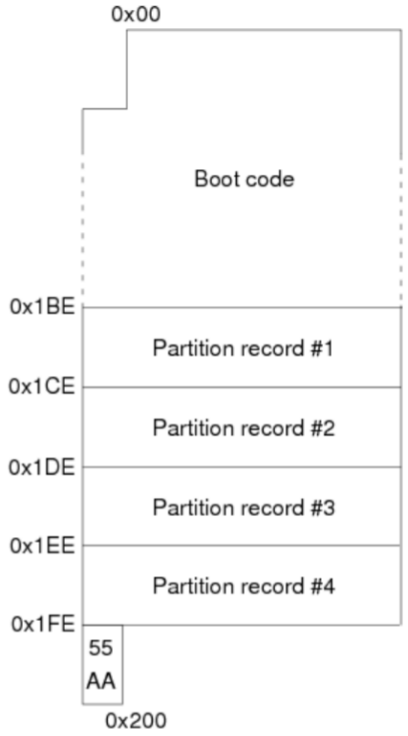
Ungraded Pre-Lecture Quiz

- ***Master Boot Record (MBR)*** is used in older disks by BIOS. Why do the newer **UEFI-based GPT disks** also contain MBR in their first sector?



From: Bruce J. Nikkel, "Forensic Analysis of GPT Disks and GUID Partition Tables", https://www.digitalforensics.ch/nikkel09.pdf

Some Quick Reminders/Announcements

- **A1** is out
- **Graded Lab Tasks #3**: from today's Lab 5
- Please register your **group** (of 4 members) by **19 February** by referring to my Canvas announcement
- After that, I will randomly-assign those unassigned and possibly merge partial groups by **26 February**

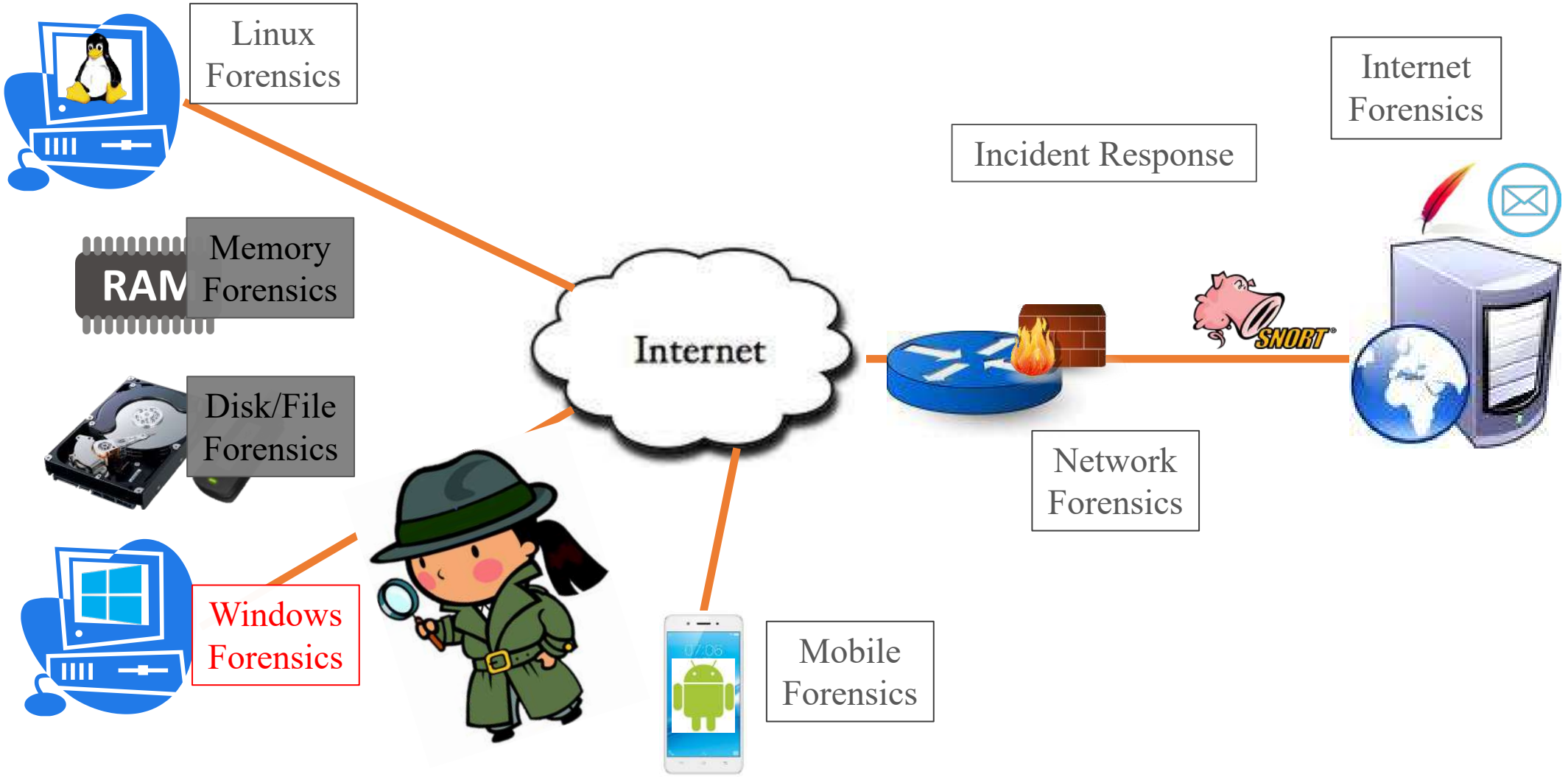
IFS4102: Digital Forensics

Lecture 5: Windows Forensics (Part 1)

Outline

- Intro to Windows forensics
- File system analysis: FAT
- File system analysis : NTFS & ADS
- MAC times in Windows
- Live/online Windows analysis
- Windows registry analysis
- Lab 5 exercises

This Lecture's Focus



Intro to Windows Forensics

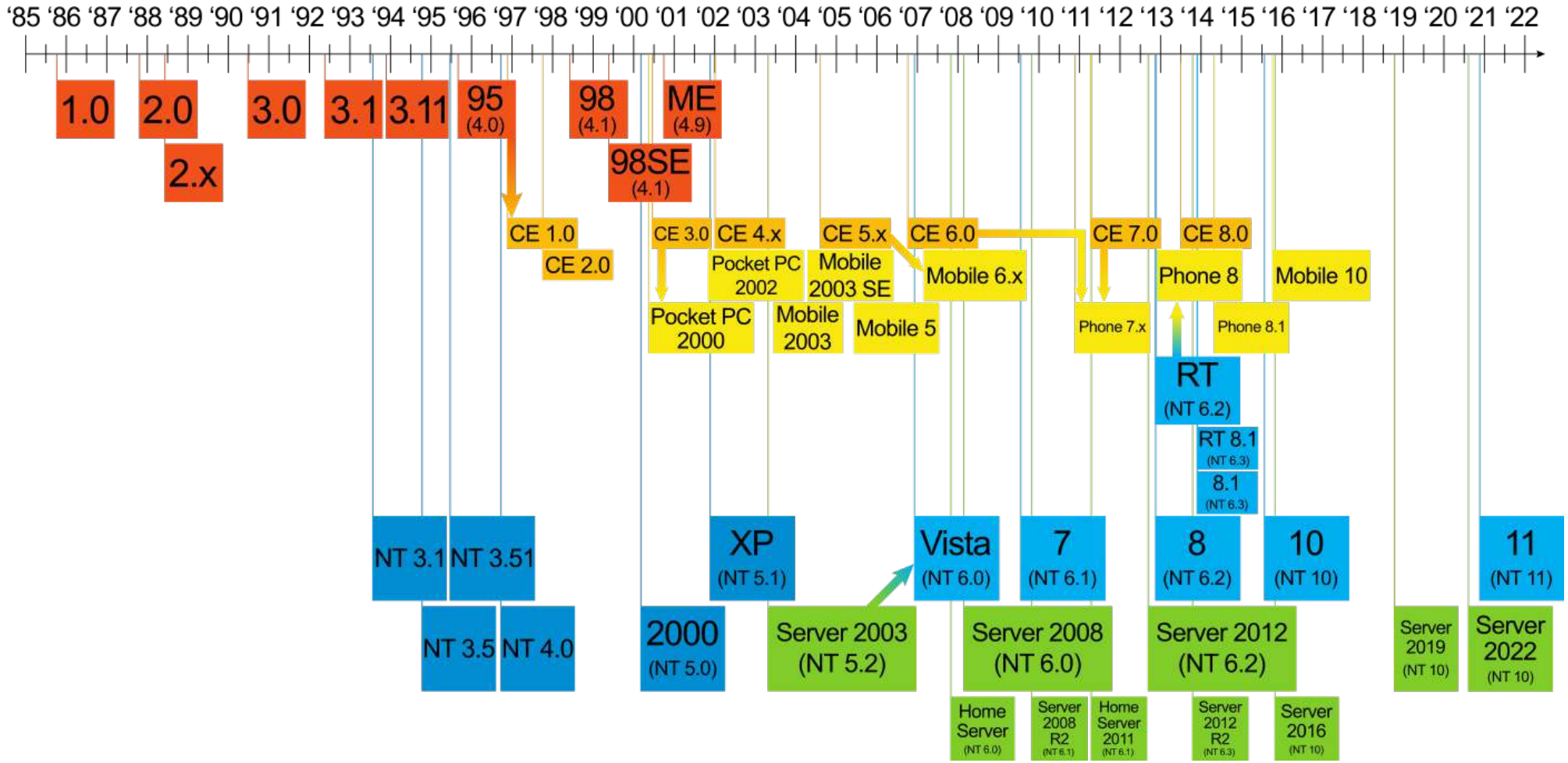
Why Windows Forensics?

- Windows is still **the most popular** desktop/laptop OS:
  - Net Applications & StatCounter track OSes in devices active on the Web: the most used OS family on PCs: **76.12%-86.19%** usage share [Wikipedia]
- Two widely-popular file systems to store user-stored data:
  - File Allocation Table (**FAT**): also used in **USB flash drives**, **small storage media**, embedded/IoT devices
  - New Technology File System (**NTFS**)
- Various **computer-generated artefacts** for forensic analysis:
  - Registry, prefetch files, shortcuts & jump lists, thumbnail cache, …
  - Event logs (*in Lecture 6*)
  - Network & browser information (*in Lecture 7*)

Windows Family Overview

- **Windows NT**: started with Windows NT 3.1, for servers and workstations
  - Windows: **Windows XP, Vista, 7, 8, 8.1, 10**
  - Windows **Server**: …, Windows Server 2019
  - Windows **PE** (a live operating system): …, Windows PE 10
- Windows **IoT** (previously Windows Embedded)
- Obsolete:
  - Windows 9x
  - Windows Mobile
  - Windows Phone

Windows Family Tree

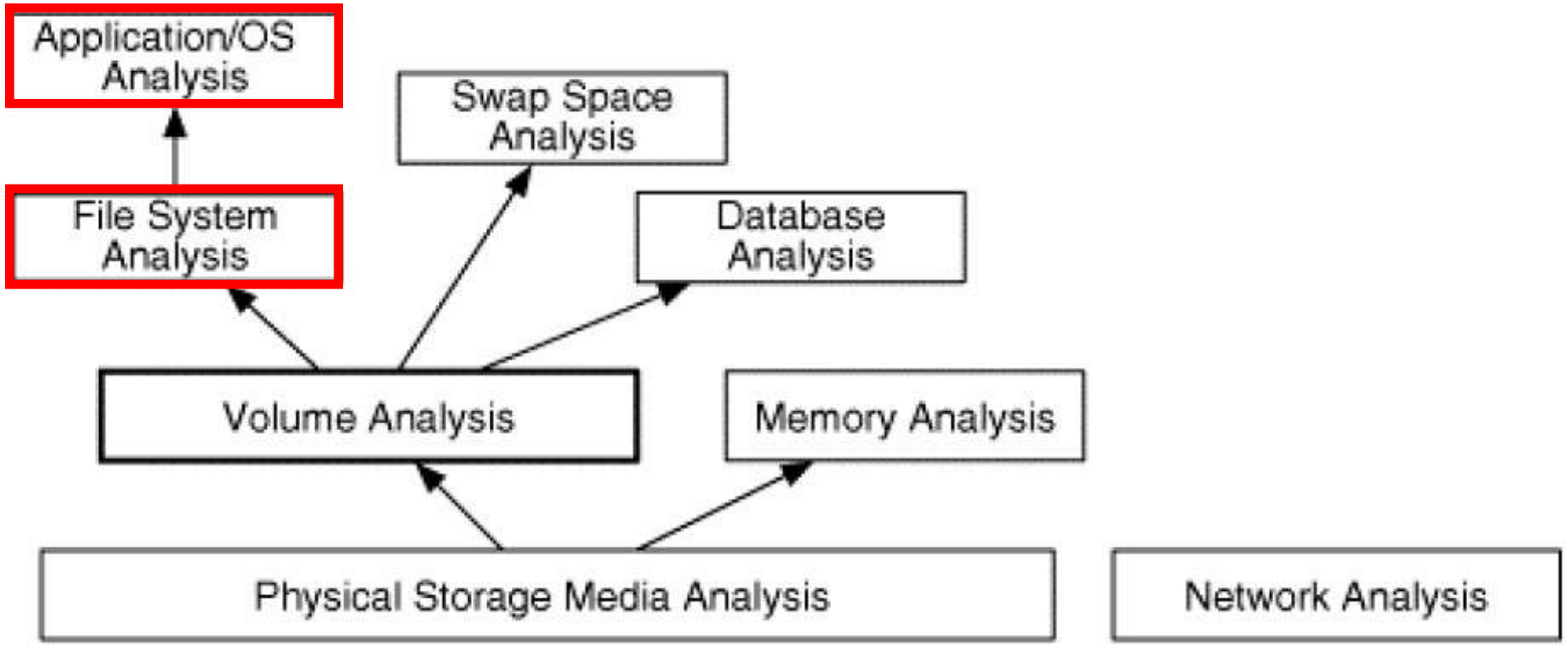


From: Wikipedia

Potential Challenges in Windows Forensics

- Windows environment is **complex**,
  and poses a number of **challenges** for Digital Forensic Investigator
- Windows **version differentiation**:
  important with analysis as different artefacts reside in
  different locations
- Different versions of **file system types**: FAT12, FAT16, FAT32, …
- **Invasive** characteristics of the Windows environment:
  it does *not* mount hard disk drives as read-only
- Possible **obsolete legacy** Windows systems & versions

Layers of Disk & File Analysis

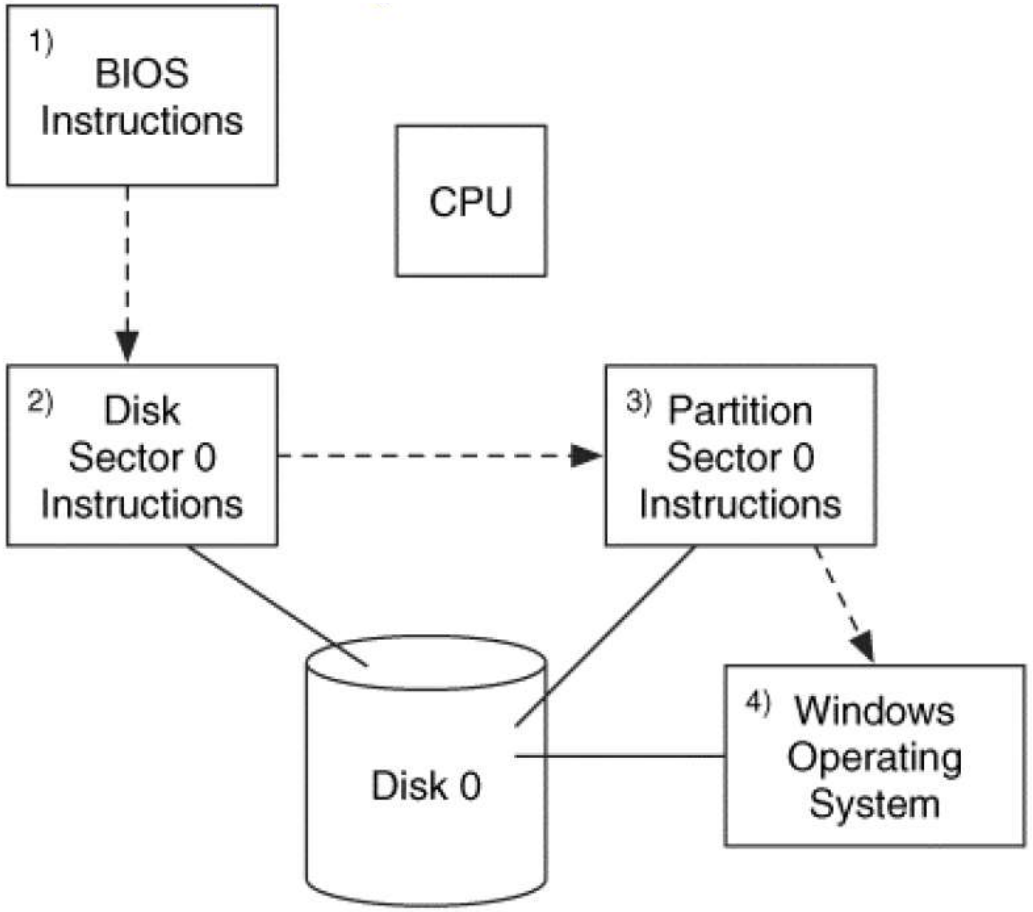


From: Brian Carrier, “File System Forensic Analysis”

File System Analysis: FAT

Sample Boot Sequence (Using BIOS & MBR)

Review

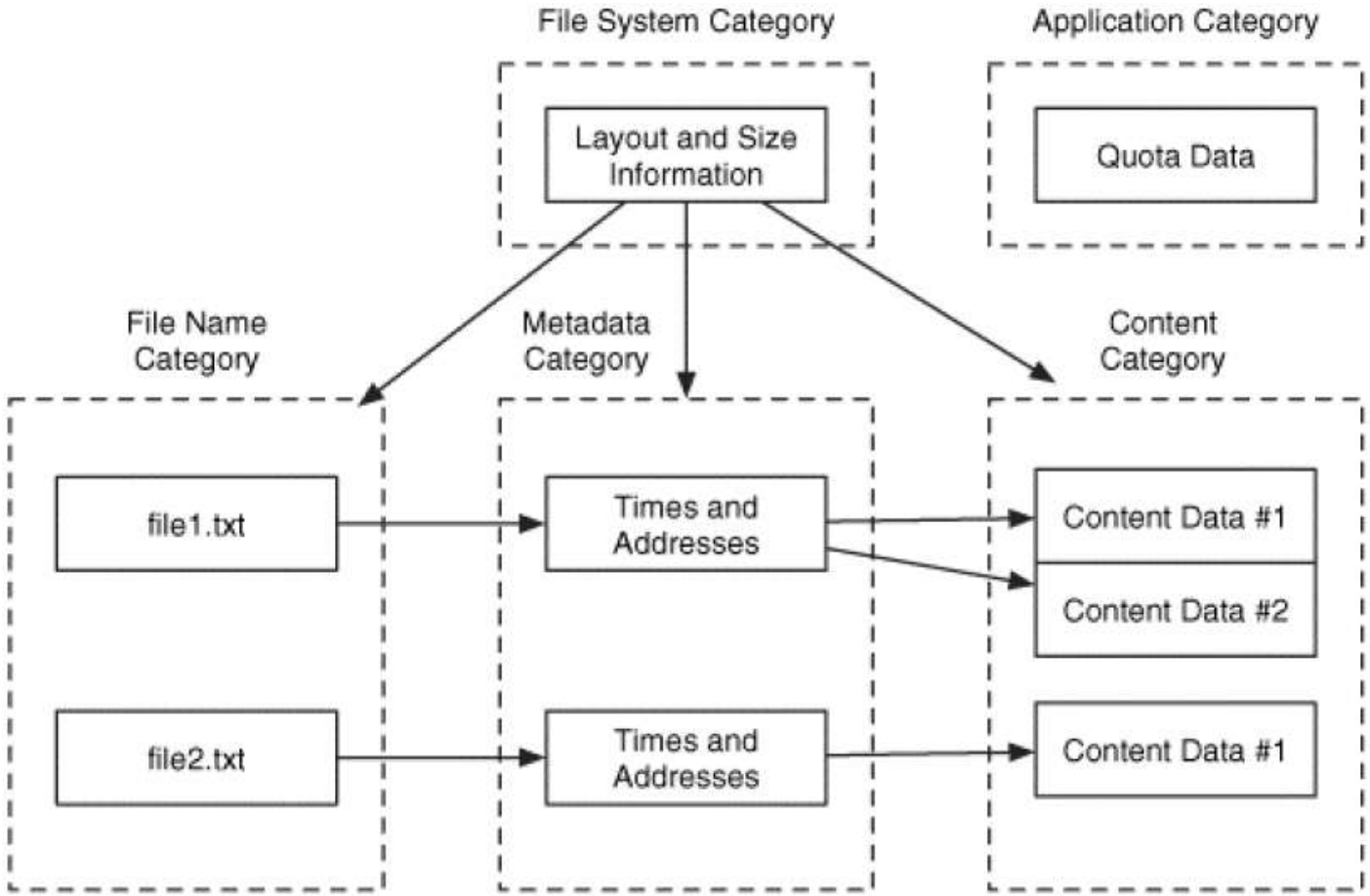


From: Brian Carrier, "File System Forensic Analysis"

