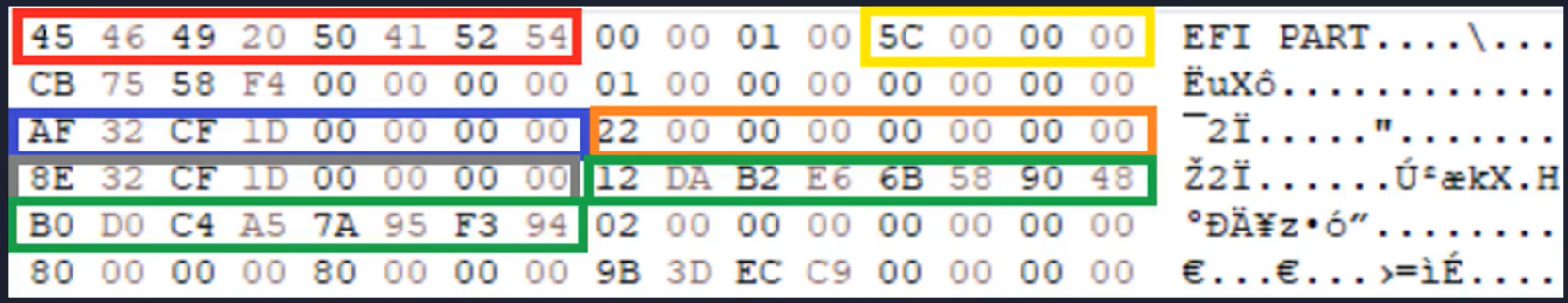
DISK LAYOUT - PARTITION STYLES: GPT

GPT layout consists of 5 major components, that store all partitioning related metadata:

1. *The “Protective MBR”*
2. *Primary “GUID Partition Table” with GPT header*
3. *GUID partition entries (can accommodate to 128 partitions on the disk)*
4. *Partition area*
5. *Backup area (copies of GUID partition table header and partition entries)*

the gpt partitioning will be created
gpta layout is a bit different

GPT HEADER STRUCTURE:



0x00 (8) – signature /x45/x46/x49/x20/x50/x41/x52/x54 – Signature „EFI PART“

0x08 (4) – Revision number for this header

0x0C (4) – GPT header size in bytes (at least 92 bytes, 0x5C)

0x10 (4) – CRC32 checksum of GPT header

0x14 (4) – reserved

0x18 (8) – LBA of the current GPT header structure (sector 1)

0x20 (8) – LBA of backup copy of GPT

0x28 (8) – LBA of start of partition area

0x30 (8) – LBA of the end of partition area (last used partition sector)

0x38 (16) – disk Global Unique Identifier for this disk

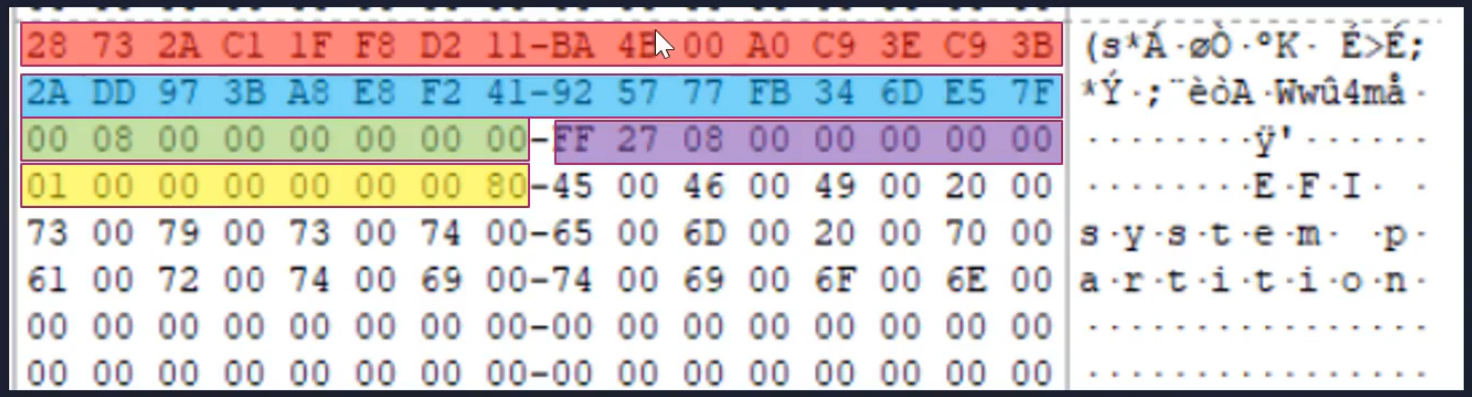
0x48 (8) – LBA of the start of the partition tables (usually sector 2)

0x50 (4) – number of possible partition enrties (set to 128)

0x54 (4) – size of each entry in the partition table in bytes (set to 128)

0x58 (4) – CRC32 checksum of the partition table

GPT PARTITION ENTRY STRUCTURE:



0x00 (16) – partition type GUID

0x10 (16) – Unique partititon GUID

0x20 (8) – starting LBA for the partition

0x28 (8) – ending LBA for the partition

0x30 (8) – partition attribute flags

0x38 (72) – partition name in Unicode

0x30 (8) – partition attribute flags:

0x00 – GPT attribute platform required

0x60 – read only

0x61 – the partition is shadow copy of another partition

0x62 – hidden partition (inaccessible by user)

0x63 – no drive letter assigned

File system

FILE SYSTEM:

- Is a method of organizing (creating changing and retrieving) data on a storage media
- FS provide a method for users to store data in a hierarchy of files and directories.
- Has a defined layout and structure, that consists of:
- - FS structural data – data used by an operating system to search and handle user data (information about file system layout)
- - File System level metadata for the files and directories – stores file or folder parameters and description (e.g., name, timestamps, size, access parameters, etc.)
- - File Content – user/system file content

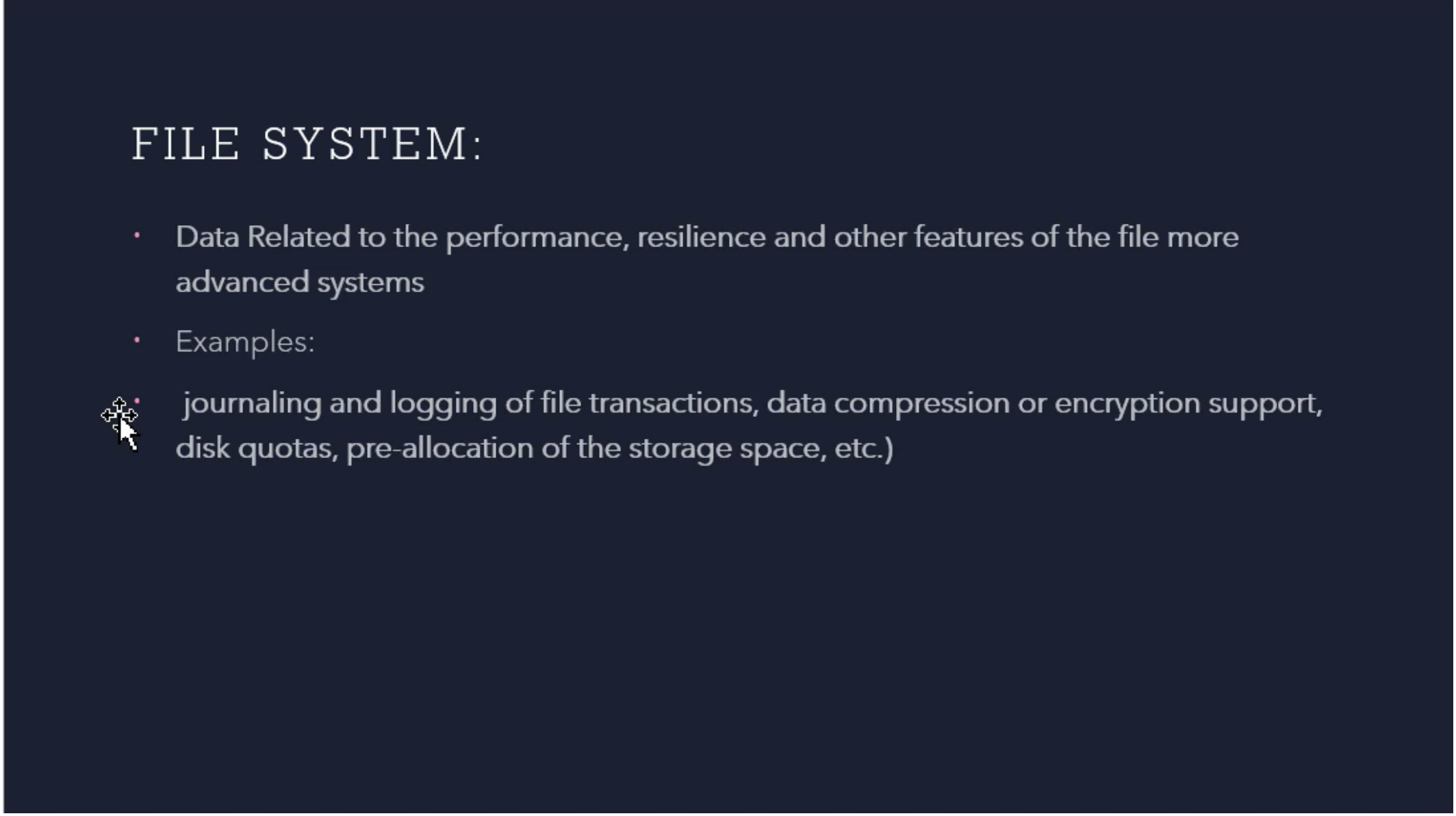
FILE SYSTEM:

- **Structural Data** - contains the general file system information that defines the file system layout & parameters (e.g., data unit size(cluster or block), volume layout, media descriptors, allocation bitmaps, etc..)
- Structural Data is usually placed in the beginning of the formatted logical volume (disk)
- Examples:
- FAT, NTFS file systems – 1st sector of a logical volume – **Logical Sector 0,** sometimes called a **Volume Boot Record**
- Linux native ext or xfs filesystems - **Superblock**

FILE SYSTEM:

- File and folder metadata – the data category that describe a file or directory – "data about data" (physical location of the file, LW, LR, LA timestamps, file size, access control information, descriptors, allocation information, a file name)

- Examples:
- FAT file systems – Directory Entry, File Allocation Table
- NTFS - $MFT (Master File Table)
- Linux native ext filesystems - Directory Entry, Inode Table



Disk formatting

DISK FORMATTING:

- Formatting is the process of creating a new file system on a disk partition
- Partitions can be formatted using the quick or full or normal formatting

- During the formatting User specifies :
    1. the file system type of the logical volume (NTFS, FAT32, ext4, xfs, APFS, etc)
    2. the allocation unit size (cluster, or block) – multiple of 512 or sector size

If no cluster size specified by the user, default value is selected based on the size of the partition

assignments
first of all what is the disk formatting

DISK FORMATTING:

During the full formatting:

1. all logical sectors of the partition will be zeroed out –

   all previously stored data is gone

2. Structural data and file system metadata (metafiles) are created accordingly to file system type and allocation unit (cluster) size

During the quick formatting:

1. Structural data and file system metadata (metafiles) are created accordingly to file system type and allocation unit (cluster) size
2. Old data is not completely gone, but only partially overwritten with the new file system metadata

FILE SYSTEM ALLOCATION UNITS:

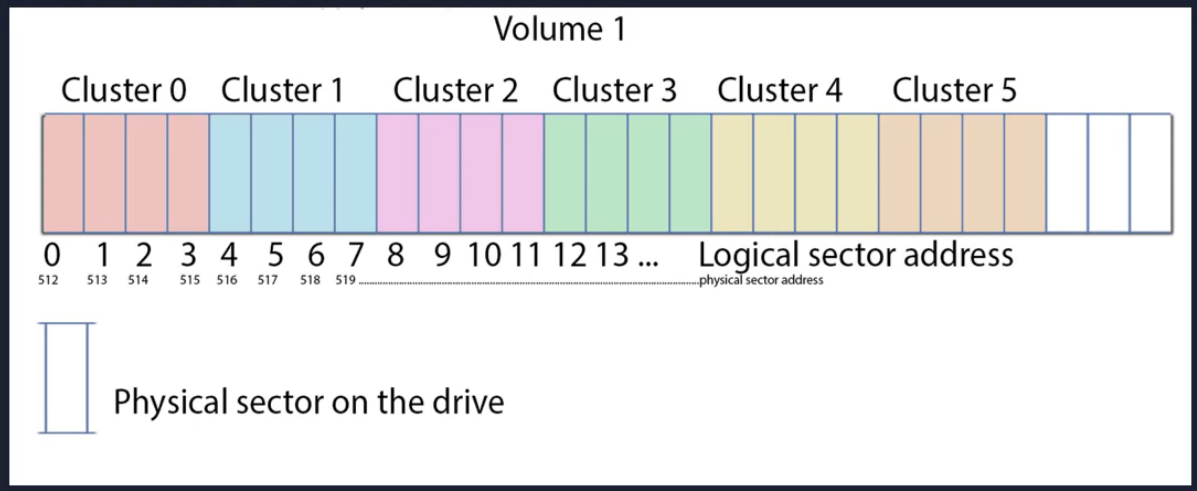
Called clusters or blocks:

combine several consecutive logical sectors into one addressable unit

cluster or block size can be equal to 1, 2, 4, 8, 16, 32, 64, 128 sectors, or

512b, 1024b, 2048b, 4096b, 8192b, 16384b, 32768b, 65536b

Most commonly 4096 (or 4k) clusters are used in Windows and Linux



FILE SYSTEM ALLOCATION UNITS:

Cluster size represents the smallest amount of disk space that can be used to hold a file.

Example:

- FS cluster size is 4096 bytes
- User creates a text file with one letter of text (1b) – file logical size
- FS allocates 1 cluster to store this file – 4096b – file physical size
- 4095 bytes are wasted on the disk, and no other file can place its data into the same cluster, until this text file is deleted by the user

FAT file system

FAT FILE SYSTEMS:

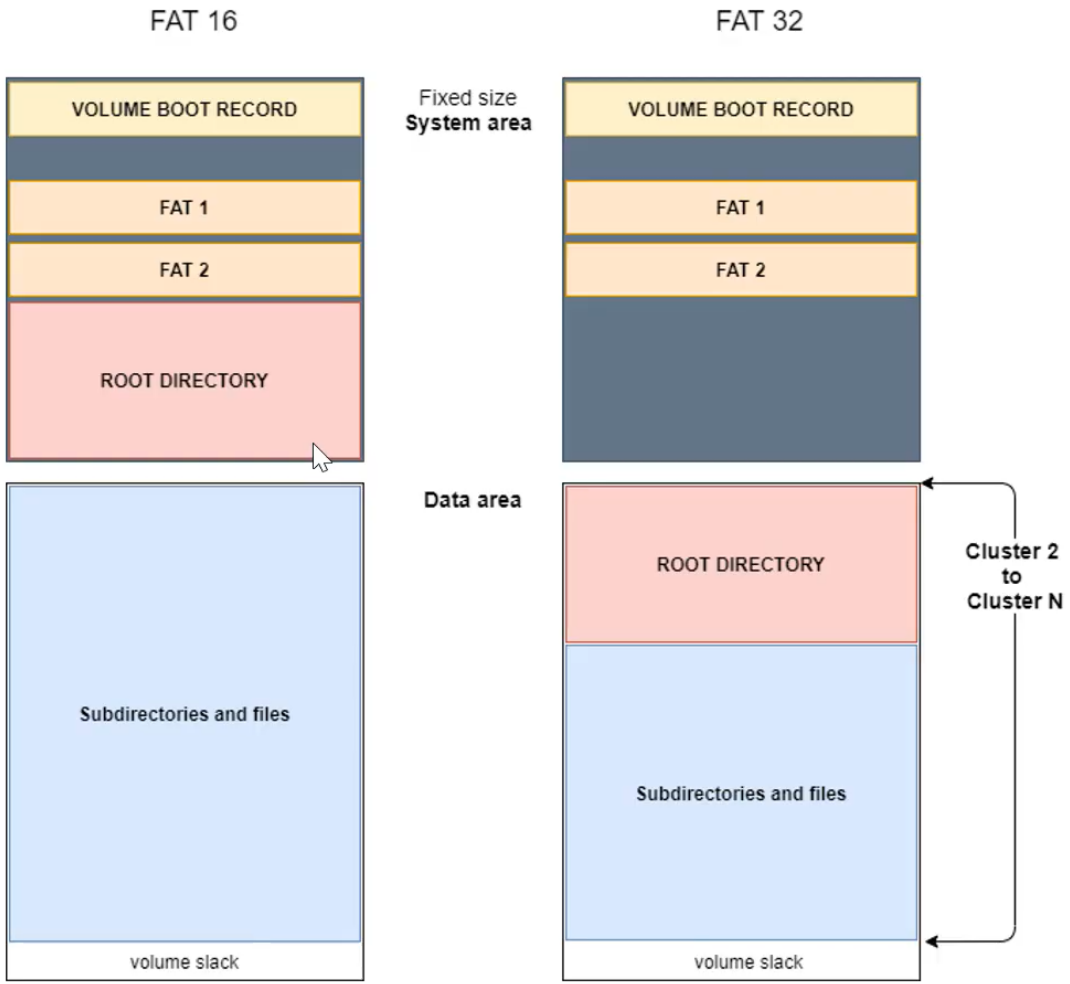
- FAT – stands for File Allocation Table
- FAT 12/16 – legacy file system, sometimes used on USB memory sticks up to 1GB
- FAT32 – external storage media devices (USB memory sticks, SD cards,
- exFAT (extended FAT) – larger addressable space than in FAT32, more timestamps and metadata for the files, space pre allocation functionality, possibility to store larger files