Review

File System Management Categories

- A reference model with **5 categories**, which is also used by TSK

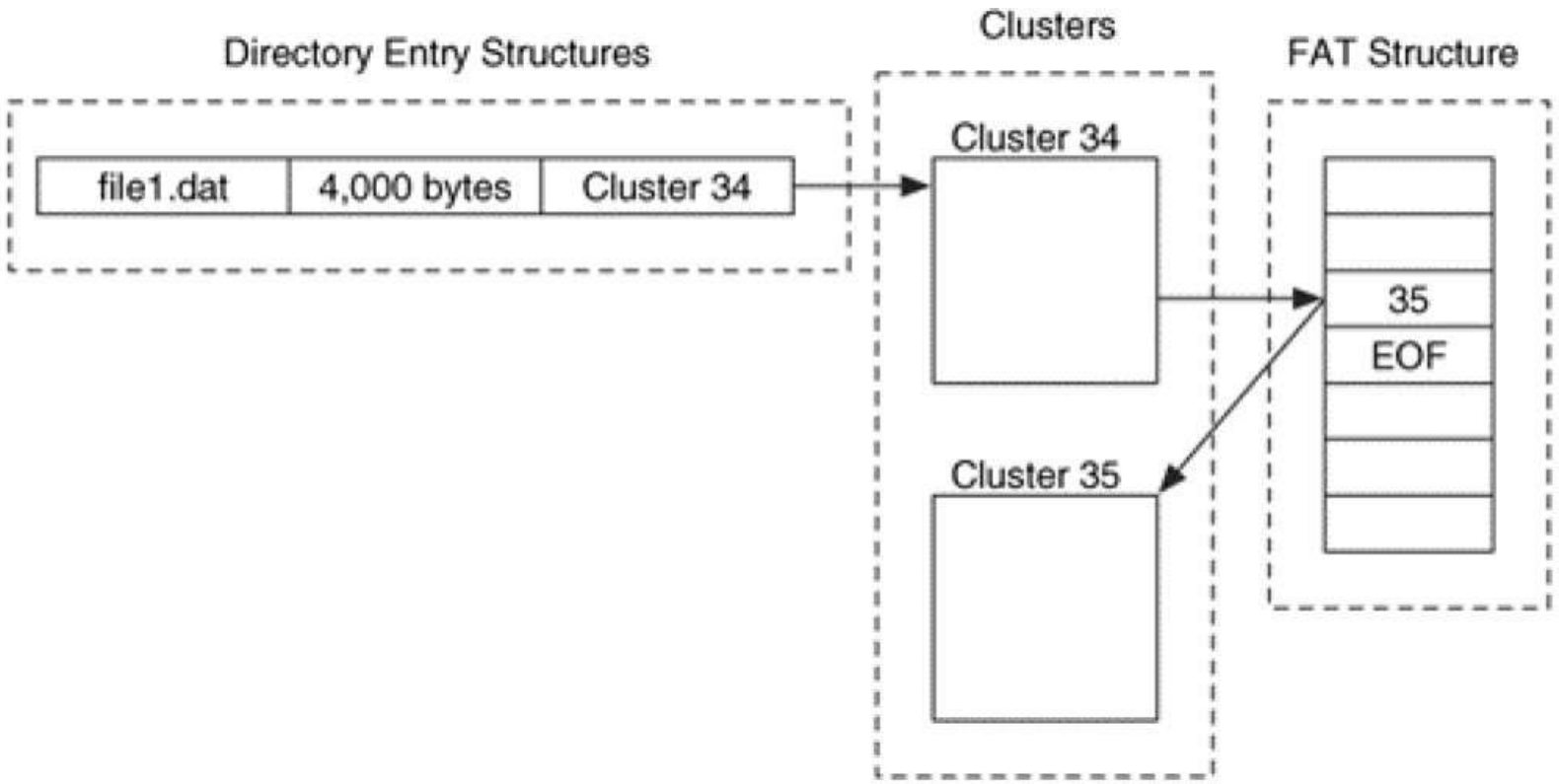


From: Brian Carrier, "File System Forensic Analysis"

File Allocation Table (FAT) File System

- The primary file system of the **Microsoft DOS** & **Windows 9x**
- Three major **variants**: FAT12, FAT16, FAT32
  - (*8-bit FAT*): FAT *precursor*, originally designed in the late 1970s for use on floppy disks and early OS
  - **FAT12**: Started to emerge ~1980, in early DOS & early Windows
  - **FAT16**: Emerged in mid 1980's, in DOS 3.1 through to Windows 95
  - **FAT32**: Emerged in late 1996 with Windows 95 OSR2
- The FAT **version number**:
  relates to the ***size of the entries*** in the **FAT structure**
  → also corresponds to the number of **addressable data clusters**

File Allocation Table (FAT) File System: Overview



From: Brian Carrier, “File System Forensic Analysis”

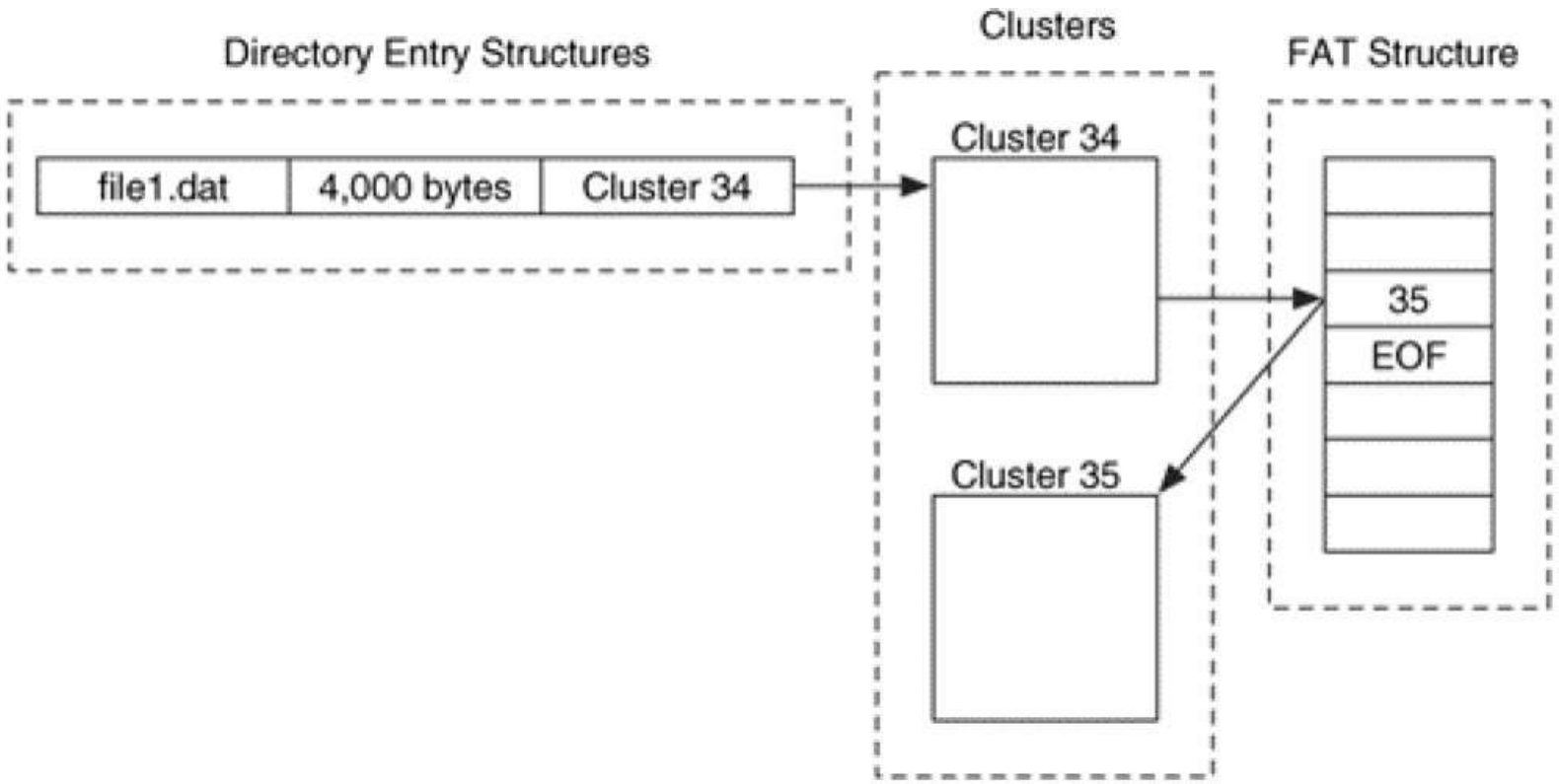
FAT Strengths and Shortcomings

- **Good** aspects:
  - ***Simple*** file-system layout:
    good performance even in lightweight implementations
  - Relatively ***robust*** for storing data: a FAT backup/duplicate
- **Problems**:
  - Limited ability to store **detailed information** about a file:
    no file owner, file permission or ACL, hard/symbolic links, quotas, …
  - Cannot deliver the same performance, reliability and scalability
    as **newer modern** file systems
- **No longer** the default file system for usage on Windows computers:
  NT, XP and later use **NTFS**

Relevancy of FAT File System Nowadays

- Yet, FAT is still **commonly found** on floppy disks, flash & other solid-state memory cards and modules, small storage media, gaming/embedded/IoT devices
- The standard file system for **digital cameras** per the ***Design rule for Camera File system (DCF)*** specification
- Still **supported** (for compatibility reasons) by nearly **all** currently developed OSes for PCs (including UNIX/Linux), many **mobile devices**, and **embedded systems**
- *The point is*: FAT is still ***relevant*** to modern digital forensics!

File Allocation Table (FAT) File System: Overview

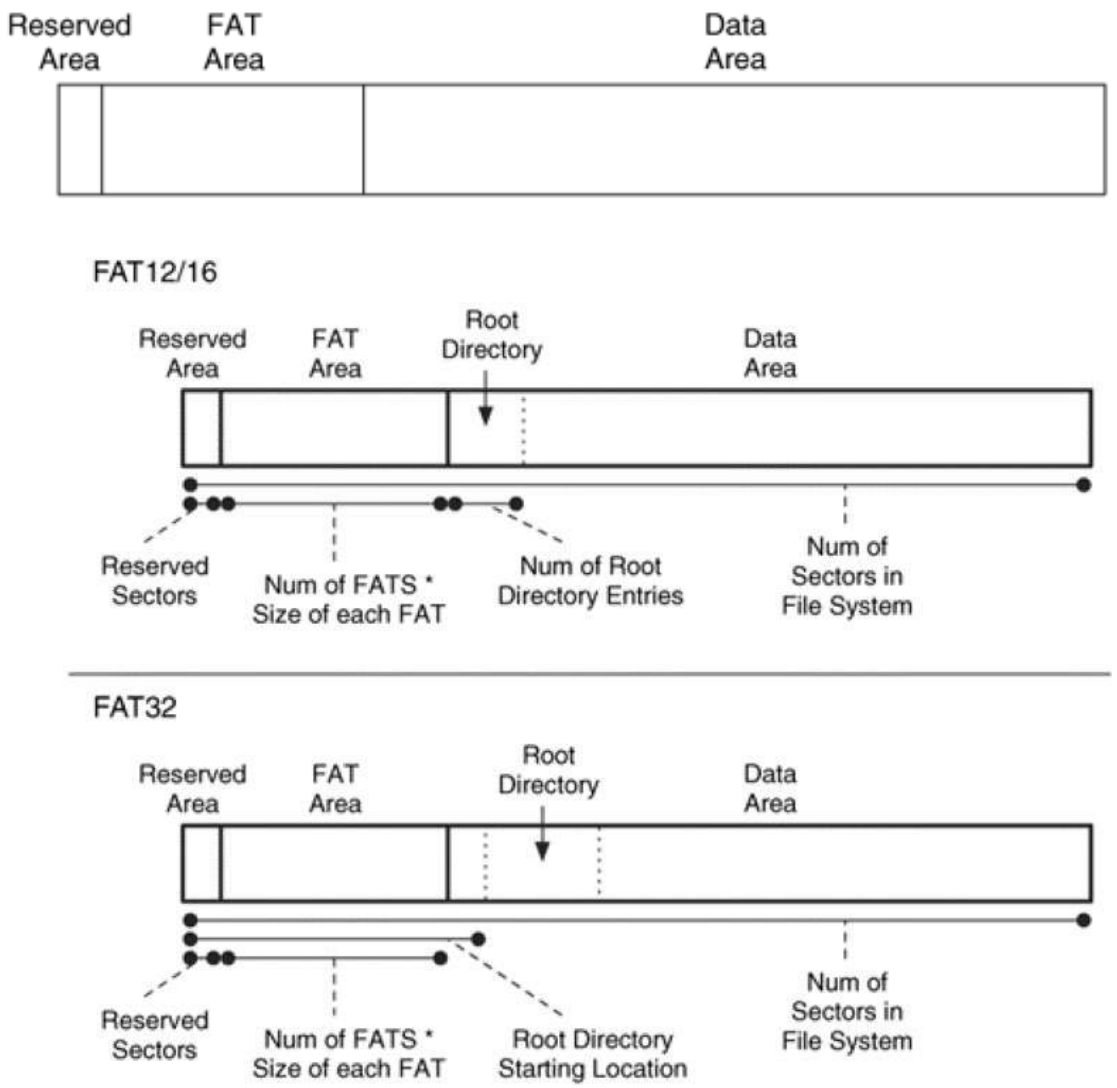


From: Brian Carrier, “File System Forensic Analysis”

FAT Key Concepts: Overview

- Each **file** and **directory** is allocated a data structure called a ***directory entry***
- A directory entry has information of the ***first data cluster***
- If a file/directory has **multiple** allocated data clusters, the ***FAT structure*** indicate the ***next data clusters***
- "***Cluster chains***": all allocated data clusters of a file/directory
- The FAT structure also indicates the **allocation status** of all data clusters in the FAT file system, including damaged ones
- FAT32 file-system **boot sector**: contains **additional data**, e.g. **sector address** of boot sector's backup copy, root directory, FSINFO (which gives the next available cluster, no of free clusters)

FAT Layout



From: Brian Carrier, "File System Forensic Analysis"

FAT Layout

- Can be inspected using TSK's tool **fsstat**: e.g. `fsstat -f fat fat-4.dd`

Overview of the order of structures in a FAT partition or disk

Region	Size in sectors	Contents
Reserved sectors	(number of reserved sectors)	Boot Sector
		FS Information Sector (FAT32 only)
		More reserved sectors (optional)
FAT Region	(number of FATs) * (sectors per FAT)	File Allocation Table #1
		File Allocation Table #2 ... (optional)
Root Directory Region	(number of root entries * 32) / (bytes per sector)	Root Directory (FAT12 and FAT16 only)
Data Region	(number of clusters) * (sectors per cluster)	Data Region (for files and directories) ... (to end of partition or disk)

From: Wikipedia

FAT Layout

The tool used to make the FS: "MSDOS5.0" → WinXP

A 4-byte value based on the FS creation time

```
# fsstat -f fat fat-4.dd
FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: FAT
OEM Name: MSDOS5.0
Volume ID: 0x4c194603
Volume Label (Boot Sector): NO NAME
Volume Label (Root Directory): FAT DISK
File System Type Label: FAT32

Backup Boot Sector Location: 6
FS Info Sector Location: 1
```

From: Brian Carrier, "File System Forensic Analysis"

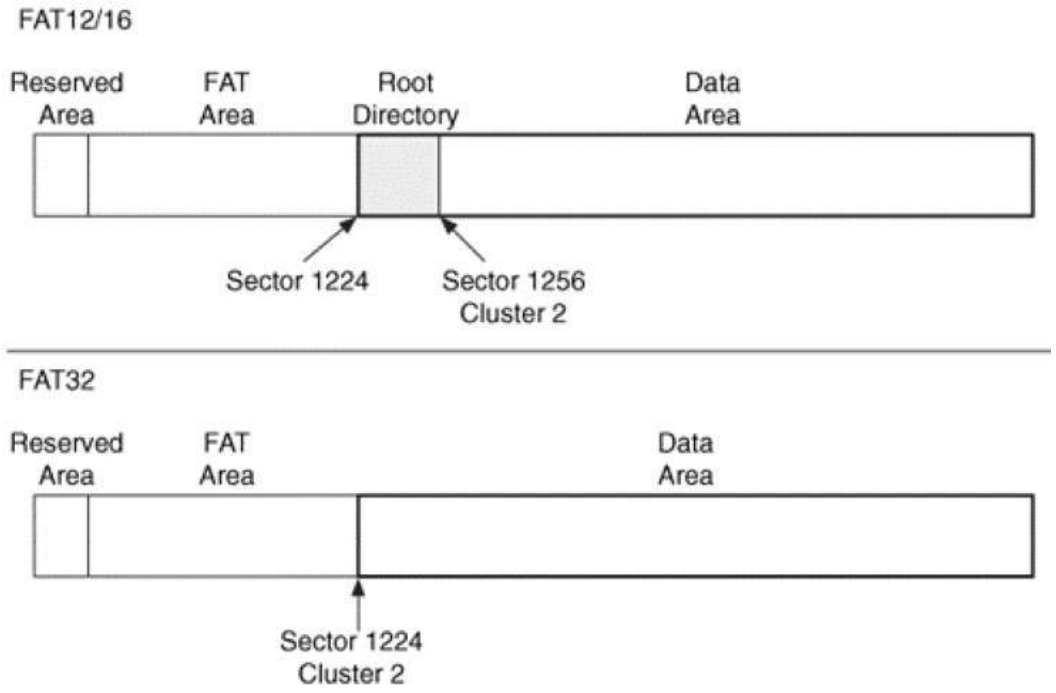
FAT Layout

```
Next Free Sector (FS Info): 1778
Free Sector Count (FS Info): 203836
Sectors before file system: 100800

File System Layout (in sectors)
Total Range: 0 - 205631
* Reserved: 0 - 37
** Boot Sector: 0
** FS Info Sector: 1
** Backup Boot Sector: 6
* FAT 0: 38 - 834
* FAT 1: 835 - 1631
* Data Area: 1632 - 205631
** Cluster Area: 1632 - 205631
*** Root Directory: 1632 - 1635

CONTENT-DATA INFORMATION
------------------------------------------
Sector Size: 512
Cluster Size: 1024
Total Cluster Range: 2 - 102001
[REMOVED]
```

From: Brian Carrier, “File System Forensic Analysis”

Directory

- ***Directory/folder***:
  - Represented as a **special type** of file
  - ***Root directory***: stored in a special location
  - Each file/sub-directory within a folder is represented as a ***directory entry***



From: Brian Carrier, "File System Forensic Analysis"

Directory Entry

- ***Directory entry***:
  - Each file and directory is allocated one: **fixed 32 bytes** in size
  - **Contains**:
    - **File name** & **extension**
    - **Attributes** (1 byte): Read Only, Hidden, System, Volume label (disk volume label is a directory entry), Directory (tells whether the entry is a directory or a file), Archive, …
    - **Created time & created date**: the value is to be added to **1980**, giving a possible year range of 1980 to 2107
    - Last **modified time**, last **modified date**
    - Last **accessed date**, but ***no*** **last accessed time**!
    - First **data cluster**
    - **File size** (in bytes)
  - See: https://en.wikipedia.org/wiki/Design_of_the_FAT_file_system#Directory_entry

Directory Entry

- Sample directory entry as reported by TSK's istat:

```
# istat -f fat fat-4.dd 4
Directory Entry: 4
Allocated
File Attributes: File, Archive
Size: 8689
Name: RESUME-1.RTF

Directory Entry Times:
Written:        Wed Mar 24 06:26:20 2004
Accessed:       Thu Apr 8 00:00:00 2004
Created:        Tue Feb 10 15:49:40 2004

Sectors:
1646 1647 1648 1649 1650 1651 1652 1653
1654 1655 1656 1657 1658 1659 1660 1661
1662 1663
```

Directory-entry address/no
(see Brian Carrier's book)

From: Brian Carrier, "File System Forensic Analysis"

Long File Name (LFN) Support

- ***Virtual FAT* (*VFAT*)**:
  - Supports **long filenames** up to 255
  - Was first used in Win 95
- A **workaround/hack** to support LFN on a file system that only allows **8+3 short file name (SFN)**:
  - Uses additional **multiple directory entries** for a file with LFN:
    - "**Special" directory entries ("LFN")**: uses a previously-invalid **file attribute** value combination, so that non-VFAT applications will ignore them
    - Uses the **first byte** in the **filename** to indicate **directory-entry sequence**
  - The standard 8+3 SFN entry is kept for backward compatibility reason