in fat file systems
fat

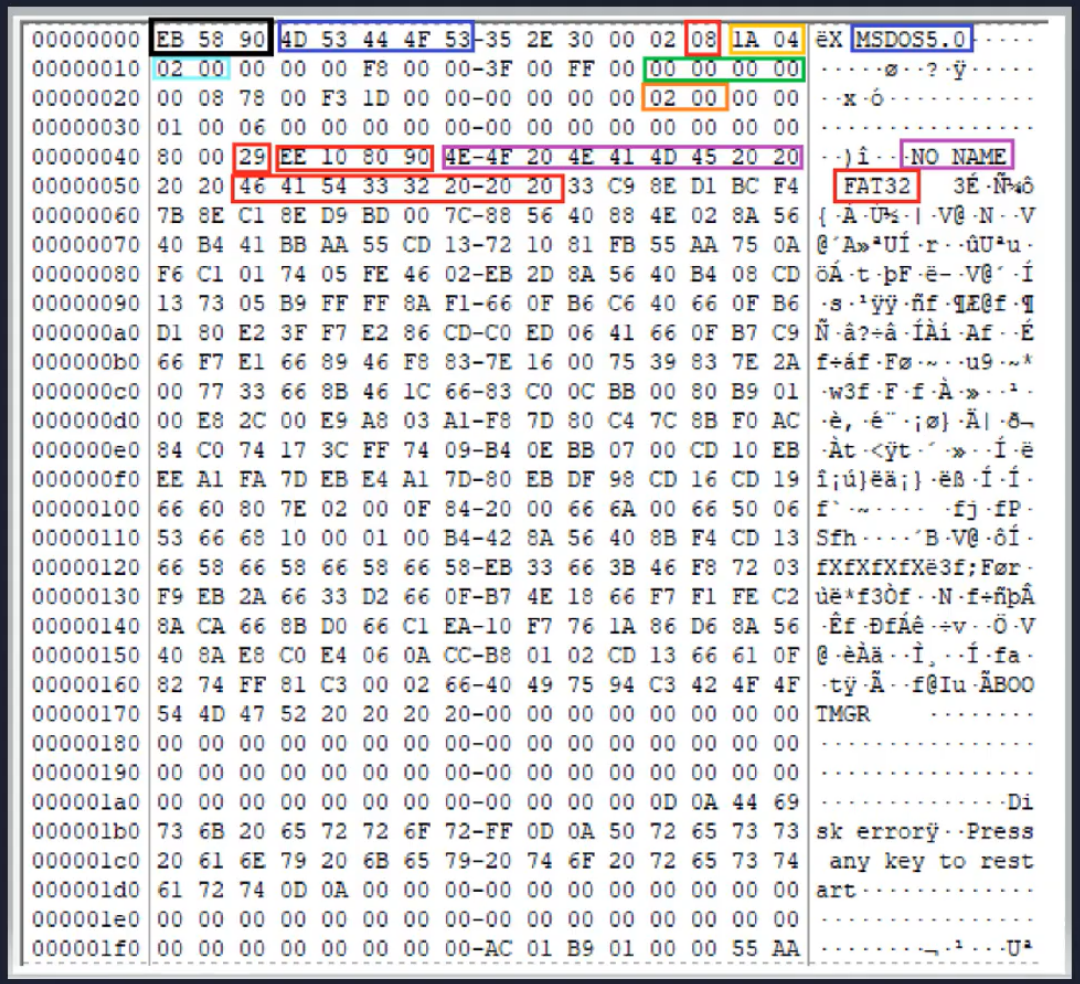
FAT16 VS FAT32:

FAT 16	FAT32
2 bytes (16 bit) are used for cluster addressing 0xFFFF	3 bytes (28 bit) are used for cluster addressing 0xFF FF FF
Max cluster number is – 65535 clusters, cluster numbering starts from 2	Max cluster number 16,777,214, cluster numbering starts from 2
System Area contains Volume Boot Record, two FAT tables and **Root Directory**	System Area contains Volume Boot Record, two FAT tables
Data Area contains subdirectories and file data	Data Area contains **Root Directory** , subdirectories and file data

if you compare fat 16 and fat 32 file systems in


FAT 16
FAT 32
VOLUME BOOT RECORD
FAT 1
FAT 2
ROOT DIRECTORY
Fixed size
System area
VOLUME BOOT RECORD
FAT 1
FAT 2
Data area
Subdirectories and files
volume slack
ROOT DIRECTORY
Subdirectories and files
volume slack
Cluster 2
to
Cluster N

FAT32 VOLUME BOOT RECORD:



```
00000000  EB 58 90 4D 53 44 4F 53-35 2E 30 00 02 08 1A 04  ëX MSDOS5.0·····
00000010  02 00 00 00 00 F8 00 00-3F 00 FF 00 00 00 00 00  ·····ø··?·ÿ·····
00000020  00 08 78 00 F3 1D 00 00-00 00 00 00 02 00 00 00  ··x·ó···········
00000030  01 00 06 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
00000040  80 00 29 EE 10 80 90 4E-4F 20 4E 41 4D 45 20 20  ··)î··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 56 40 88 4E 02 8A 56  {·Á·Ù½·|·V@·N··V
00000070  40 B4 41 BB AA 55 CD 13-72 10 81 FB 55 AA 75 0A  @´A»ªUÍ·r··ûUªu·
00000080  F6 C1 01 74 05 FE 46 02-EB 2D 8A 56 40 B4 08 CD  öÁ·t·þF·ë-·V@´·Í
00000090  13 73 05 B9 FF FF 8A F1-66 0F B6 C6 40 66 0F B6  ·s·¹ÿÿ·ñf·¶Æ@f·¶
000000a0  D1 80 E2 3F F7 E2 86 CD-C0 ED 06 41 66 0F B7 C9  Ñ·â?÷â·ÍÀí·Af·É
000000b0  66 F7 E1 66 89 46 F8 83-7E 16 00 75 39 83 7E 2A  f÷áf·Fø·~··u9·~*
000000c0  00 77 33 66 8B 46 1C 66-83 C0 0C BB 00 80 B9 01  ·w3f·F·f·À·»··¹·
000000d0  00 E8 2C 00 E9 A8 03 A1-F8 7D 80 C4 7C 8B F0 AC  ·è,·é¨·¡ø}·Ä|·ð¬
000000e0  84 C0 74 17 3C FF 74 09-B4 0E BB 07 00 CD 10 EB  ·Àt·<ÿt·´·»··Í·ë
000000f0  EE A1 FA 7D EB E4 A1 7D-80 EB DF 98 CD 16 CD 19  î¡ú}ëä¡}·ëß·Í·Í·
00000100  66 60 80 7E 02 00 0F 84-20 00 66 6A 00 66 50 06  f`·~···· ·fj·fP·
00000110  53 66 68 10 00 01 00 B4-42 8A 56 40 8B F4 CD 13  Sfh····´B·V@·ôÍ·
00000120  66 58 66 58 66 58 66 58-EB 33 66 3B 46 F8 72 03  fXfXfXfXë3f;Før·
00000130  F9 EB 2A 66 33 D2 66 0F-B7 4E 18 66 F7 F1 FE C2  ùë*f3Òf··N·f÷ñþÂ
00000140  8A CA 66 8B D0 66 C1 EA-10 F7 76 1A 86 D6 8A 56  ·Êf·ÐfÁê·÷v··Ö·V
00000150  40 8A E8 C0 E4 06 0A CC-B8 01 02 CD 13 66 61 0F  @·èÀä··Ì¸··Í·fa·
00000160  82 74 FF 81 C3 00 02 66-40 49 75 94 C3 42 4F 4F  ·tÿ·Ã··f@Iu·ÃBOO
00000170  54 4D 47 52 20 20 20 20-00 00 00 00 00 00 00 00  TMGR    ········
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 44 69  ··············Di
000001b0  73 6B 20 65 72 72 6F 72-FF 0D 0A 50 72 65 73 73  sk errorÿ··Press
000001c0  20 61 6E 79 20 6B 65 79-20 74 6F 20 72 65 73 74   any key to rest
000001d0  61 72 74 0D 0A 00 00 00-00 00 00 00 00 00 00 00  art·············
000001e0  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
000001f0  00 00 00 00 00 00 00 00-AC 01 B9 01 00 00 55 AA  ········¬·¹···Uª
```

Bytes 4-8 OEM ID, what OS used for the formatting

Byte 14 – how many sectors in one cluster (512*8) = 4096b cluster size

Bytes 15-16 – offset to the first FAT table

Bytes 29 -32 – sectors between physical sector 0 and this VBR

Offset 0x42 – extended boot signature flag 0x29

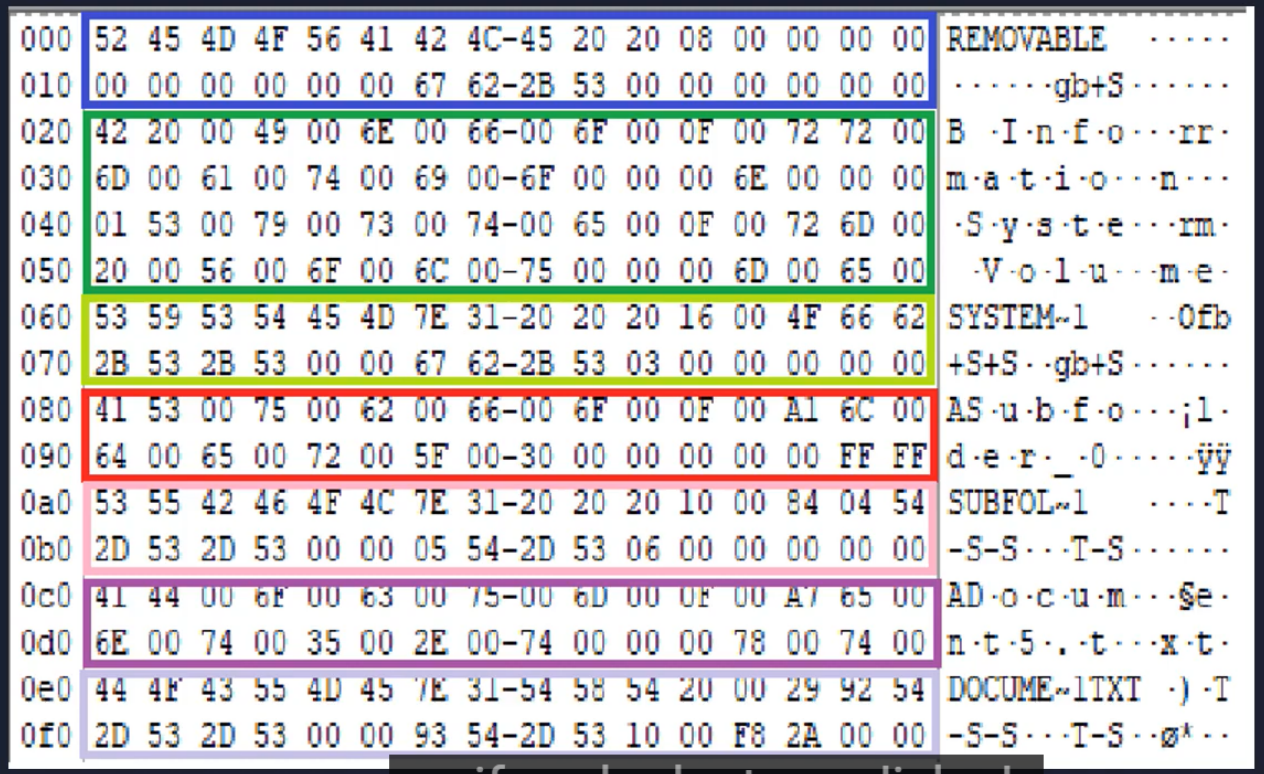
Offset 0x43 – Volume label "NO NAME" by default

Offset 52 – Filesystem type

DIRECTORY ENTRIES:

- Store metadata for the files and folders (timestamps, name, physical location, attributes and file size information
- There are 2 entries exist for each file – base entry (32 bytes, starting with file name converted in DOS 8.3 format and long filename entry – LFN. LFN store file name in UNICODE and occupies as many 32-byte entries as needed to accommodate the name of the file or folder)
- First 32 bytes in ROOT directory store the volume name specified by the user during disk formatting

DIRECTORY ENTRIES:



000	52	45	4D	4F	56	41	42	4C	45	20	20	08	00	00	00	00	REMOVABLE ·····
010	00	00	00	00	00	00	67	62	2B	53	00	00	00	00	00	00	·····gb+S·····
020	42	20	00	49	00	6E	00	66	00	6F	00	0F	00	72	72	00	B ·I·n·f·o···rr·
030	6D	00	61	00	74	00	69	00	6F	00	00	00	6E	00	00	00	m·a·t·i·o···n···
040	01	53	00	79	00	73	00	74	00	65	00	0F	00	72	6D	00	·S·y·s·t·e···rm·
050	20	00	56	00	6F	00	6C	00	75	00	00	00	6D	00	65	00	·V·o·l·u···m·e·
060	53	59	53	54	45	4D	7E	31	20	20	20	16	00	4F	66	62	SYSTEM~1 ···Ofb
070	2B	53	2B	53	00	00	67	62	2B	53	03	00	00	00	00	00	+S+S··gb+S·····
080	41	53	00	75	00	62	00	66	00	6F	00	0F	00	A1	6C	00	AS·u·b·f·o···¡l·
090	64	00	65	00	72	00	5F	00	30	00	00	00	00	00	FF	FF	d·e·r·_·0·····ÿÿ
0a0	53	55	42	46	4F	4C	7E	31	20	20	20	10	00	84	04	54	SUBFOL~1 ····T
0b0	2D	53	2D	53	00	00	05	54	2D	53	06	00	00	00	00	00	-S-S···T-S·····
0c0	41	44	00	6F	00	63	00	75	00	6D	00	0F	00	A7	65	00	AD·o·c·u·m···§e·
0d0	6E	00	74	00	35	00	2E	00	74	00	00	00	78	00	74	00	n·t·5·.·t···x·t·
0e0	44	4F	43	55	4D	45	7E	31	54	58	54	20	00	29	92	54	DOCUME~1TXT ·)·T
0f0	2D	53	2D	53	00	00	93	54	2D	53	10	00	F8	2A	00	00	-S-S···T-S··ø*··

so if we look at our disk uh in cluster

Volume Label (REMOVABLE) – 32b

LFN for "System Volume Information" folder – 2x32b

Base entry for "System Volume Information" – 32b

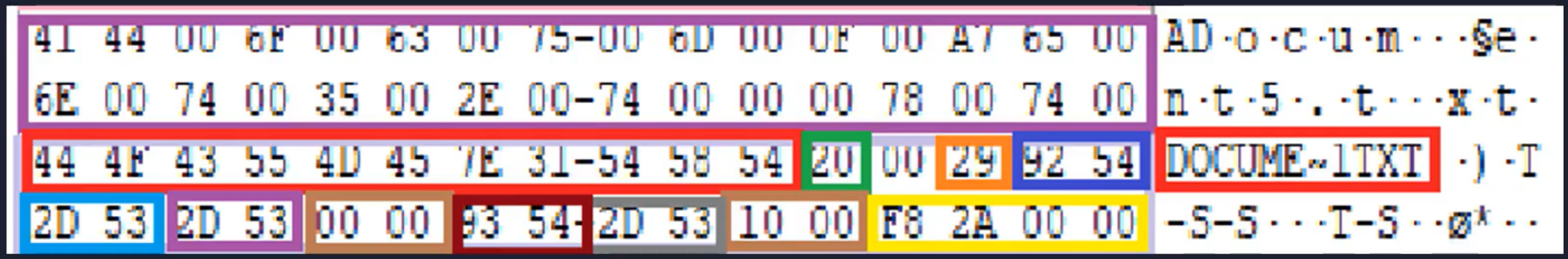
LFN for "Subfolder_1" – 32b

Base entry for Subfolder 1 – 32b

LFN for "Document5.txt" – 32b

Base entry for "Document5.txt - 32b

DIRECTORY ENTRIES –BASE ENTRY METADATA:



DOS 8.3 Converted file name (8 UC letters + 3 UC extension)

File attribute (1 byte) – 0x20 or 0010 0000 in binary)

Milliseconds of creation time – 0x29 (41 in decimal)

File Creation TIME – DOS Timestamp (Local time)

File Creation DATE - DOS (Local time)

Last Access DATE - DOS (Local time)

Starting Cluster for the file contents (Higher Bits + Lower Bits)

Content's modification TIME and DATE

FILE SIZE in BYTES – the max value is (0xFF FF FF FF = 4095MB)

File Attributes:

Need to convert to binary:

0000 0001 – read only
0000 0010 – hidden
0000 0100 – system file
0000 1000 – volume label entry
0001 0000 – directory
0010 0000 – archive or normal file

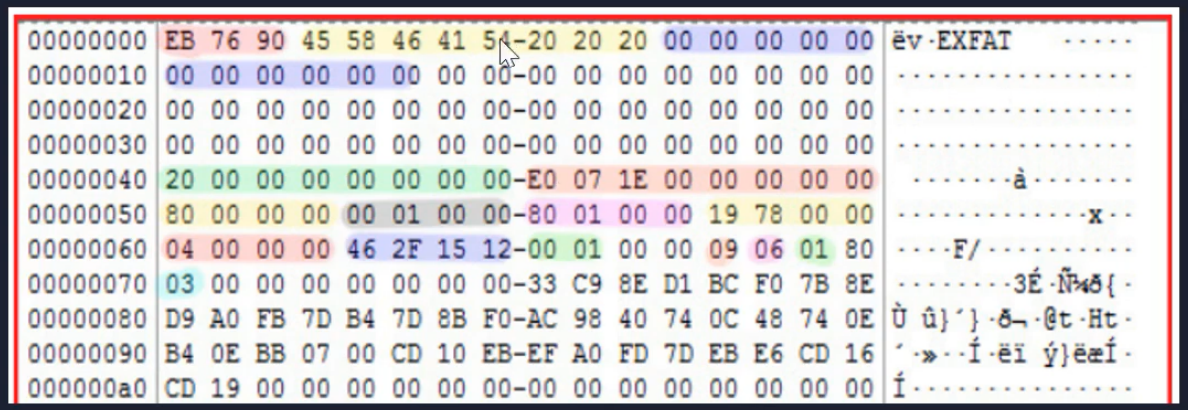
ExFAT File System

EXTENDED FAT FILE SYSTEM

- exFAT :
- 4 bytes used for the cluster addressing
- 8 bytes used for the file size
- Improved performance for free space allocation over free space bitmap
- Timestamp granularity of 10 ms for “Created” and “Modified” timestamps
- Timestamp for Last Access time, FAT32 has only date
- Time Zone value to track difference from UTC time

about uh this file system
so

EX FAT VBR (BOOT SECTOR):



```
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  20 00 00 00 00 00 00 00-E0 07 1E 00 00 00 00 00  ······à·····
00000050  80 00 00 00 00 01 00 00-80 01 00 00 19 78 00 00  ····x··
00000060  04 00 00 00 46 2F 15 12-00 01 00 00 09 06 01 80  ····F/··········
00000070  03 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···············
```

volume boot record of formatted exfat
this is a

- 0x00 - 2 Jump Code
- 0x03 -10 FS Identifier
- 0x40 - 48 – Volume offset from MBR in sectors
- 0x48 – 4F – total sectors on the volume
- 0x50 – 53 – FAT location
- 0x54 – 57 – FAT size in sectors
- 0x58 – 5B – Bitmap Location
- 0x5C – 5D – cluster count
- 0x60 – 63 – Root directory starting cluster
- 0x64 – 67 volume serial nr
- 0x6D – sectors per cluster as 2^n

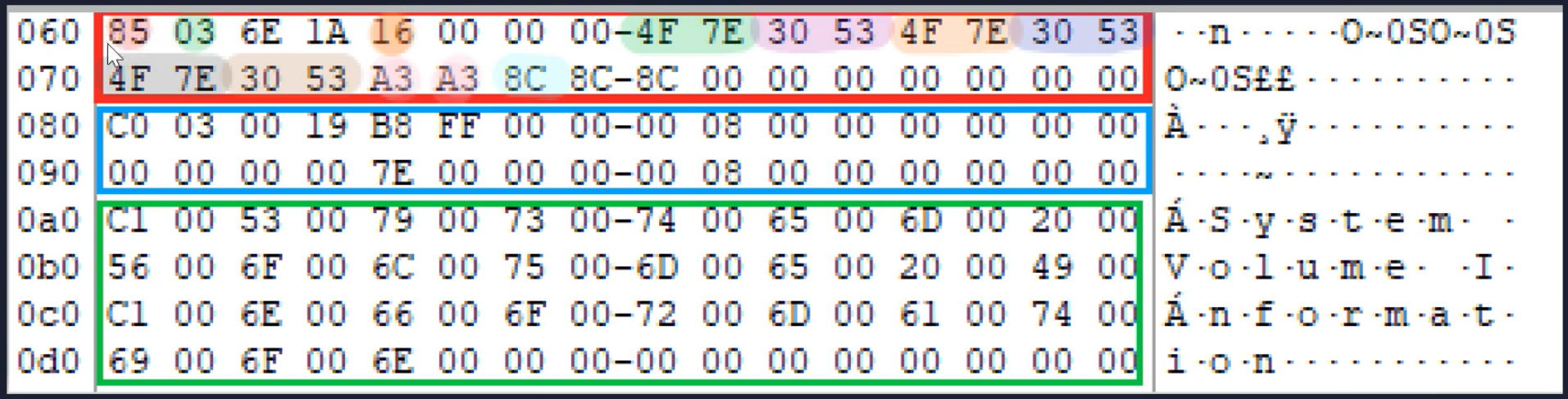
DIRECTORY ENTRY:

Offset Hex	Field Definition
x85	Directory Entry Record
x83	Volume Name Record
x82	Up-Case Table Logical Location and Size
x81	Bitmap Logical Location and Size
xC0	Stream Extension
xC1	File Name Extension

- Minimum 3 entries for file or folder:
- 1st starts from 0x85 – Directory entry record, contains timestamps
- 2nd starts from 0xC0 – Stream extension – file size , starting cluster, file name extension parameters
- 3rd starts from 0xC1 - File name extension

a look at the directory entry which is most interesting part for

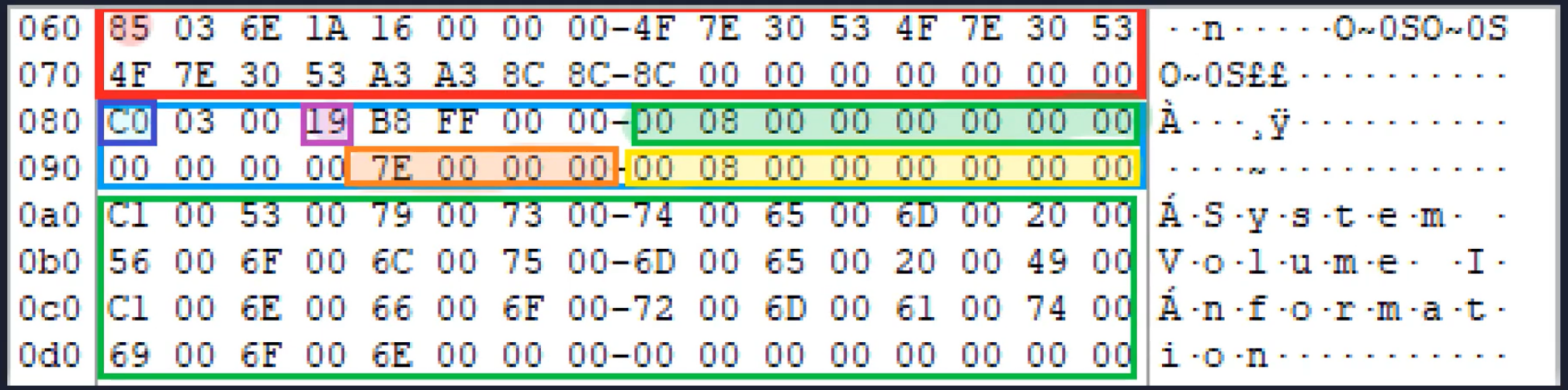
DIRECTORY ENTRY RECORD:



```
060  85 03 6E 1A 16 00 00 00-4F 7E 30 53 4F 7E 30 53   ··n·····O~0SO~0S
070  4F 7E 30 53 A3 A3 8C 8C-8C 00 00 00 00 00 00 00   O~0S££·····
080  C0 03 00 19 B8 FF 00 00-00 08 00 00 00 00 00 00   À···¸ÿ·····
090  00 00 00 00 7E 00 00 00-00 08 00 00 00 00 00 00   ····~·····
0a0  C1 00 53 00 79 00 73 00-74 00 65 00 6D 00 20 00   Á·S·y·s·t·e·m· ·
0b0  56 00 6F 00 6C 00 75 00-6D 00 65 00 20 00 49 00   V·o·l·u·m·e· ·I·
0c0  C1 00 6E 00 66 00 6F 00-72 00 6D 00 61 00 74 00   Á·n·f·o·r·m·a·t·
0d0  69 00 6F 00 6E 00 00 00-00 00 00 00 00 00 00 00   i·o·n···········
```

- Off 0x00 – Record Type 0x85
- Off 0x01 – additional 32 records count (3)
- Off 0x04 – DOS flags – (convert to bin. Hid. Sys. Dir.
- Created tt / dd Modified tt / dd Accessed tt / dd - UTC
- Off 0x14 – 10 MS increments to Created and Modified ts
- Off 0x18 – Time zone offset from UTC
- 0x8C = 1000 1100 : 1000 – TZ flag 1100 – offset in 15 minutes increments: 8+4 = 12, 12*0.25 = 3hr

DIRECTORY ENTRY RECORD, STREAM EXTENSION:



- Off 0x00 – Record Type 0xC0 – stream extension
- Off 0x03 – number of Unicode characters in file name (25)
- Off 0x08 – initialized file size in bytes
- Off 0x14 – starting cluster of a file or folder
- 0ff 0x18 – logical size of the file in bytes

- 0xC1 – File Name extensions


DIRECTORY ENTRY RECORD DELETION:
0x85 record type value is changed to 0x05
0xC0 record type value is changed to 0x40
0xC1 record type value changed to 0x41
FAT / BITMAP updated with new free clusters information
instance
so

NTFS

Intro

NTFS:

- Native file system for MS Windows
- NTFS is capable to use 64bit cluster addressing (default cluster size is 4096 for up to 16TB disk volumes)
- Windows limits that number to 32bit cluster addresses
- Includes number of advanced features

NTFS ADVANCED FEATURES:

- Journaling for better recoverability
- File and directory access control
- Alternate data streams (multiple data streams)
- Disk quotas
- Sparse files
- File compression
- Soft (symbolic) and hard links
- Encryption support
- Junction points
- Metafile backups, “self healing on the run”, software-based data redundancy

NTFS:

All metadata stored in metafiles :

Metafile names start with $ and have no file extension in the name:

$MFT, $Logfile, $UsnJrnl

Metafilies are stored in the root directory of the volume or in $MetaDirectories

Metafiles can have multiple data streams (named data streams - $J, $SDS)

Directory tree is an index B-tree (Balanced Search Tree), stored in $MFT

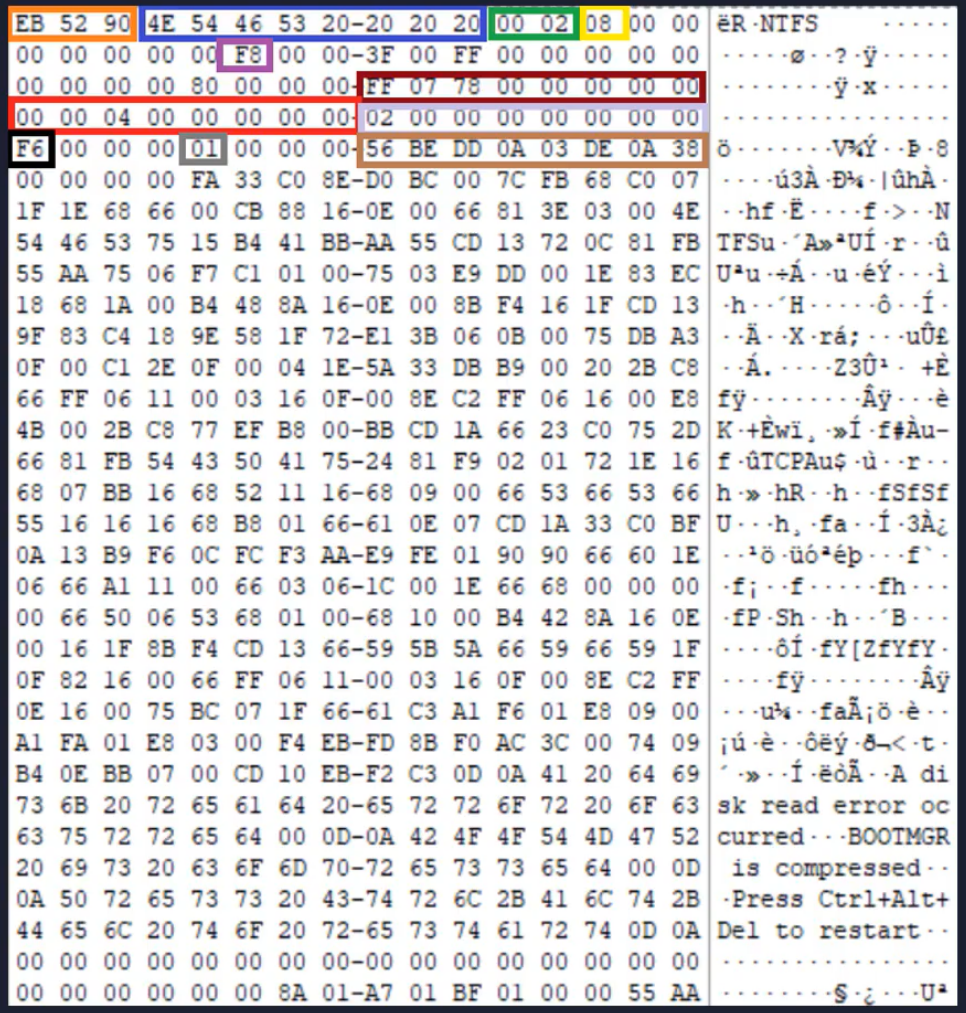

NTFS RELATED UPCOMING EPISODES:
Windows File System Forensics – intro to NTFS
Windows File System Forensics – NTFS layout and Volume Boot Record
Windows File System Forensics - Master File Table - $MFT
Windows File System Forensics - Alternate Data Streams in Windows
Windows File System Forensics - $LogFile
Windows File System Forensics – analyisis of journal $UsnJrnl
Windows File System Forensics – NTFS Timeline

Boot record

NTFS VOLUME BOOT RECORD:

- Volume starts with a logical sector 0, that stores a Volume Boot Record (part of $Boot metafile contents)
- Consecutive Logical sectors are combined into clusters
- All space is addressable by means of Logical Cluster Numbers (LCNs)
- $Boot is in LCN0 and occupies 1 cluster (4096b by default)
- $Boot has a backup (in very last logical sector of the partition)