Long File Name (LFN) Support

- LFN entry format:

Byte offset	Length (bytes)	Description
0x00	1	Sequence Number (bit 6: last logical, first physical LFN entry, bit 5: 0; bits 4-0: number 0x01..0x14 (0x1F), deleted entry: 0xE5)
0x01	10	Name characters (five UCS-2 characters)
0x0B	1	Attributes (always 0x0F)
0x0C	1	Type (always 0x00 for VFAT LFN, other values reserved for future use; for special usage of bits 4 and 3 in SFNs see further up)
0x0D	1	Checksum of DOS file name
0x0E	12	Name characters (six UCS-2 characters)
0x1A	2	First cluster (always 0x0000)
0x1C	4	Name characters (two UCS-2 characters)

From: Wikipedia

Long File Name (LFN) Support: Example

```
Atr: File  Name: RESUME-1.RTF Cluster: 9
Atr: LFN   Seq: 2  CSum: 0xdf  Name: Name.rtf
Atr: LFN   Seq: 1  CSum: 0xdf  Name: My Long File
Atr: File  Name: MYLONG~1.RTF Cluster: 26
Atr: File  Name: _ILE6.TXT Cluster: 48
```

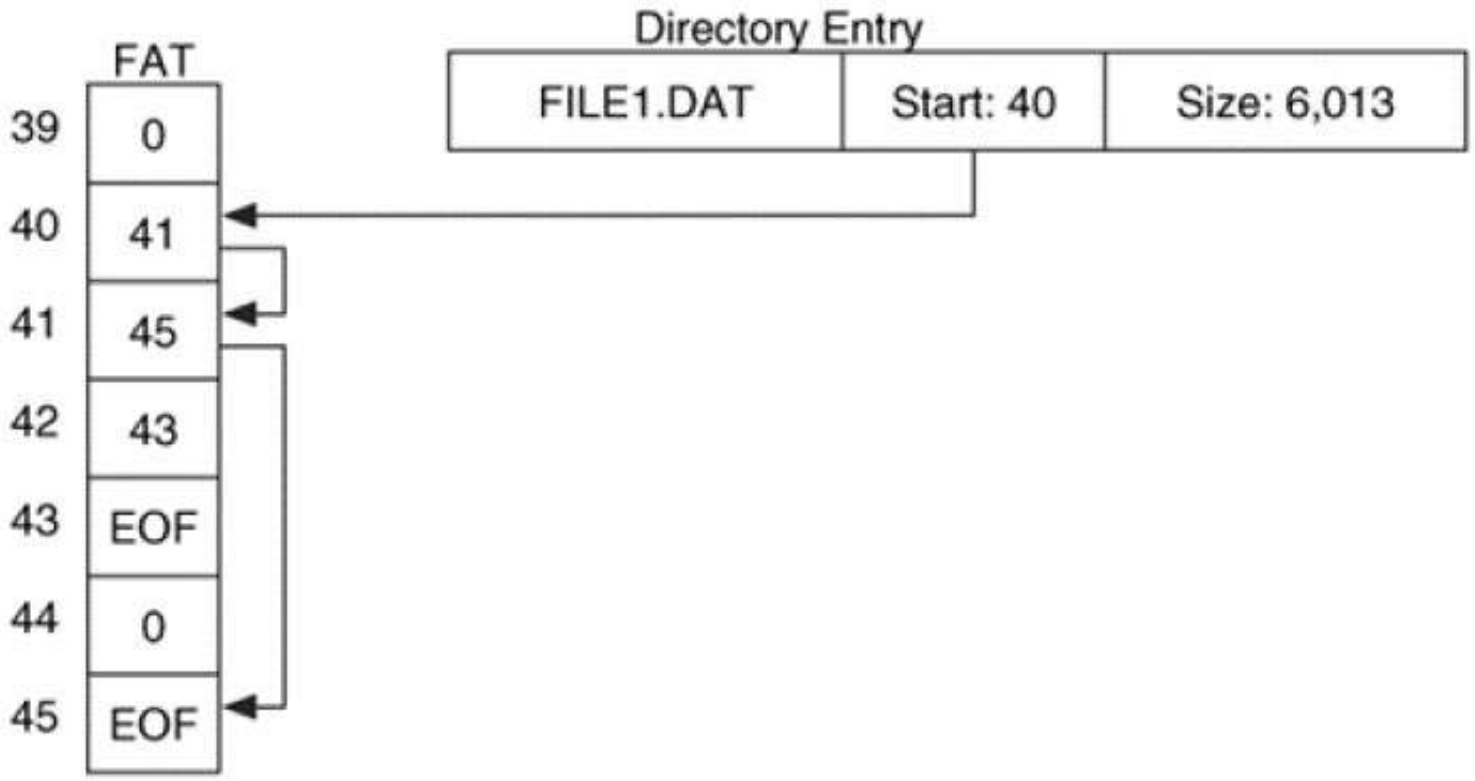
From: Brian Carrier, "File System Forensic Analysis"

```
# fls -f fat fat-2.dd
r/r 3: FAT DISK      (Volume Label Entry)
r/r 4: RESUME-1.RTF
r/r 7: My Long File Name.rtf (MYLONG~1.RTF)
r/r * 8: _ile6.txt
```

Cluster Chain: Definition & Representation

- ***Data cluster***: a data unit of file and directory content
- File data is allocated to **a number** of data clusters
- The clusters is kept as a **linked list**, forming ***cluster chain***:
  - The pointer to the **first cluster** is kept in the file's directory entry
  - The pointers to **subsequent clusters** are kept in the File Allocation Table (FAT) structure
- A ***FAT entry*** contains either *(see the next slide for an example)*:
  - **FREE (0)**: its data cluster is unused/free
  - **EOF**: used, a NULL pointer (its cluster is the **last cluster** in the chain)
  - **Some number**: used, with the number as the **next cluster** in the chain
  - **BAD**: its cluster is unusable (i.e. due to disk error)

FAT File System Structure: Cluster Chain



From: Brian Carrier, “File System Forensic Analysis”

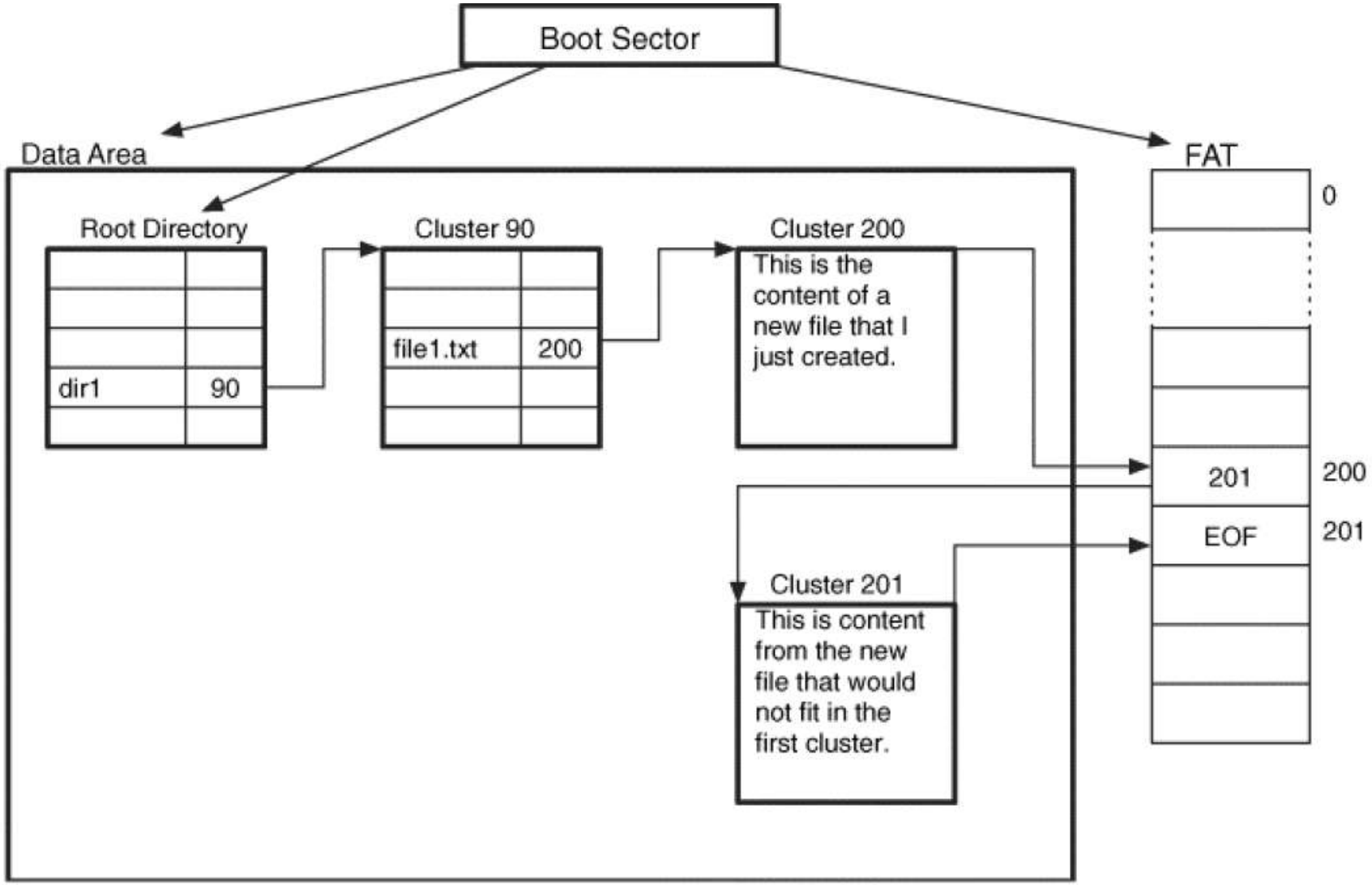
FAT File System Structure: Cluster Chain

- Sample TSK output dumping the contents of the FAT structure:

```
# fsstat -f fat fat-4.dd
[REMOVED]
1642-1645 (4) -> EOF
1646-1663 (18) -> EOF
1664-1681 (18) -> EOF
[REMOVED]
```

From: Brian Carrier, “File System Forensic Analysis”

FAT File System: Sample Layout & File

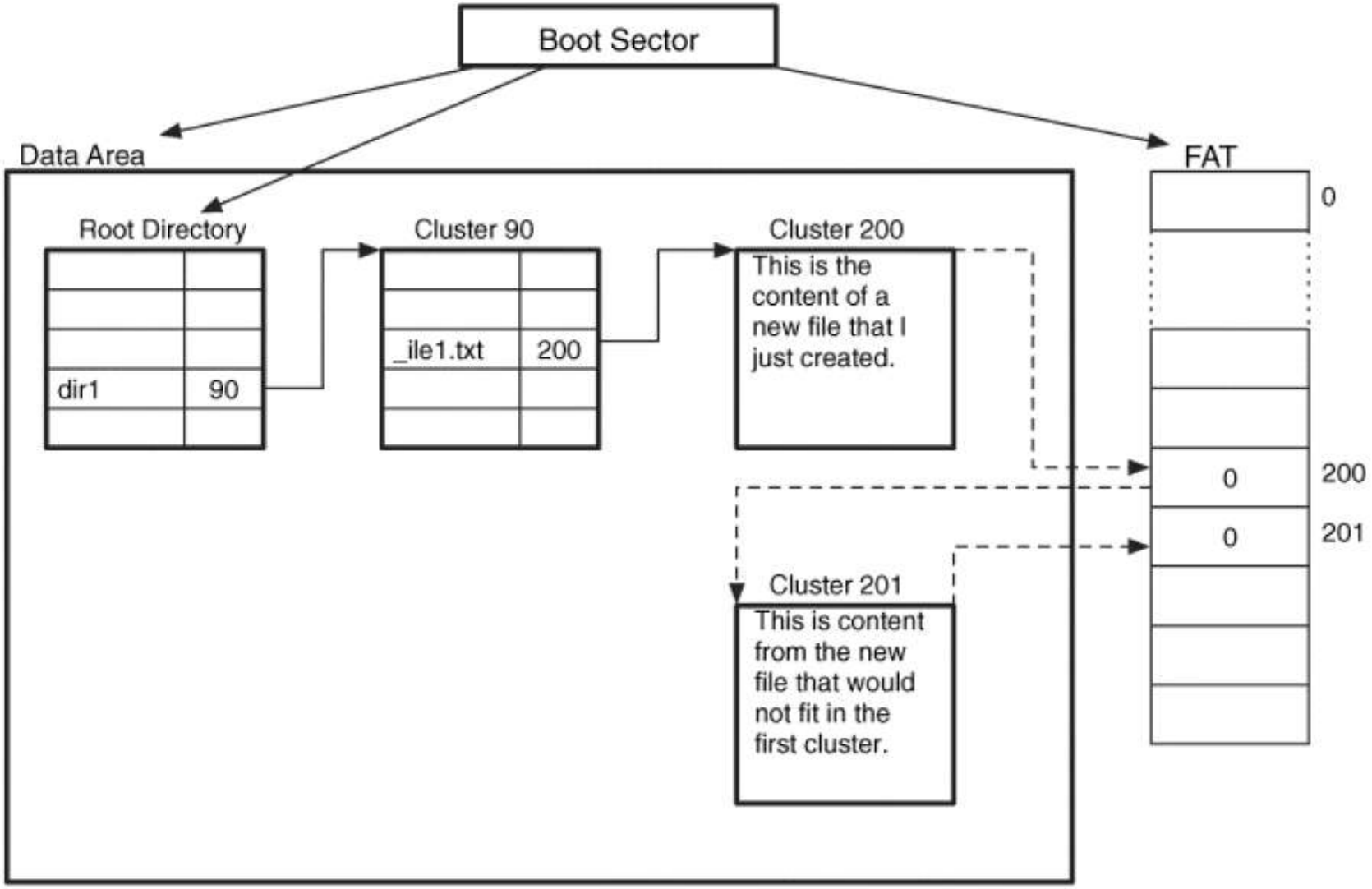


From: Brian Carrier, "File System Forensic Analysis"

File Deletion

- Several **steps** below are performed
- Delete the **directory entry**:
  - Set the ***first letter*** in the filename to a special value **0xe5**
- Free the **allocated data clusters**:
  - Set the corresponding **FAT entries** in the cluster chain to **FREE (0)**
- The **actual file content** inside the previously-allocated clusters?
  - Remains ***intact***: can *possibly* be recovered,
    but it depends on file system activity (not always possible)
- **Example**: the deletion of file "dir1\file1.txt" *(see the next slide)*

FAT File System: Sample Layout & Deletion

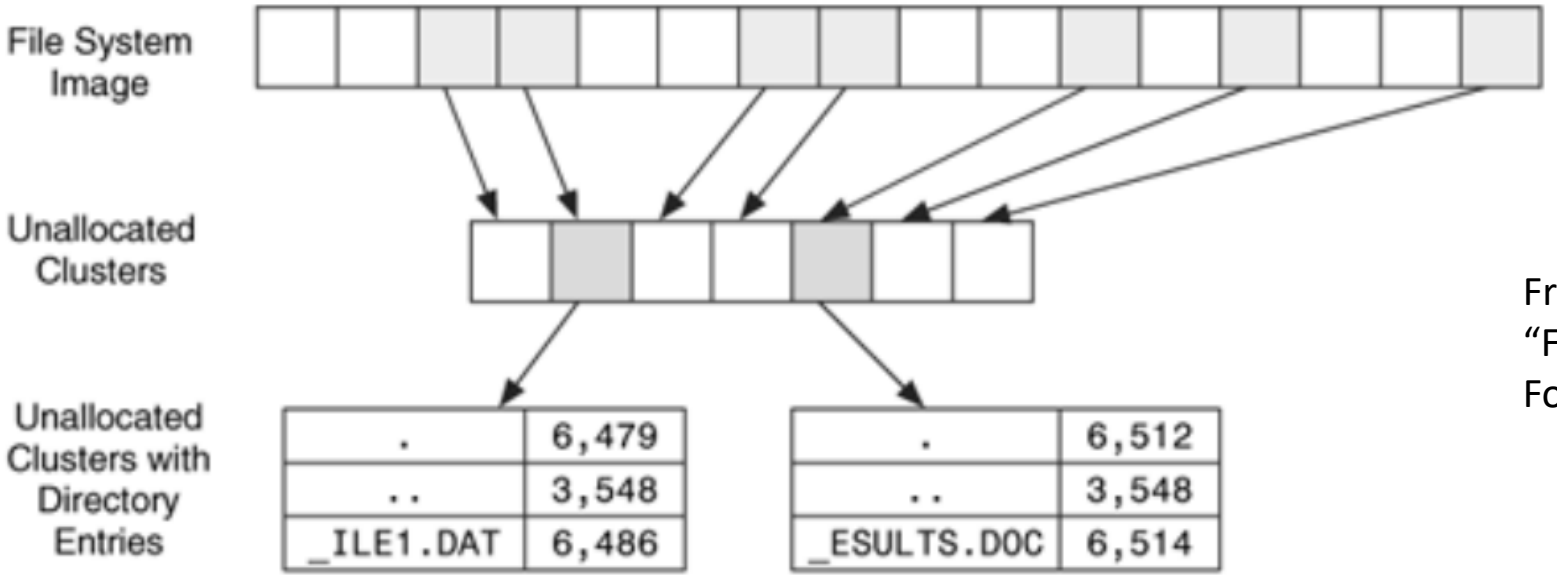


From: Brian Carrier, "File System Forensic Analysis"

File Recovery

Question: How is the recovery of a deleted file done?

- If the directory is still allocated: the **first cluster no** is known
- If the directory is already removed: need to check the unallocated clusters (see an illustration below)

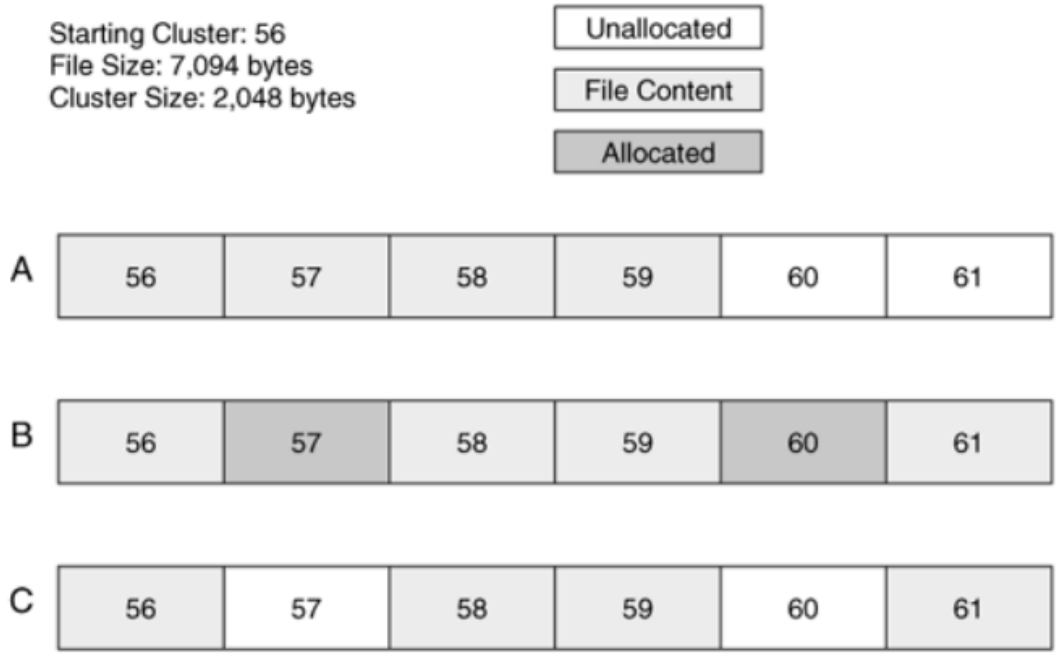


From: Brian Carrier, "File System Forensic Analysis"

File Recovery

becareful of false inclusion & fasle omiss

- *How about the next cluster numbers?*
  → depends on whether the clusters are contiguous or fragmented
  (see some possible cases and respective issues below)
  → the clusters must be still unallocated



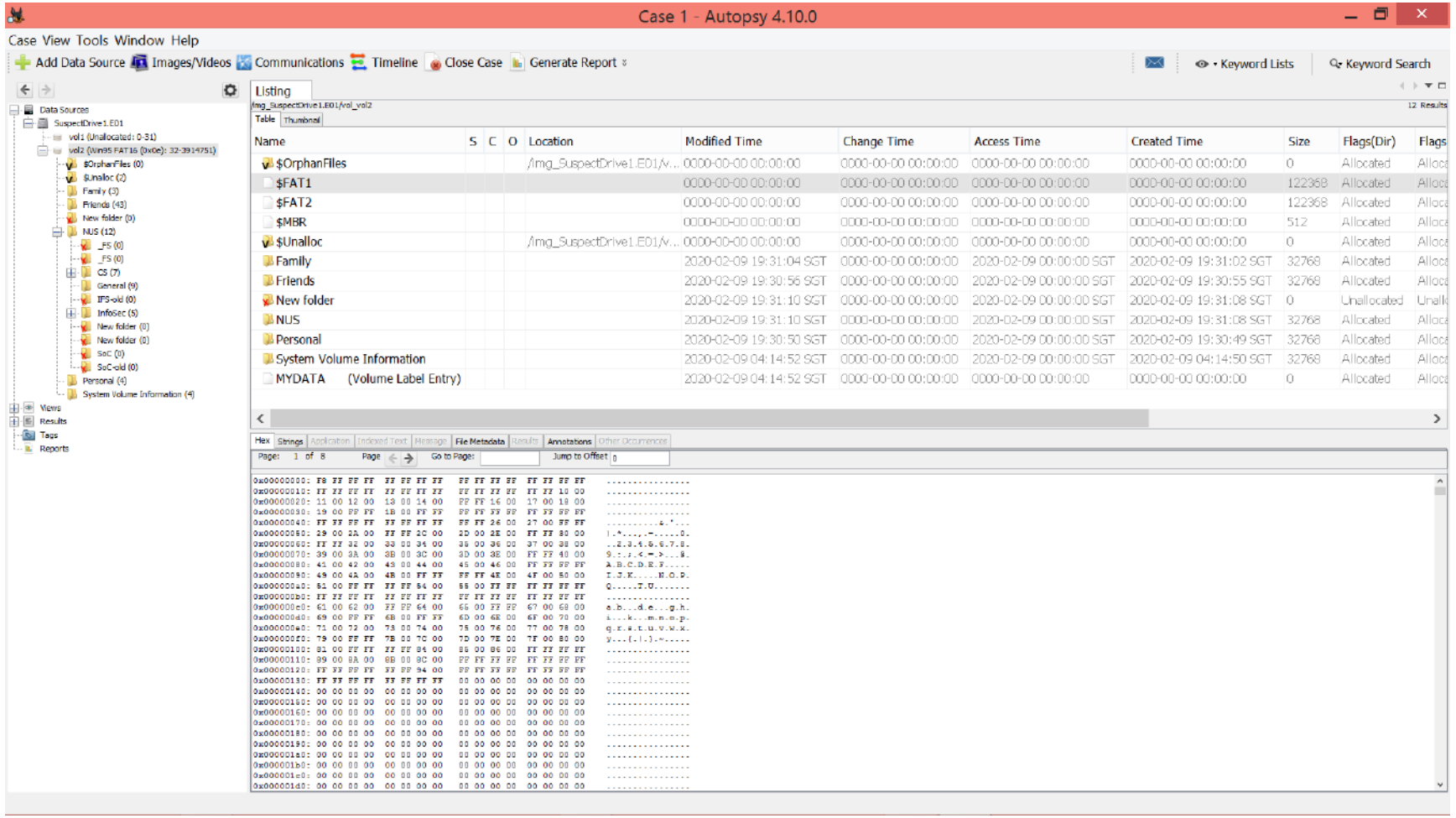
From: Brian Carrier, "File System Forensic Analysis"

Formatting of FAT File System

- Two types of format with different operations conducted
- A ***full*** format:
  - Writes the whole disk with **zero** or **0xf6**
- A ***quick*** format:
  - Zeroes out the **FAT area**
  - Zeroes out the **root directory's entries**
  - The FAT file system's **data area** is ***left untouched!***
    → forensics tools *can extract* valuable information!

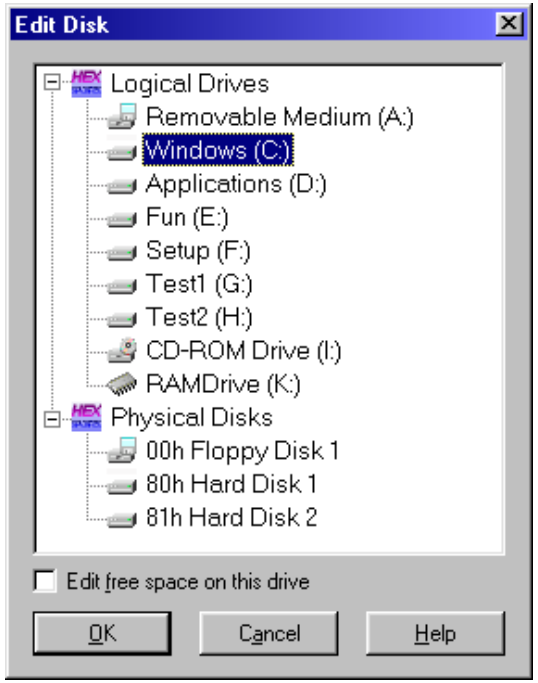
Tools for Inspecting FAT File System

- *Autopsy?*



Tools for Inspecting FAT File System

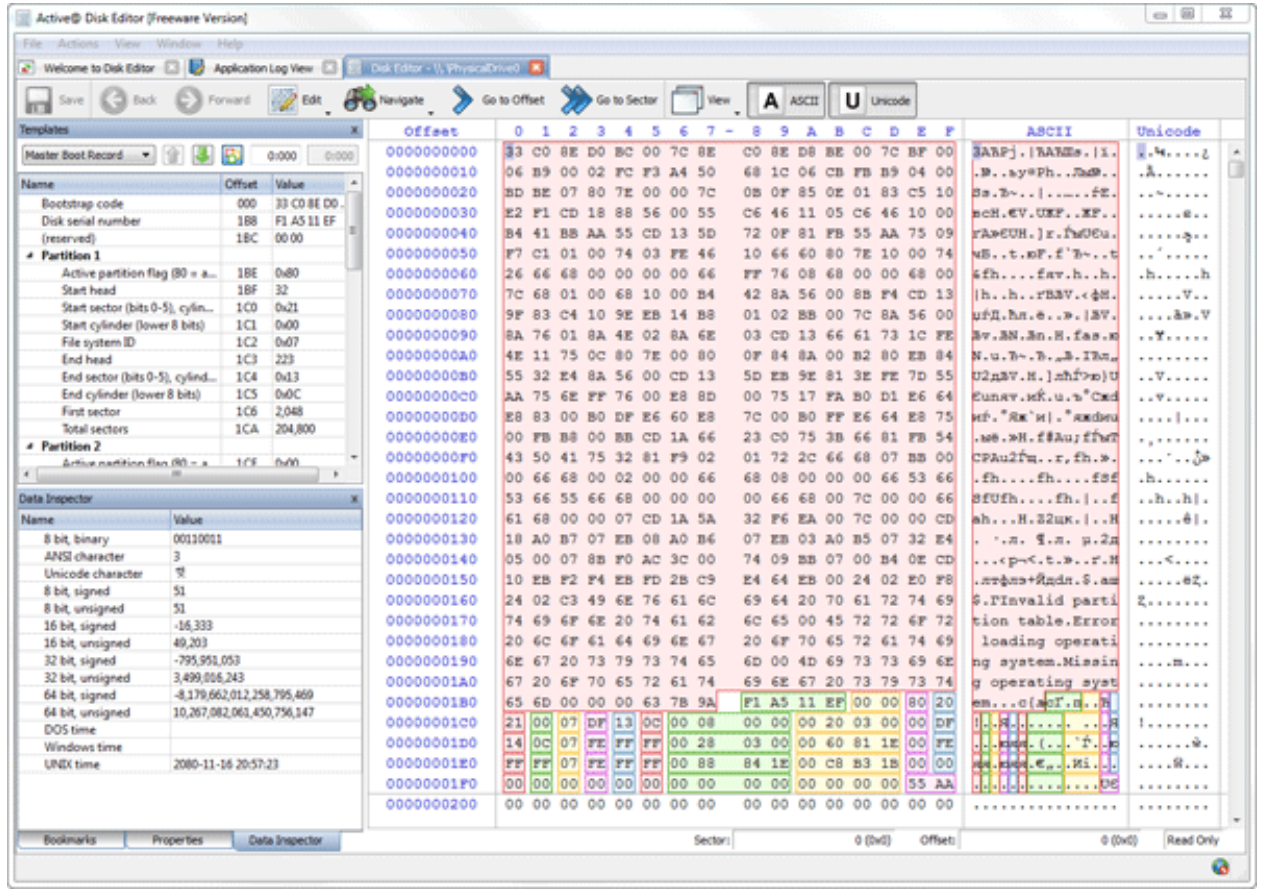
- *Win-Hex hex editor?*
  - **Disk editing** feature: http://www.winhex.com/winhex/index-m.html
  - Available only in certain license types, see: http://www.winhex.com/winhex/comparison.html

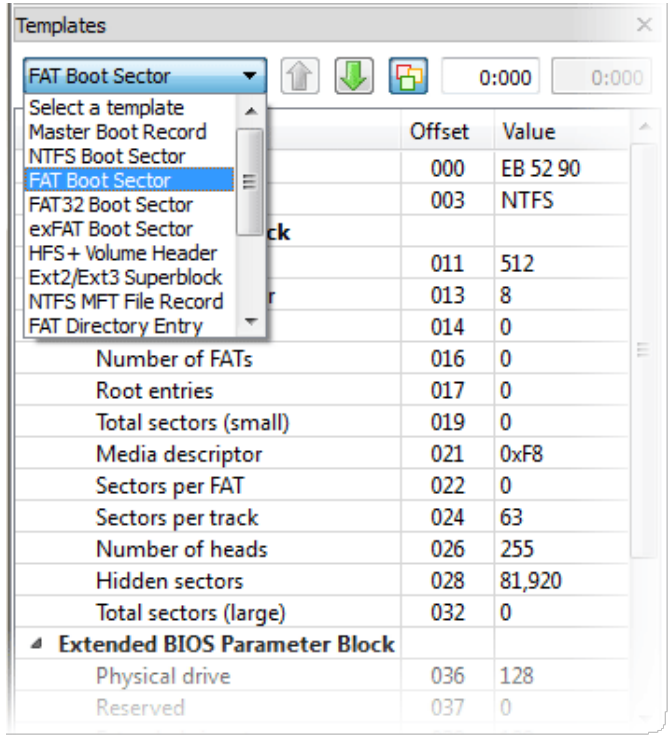


Source: http://www.winhex.com/winhex/index-m.html

Full-Featured Disk Editor

- One popular product: Active@ (https://www.disk-editor.org/)





Additional Notes: FAT Derivatives & References

- **FAT Derivatives**:
  - Turbo FAT: NetWare File System (NWFS) Novell
  - FATX: designed for Microsoft's Xbox video game console hard disk drives and memory cards
  - exFAT: intended for use on flash drives (e.g. SDXC and Memory Stick XC)
  - FAT+: for storing larger files
- **References**:
  - Brian Carrier, “File System Forensic Analysis”
  - https://en.wikipedia.org/wiki/File_Allocation_Table
  - https://en.wikipedia.org/wiki/Design_of_the_FAT_file_system

File System Comparison

	File System	Content	Metadata	File Name	Application
ExtX	Superblock, group descriptor	Blocks, block bitmap	Inodes, inode bitmap, extended attributes	Directory entries	Journal
FAT	Boot sector, FSINFO	Clusters, FAT	Directory entries, FAT	Directory entries	N/A
NTFS	$Boot, $Volume, $AttrDef	Clusters, $Bitmap	$MFT, $MFTMirr, $STANDARD_ INFORMATION, $DATA, $ATTRIBUTE_LIST, $SECURITY_DESCRIPTOR	$FILE_NAME, $IDX_ROOT, $IDX_ALLOCATION, $BITMAP	Disk Quota, Journal, Change Journal
UFS	Superblock, group descriptor	Blocks, fragments, block bitmap, fragment bitmap	Inodes, inode bitmap, extended attributes	Directory entries	N/A

From: Brian Carrier, “File System Forensic Analysis”

File System Analysis: NTFS & ADS

NTFS: General

- **NTFS**: New Technologies File System
- Emerged in 1993 with Windows NT3.1
- Now the most **commonly-used** file system on end-user computers
- File locations are stored in the ***Master File Table*** (***MFT***)
- Has a lot **more details** that FAT: the size of the MFT can grow as needed in order to accommodate more fie entries
- Our lecture only **covers *some parts*** of the file system only!

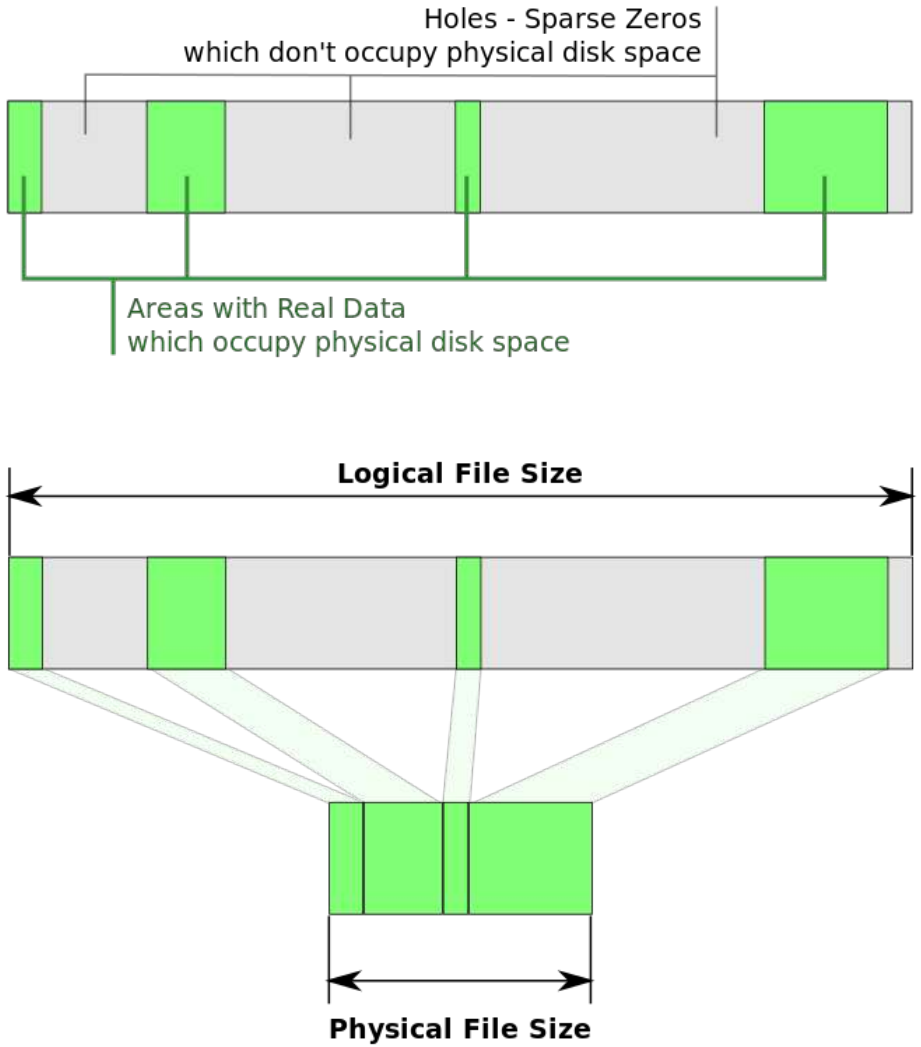
NTFS Design Goals

- **Reliability**: a journaling file system
  - The **NTFS Log ($LogFile)** to record metadata changes to the volume
  - In case of system crashes or data moves performed by the defragmentation API, the FS internal data structures will remain **consistent**
  - Allow for **easy rollback** of uncommitted changes when the volume is remounted
- **Security**: by access control information
  - Each file/folder is assigned a **security descriptor** that defines its owner
  - Contains 2 **ACLs**: discretionary access control list (DACL) & system access control list (SACL)

NTFS Design Goals, Features

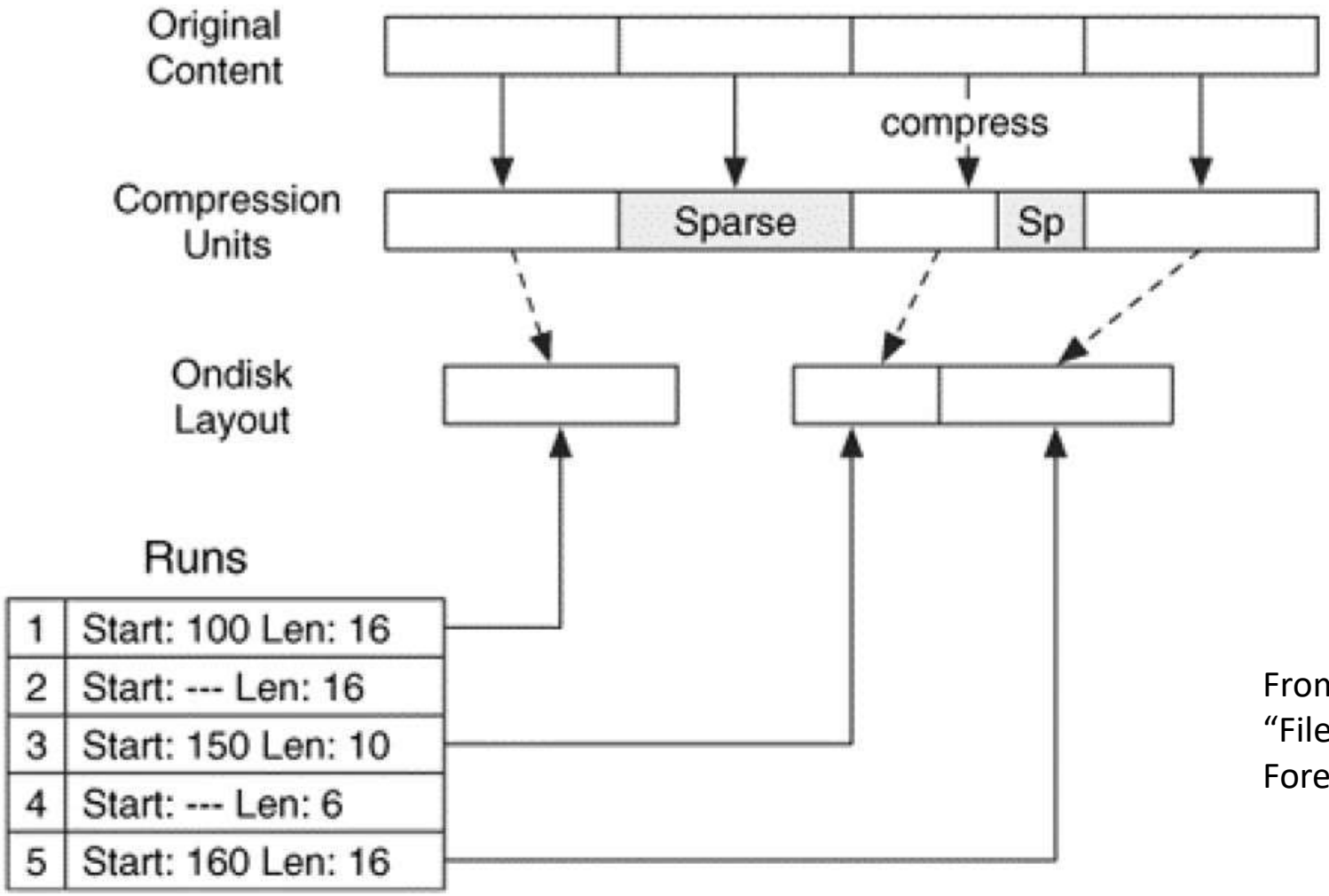
- **Scalability**:
  - Optimized for 4 KB clusters, but supports a max cluster size of 2MB
  - Maximum supported volume size: $2^{64}-1$ clusters
- **Alternate data streams** (ADS): *more later!*
- **File compression**: using LZNT1 algorithm (a variant of LZ77)
- Support for **sparse files**:
  - A ***sparse file***: a file interspersed with *empty segments* for which **no actual storage space** is used
  - May be used by database applications

Sparse File Illustration



From:
Wikipedia

Sparse File in NTFS & Cluster Runs



From: Brian Carrier, "File System Forensic Analysis"

NTFS Design Paradigm

- Design paradigm: ***"everything is a file"***
  - Each byte of **an NTFS file system** belongs to a file
  - **File system data & *file system metadata*** are also located in files
  - File system metadata files ("***metafiles***"): names start with dollar character $ (except for '.' which contains the root directory)
- For a list and of metafiles and respective purposes:
  - See https://en.wikipedia.org/wiki/NTFS#Metafiles