VOLUME BOOT RECORD STRUCTURE:



```
EB 52 90 4E 54 46 53 20-20 20 20 00 02 08 00 00  ëR·NTFS     ·····
00 00 00 00 00 F8 00 00-3F 00 FF 00 00 00 00 00  ·····ø··?·ÿ·····
00 00 00 00 80 00 00 00-FF 07 78 00 00 00 00 00  ·····ÿ·x·····
00 00 04 00 00 00 00 00-02 00 00 00 00 00 00 00  ················
F6 00 00 00 01 00 00 00-56 BE DD 0A 03 DE 0A 38  ö·····V¾Ý··Þ·8
00 00 00 00 FA 33 C0 8E-D0 BC 00 7C FB 68 C0 07  ····ú3À·Ð¼·|ûhÀ·
1F 1E 68 66 00 CB 88 16-0E 00 66 81 3E 03 00 4E  ··hf·Ë····f·>··N
54 46 53 75 15 B4 41 BB-AA 55 CD 13 72 0C 81 FB  TFSu·´A»ªUÍ·r··û
55 AA 75 06 F7 C1 01 00-75 03 E9 DD 00 1E 83 EC  Uªu·÷Á··u·éÝ···ì
18 68 1A 00 B4 48 8A 16-0E 00 8B F4 16 1F CD 13  ·h··´H·····ô··Í·
9F 83 C4 18 9E 58 1F 72-E1 3B 06 0B 00 75 DB A3  ··Ä··X·rá;···uÛ£
0F 00 C1 2E 0F 00 04 1E-5A 33 DB B9 00 20 2B C8  ··Á.····Z3Û¹· +È
66 FF 06 11 00 03 16 0F-00 8E C2 FF 06 16 00 E8  fÿ·····Âÿ···è
4B 00 2B C8 77 EF B8 00-BB CD 1A 66 23 C0 75 2D  K·+Èwï¸·»Í·f#Àu-
66 81 FB 54 43 50 41 75-24 81 F9 02 01 72 1E 16  f·ûTCPAu$·ù··r··
68 07 BB 16 68 52 11 16-68 09 00 66 53 66 53 66  h·»·hR··h··fSfSf
55 16 16 16 68 B8 01 66-61 0E 07 CD 1A 33 C0 BF  U···h¸·fa··Í·3À¿
0A 13 B9 F6 0C FC F3 AA-E9 FE 01 90 90 66 60 1E  ··¹ö·üó²éþ···f`·
06 66 A1 11 00 66 03 06-1C 00 1E 66 68 00 00 00  ·f¡··f·····fh···
00 66 50 06 53 68 01 00-68 10 00 B4 42 8A 16 0E  ·fP·Sh··h··´B···
00 16 1F 8B F4 CD 13 66-59 5B 5A 66 59 66 59 1F  ····ôÍ·fY[ZfYfY·
0F 82 16 00 66 FF 06 11-00 03 16 0F 00 8E C2 FF  ····fÿ·····Âÿ
0E 16 00 75 BC 07 1F 66-61 C3 A1 F6 01 E8 09 00  ···u¼··faÃ¡ö·è··
A1 FA 01 E8 03 00 F4 EB-FD 8B F0 AC 3C 00 74 09  ¡ú·è··ôëý·ð¬<·t·
B4 0E BB 07 00 CD 10 EB-F2 C3 0D 0A 41 20 64 69  ´·»··Í·ëòÃ··A di
73 6B 20 72 65 61 64 20-65 72 72 6F 72 20 6F 63  sk read error oc
63 75 72 72 65 64 00 0D-0A 42 4F 4F 54 4D 47 52  curred···BOOTMGR
20 69 73 20 63 6F 6D 70-72 65 73 73 65 64 00 0D   is compressed··
0A 50 72 65 73 73 20 43-74 72 6C 2B 41 6C 74 2B  ·Press Ctrl+Alt+
44 65 6C 20 74 6F 20 72-65 73 74 61 72 74 0D 0A  Del to restart··
00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ················
00 00 00 00 00 00 8A 01-A7 01 BF 01 00 00 55 AA  ········§·¿···Uª
```

0x00-02 Jump instructions
0x03-0A – File system name – NTFS
0x0B – 0C – sector size in bytes
0x0D – sectors per cluster
0x28 – 2F – Total number of logical sectors
0x30 – 38 - $MFT starting logical sector
0x38 – 3F - $MFT Mirr starting logical sector
0x40 - $MFT record size, if F6, then 1024b

0x44 – clusters per Index Buffer

0x48 – 4F – Volume Serial Number
0x1FE – Boot signature 0xAA55

MFT

MASTER FILE TABLE - $MFT

- In NTFS all data stored on a volume is contained in files
- It allows the file system to easily locate and maintain the data and protect all files by a security descriptor
- If sectors goes bad the NTFS can relocate the metafiles
- Master File Table – is the heart of NTFS volume structure
- $MFT is implemented as an array of file records, starting with signature “FILE0”
- Each File Record is 1024b (4096b can be also defined at volume format time)

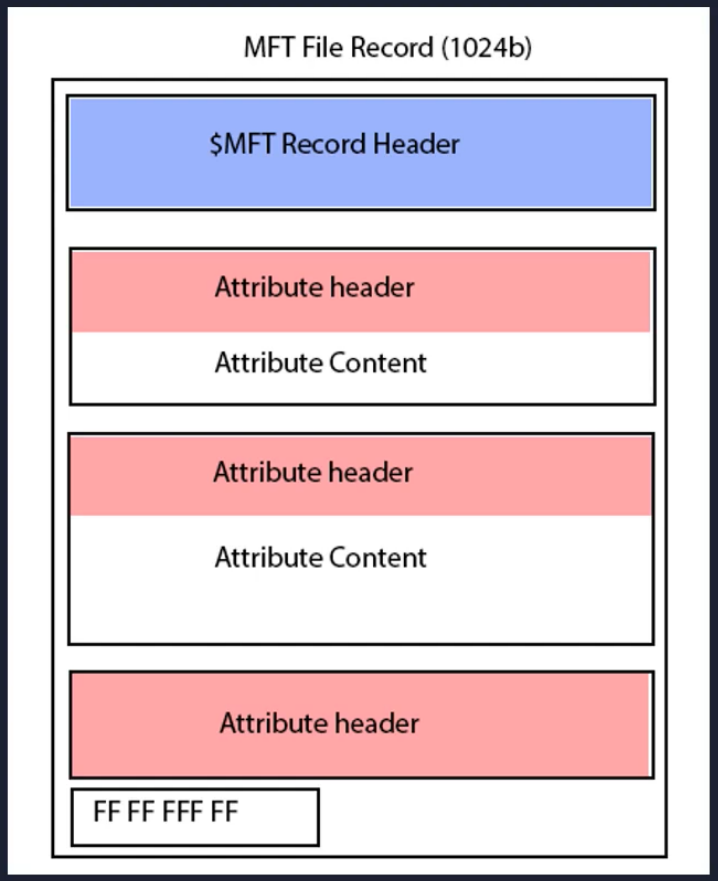
MASTER FILE TABLE – FILE RECORDS

0	$MFT - MFT
1	$MFTMirr - MFT mirror
2	$LogFile - Log file
3	$Volume - Volume file
4	$AttrDef - Attribute definition table
5	\ - Root directory
6	$BitMap - Volume cluster allocation file
7	$Boot - Boot sector
8	$BadClus - Bad-cluster file
9	$Secure - Security settings file
10	$UpCase - Uppercase character mapping
11	$Extend - Extended metadata directory

23	Unused
24	$Extend\$Quota - Quota information
25	$Extend\$ObjId - Distributed link tracking information
26	$Extend\$Reparse - Back references to reparse points
27	$Extend\$RmMetadata - RM metadata directory
28	$Extend\$RmMetadata\$Repair - RM repair information
29	$Extend\$Deleted - POSIX deleted files
30	$Extend\$RmMetadata\$TxfLog - TxF log directory
31	$Extend\$RmMetadata\$Txf - TxF metadata directory
32	$Extend\$RmMetadata\$TxfLog\$Tops - TOPS file
33	$Extend\$RmMetadata\$TxfLog\$TxfLog.blf - TxF BLF
34	$TxfLogContainer00000000000000000001
35	$TxfLogContainer00000000000000000002

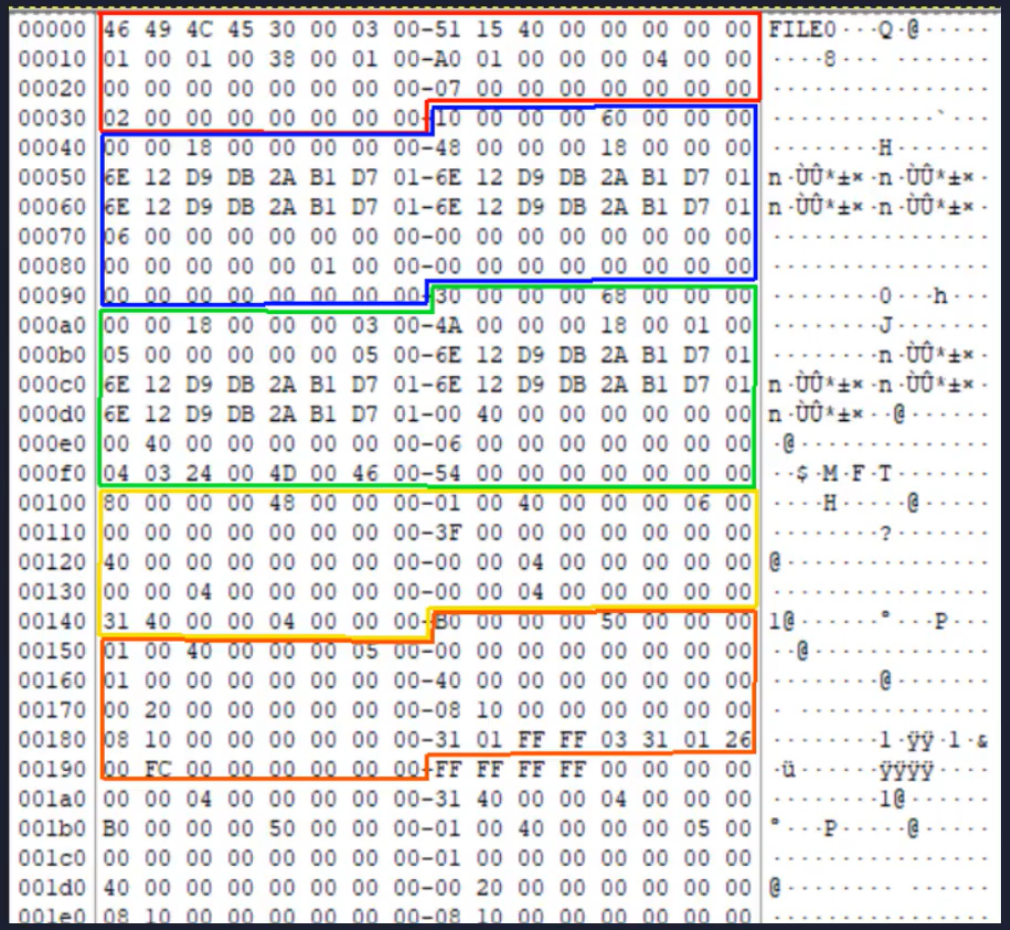
- All records are numbered starting from 0
- 1-st record in $MFT describes $MFT itself
- Records 1-12, 23-35 are used for other NTFS metafiles on the volume

MASTER FILE TABLE – FILE RECORD STRUCTURE

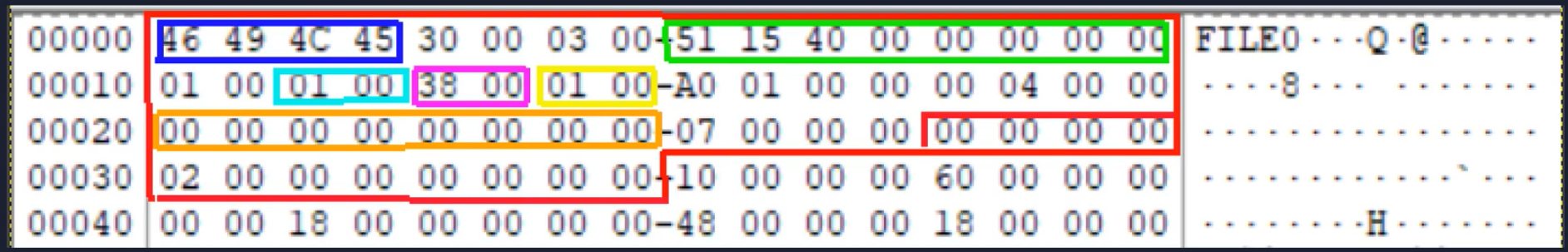


- 1024b, starting from "FILE0"
- Stores all the metadata for the file
- In a form File Record header and set of "Attributes"
- If metadata size < 1024b, then only one File Record is used for the file
- If metadata size > 1024b – then several File Records are used – Base Record + Additional Record(s)
- If additional records are used the Attribute content is relocated into additional Record and it's called non-resident attribute

$MFT – FILE RECORD STRUCTURE:



$MFT – FILE RECORD HEADER:



00000	46	49	4C	45	30	00	03	00	51	15	40	00	00	00	00	00	FILE0···Q·@·····
00010	01	00	01	00	38	00	01	00	A0	01	00	00	00	04	00	00	····8··· ·····
00020	00	00	00	00	00	00	00	00	07	00	00	00	00	00	00	00	····
00030	02	00	00	00	00	00	00	00	10	00	00	00	60	00	00	00	····`···
00040	00	00	18	00	00	00	00	00	48	00	00	00	18	00	00	00	····H···

SIGNATURE FILE or BAAD

TRANSACTION ENTRY NUMBER IN $LogFile

COUNT OF THE FILENAME ATTRIBUTES (HARD LINKS COUNT)

HEADER LENGTH IN BYTES

ALLOCATION STATUS

0x01 – allocated file,
0x00 – deleted file,
0x03 – allocated directory,
0x02 deleted directory

REFERENCE to the BASE RECORD in EXTENDED RECORD

MFT RECORD NUMBER

Timestamps, Standard Information and Filename attributes

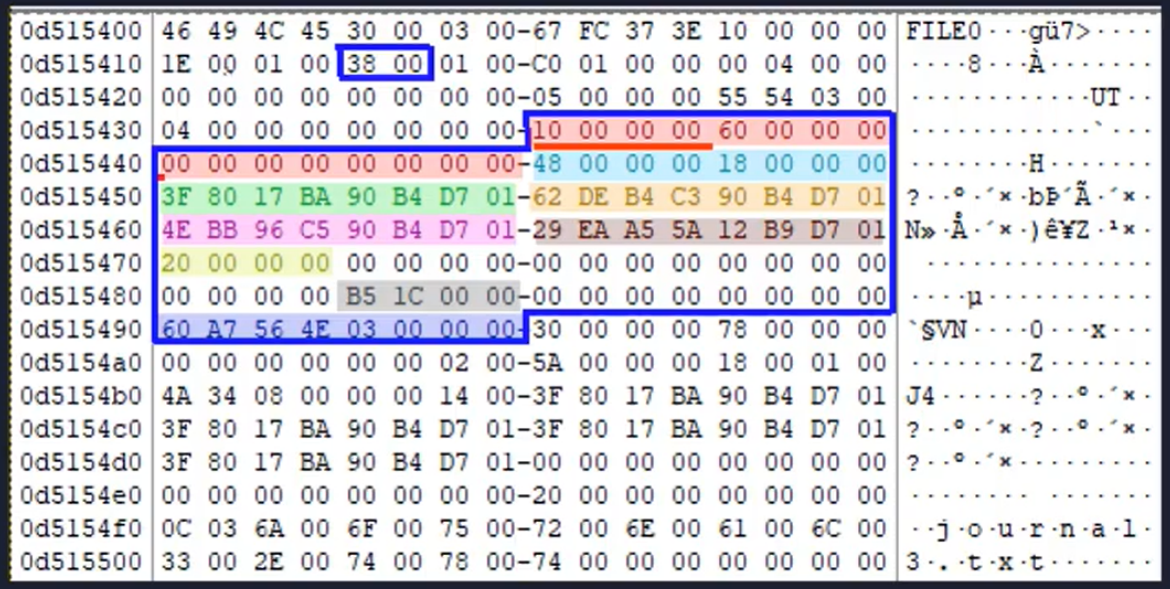
$MFT RECORD ATTRIBUTES:

- Resident attribute consists of:
- Attribute Header (16 bytes)
- Resident value header (6 bytes)
- Value (content) – file metadata

ATTRIBUTES

Attribute ID	Attribute Name	Attribute description & content
10 00 00 00	**$Standard_Information**	File permissions, time stamps, security and administrative information,**always resident**
20 00 00 00	$Attribute_List	Locations of non-resident attributes
30 00 00 00	**$File_Name**	The name of the file – **always resident**
40 00 00 00	$Object_ID	GUID (16b) – Globally Unique Identifier
80 00 00 00	**$Data**	The actual file's data or pointers to the file's data
90 00 00 00	$Index_Root	The top-level entry of a sorted tree that lists directory's child files – **always resident**
A0 00 00 00	$Index_Allocation	Points to the location of the Index Buffers of a large directory
C0 00 00 00	$Reparse_Point	Soft link (reparse points), for NTFS junctions and reparse points
00 01 00 00	$Logged_Utility_Stream	Contains information and keys for encrypted attributes. Used for various purposes, mainly for file encryption management ($EFS), or transactional NTFS data for the file $TXF_DATA