NTFS "File System" Metadata Files

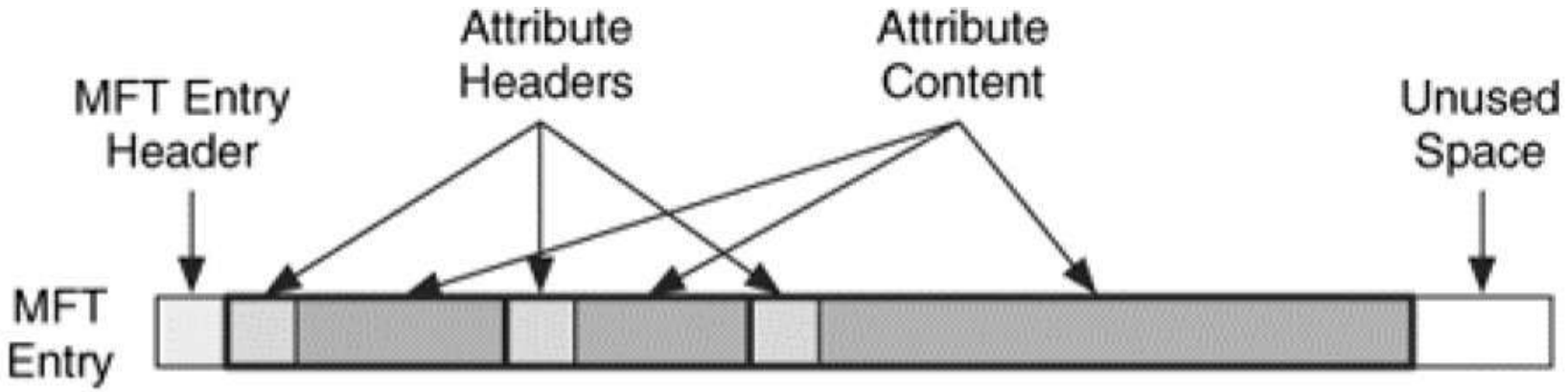
Entry	File Name	Description
0	$MFT	The entry for the MFT itself.
1	$MFTMirr	Contains a backup of the first entries in the MFT. See the "File System Category" section in Chapter 12.
2	$LogFile	Contains the journal that records the metadata transactions. See the "Application Category" section in Chapter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

From: Brian Carrier, "File System Forensic Analysis"

NTFS Master File Table (MFT)

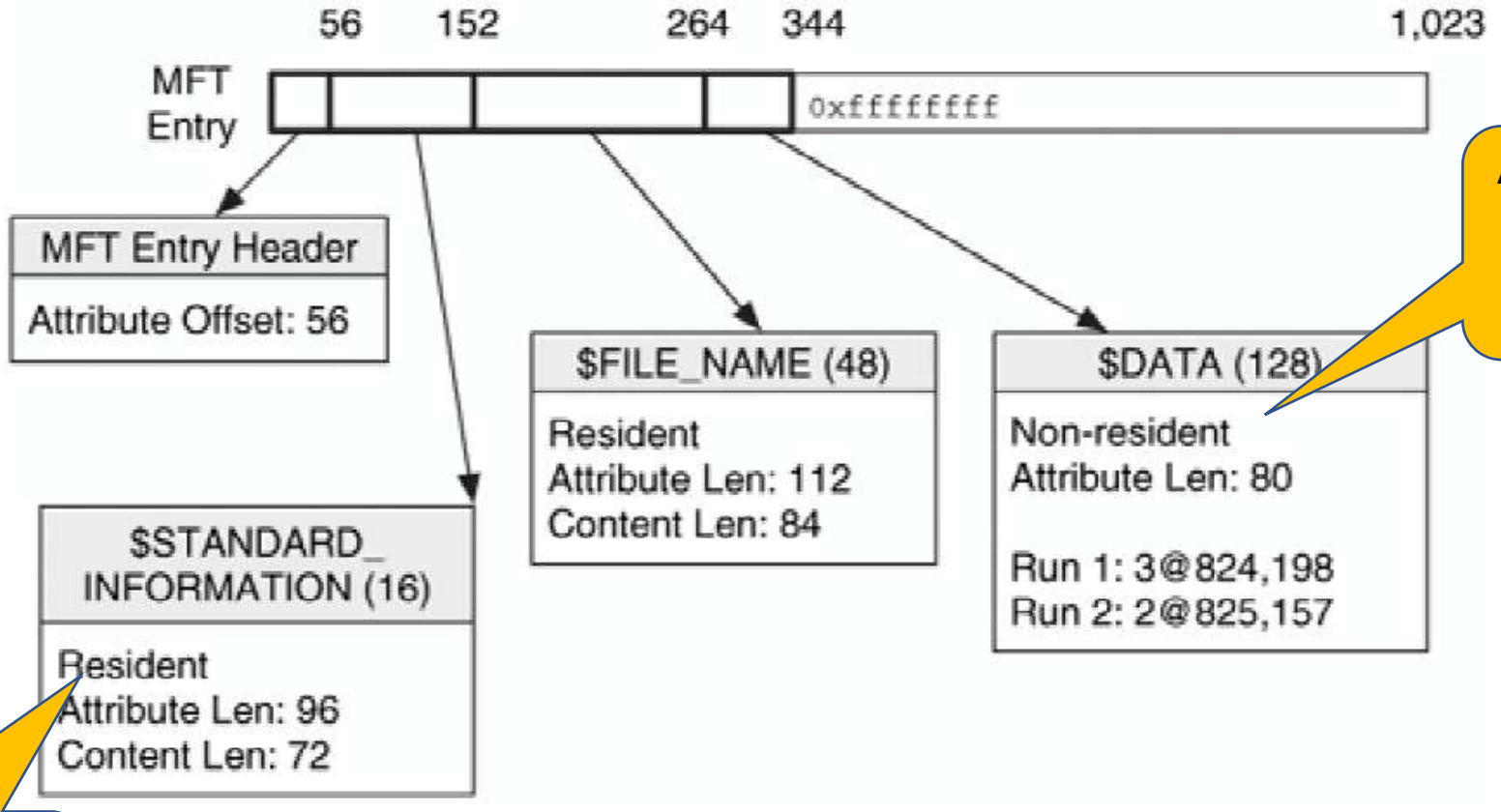
- Stores **all the information** about all files and folders
- A **relational database** that consists of **rows** of ***file records***, and **columns** of ***file attributes***:
  - ***MFT entry header*** (structured, 42 bytes): contains signature (either string 'FILE' or 'BAAD'), allocation status, the used bytes in the MFT record, ...
  - ***MFT entry body*** (unstructured, 982 bytes): contains **attributes** for file name, file metadata, content data
- Example: see next few slides
- NTFS references:
  - Brian Carrier, "File System Forensic Analysis"
  - https://en.wikipedia.org/wiki/NTFS

MFT Entry Example



From: Brian Carrier, “File System Forensic Analysis”

MFT Entry Example



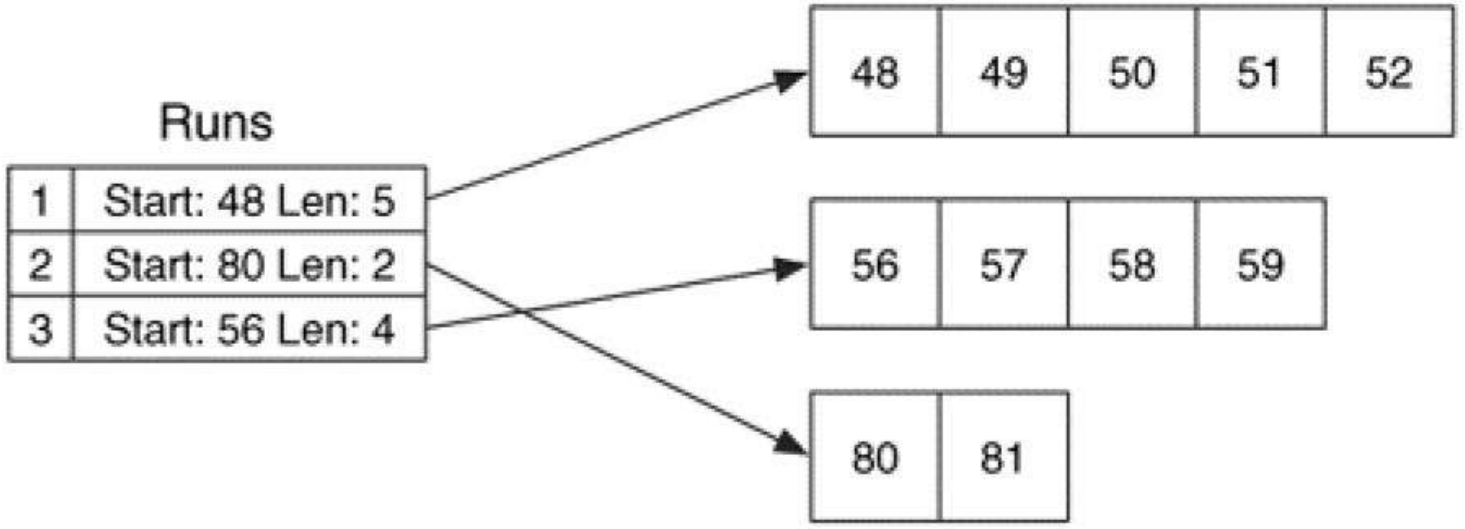
Attribute's content is stored **outside** of the MFT entry, i.e. external cluster(s) in the file system

Attribute's content is **fully stored inside** the MFT entry

From: Brian Carrier, "File System Forensic Analysis"

Data Cluster Allocation: Cluster Runs

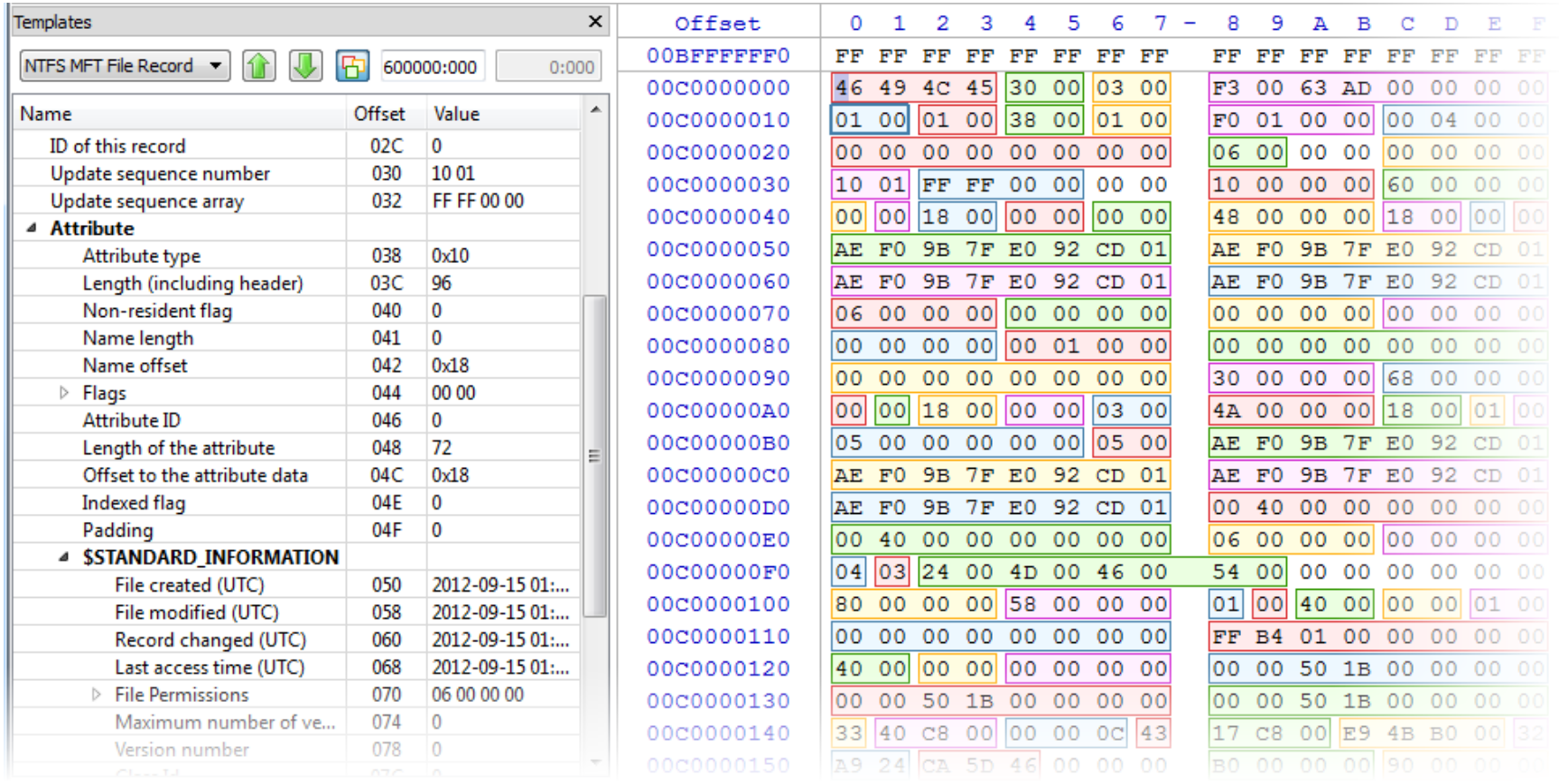
- *A **cluster run***:
  - A block of **consecutive data clusters**
  - Referred to by the ***starting cluster address*** and ***run length***
- **Cluster runs**:
  - Store the content of a **non-resident attribute**



From: Brian Carrier, "File System Forensic Analysis"

Inspecting NTFS File System

- Active@ disk editor (https://www.disk-editor.org/)



Source: https://www.disk-editor.org/index.html

MFTECmd and MFT Explorer

- Two tools by Eric Zimmerman (among many of his tools)
- Can be used to inspect $MFT and other metafiles of NTFS
- **MFTECmd** (https://github.com/EricZimmerman/MFTECmd):
  - A CLI-based tool MFT parser
  - See: https://binaryforay.blogspot.com/2018/06/introducing-mftecmd.html, https://aboutdfir.com/toolsandartifacts/windows/mft-explorer-mftecmd/

```
MFTECmd version 0.2.5.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/MFTECmd

        f               File to process. Either this or -d is required
        csv             Directory to save CSV formatted results to. Be sure to include the full path in double quotes. Required unless --de is specified
        de              Dump full details for the entry/sequence number provided. Format is 'Entry-Seq' as decimal or hex. Example: 624-5 or 0x270-0x5

        dt              The custom date/time format to use when displaying time stamps. Default is: yyyy-MM-dd HH:mm:ss.fffffff
        sn              Include DOS file name types. Default is false
        vl              Verbose log messages. 1 == Debug, 2 == Trace

Examples: MFTECmd.exe -f "C:\Temp\SomeMFT"
          MFTECmd.exe -f "C:\Temp\SomeMFT" --csv "c:\temp\out"
          MFTECmd.exe -f "C:\Temp\SomeMFT" --de 5-5

          Short options (single letter) are prefixed with a single dash. Long commands are prefixed with two dashes
```

Source: https://binaryforay.blogspot.com/2018/06/introducing-mftecmd.html

MFTECmd

```
PS C:\Tools> .\MFTECmd.exe -f 'D:\SynologyDrive\MFTs\nromanoff\$MFT' --csv C:\Temp\
MFTECmd version 0.2.5.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/MFTECmd

Command line: -f D:\SynologyDrive\MFTs\nromanoff\$MFT --csv C:\Temp\

Warning: Administrator privileges not found!


Processed 'D:\SynologyDrive\MFTs\nromanoff\$MFT' in 3.3311 seconds

CSV output will be saved to 'C:\Temp\20180620102304_MFTECmd_Output.csv'
```

```
    0·····10·····20·····30·····40·····50·····60·····70·····80·····90·····100····110····120····130····140····150····1
0   EntryNumber,SequenceNumber,InUse,ParentEntryNumber,ParentSequenceNumber,ParentPath,FileName,Extension,FileSize,ReferenceCount,ReparseTarget,IsDirectory,HasAds,Is
1   0,1,True,5,5,.,$MFT,,118751232,1,,False,False,False,False,False,False,"Hidden, System",DosWindows,2010-11-10 17:37:26.4843750,,2010-11-10 17:37:26.4843750,,2010-
2   1,1,True,5,5,.,$MFTMirr,,4096,1,,False,False,False,False,False,False,"Hidden, System",DosWindows,2010-11-10 17:37:26.4843750,,2010-11-10 17:37:26.4843750,,2010-1
3   2,2,True,5,5,.,$LogFile,,67108864,1,,False,False,False,False,False,False,"Hidden, System",DosWindows,2010-11-10 17:37:26.4843750,,2010-11-10 17:37:26.4843750,,20
4   3,3,True,5,5,.,$Volume,,0,1,,False,False,False,False,False,False,"Hidden, System",DosWindows,2010-11-10 17:37:26.4843750,,2010-11-10 17:37:26.4843750,,2010-11-10
5   4,4,True,5,5,.,$AttrDef,,2560,1,,False,False,False,False,False,False,"Hidden, System",DosWindows,2010-11-10 17:37:26.4843750,,2010-11-10 17:37:26.4843750,,2010-1
6   5,5,True,5,5,.,.,,0,1,,True,False,False,True,False,False,"Hidden, System",DosWindows,2009-07-14 02:07:54.8957117,2010-11-10 17:37:26.4843750,2012-04-04 11:47:38.
7   6,6,True,5,5,.,$Bitmap,,811104,1,,False,False,False,False,False,False,"Hidden, System",DosWindows,2010-11-10 17:37:26.4843750,,2010-11-10 17:37:26.4843750,,2010-
8   7,7,True,5,5,.,$Boot,,8192,1,,False,False,False,False,False,False,"Hidden, System",DosWindows,2010-11-10 17:37:26.4843750,,2010-11-10 17:37:26.4843750,,2010-11-1
9   8,8,True,5,5,.,$BadClus,,0,1,,False,True,False,False,False,False,"Hidden, System",DosWindows,2010-11-10 17:37:26.4843750,,2010-11-10 17:37:26.4843750,,2010-11-10
0   8,8,True,5,5,.,$BadClus:$Bad,,26578251776,1,,False,False,True,False,False,False,"Hidden, System",DosWindows,2010-11-10 17:37:26.4843750,,2010-11-10 17:37:26.4843
1   9,9,True,5,5,.,$Secure,,964632,1,,False,True,False,False,False,False,"Hidden, System, IsIndexView",DosWindows,2010-11-10 17:37:26.4843750,,2010-11-10 17:37:26.48
2   9,9,True,5,5,.,$Secure:$SDS,,964632,1,,False,False,True,False,False,False,"Hidden, System, IsIndexView",DosWindows,2010-11-10 17:37:26.4843750,,2010-11-10 17:37:
3   10,10,True,5,5,.,$UpCase,,131072,1,,False,False,False,False,False,False,"Hidden, System",DosWindows,2010-11-10 17:37:26.4843750,,2010-11-10 17:37:26.4843750,,201
4   11,11,True,5,5,.,$Extend,,0,1,,True,False,False,False,False,False,"Hidden, System",DosWindows,2010-11-10 17:37:26.4843750,,2010-11-10 17:37:26.4843750,,2010-11-1
5   24,1,True,11,11,.\$Extend,$Quota,,0,1,,False,False,False,False,False,False,"Hidden, System, Archive, IsIndexView",Posix,2010-11-10 17:37:58.0937500,,2010-11-10 1
6   25,1,True,11,11,.\$Extend,$ObjId,,0,1,,False,False,False,False,False,False,"Hidden, System, Archive, IsIndexView",Posix,2010-11-10 17:37:58.0937500,,2010-11-10 1
7   26,1,True,11,11,.\$Extend,$Reparse,,0,1,,False,False,False,False,False,False,"Hidden, System, Archive, IsIndexView",Posix,2010-11-10 17:37:58.0937500,,2010-11-10
8   27,1,True,11,11,.\$Extend,$RmMetadata,,0,1,,True,False,False,False,False,False,"Hidden, System",Posix,2010-11-10 17:37:58.0937500,,2010-11-10 17:37:58.0937500,,2
9   28,1,True,27,1,.\$Extend\$RmMetadata,$Repair,,0,1,,False,True,False,False,False,False,"Hidden, System, Archive",Posix,2010-11-10 17:37:58.0937500,,2010-11-10 17:
0   28,1,True,27,1,.\$Extend\$RmMetadata,$Repair:$Config,8,1,,False,False,True,False,False,False,"Hidden, System, Archive",Posix,2010-11-10 17:37:58.0937500,,2010-1
```

Source: https://binaryforay.blogspot.com/2018/06/introducing-mftecmd.html

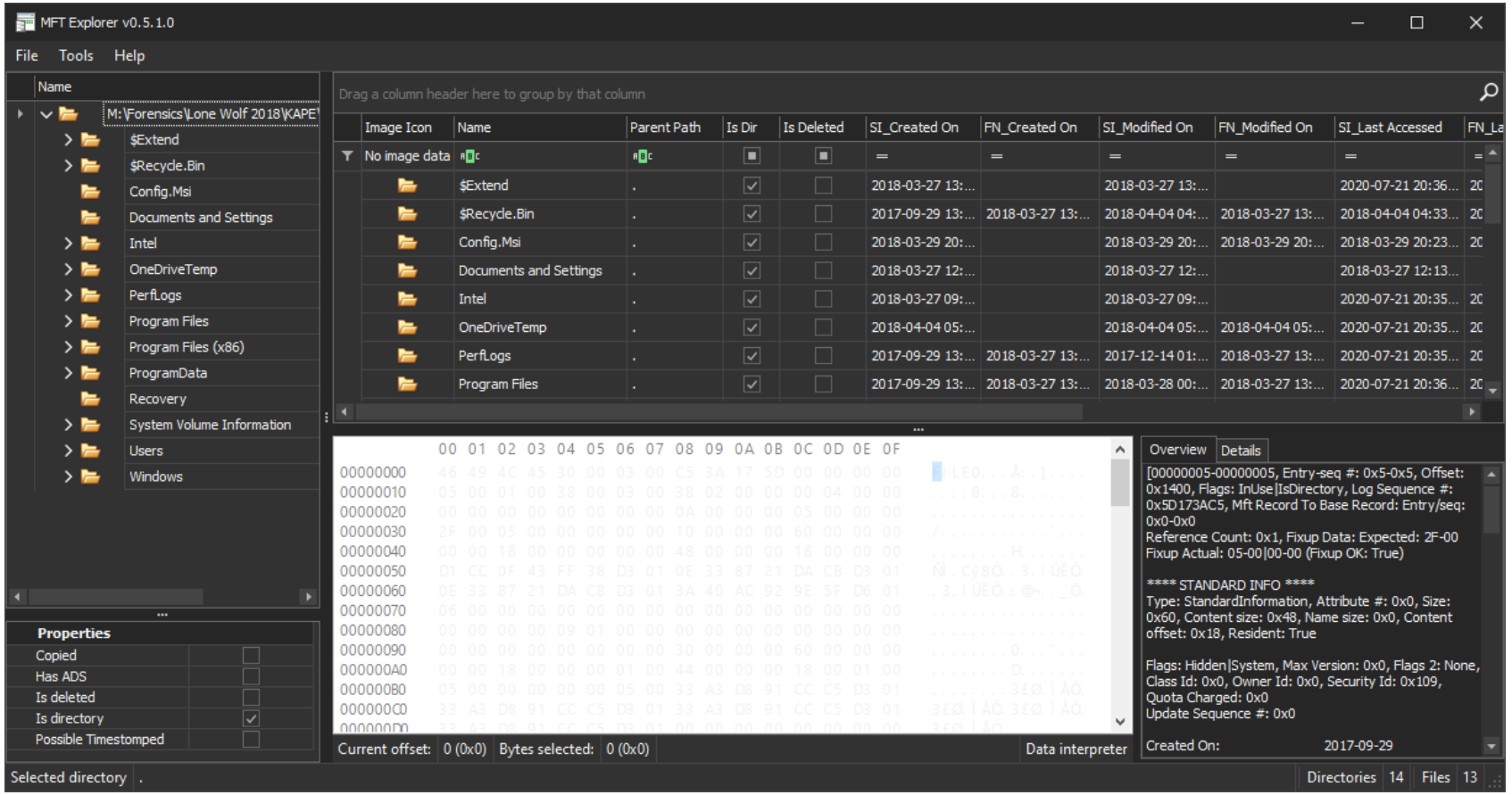
MFTECmd

```
PS C:\Tools> .\MFTECmd.exe -f 'D:\SynologyDrive\MFTs\nromanoff\$MFT' --de 16519-4
MFTECmd version 0.2.5.0

Author: Eric Zimmerman (saericzimmerman@gmail.com)
https://github.com/EricZimmerman/MFTECmd

Command line: -f D:\SynologyDrive\MFTs\nromanoff\$MFT --de 16519-4

Warning: Administrator privileges not found!


Processed 'D:\SynologyDrive\MFTs\nromanoff\$MFT' in 3.3231 seconds

Dumping details for file record with key '00004087-00000004'

Entry/seq #: 0x4087/0x4 Offset: 0x1021C00 Flags: InUse LogSequenceNumber: 0x2031B879C MftRecordToBaseRecord: Entry: 0x0, Seq: 0x0
ReferenceCount: 0x1 FixupData: Expected: 05-00 FixupActual: 00-00|00-00 (Fixup OK: True)

**** STANDARDINFO ****
Type: StandardInformation, Attr #: 0x0 Size: 0x60, Content size: 0x48, Name size: 0x0, Content offset: 0x18, Resident: True

Flags: Hidden, System, Archive MaxVersion: 0x0 VersionNumber: 0x0, ClassId: 0x0 OwnerId: 0x0 SecurityId: 0x6B4, QuotaCharged: 0x0
UpdateSequenceNumber: 0x7267A550
CreatedOn: 2012-03-15 20:48:24.0000000
ContentModifiedOn: 2012-03-16 20:25:36.0388590
RecordModifiedOn: 2012-04-04 15:21:06.9097813
LastAccessedOn: 2012-03-15 20:48:24.0000000

**** FILENAME ****
Type: FileName, Attr #: 0x2 Size: 0x70, Content size: 0x54, Name size: 0x0, Content offset: 0x18, Resident: True


File name: Thumbs.db (Len:0x9)
Flags: Archive, NameType: DosWindows ReparseValue: 0 PhysicalSize: 0x0, LogicalSize: 0x0
ParentMftRecord: Entry: 0x4028, Seq: 0x6
CreatedOn: 2012-03-16 20:25:36.0115170
ContentModifiedOn: 2012-03-16 20:25:36.0115170
RecordModifiedOn: 2012-03-16 20:25:36.0115170
LastAccessedOn: 2012-03-16 20:25:36.0115170


**** DATA ****
Type: Data, Attr #: 0x3 Size: 0x48, Content size: 0x0, Name size: 0x0, Content offset: 0x0, Resident: False

NonResidentData

StartingVirtualClusterNumber: 0x0 EndingVirtualClusterNumber: 0xC AllocatedSize: 0xD000 ActualSize: 0xC600 InitializedSize: 0xC600

DataRuns Entries
Cluster offset: 0x209108, # clusters: 0xD


**** DATA ****
Type: Data, Attr #: 0x4 Size: 0x58, Content size: 0x1A, Name size: 0xF, Name: Zone.Identifier, Content offset: 0x38, Resident: True

ResidentData
Data: 5B-5A-6F-6E-65-54-72-61-6E-73-66-65-72-5D-0D-0A-5A-6F-6E-65-49-64-3D-33-0D-0A
```

Source: https://binaryforay.blogspot.com/2018/06/introducing-mftecmd.html

MFT Explorer

- MFT Explorer: a GUI-based tool for visualizing the $MFT



From: https://aboutdfir.com/toolsandartifacts/windows/mft-explorer-mftecmd/2/

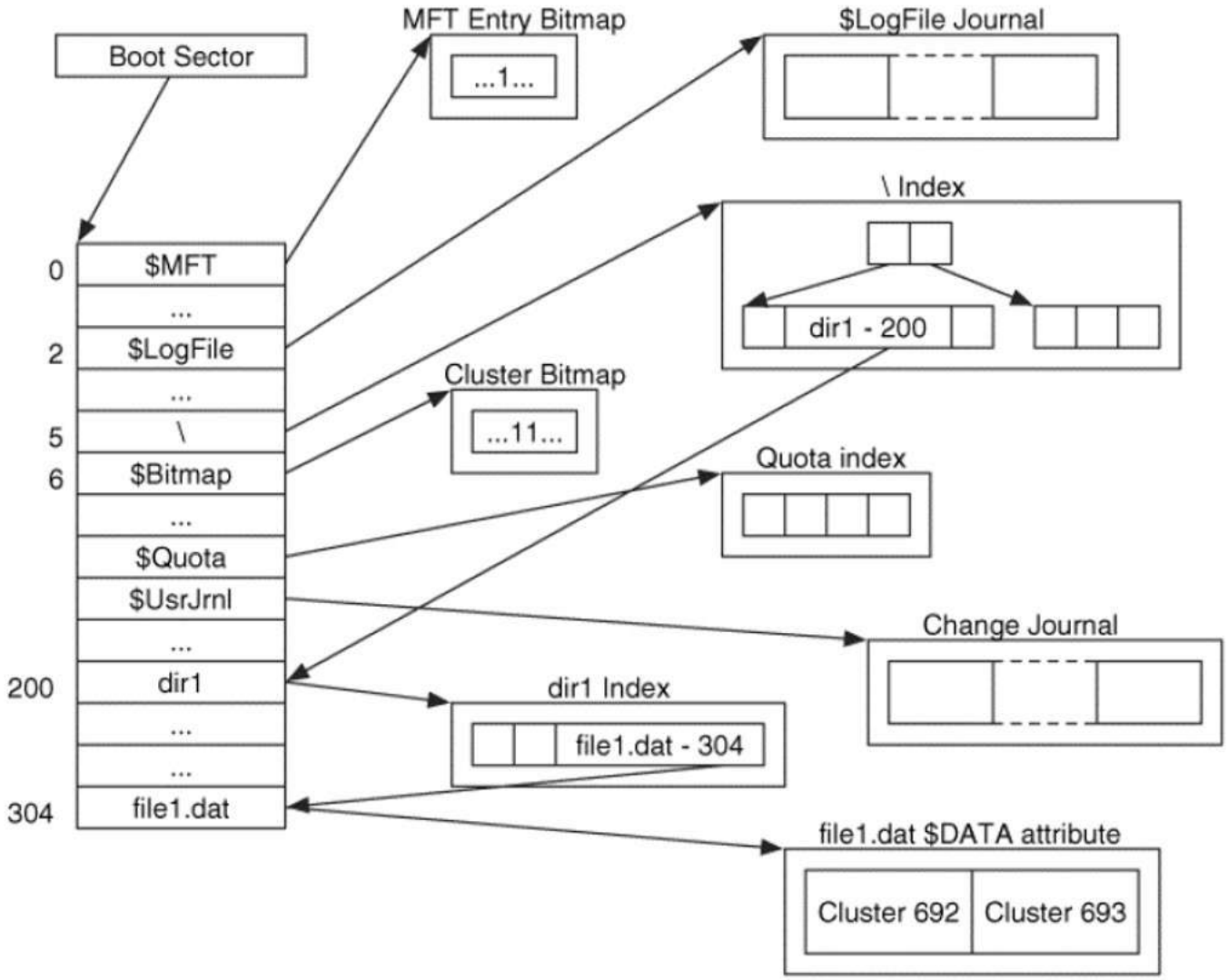
Optional

NTFS Journaling

- **Journaling** in NTFS:
  - NTFS is a journaling file system
  - File changes are logged
  - Old files retain their validity until writing is marked as successful in the log file
  - Following a system failure, restore operations are automatically executed
- **Journal files**:
  - **$LogFile**: NTFS journal file, which records metadata transactions
  - **\$Extend\$UsrJrnl**: change journal, which records changes to files

Optional

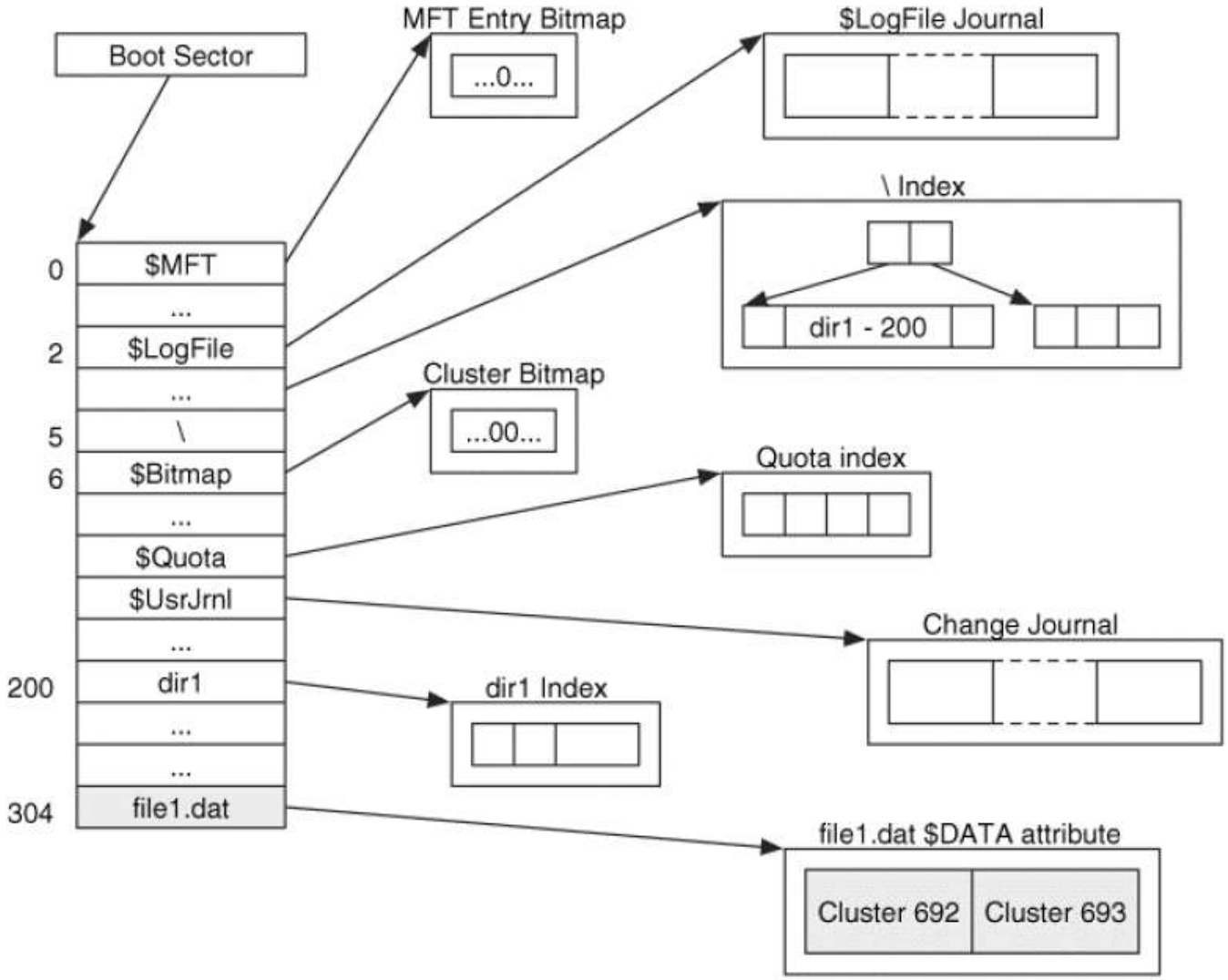
NTFS File System: Sample Layout & File



From: Brian Carrier, "File System Forensic Analysis"

Optional

NTFS File System: Sample Layout & Deletion



From: Brian Carrier, "File System Forensic Analysis"

Break!

NTFS Alternate Data Streams (ADS)

- Introduced into NTFS since Windows NT 3.1
- Allow for **more than one *data streams*** to be associated with a file
- Each extra stream is stored as an additional **$DATA attribute**
- A file on NTFS may therefore have **multiple streams**:
  - File system by default only shows **1 stream**: the unnamed $DATA attribute
  - All other streams are ***not* visible**, and ***not* indicated** by the Explorer, and ***not* considered** in the file's size
  - When copying a file from an NTFS to a FAT partition, ***only* the main stream** will be copied

ADS: Example and Detection

- **Create** two ADS streams on `file.txt`:
  - `C:> type hidden.txt > file.txt:hidden.txt`
  - `C:> type starwars.jpg > file.txt:starwars.jpg`
- **Display** the ADS streams:
  - `C:> notepad file.txt:hidden.txt`
  - `C:> mspaint file.txt:starwars.jpg`
- ADS stream **detection**:
  - Windows 7 and later: `dir /R` command
  - TSK: `fls` command
  - Various ADS tracking/scanning tools

ADS: Security & Forensics Implications

- **Executable files** embedded in ADS:
  exactly like normal executable files
- They can be executed with a start command
- ADS is thus important for **malware analysis**:
  - Many root kits, viruses, hacking tools can hide executable files in the extra streams
- ADS can also be used to **hide evidence/data files**:
  - Relevant to forensic investigation!
- See Task 1 of Lab 6 (next week)

File System Comparison

	File System	Content	Metadata	File Name	Application
ExtX	Superblock, group descriptor	Blocks, block bitmap	Inodes, inode bitmap, extended attributes	Directory entries	Journal
FAT	Boot sector, FSINFO	Clusters, FAT	Directory entries, FAT	Directory entries	N/A
NTFS	$Boot, $Volume, $AttrDef	Clusters, $Bitmap	$MFT, $MFTMirr, $STANDARD_ INFORMATION, $DATA, $ATTRIBUTE_LIST, $SECURITY_DESCRIPTOR	$FILE_NAME, $IDX_ROOT, $IDX_ALLOCATION, $BITMAP	Disk Quota, Journal, Change Journal
UFS	Superblock, group descriptor	Blocks, fragments, block bitmap, fragment bitmap	Inodes, inode bitmap, extended attributes	Directory entries	N/A

From: Brian Carrier, “File System Forensic Analysis”

MAC Times in Windows

MAC Times: Notes on Possible Issues (NIST)

- Among the reasons for ***time inaccuracies*** are the following:
  - The computer's clock does not have **the correct time**
  - The time may not be recorded with the **expected level of detail**, such omitting the seconds or minutes
  - An attacker ***may have altered*** the recorded file times: general issue of metadata reliability!

MAC Times: Overview

- **MAC** (common or standard definition):
  - **M**: *modified/modification* time
  - **A**: *accessed/access* time
  - **C**: *created/creation* time
- **C**: ***created*** or ***changed*** time?
  - **Windows** (FAT/NTFS): **created** time
  - **Linux/UNIX**: **changed** time, i.e. the last time a file metadata was changed

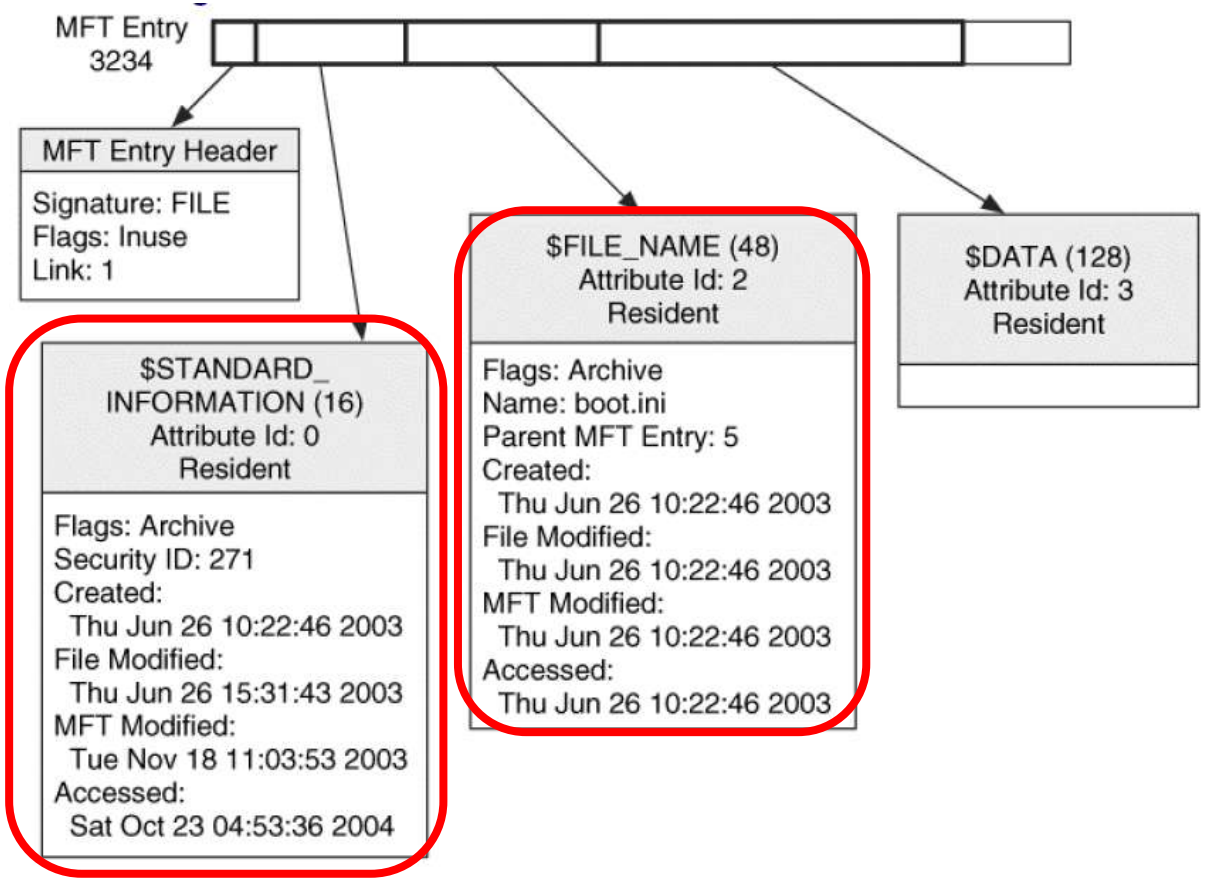
File System	M	A	C	B
Ext2/3	Modified	Accessed	Inode Changed	
FAT	Modified	Accessed Date		Created
NTFS	Modified	Accessed	MFT Modified	Created

From: https://digital-forensics.sans.org/blog/2009/02/24/digital-forensic-sifting-registry-and-filesystem-timeline-creation/

- **MACB** times:
  Modification time, **Access** time, **Change** time, **Birth** (creation) time

NTFS Timestamps

- **Time stamps** in NTFS (stored in the **$STANDARD_INFORMATION** attribute):
  - **Created time**: file was created
  - **File Modified time**: last write access (on the content of the $DATA or $INDEX attributes)
  - **MFT Modified time**: last update of MFT record, which stores the file's metadata
  - **Accessed time**: last read access on the file's content



From: Brian Carrier, "File System Forensic Analysis"

Date and Time in FAT & NTFS (Microsoft)

- **Issue**: MAC times and **file operations**, including ***between media***
- Some **scenarios** involving a file named **`D:\NTFS\file.txt`**:
  - **Copying** the file to `D:\NTFS\SUBDIR` or `E:\`
  - **Moving** the file to `D:\NTFS\SUBDIR` or `E:\`
  - **Renaming** the file into `newfile.txt`
- **Important questions**:
  - What should be the ***created time (ctime)*** & ***modified time (mtime)*** of the **new file**?
  - Should the **ctime** becomes the **time of** copy/move/rename **operation**?
  - How about the **mtime** of the new file?