STANDARD INFORMATION ATTRIBUTE 0X10



0d515400	46	49	4C	45	30	00	03	00	67	FC	37	3E	10	00	00	00	FILE0···gü7>····
0d515410	1E	00	01	00	38	00	01	00	C0	01	00	00	00	04	00	00	····8···À·····
0d515420	00	00	00	00	00	00	00	00	05	00	00	00	55	54	03	00	···········UT··
0d515430	04	00	00	00	00	00	00	00	10	00	00	00	60	00	00	00	···········`···
0d515440	00	00	00	00	00	00	00	00	48	00	00	00	18	00	00	00	·········H·····
0d515450	3F	80	17	BA	90	B4	D7	01	62	DE	B4	C3	90	B4	D7	01	?··º·´×·bÞ´Ã·´×·
0d515460	4E	BB	96	C5	90	B4	D7	01	29	EA	A5	5A	12	B9	D7	01	N»·Å·´×·)ê¥Z·¹×·
0d515470	20	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	···············
0d515480	00	00	00	00	B5	1C	00	00	00	00	00	00	00	00	00	00	····µ···········
0d515490	60	A7	56	4E	03	00	00	00	30	00	00	00	78	00	00	00	`§VN····0···x···
0d5154a0	00	00	00	00	00	00	02	00	5A	00	00	00	18	00	01	00	·········Z·····
0d5154b0	4A	34	08	00	00	00	14	00	3F	80	17	BA	90	B4	D7	01	J4·····?··º·´×·
0d5154c0	3F	80	17	BA	90	B4	D7	01	3F	80	17	BA	90	B4	D7	01	?··º·´×·?··º·´×·
0d5154d0	3F	80	17	BA	90	B4	D7	01	00	00	00	00	00	00	00	00	?··º·´×··········
0d5154e0	00	00	00	00	00	00	00	00	20	00	00	00	00	00	00	00	········ ·····
0d5154f0	0C	03	6A	00	6F	00	75	00	72	00	6E	00	61	00	6C	00	··j·o·u·r·n·a·l·
0d515500	33	00	2E	00	74	00	78	00	74	00	00	00	00	00	00	00	3·.·t·x·t·····

0x0001 – read only
0x0002 – hidden
0x0004 – system file
0x0020, 0x0080 – Archive (normal file)
0x0200 – sparse file
0x0800 – compressed
0x4000 – encrypted (EFS)

0x00 (8) – Date/Time file created on volume

0x08 (8) - Date/Time file modified on volume

0x10 (8) – Date/Time $MFT content changed

0x18 (8) - Date/Time file last accessed

0x20 (4) - File type flag

0x34 (4) – Security ID (index of permission settings (key to $SII Index $SDS data stream in $Secure)

0x40 (8) – Last Update Sequence Number ($USN journal), if 0x00 – then journaling is disabled

TIMESTAMSPS IN NTFS

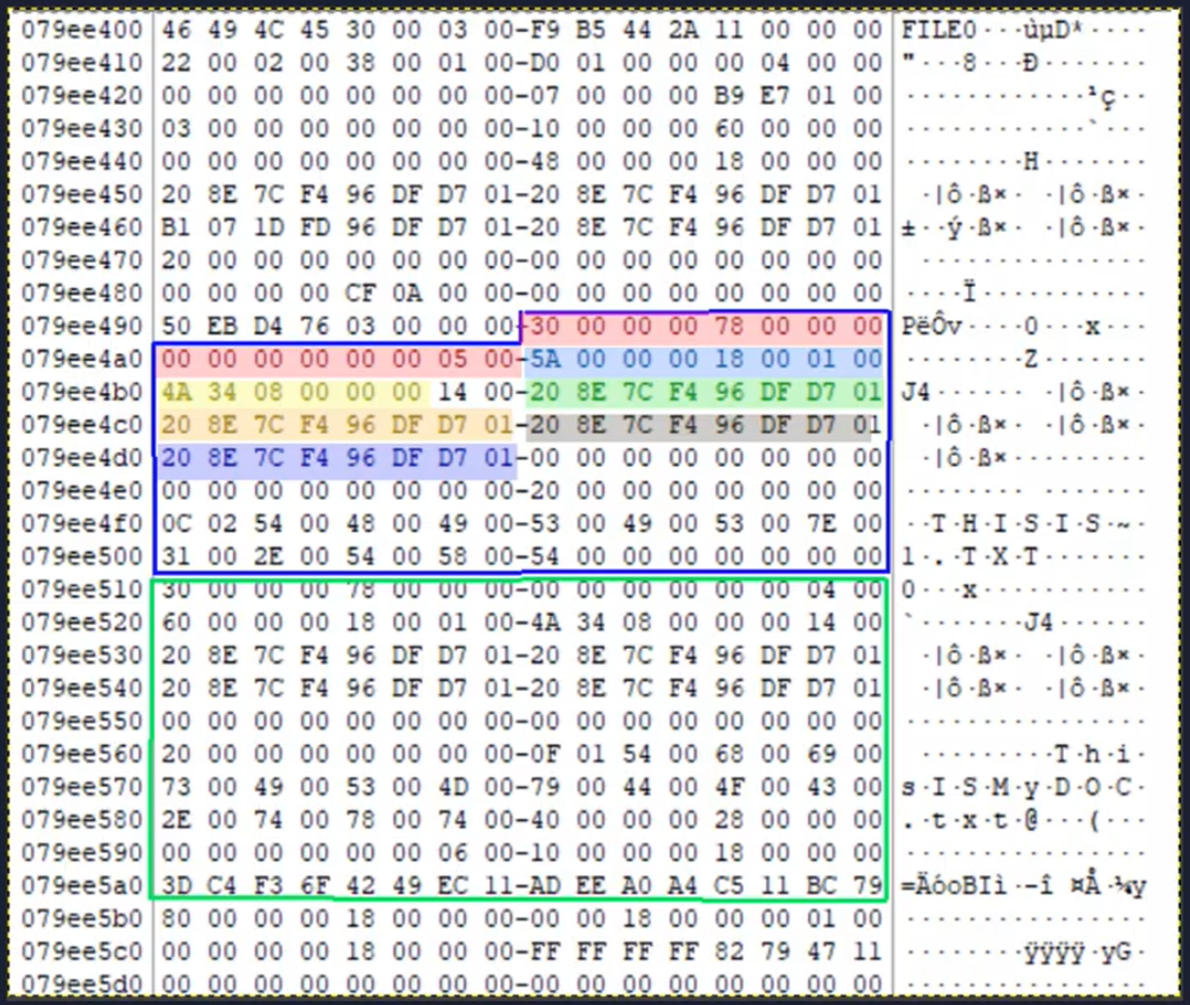
- Called FILETIME
- 64-bit unsigned value
- UTC (coordinated universal time)
- Represents the 100 nanosecond intervals since January 1st, 1601
- You need to know the system time zone information to convert it into local time

FILE NAME ATTRIBUTE – 0X30

- There are usually at least 2 FN attributes on system volume:
  1. *POSIX (DOS 8.3)*
  2. *Long File Name – Unicode*

  On external volumes POSIX names are disabled by default, can be enabled

FILE NAME ATTRIBUTE – 0X30



```
079ee400 46 49 4C 45 30 00 03 00-F9 B5 44 2A 11 00 00 00  FILE0···ùµD*····
079ee410 22 00 02 00 38 00 01 00-D0 01 00 00 00 04 00 00  "···8···Ð·······
079ee420 00 00 00 00 00 00 00 00-07 00 00 00 B9 E7 01 00  ············¹ç··
079ee430 03 00 00 00 00 00 00 00-10 00 00 00 60 00 00 00  ············`···
079ee440 00 00 00 00 00 00 00 00-48 00 00 00 18 00 00 00  ········H·······
079ee450 20 8E 7C F4 96 DF D7 01-20 8E 7C F4 96 DF D7 01   ·|ô·ß×· ·|ô·ß×·
079ee460 B1 07 1D FD 96 DF D7 01-20 8E 7C F4 96 DF D7 01  ±··ý·ß×· ·|ô·ß×·
079ee470 20 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ···············
079ee480 00 00 00 00 CF 0A 00 00-00 00 00 00 00 00 00 00  ····Ï···········
079ee490 50 EB D4 76 03 00 00 00-30 00 00 00 78 00 00 00  PëÔv····0···x···
079ee4a0 00 00 00 00 00 00 05 00-5A 00 00 00 18 00 01 00  ········Z·······
079ee4b0 4A 34 08 00 00 00 14 00-20 8E 7C F4 96 DF D7 01  J4······ ·|ô·ß×·
079ee4c0 20 8E 7C F4 96 DF D7 01-20 8E 7C F4 96 DF D7 01   ·|ô·ß×· ·|ô·ß×·
079ee4d0 20 8E 7C F4 96 DF D7 01-00 00 00 00 00 00 00 00   ·|ô·ß×··········
079ee4e0 00 00 00 00 00 00 00 00-20 00 00 00 00 00 00 00  ········ ·······
079ee4f0 0C 02 54 00 48 00 49 00-53 00 49 00 53 00 7E 00  ··T·H·I·S·I·S·~·
079ee500 31 00 2E 00 54 00 58 00-54 00 00 00 00 00 00 00  1·.·T·X·T·······
079ee510 30 00 00 00 78 00 00 00-00 00 00 00 00 00 04 00  0···x···········
079ee520 60 00 00 00 18 00 01 00-4A 34 08 00 00 00 14 00  `·······J4·····
079ee530 20 8E 7C F4 96 DF D7 01-20 8E 7C F4 96 DF D7 01   ·|ô·ß×· ·|ô·ß×·
079ee540 20 8E 7C F4 96 DF D7 01-20 8E 7C F4 96 DF D7 01   ·|ô·ß×· ·|ô·ß×·
079ee550 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ···············
079ee560 20 00 00 00 00 00 00 00-0F 01 54 00 68 00 69 00   ·········T·h·i·
079ee570 73 00 49 00 53 00 4D 00-79 00 44 00 4F 00 43 00  s·I·S·M·y·D·O·C·
079ee580 2E 00 74 00 78 00 74 00-40 00 00 00 28 00 00 00  .·t·x·t·@···(···
079ee590 00 00 00 00 00 00 06 00-10 00 00 00 18 00 00 00  ···············
079ee5a0 3D C4 F3 6F 42 49 EC 11-AD EE A0 A4 C5 11 BC 79  =ÄóoBIì·-î ¤Å·¼y
079ee5b0 80 00 00 00 18 00 00 00-00 00 18 00 00 00 01 00  ···············
079ee5c0 00 00 00 00 18 00 00 00-FF FF FF FF 82 79 47 11  ·····ÿÿÿÿ·yG·
079ee5d0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ···············
```

Offset in the attribute value field:

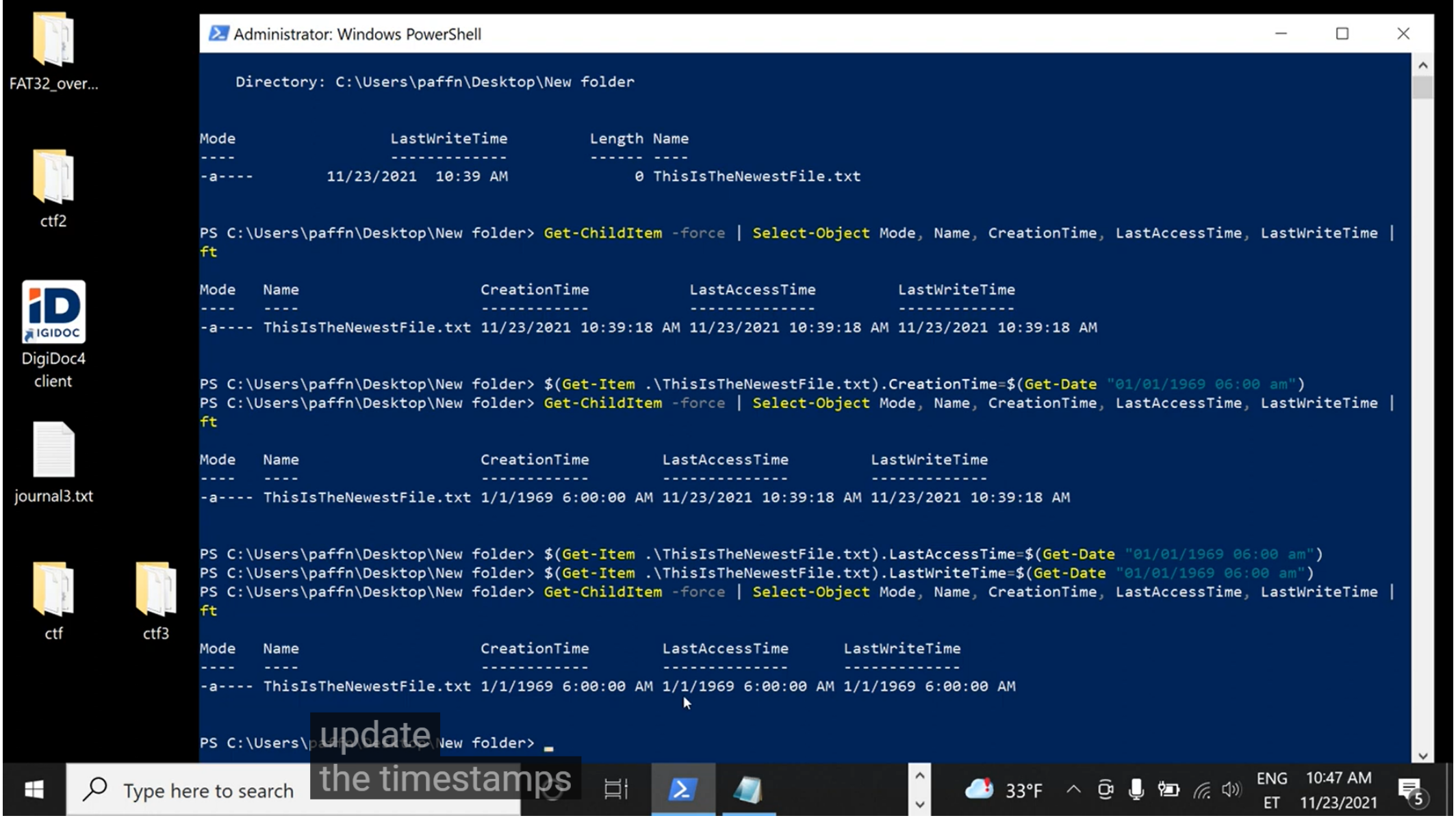
0x00 (6) – $MFT record number of the parent directory

0x08 (8) – File name creation timestamp

0x10 (8) – File name modification timestamp

0x18 (8) - $MFT Modification Time

0x20 (8) - Last access time

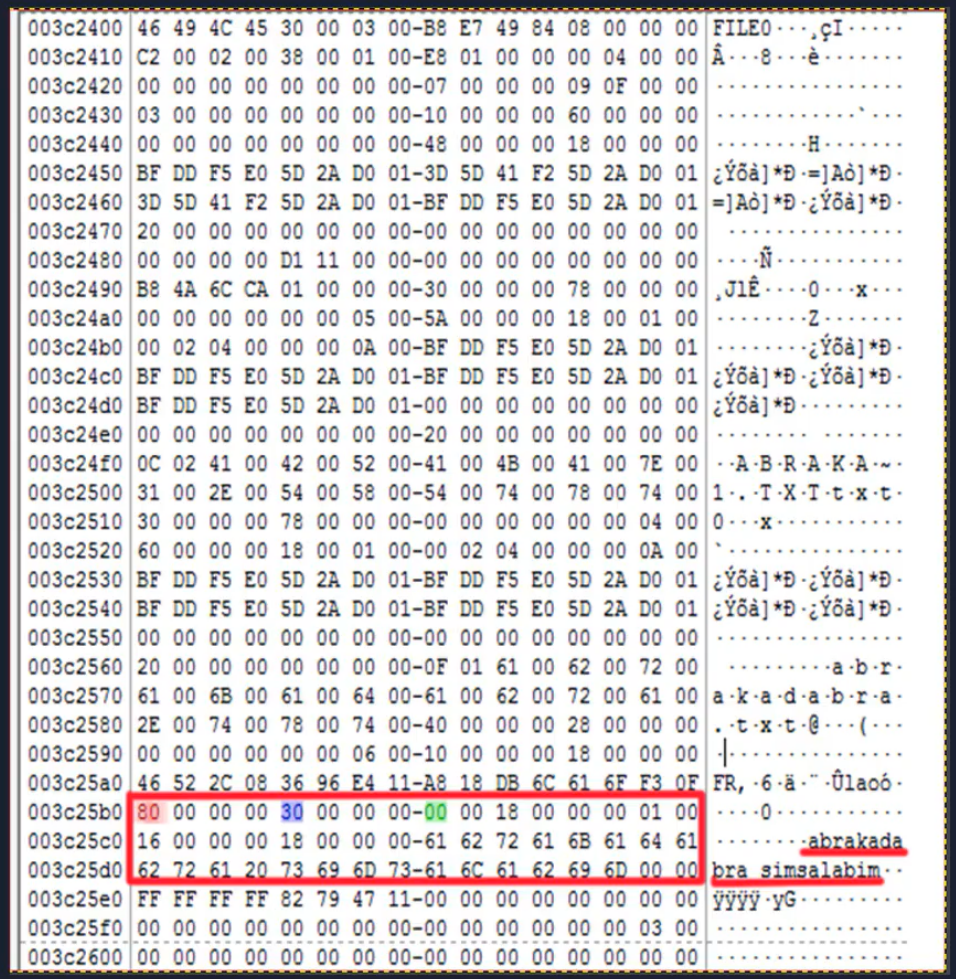


Understanding where file is located using Data and Attr list attribute

DATA ATTRIBUTE – 0X80

- Header – 0x80 00 00 00
- Used to store information about file size and its content location
- Attribute value can be **resident** – file content is stored directly in the attribute (e.g., small files < 500 b in size)
- or **non-resident** – content location information stored in the "run-lists"

DATA ATTRIBUTE WITH RESIDENT CONTENT



```
003c2400 46 49 4C 45 30 00 03 00-B8 E7 49 84 08 00 00 00 FILE0···¸çI·····
003c2410 C2 00 02 00 38 00 01 00-E8 01 00 00 00 04 00 00 Â···8···è·······
003c2420 00 00 00 00 00 00 00 00-07 00 00 00 09 0F 00 00 ················
003c2430 03 00 00 00 00 00 00 00-10 00 00 00 60 00 00 00 ············`···
003c2440 00 00 00 00 00 00 00 00-48 00 00 00 18 00 00 00 ········H·······
003c2450 BF DD F5 E0 5D 2A D0 01-3D 5D 41 F2 5D 2A D0 01 ¿Ýõà]*Ð·=]Aò]*Ð·
003c2460 3D 5D 41 F2 5D 2A D0 01-BF DD F5 E0 5D 2A D0 01 =]Aò]*Ð·¿Ýõà]*Ð·
003c2470 20 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ···············
003c2480 00 00 00 00 D1 11 00 00-00 00 00 00 00 00 00 00 ····Ñ···········
003c2490 B8 4A 6C CA 01 00 00 00-30 00 00 00 78 00 00 00 ¸JlÊ····0···x···
003c24a0 00 00 00 00 00 00 05 00-5A 00 00 00 18 00 01 00 ········Z·······
003c24b0 00 02 04 00 00 00 0A 00-BF DD F5 E0 5D 2A D0 01 ········¿Ýõà]*Ð·
003c24c0 BF DD F5 E0 5D 2A D0 01-BF DD F5 E0 5D 2A D0 01 ¿Ýõà]*Ð·¿Ýõà]*Ð·
003c24d0 BF DD F5 E0 5D 2A D0 01-00 00 00 00 00 00 00 00 ¿Ýõà]*Ð·········
003c24e0 00 00 00 00 00 00 00 00-20 00 00 00 00 00 00 00 ········ ·······
003c24f0 0C 02 41 00 42 00 52 00-41 00 4B 00 41 00 7E 00 ··A·B·R·A·K·A·~·
003c2500 31 00 2E 00 54 00 58 00-54 00 74 00 78 00 74 00 1·.·T·X·T·t·x·t·
003c2510 30 00 00 00 78 00 00 00-00 00 00 00 00 00 04 00 0···x···········
003c2520 60 00 00 00 18 00 01 00-00 02 04 00 00 00 0A 00 `···············
003c2530 BF DD F5 E0 5D 2A D0 01-BF DD F5 E0 5D 2A D0 01 ¿Ýõà]*Ð·¿Ýõà]*Ð·
003c2540 BF DD F5 E0 5D 2A D0 01-BF DD F5 E0 5D 2A D0 01 ¿Ýõà]*Ð·¿Ýõà]*Ð·
003c2550 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
003c2560 20 00 00 00 00 00 00 00-0F 01 61 00 62 00 72 00  ·········a·b·r·
003c2570 61 00 6B 00 61 00 64 00-61 00 62 00 72 00 61 00 a·k·a·d·a·b·r·a·
003c2580 2E 00 74 00 78 00 74 00-40 00 00 00 28 00 00 00 .·t·x·t·@···(···
003c2590 00 00 00 00 00 00 06 00-10 00 00 00 18 00 00 00 ·┤··············
003c25a0 46 52 2C 08 36 96 E4 11-A8 18 DB 6C 61 6F F3 0F FR,·6·ä·¨·Ûlaoó·
003c25b0 80 00 00 00 30 00 00 00-00 00 18 00 00 00 01 00 ····0···········
003c25c0 16 00 00 00 18 00 00 00-61 62 72 61 6B 61 64 61 ········abrakada
003c25d0 62 72 61 20 73 69 6D 73-61 6C 61 62 69 6D 00 00 bra simsalabim··
003c25e0 FF FF FF FF 82 79 47 11-00 00 00 00 00 00 00 00 ÿÿÿÿ·yG·········
003c25f0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 03 00 ················
003c2600 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
```

Offset in the attribute value field:

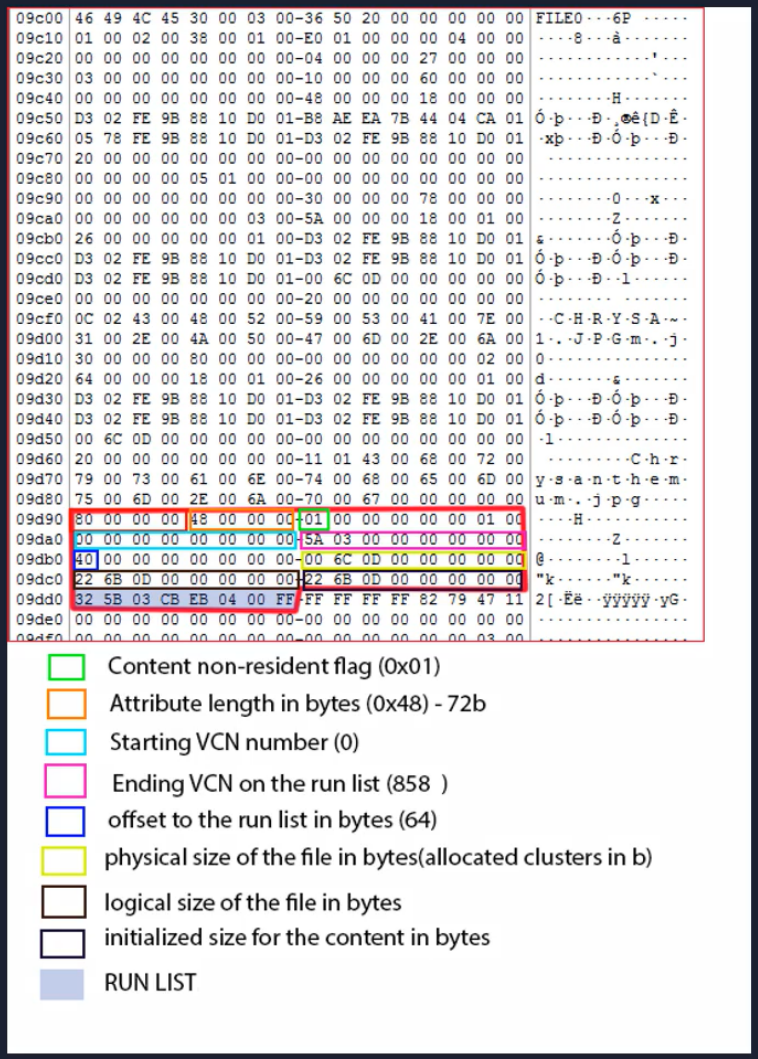
0x00 (4) – Data attribute ID – 0x80

0x05 (2) – size of attribute in bytes

0x08 (1) – content flag:

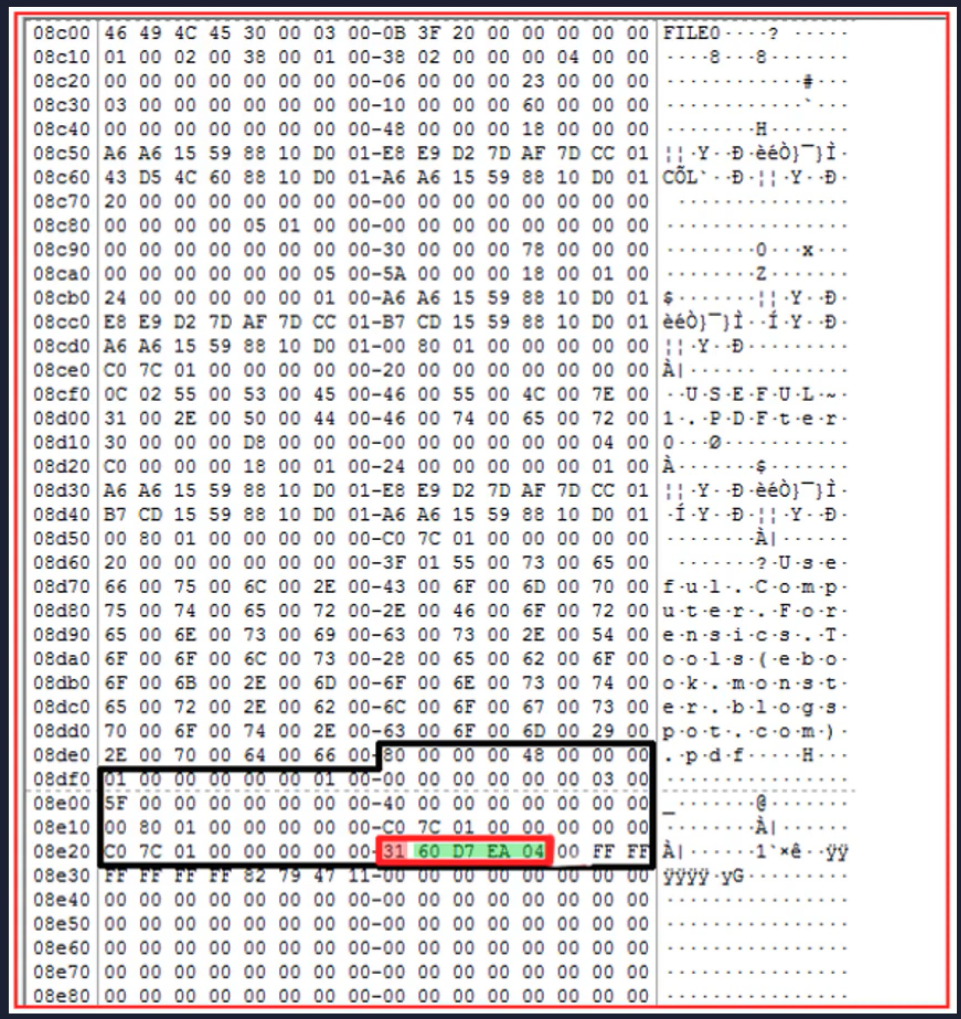
- 0x00 – resident
- 0x01 – nonresident

DATA ATTRIBUTE WITH NONRESIDENT CONTENT



- Flag 0x01 In the attribute header offset 0x08 points to non-residend content
- Content structure:
- 0x10 (8) – Starting VCN of the run list
  (Virtual Cluster Number relative to start of data)
- 0x18 (8) – Ending VCN of the run list
- 0x20 (2) – Offset to the run list from the start of the attribute
- 0x28 (8) – Physical size of the file
- 0x30 (8) - Logical size of the file
- 0x38 (8) – Initialized size of the file
- ~ Run list

$DATA ATTRIBUTE RUN LISTS



```
08c00  46 49 4C 45 30 00 03 00-0B 3F 20 00 00 00 00 00  FILE0····? ·····
08c10  01 00 02 00 38 00 01 00-38 02 00 00 00 04 00 00  ····8···8·······
08c20  00 00 00 00 00 00 00 00-06 00 00 00 23 00 00 00  ············#···
08c30  03 00 00 00 00 00 00 00-10 00 00 00 60 00 00 00  ············`···
08c40  00 00 00 00 00 00 00 00-48 00 00 00 18 00 00 00  ········H·······
08c50  A6 A6 15 59 88 10 D0 01-E8 E9 D2 7D AF 7D CC 01  ¦¦·Y··Ð·èéÒ}¯}Ì·
08c60  43 D5 4C 60 88 10 D0 01-A6 A6 15 59 88 10 D0 01  CÕL`··Ð·¦¦·Y··Ð·
08c70  20 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ···············
08c80  00 00 00 00 05 01 00 00-00 00 00 00 00 00 00 00  ···············
08c90  00 00 00 00 00 00 00 00-30 00 00 00 78 00 00 00  ········0···x···
08ca0  00 00 00 00 00 00 05 00-5A 00 00 00 18 00 01 00  ········Z·······
08cb0  24 00 00 00 00 00 01 00-A6 A6 15 59 88 10 D0 01  $·······¦¦·Y··Ð·
08cc0  E8 E9 D2 7D AF 7D CC 01-B7 CD 15 59 88 10 D0 01  èéÒ}¯}Ì··Í·Y··Ð·
08cd0  A6 A6 15 59 88 10 D0 01-00 80 01 00 00 00 00 00  ¦¦·Y··Ð·········
08ce0  C0 7C 01 00 00 00 00 00-20 00 00 00 00 00 00 00  À|······ ·······
08cf0  0C 02 55 00 53 00 45 00-46 00 55 00 4C 00 7E 00  ··U·S·E·F·U·L·~·
08d00  31 00 2E 00 50 00 44 00-46 00 74 00 65 00 72 00  1·.·P·D·F·t·e·r·
08d10  30 00 00 00 D8 00 00 00-00 00 00 00 00 00 04 00  0···Ø···········
08d20  C0 00 00 00 18 00 01 00-24 00 00 00 00 00 01 00  À·······$·······
08d30  A6 A6 15 59 88 10 D0 01-E8 E9 D2 7D AF 7D CC 01  ¦¦·Y··Ð·èéÒ}¯}Ì·
08d40  B7 CD 15 59 88 10 D0 01-A6 A6 15 59 88 10 D0 01  ·Í·Y··Ð·¦¦·Y··Ð·
08d50  00 80 01 00 00 00 00 00-C0 7C 01 00 00 00 00 00  ·········À|·····
08d60  20 00 00 00 00 00 00 00-3F 01 55 00 73 00 65 00  ·······?·U·s·e·
08d70  66 00 75 00 6C 00 2E 00-43 00 6F 00 6D 00 70 00  f·u·l·.·C·o·m·p·
08d80  75 00 74 00 65 00 72 00-2E 00 46 00 6F 00 72 00  u·t·e·r·.·F·o·r·
08d90  65 00 6E 00 73 00 69 00-63 00 73 00 2E 00 54 00  e·n·s·i·c·s·.·T·
08da0  6F 00 6F 00 6C 00 73 00-28 00 65 00 62 00 6F 00  o·o·l·s·(·e·b·o·
08db0  6F 00 6B 00 2E 00 6D 00-6F 00 6E 00 73 00 74 00  o·k·.·m·o·n·s·t·
08dc0  65 00 72 00 2E 00 62 00-6C 00 6F 00 67 00 73 00  e·r·.·b·l·o·g·s·
08dd0  70 00 6F 00 74 00 2E 00-63 00 6F 00 6D 00 29 00  p·o·t·.·c·o·m·)·
08de0  2E 00 70 00 64 00 66 00-80 00 00 00 48 00 00 00  .·p·d·f·····H···
08df0  01 00 00 00 00 00 01 00-00 00 00 00 00 00 03 00  ···············
08e00  5F 00 00 00 00 00 00 00-40 00 00 00 00 00 00 00  _······@······
08e10  00 80 01 00 00 00 00 00-C0 7C 01 00 00 00 00 00  ······À|·····
08e20  C0 7C 01 00 00 00 00 00-31 60 D7 EA 04 00 FF FF  À|·····1`×ê··ÿÿ
08e30  FF FF FF FF 82 79 47 11-00 00 00 00 00 00 00 00  ÿÿÿÿ·yG·········
08e40  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ···············
08e50  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ···············
08e60  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ···············
08e70  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ···············
08e80  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00  ···············
```