Date and Time in FAT & NTFS (Microsoft)

	File Creation	File Access	File Modification	File Rename	File Copy	(Local) File Move
***M*odified time (mtime)**	Time of file creation	Unchanged	**Time of modification**	?	?	?
***A*ccessed time (atime)**	Time of file creation	**Time of access***	Unchanged	?	?	?
Metadata changed time	Time of file creation	Unchanged	**Time of modification**	?	?	?
***C*reated time (ctime)**	Time of file creation	Unchanged	Unchanged	?	?	?

For the **existing** file

For the new **directory entry** of the "**new**" file

Note:

* Unchanged/disabled in NTFS Win 7+

See: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc959914(v=technet.10)

Date and Time in FAT & NTFS (Microsoft)

- **Microsoft Knowledge Base** article 299648:
  - Describes how **file times** are affected by **copy** or **move** operations from one **media** to **another**

- The **created time (ctime)**:
  - Is **set** when Windows allocates a ***new directory entry*** for a ***new file***
  - A **new file** means **a new chain of data clusters** is being allocated
  - If a new directory entry is allocated for an **already-existing** file, even if the original location was on a different disk, then the **original** ctime is kept
  - Different effects of **file operations** that create **a new directory entry** on the file's ctime: see cases below

Date and Time in FAT & NTFS (Microsoft)

- **ctime** and **file operations**:
  - **Rename** or **move**: **ctime = *original*** ctime (with an **exception** below)
    - The ctime of the **new directory entry** is set to the **original** entry
    - Reason: **no** new file (i.e. chain of data clusters) is allocated
    - An **exception**: If the move is done from the **command line** of a 2000/XP system to a ***different volume***, the ctime is the ***time of the move***
  - **Copy**: **ctime = *new*** ctime
    - Reason: a **new file** (i.e. chain of data clusters) is being created
    - The ctime of the **new directory entry**:
      the time when the **(first) directory entry** was allocated

Date and Time in FAT & NTFS (Microsoft)

	File Creation	File Access	File Modification	File Rename	File Copy	(Local) File Move
***M*odified time (mtime)**	Time of file creation	Unchanged	Time of modification	?	?	?
***A*ccessed time (atime)**	Time of file creation	Time of access*	Unchanged	?	?	?
Metadata changed time	Time of file creation	Unchanged	Time of modification	?	?	?
***C*reated time (ctime)**	Time of file creation	Unchanged	Unchanged	**Unchanged**	**Time of file copy**	**Unchanged**

For the **existing** file

For the new **directory entry** of the "**new**" file

Date and Time in FAT & NTFS (Microsoft)

- The **Modified/written time (mtime)**:
  - Is set when Windows writes **new *file content***
  - Is ***content-based***, and not directory entry-based
- **mtime** and **file operations**:
  - **Move** or **copy**: **mtime = *original*** mtime
    - The new directory entry has the mtime from the **original** file
    - *"mtime follows data as the file is **copied/moved** around"*
  - **Changing a file's attributes or name**: **no** changes
    - Does **not** result in an update to mtime
  - When an application **writes content** to the file: ***new*** mtime
    - The mtime gets **updated**

Date and Time in FAT & NTFS (Microsoft)

	File Creation	File Access	File Modification	File Rename	File Copy	(Local) File Move
***M*odified time (mtime)**	Time of file creation	Unchanged	Time of modification	**Unchanged**	**Unchanged**	**Unchanged**
***A*ccessed time (atime)**	Time of file creation	Time of access*	Unchanged	?	?	?
Metadata changed time	Time of file creation	Unchanged	Time of modification	?	?	?
***C*reated time (ctime)**	Time of file creation	Unchanged	Unchanged	**Unchanged**	**Time of file copy**	**Unchanged**

For the **existing** file

For the new **directory entry** of the "**new**" file

Note: Unchanged can also mean the **original** time

Summary: MAC Time & File Copy/Move

- The following statements **summarize** how a file copy & move affect the file's modified/written and created times
- If you **copy** a file from D:\NTFS to D:\NTFS\SUB:
  - The modified date and time are kept **the same**
  - The created date and time are **changed** to the current date and time
- If you **move** a file from D:\NTFS to D:\NTFS\SUB:
  - The modified date and time are kept **the same**
  - The created date and time are kept **the same**, **unless** if the move is to a different volume & the command line is used
- If an application **writes content** to a file:
  - The modified time gets **updated**

Date and Time in FAT & NTFS (Microsoft)

	File Creation	File Access	File Modification	File Rename	File Copy	(Local) File Move
***M*odified time (mtime)**	Time of file creation	Unchanged	Time of modification	**Unchanged**	**Unchanged**	**Unchanged**
***A*ccessed time (atime)**	Time of file creation	Time of access*	Unchanged	**Unchanged**	**Time of file copy**	**Unchanged**
Metadata changed time	Time of file creation	Unchanged	Time of modification	?	?	?
***C*reated time (ctime)**	Time of file creation	Unchanged	Unchanged	**Unchanged**	**Time of file copy**	**Unchanged**

For the **existing** file

For the new **directory entry** of the "**new**" file

Date and Time in FAT & NTFS (Microsoft)

	File Creation	File Access	File Modification	File Rename	File Copy	(Local) File Move
***M*odified time (mtime)**	Time of file creation	Unchanged	Time of modification	**Unchanged**	**Unchanged**	**Unchanged**
***A*ccessed time (atime)**	Time of file creation	Time of access*	Unchanged	**Unchanged**	**Time of file copy**	**Unchanged**
Metadata changed time	Time of file creation	Unchanged	Time of modification	**Time of rename**	**Time of file copy**	**Time of (local) file move**
***C*reated time (ctime)**	Time of file creation	Unchanged	Unchanged	**Unchanged**	**Time of file copy**	**Unchanged**

For the **existing** file

For the new **directory entry** of the "**new**" file

Date and Time in FAT & NTFS (Microsoft)

- For volume file move (using CLI & Explorer), file deletion:

	File Rename	File Copy	(Local) File Move	Volume File Move (CLI)	Volume File Move (Explorer)	File Deletion
***M*odified time (mtime)**	**Unchanged**	**Unchanged**	**Unchanged**	**Unchanged**	**Unchanged**	**Unchanged**
***A*ccessed time (atime)**	**Unchanged**	**Time of file copy**	**Unchanged**	**Time of file move**	**Time of file move**	**Unchanged**
Metadata changed time	**Time of rename**	**Time of file copy**	**Time of (local) file move**	**Unchanged**	**Unchanged**	**Unchanged**
***C*reated time (ctime)**	**Unchanged**	**Time of file copy**	**Unchanged**	**Time of file move**	**Unchanged**	**Unchanged**

For the new **directory entry** of the "**new**" file

MACB Times: Visual Summary

From: https://www.sans.org/security-resources/posters/windows-forensic-analysis/170/download

MACB Times Differences in NTFS

- MAC time **differences** within $STANDARD_INFORMATION and $FILE_NAME?
- The **$STANDARD_INFORMATION** attribute:
  - Used by **Windows API**
  - Most frequently **changed** as a result of file activity
  - The timestamp **collected** by Windows Explorer, fls, mactime, timestomp, find, …
  - Can also be modified by user-level processes like timestomp
- The **$FILE_NAME** attribute:
  - Most often correspond to the **file creation time**
  - ***Rarely updated!***
  - Can only be modified by the system kernel
- *Question: What if the ctimes in both attributes are **different**?*
  There could be a possible **anti-forensic** operation done, e.g. using timestomp
- See: https://andreafortuna.org/2017/10/06/macb-times-in-windows-forensic-analysis/

Live/Online Windows Analysis

Live/Online Windows Analysis

- **Live analysis** on **live accessible** Windows machine
- ***Not*** generally advisable for forensic purposes:
  it will **change** the evidence PC's state
- Done in **incident response**, or included if a forensic analysis is done as a part of **incident response**:
  to detect, contain, eradicate, recover, post-analyse an incident
  (See: "Computer Security Incident Handling Guide", NIST 2012)

What Information to Collect?

- **Goal**: To acquire useful information from ***volatile evidence***, such as:
  - System information
  - Running processes
  - Memory usage
  - Logged in users
  - Network interface configuration (is it the promiscuous mode?)
  - Routing information
  - Network connections
  - Open files
  - …

Live/Online Windows Analysis: Some Tools

- Various **Windows shell** (cmd.exe and PowerShell) **commands**:
  - Windows commands: https://en.wikipedia.org/wiki/Category:Windows_commands
  - PowerShell commands: https://en.wikipedia.org/wiki/PowerShell
- Various other **useful tools**:
  - **Process Explorer**: https://en.wikipedia.org/wiki/Process_Explorer
  - **Autoruns**: https://docs.microsoft.com/en-us/sysinternals/downloads/autoruns
  - Many **others**: https://docs.microsoft.com/en-us/sysinternals/
  - …
- Some will be covered in the **incident response** segment *later*

Windows Registry Analysis

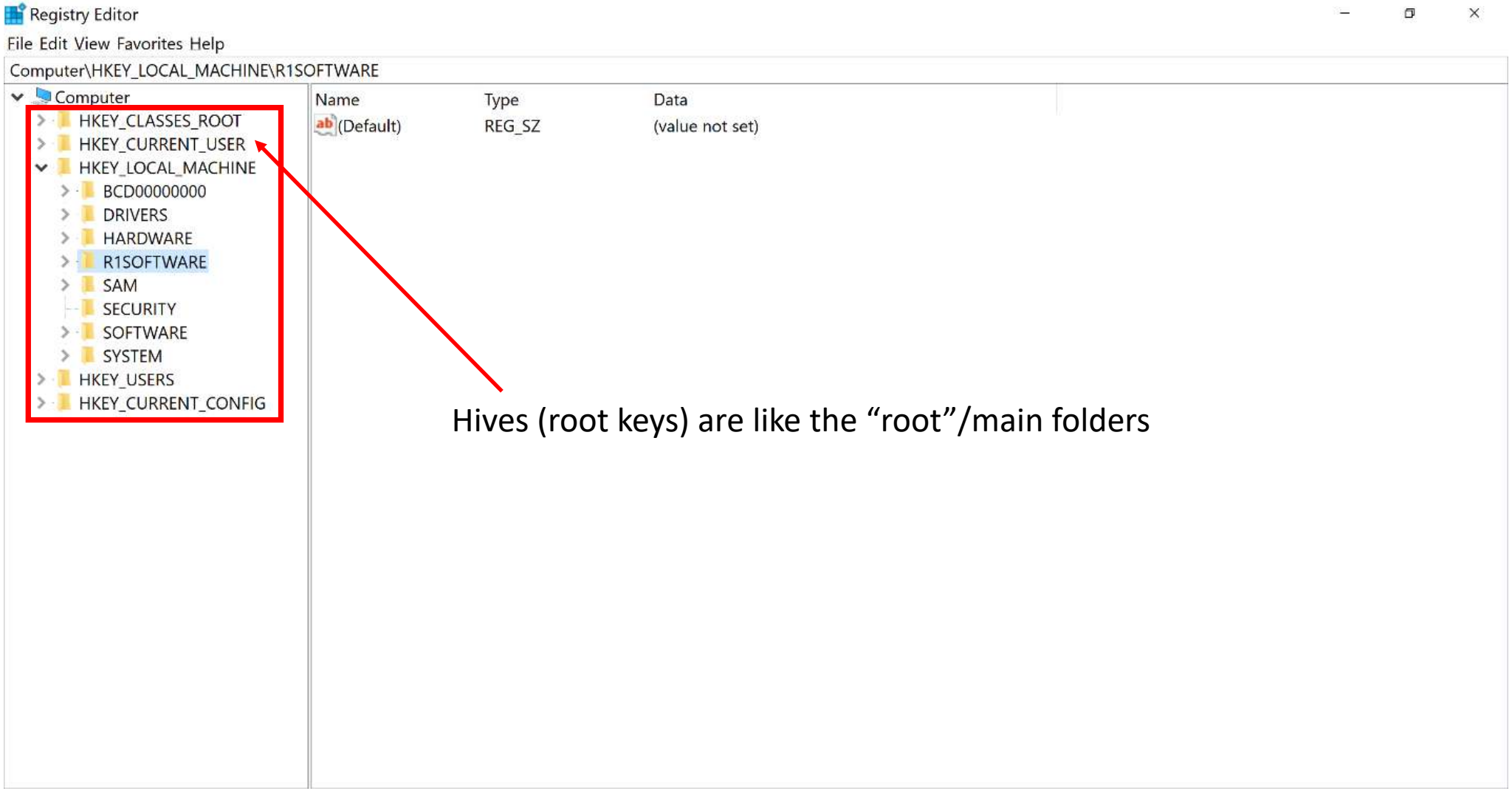
Windows Registry

- What is **Windows registry**?
  - A **repository** in a Windows system
  - Central place for storing **most *configuration settings*** of OS, applications, users, and devices
  - Introduced in Windows 95 and NT
  - Replaces most (not all) .INI files
  - A **tree-structured hierarchical database**, stored in at least **5 hive files**
  - A ***wealth of important data*** pertaining to system and users: very useful to forensic investigations, as well as malware analysis

Windows Registry Hives and Contents

- **Five main *hives* (*root keys*)** and their contents:
  - **HKEY_LOCAL_MACHINE** (HLM):
    settings that are specific to the **local computer** (consists of SAM, Security, System, and Software, …)
  - **HKEY_CURRENT_CONFIG** (HCC): **current system configuration**, which is regenerated at boot time and gathered at runtime
  - **HKEY_CLASSES_ROOT** (HCR):
    registered **application**, including program shortcuts, UI information
  - **HKEY_USERS** (HKU): profiles of **all users**, including preferences
  - **HKEY_CURRENT_USER** (HCU): information about the **currently logged-on user**
- Refs: https://en.wikipedia.org/wiki/Windows_Registry, https://technet.microsoft.com/en-us/library/cc959046.aspx

Windows Registry Hives and Contents



Hives (root keys) are like the “root”/main folders

Where Are the Registry Files Stored?

- **Binary format** files on the file system, which can be exported, loaded & unloaded by the **Registry Editor**
- **System-wide files** in %SystemRoot%`\System32\Config\` (for Windows NT):
  - `SAM`: HKEY_LOCAL_MACHINE\SAM (Security Account Manager), for users' settings (e.g.: info, group) and hashed passwords
  - `SECURITY`: HKEY_LOCAL_MACHINE\SECURITY, for system security settings
  - `SOFTWARE`: HKEY_LOCAL_MACHINE\SOFTWARE, for Windows and application configurations
  - `SYSTEM`: HKEY_LOCAL_MACHINE\SYSTEM, for system and connected devices (e.g. USBStor)

Registry Files

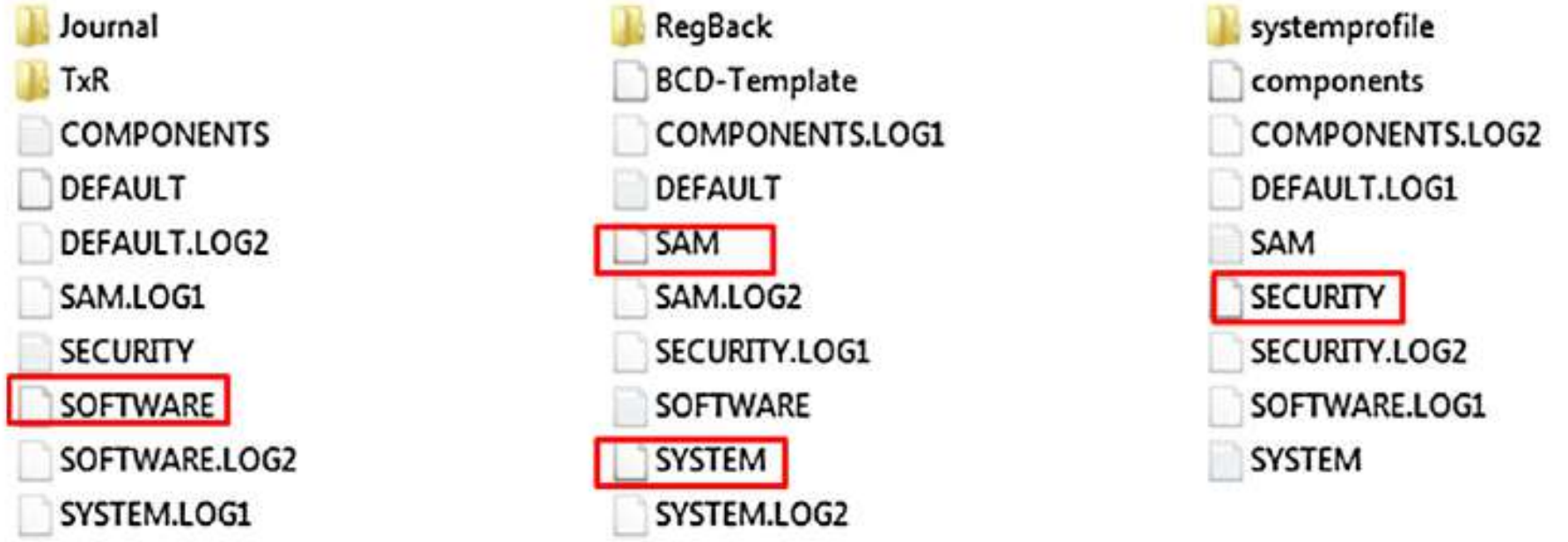


Figure 1.3 Registry hive files in the *Windows\system32\config* folder (Windows 7).

From: Harlan Carvey, “Windows Registry Forensics", 2nd Edition

Where Are the Registry Files Stored?