Run list consists of:

Header – 1 byte

Header's right nibble (1) – number of bytes used to store the fragment length in clusters

Header's left nibble (3) – number of bytes used to store starting cluster number for the file content

In our example:
There is only one run list – file content is not fragmented
File data length in clusters – 0x60 or 96 clusters
File starts at 0x4EAD7 or 322263 cluster

ObjectID attribute

OBJECT ID ATTRIBUTE -0X40

- Header 0x40 00 00 00
- Contains GUID-1 value
- Resident, but optional (not always there)
- NTFS link-tracking support is based on object ID attribute (files can be opened by their object IDs rather than their file name
- An application can assign an object ID to a file with file system control code FSCTL_SET_OR_GET_OBJECT_ID, or delete it with FSCTL_DELETE_OBJECT_ID
- File object ID-s are also stored in $ObjId metafile (object identifier file)

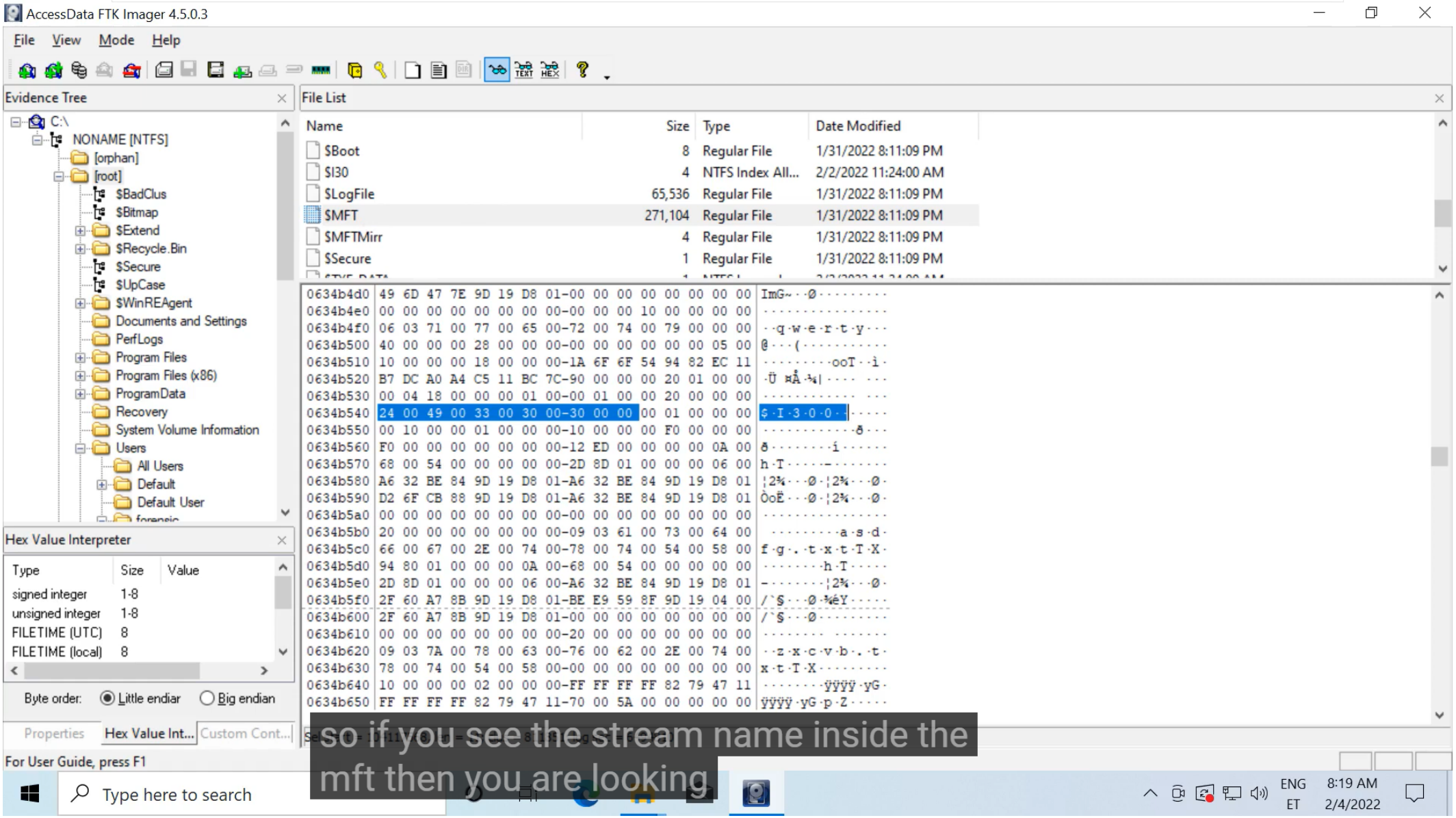
OBJECT ID ATTRIBUTE -0X40

- ObjectID attribute stores 16 byte long GUID value for a file or folder
- GUID-1 value of file's object ID can be converted into MAC address and GUID timestamp (OS boot time and date prior the file creation)
- NTFS keeps the correspondence between Object IDs and $MFT record number of the file in $O index of $ObjId metafile

Directory Tree Indexes

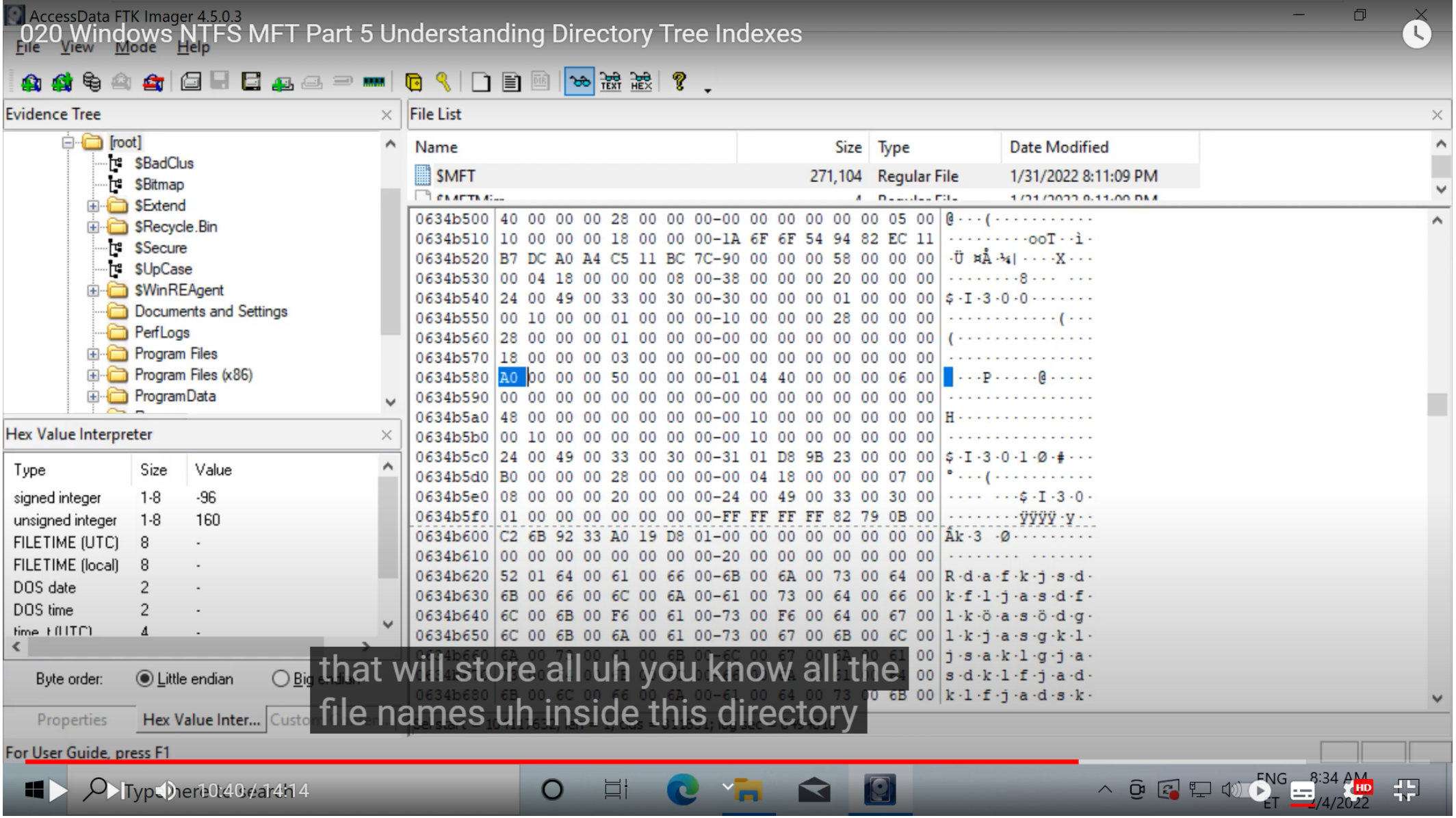
DIRECTORY TREE STRUCTURE IN NTFS

- NTFS uses indexes (attributes that are stored in a sorted order) to compose the directory tree and track changes in the directories
- The file system elements that store the NTFS directory indexes are $Index_Root, $Index_Allocation attributes and $Index_Buffers
- Directory tree is organized using a B-Tree concept based on the filenames
- Indexes store the directory names along with directory related timestamps



Index_Root: $I30 (0x900000)

Timestamp of File: offset



Index_Allocation: (0xA00000)
offset 32 (0x48): size of attribute
after that is runlist

- Index_Allocation point to Index_Buffer
  orlikoski/CyLR: CyLR - Live Response Collection Tool (github.com)

blueangel's ForensicNote (google.com)

Run FTK Imager from a flash drive (Imager Lite) : Support Portal (freshdesk.com)

Extracted_metafiles - Google Drive

``

EXT

ext2 ext3 and ext4 filesystems

EXT(ENDED) FILE SYSTEMS

EXT2 – 1993 , first stable EXT FS

- *up to 2 TB files and 32 TB file systems*
- uses block groups and allocation bitmaps to limit file system fragmentation
- E2fsprogs – ext2 utilities
- fsck – FS check and repair utility
- libext2fs – general library to access ext2

let's
first take a look at some facts related

EXT(ENDED) FILE SYSTEMS

ext3 FS – 2001 (new version of ext2)

Added features:

- Journalling
- Hashed btrees
- ACL's and Extended Attributes
- backward and forward compability with ext2

it's not suitable for the modern operating systems

EXT(ENDED) FILE SYSTEMS

ext4 FS

Added features:

- Use of extents instead of indirect blocks
- Delayed Allocation (lazy write)
- fallocate() - allocation blocks in advance (free blocks indication without wiping)
- subsecond timestamps
- option to enable or disable FS journal
- tons of configurable options

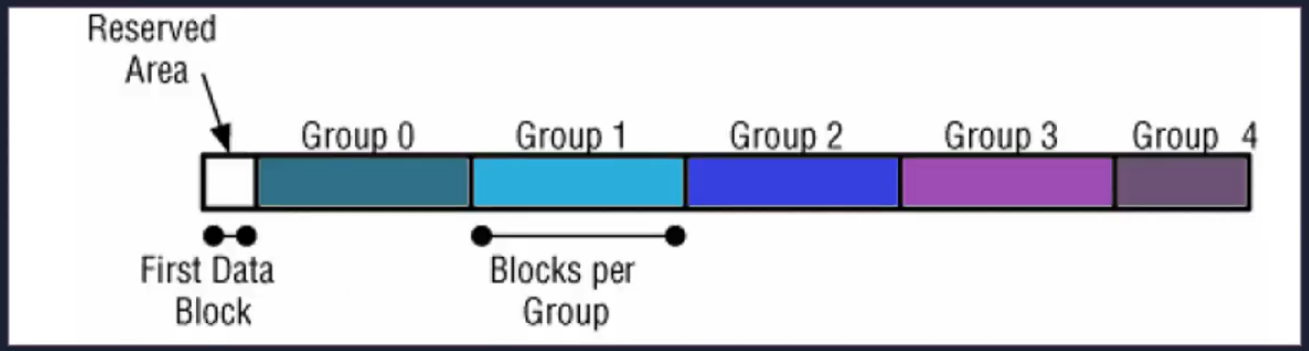
actually for for for tracking the file
data that's why the x4 file system was

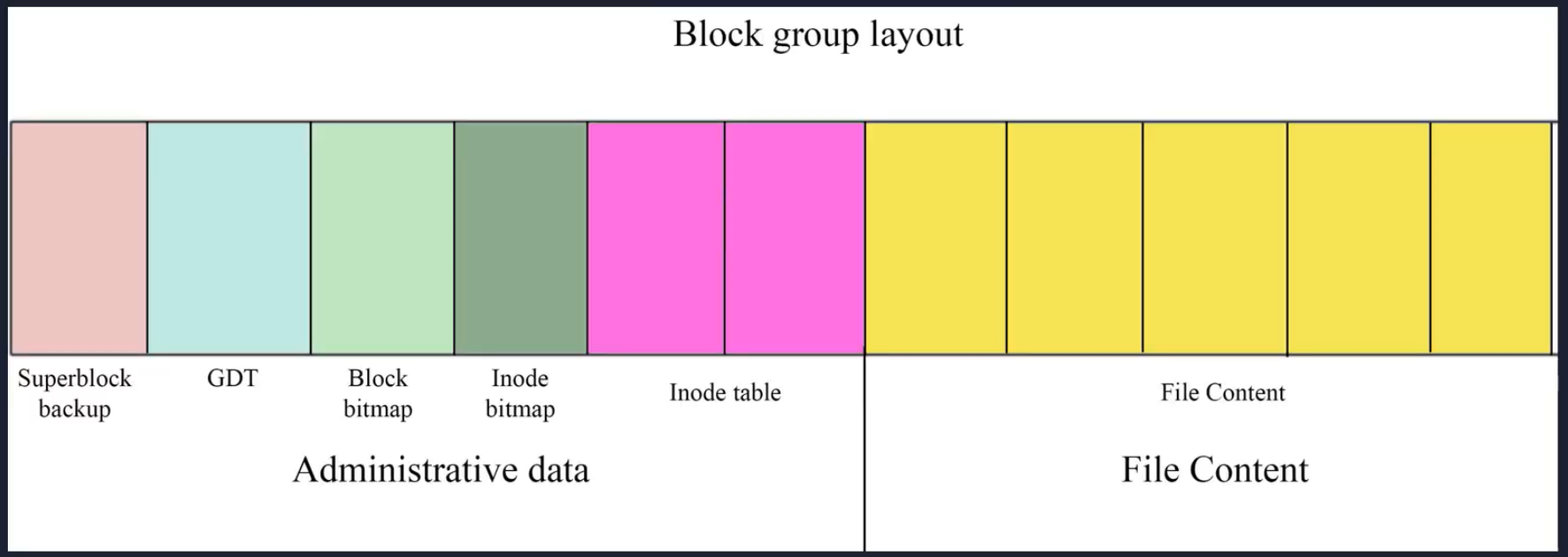
EXT FILE SYSTEM ELEMENTS

- Main elements:
- Blocks = clusters
- Block groups
- Superblock
- Group Descriptor Table
- Inodes
- Directory Entries
- Block Bitmap

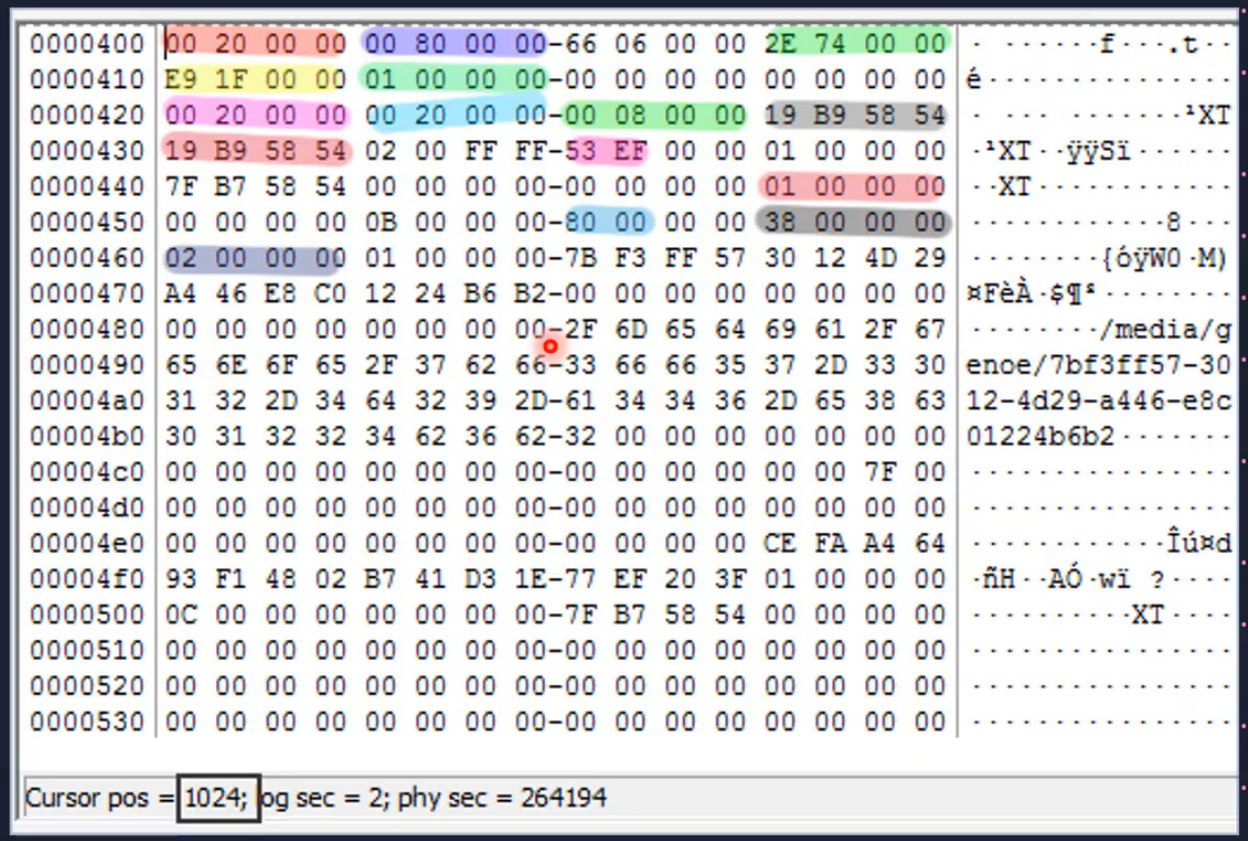
EXT FILE SYSTEM ELEMENTS

- The layout of the file system starts with the **optional reserved area**, and the remainder of the allocated space devided into sections, which are called ***block groups***.
- ***Block groups*** *(except for the last)* contain the same number of blocks, which are used to store file names, metadata and file content
- The extX file system layout is stored in **superblock** in the beginning of the file system


Reserved
Area
Group 0
Group 1
Group 2
Group 3
Group 4
First Data
Block
Blocks per
Group


Block group layout
Superblock
backup
GDT
Block
bitmap
Inode
bitmap
Inode table
File Content
Administrative data
File Content

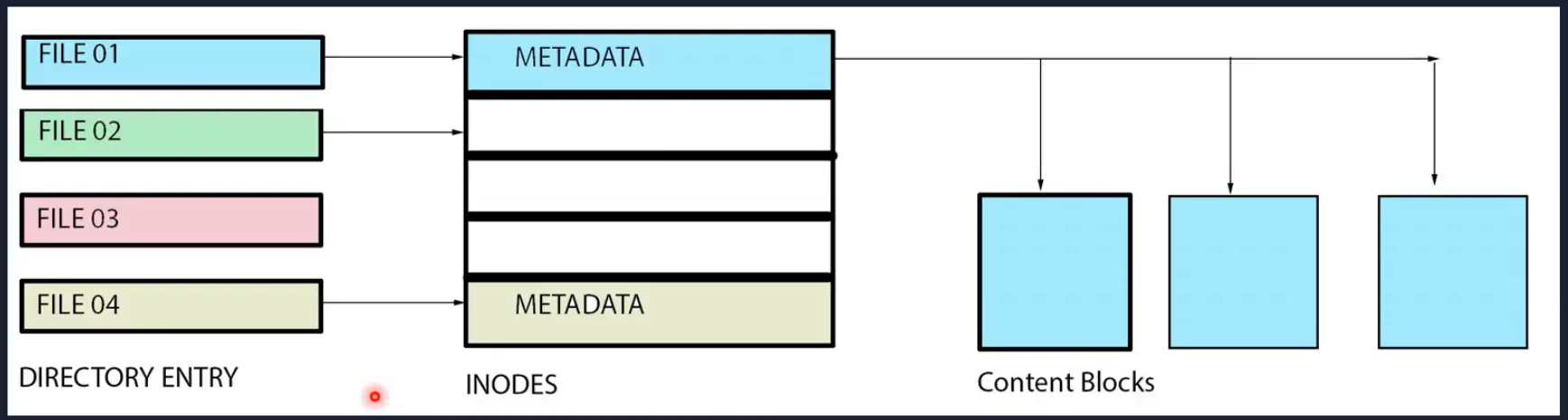
SUPERBLOCK



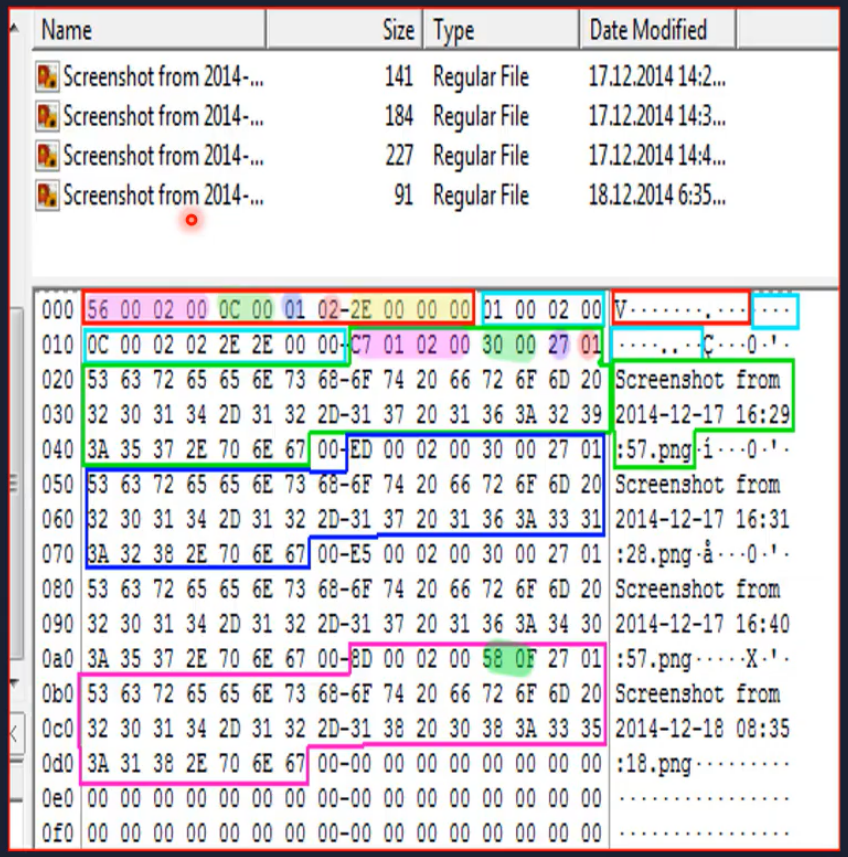
```
0000400 00 20 00 00 00 80 00 00-66 06 00 00 2E 74 00 00 · ·····f···.t··
0000410 E9 1F 00 00 01 00 00 00-00 00 00 00 00 00 00 00 é···············
0000420 00 20 00 00 00 20 00 00-00 08 00 00 19 B9 58 54 · ··· ·····¹XT
0000430 19 B9 58 54 02 00 FF FF-53 EF 00 00 01 00 00 00 ·¹XT··ÿÿSï·····
0000440 7F B7 58 54 00 00 00 00-00 00 00 00 01 00 00 00 ··XT············
0000450 00 00 00 00 0B 00 00 00-80 00 00 00 38 00 00 00 ············8···
0000460 02 00 00 00 01 00 00 00-7B F3 FF 57 30 12 4D 29 ········{óÿW0·M)
0000470 A4 46 E8 C0 12 24 B6 B2-00 00 00 00 00 00 00 00 ¤FèÀ·$¶²········
0000480 00 00 00 00 00 00 00 00-2F 6D 65 64 69 61 2F 67 ········/media/g
0000490 65 6E 6F 65 2F 37 62 66-33 66 66 35 37 2D 33 30 enoe/7bf3ff57-30
00004a0 31 32 2D 34 64 32 39 2D-61 34 34 36 2D 65 38 63 12-4d29-a446-e8c
00004b0 30 31 32 32 34 62 36 62-32 00 00 00 00 00 00 00 01224b6b2·······
00004c0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 7F 00 ················
00004d0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
00004e0 00 00 00 00 00 00 00 00-00 00 00 00 CE FA A4 64 ············Îú¤d
00004f0 93 F1 48 02 B7 41 D3 1E-77 EF 20 3F 01 00 00 00 ·ñH·AÓ·wï ?····
0000500 0C 00 00 00 00 00 00 00-7F B7 58 54 00 00 00 00 ··········XT····
0000510 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
0000520 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················
0000530 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 ················

Cursor pos = 1024; log sec = 2; phy sec = 264194
```

- 0x00 (4) - Number of inodes in file system
- 0x04 (4) - Number of blocks in file system
- 0x08 (4) - Number of blocks reserved to prevent file system from filling up
- 0x0C (4) - Number of unallocated blocks
- 0x10 (4) - Number of unallocated inodes
- 0x14 (4) - Block where block group 0 starts
- 0x18 (4) - Block size (saved as the number of places to shift 1,024 to the left)
- 0x1C (4) – log cluster size (saved as the number of bits to shift 1,024 to the left)
- 0x20 (4) - Number of blocks in each block group
- 0x24 (4)- Number of fragments in each block group
- 0x28 (4) - Number of inodes in each block group
- 0x2C (4) - Last mount time
- 0x30 (4) - Last written time
- 0x34 (2) Current mount count
- 0x38 (2) – Signature 0xEF53

DIRECTORY ENTRIES

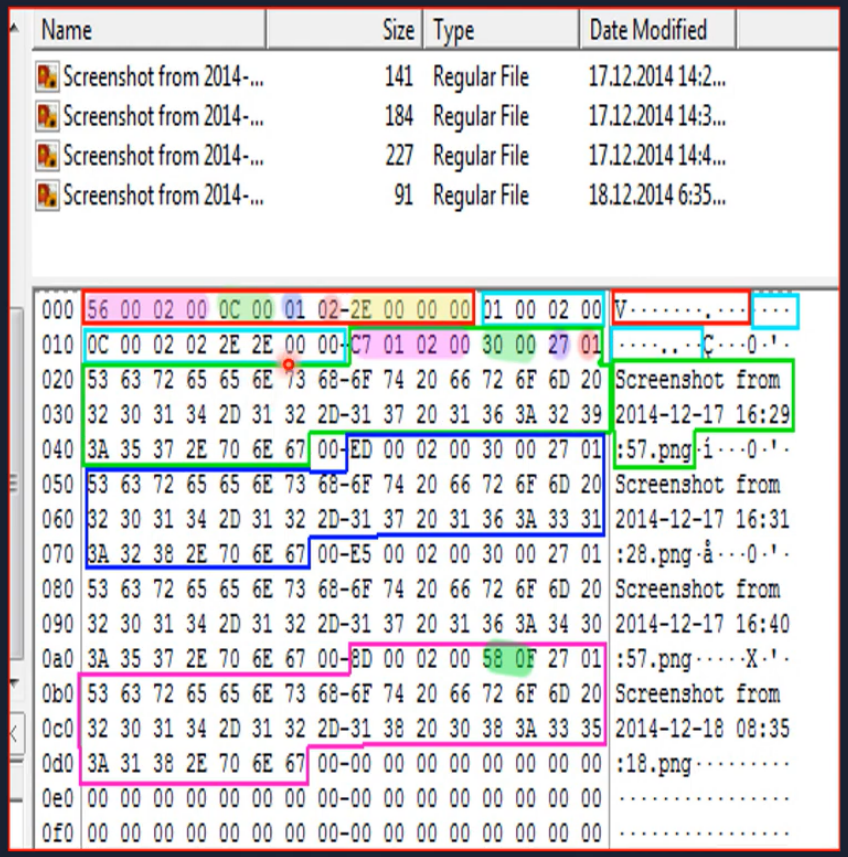


DIRECTORY ENTRIES



Every directory starts with "dot-two dots" directory entries ("." "..") for current and parent directories (analog with FAT subdirectories),
followed by file or subfolder entries

DIRECTORY ENTRIES

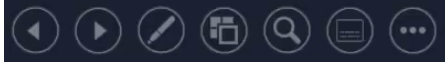


0x00 (4) - Inode number
0x04 (2) - directory entry size in bytes
0x06 (1) – name length (b)
0x07 (1) – file type (01 – regular file, 02 – directory, 03 – character device, 04 – block device, 05 – FIFO, 06 – socket, 07- symbolic link)
0x8 (~) - file name in ASCII

(note, that last file name length is 3928 (b) – points to the end of the block

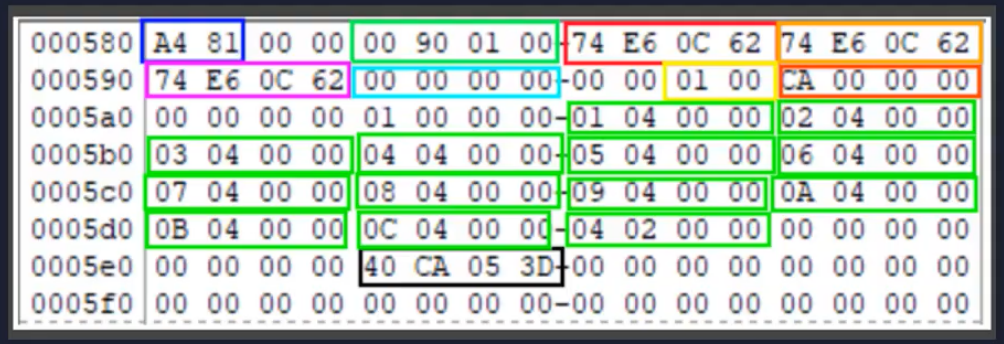
THE INODES

- In ExtX, a file's primary metadata is stored in an inode structure. Additional metadata can be stored in extended attributes
- Superblock defines the size of an inode (128 or 256 bytes)
- Inodes are counted in the Inode Table starting from 1
- Inodes 1 – 11 are typically reserved, inode 2 is used for the root directory
- Ext2/Ext3 uses “indirect pointers” to track the file data, Ext4 uses “extents”



offset: 0x58 (inode size)

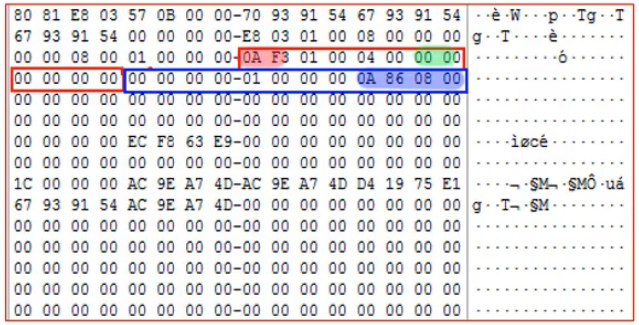
EXT2/EXT3 INODE STRUCTURE



000580	A4	81	00	00	00	90	01	00	74	E6	0C	62	74	E6	0C	62
000590	74	E6	0C	62	00	00	00	00	00	00	01	00	CA	00	00	00
0005a0	00	00	00	00	01	00	00	00	01	04	00	00	02	04	00	00
0005b0	03	04	00	00	04	04	00	00	05	04	00	00	06	04	00	00
0005c0	07	04	00	00	08	04	00	00	09	04	00	00	0A	04	00	00
0005d0	0B	04	00	00	0C	04	00	00	04	02	00	00	00	00	00	00
0005e0	00	00	00	00	40	CA	05	3D	00	00	00	00	00	00	00	00
0005f0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

- 0x00 (2) - File mode (ACL)
- 0x4 (4) file size (in bytes)
- 0x8 (4) – last access time (Unix 32 bit, seconds since Jan 1, 1970 UTC)
- 0xC (4) -last inode change time
- 0x10 (4) – last data modification time
- 0x14 (4) – deletion time
- 0x1A (2) – hard link count
- 0x1C (4) – file size in blocks
- 0x28 – start of indirect pointers

EXT4 INODE STRUCTURE



Extent tree header (12 bytes) with signature 0xF30A

offset 0x06 (2b) - depth of extent tree, if =0, then extent node points to data blocks

Extent node (leaf node, 12bytes)

offset 0x06 (2) - higher 16-bits of the starting block number

offset 0x08 (4) - lower 32 bits of the starting block number)

EXT4 INODE STRUCTURE