- **User-dependant files** in each **user's profile folder** (%USERPROFILE%\):
  - `NTUSER.DAT`: HKEY_USERS\<*User SID*>
    (linked to by HKEY_CURRENT_USER), for user activities
    (e.g. open/save MRU list, last-visited MRU list, UserAssist, recent files)
  - `USRCLASS.DAT`
- **Notes on environment variables** in Windows:
  - ***System-path variables***: locations of critical OS resources,
    e.g. %OS%, %SystemRoot%
  - ***User-management variables***: resources and settings owned by various user profiles, e.g. %USERPROFILE%
- Reference: https://en.wikipedia.org/wiki/Environment_variable#Windows

Registry Elements

- Two basic registry **elements**:
  - ***Keys***:
    - **Container** objects similar to **folders**
    - May contain values and subkeys
  - ***Values***:
    - **Non-container** objects similar to **files**
    - Each has its ***type*** and ***data***
    - Registry value types:
      see https://docs.microsoft.com/en-us/windows/win32/sysinfo/registry-value-types

Windows Registry

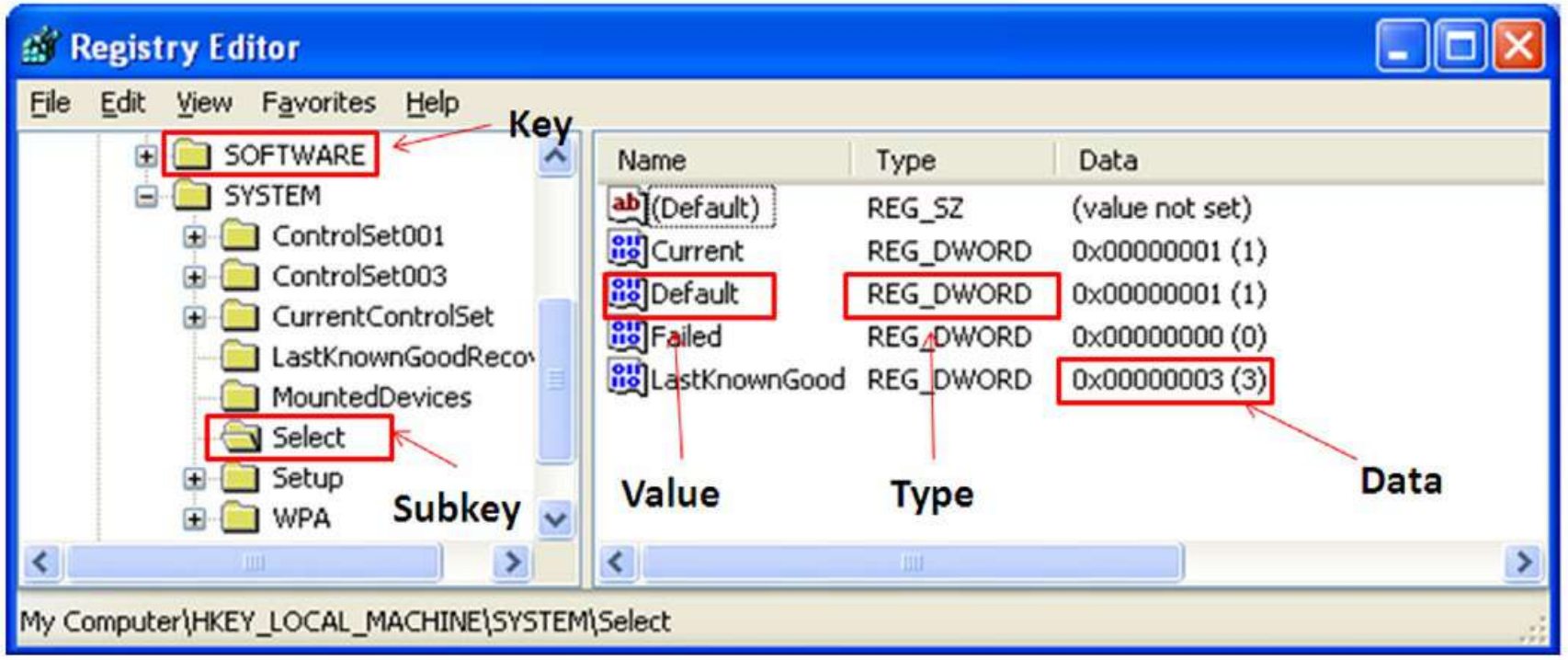


Figure 1.4 Registry nomenclature.

From: Harlan Carvey, “Windows Registry Forensics", 2nd Edition

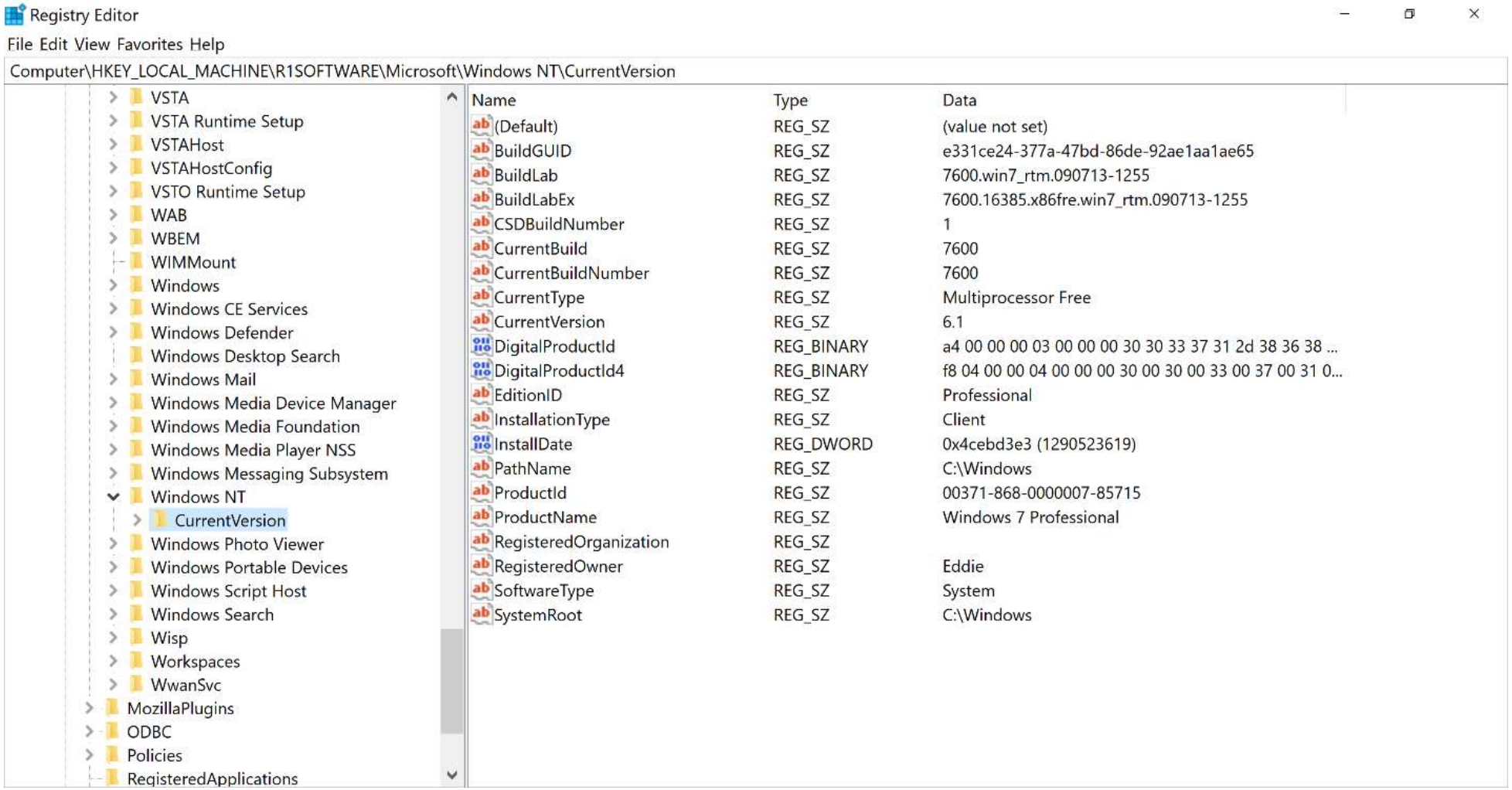
ControlSet00*n*

- **ControlSet00*n*** subkeys represent **a control set** for the system:
  - The **numbered** ControlSet00*n* subkeys (e.g. ControlSet001 & ControlSet002) contain control sets that can be used to **start & run Windows**
  - Usually **2 numbered control sets**: an **original** & a **backup** copy of a control set that has been used to start the system successfully
  - But can be as many as **4** control sets
  - **Backup copies**: maintained to undo configuration changes that might otherwise prevent you from starting the system
- The **`Select` subkey**: stores the control set that was last used successfully, the current control set, and the default control set
- Ref: https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc960234(v=technet.10)

Registry Editing

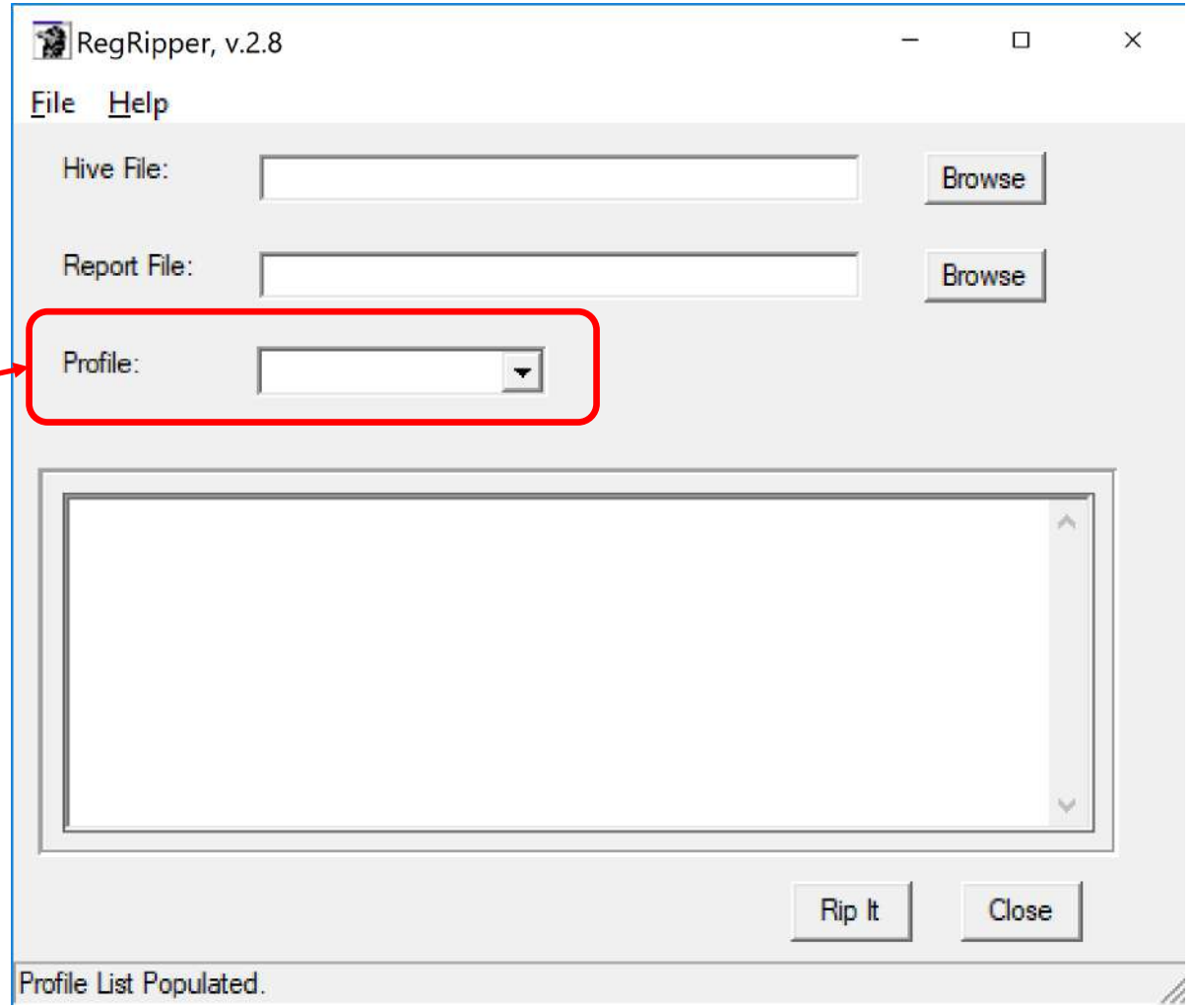
- **Registry access** (viewing/editing) on Windows:
  - **RegEdit** (available on Windows)
  - Other tools: RegRipper, MiTeC's Windows Registry Recovery, EricZimmerman's **RECmd & Registry Explorer**
- **RegRipper**: a popular online and offline registry analysis tool
  - Developed by Harlan Carvey
  - Can be used to perform an analysis on:
    - A **live** Windows machine: use the machine's registry files
    - An **offline** target machine: use the captured registry files
  - Has both CLI (rip.exe) and GUI (rr.exe) versions
  - Numerous plugins
- See Lab 5, Tasks 1-A and 1-B

Windows RegEdit



Registry-Analysis Tool: RegRipper

Removed in RegRipper 3.0



Registry-Analysis Tools: Registry Explorer & RECmd

- From Eric Zimmerman
- **Quite popular** in Digital Forensics & Incident Response
- **Registry Explorer**:
  - A GUI-based tool
  - Extra **features**: searching, filtering, bookmarking, other visualization
- **RECmd**:
  - A CLI-based tool
  - Script access to registry hives:
    can automate searching across multiple registry hives at once
- Reference: https://aboutdfir.com/toolsandartifacts/windows/registry-explorer-recmd/

Sample Registry Keys & Useful Information

- HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\WindowsNT\ **`CurrentVersion`** key:
  - Windows version (product name): the data of its **`ProductName`** value
  - Windows product ID: the data of its **`ProductId`** value
  - The registered owner and organization: the data of its **`RegisteredOwner`** and **`RegisteredOrganization`** values, respectively
- For IE typed URLs information: NTUSER.DAT\Software\Microsoft\ Internet Explorer**`Typed URLs`** key
- For USB devices information: SYSTEM\Enum**`USBSTOR`** key
- See Lab 5 Tasks 4-A and 4-B

Other Useful Information: MRU Files

- Various ***Most Recently Used (MRU)*** files of Windows applications
- From NTUSER.DAT registry file:
  - **Excel's MRU**:
    e.g. NTUSER.DAT\Software\Microsoft\Office*version*\Excel\`File MRU`
  - **Word's MRU**:
    e.g. NTUSER.DAT\Software\Microsoft\Office*version*\Word\`File MRU`
  - **File Explorer's** *recently-accessed* document folders: e.g.
    NTUSER.DAT\Software\Microsoft\Windows\CurrentVersion\Explorer\
    RecentDocs\`Folder`
    (Note: the folder access order is kept in `MRUListEx`)

Registry Analysis: USB Device's Timestamps

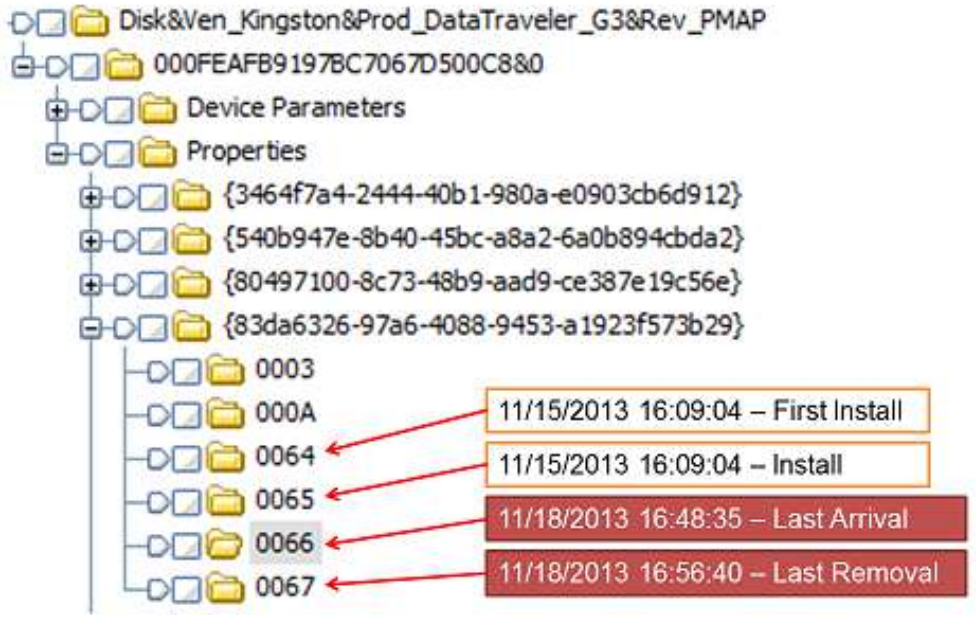
TABLE 5—*MSC devices—comparison of artifacts in different operating systems.*

	Key/Subkey	Win 7	Win 8	Win 10
First insertion time from DeviceClasses key in System Hive	10497b1b-ba51-44e5-8318-a65c837b6661	✓	✓	✓
	53f56307-b6bf-11d0-94f2-00a0c91efb8b	✓	✓	✓
	53f5630d-b6bf-11d0-94f2-00a0c91efb8b	✓	✓	✓
	65a9a6cf-64 cd-480b-843e-32c86e1ba19f	✓		
	6ac27878-a6fa-4155-baS5-f9Sf491d4f33	✓	✓	✓
	7f108a28-9833-4b3b-b780-2c6b5fa5c062		✓	✓
	7fccc86c-228a-40ad-8a58-f590af7bfdce		✓	✓
	a5dcbf10-6530-11d2-901f-00c04fb951ed	✓	✓	✓
	EEC5AD98-8080-425f-922A-DABF3DE3F69A	✓		
	f33fde04-d1ac-4e8e-9a30-19bbd4b108ae	✓	✓	✓
First insertion time from System Hive Under USBSTOR Property Key	0003	✓	✓	✓
	000A		✓	✓
	0064	✓	✓	✓
	0065	✓	✓	✓
Last insertion time from System Hive Under USBSTOR Property Key	0066		✓	✓
Last removal time from System Hive Under USBSTOR Property Key	0067		✓	✓
First insertion time from System Hive under USB Key	VID_[VendorID]&PID_[ProductID]\[SerialNo]\DeviceParameters \e5b3b5ac-9725-4f78-963f-03dfb1d828c7			✓
Last insertion time from System Hive under USB Key	VID_[VendorID]&PID_[ProductID]\[SerialNo]\	✓	✓	✓
First insertion time from Software Hive	Microsoft\Windows Portable Devices\Devices \SWD#WPDBUSENUM#_??_ USBSTOR#DISK&VEN_[VenderName] &PROD_[ProductName]&REV_PMAP#[SerialNo]#{53F56307-B6BF-11D0-94F2-00A0C91EFB8B}	✓	✓	✓

From: Arshad et al. "*USB Storage Device Forensics for Windows 10*", 2017

Registry Analysis: USB Device's Timestamps

- Example of relevant keys:



From:http://www.swiftforensics.com/2013/11/windows-8-new-registry-artifacts-part-1.html

- Additional reference:
  - https://forensics.wiki/usb_history_viewing/

Registry Analysis: USB Device's Timestamps

TABLE 6—*MTP and PTP devices—comparison of artifacts in different operating systems.*

	Key/Subkey	Win 7	Win 8	Win 10
First insertion Time from DeviceClasses key in System Hive	10497b1b-ba51-44e5-8318-a65c837b6661	✓	✓	✓
	6ac27878-a6fa-4155-baS5-f9Sf491d4f33	✓	✓	✓
	6bdd1fc6-810f-11d0-bec7-08002be2092f	✓	✓	✓
	a5dcbf10-6530-11d2-901f-00c04fb951ed	✓	✓	✓
	EEC5AD98-8080-425f-922A-DABF3DE3F69A	✓		
	f33fdc04-d1ac-4e8e-9a30-19bbd4b108ae	✓	✓	✓
First insertion time from System Hive Under USB Property Key	0003	✓	✓	✓
	0007		✓	✓
	0008		✓	✓
	0009		✓	✓
	000A		✓	✓
	0064	✓	✓	✓
	0065	✓	✓	✓
Last insertion time from System Hive Under USB Property Key	0066		✓	✓
Last removal time from System Hive Under USB Property Key	0067		✓	✓
First insertion time from System Hive under USB Key	VID_[VendorID]&PID_[ProductID]\[SerialNo]\DeviceParameters \e5b3b5ac-9725-4f78-963f-03dfb1d828c7			✓
Last insertion time from System Hive under USB Key	VID_[VendorID]&PID_[ProductID]\[SerialNo]\	✓	✓	✓

From: Arshad et al. "*USB Storage Device Forensics for Windows 10*", 2017

Registry for Incident Response

- **Run & RunOnce registry keys:**
  - The keys cause programs to run each time that a user logs on
  - Used by attackers to achieve persistent access
- **Locations**:
  - HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Run
  - HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
  - HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\RunOnce
  - HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce
- For more details: https://docs.microsoft.com/en-us/windows/win32/setupapi/run-and-runonce-registry-keys
- Can be inspected by: Autoruns from SysInternals (Microsoft)

Useful Registry Keys for Forensics

- **Question:** *Which registry keys to check for forensic purposes?*
- Some handy online **references**:
  - ***Microsoft Registry Reference***: https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc974061(v=msdn.10)
  - AcessData's Registry Quick Find Chart: uploaded to Canvas
  - Derrick J. Farmer, "*A Windows Registry Quick Reference: For the Everyday Examiner*": uploaded to Canvas
  - A forensic cheat sheet: https://www.13cubed.com/downloads/dfir_cheat_sheet.pdf
  - Its accompanying video demo: https://www.youtube.com/watch?v=VYROU-ZwZX8
  - Another video: https://www.youtube.com/watch?v=hPpqWILbrPI

Additional References on Windows Registry

General registry analysis:

- Harlan Carvey, "*Windows Registry Forensics: Advanced Digital Forensic Analysis of the Windows Registry*", 2nd Edition, 2016
- Alghafli et al., "*Forensic Analysis of the Windows 7 Registry*", Journal of Digital Forensics, Security and Law, Vol. 5, No. 4, 2010

USB devices and registry analysis:

- Carvey and Altheide, "*Tracking USB storage: Analysis of windows artifacts generated by USB storage devices*", Digital Investigation (2005) 2, 94-100
- Arshad et al., "*USB Storage Device Forensics for Windows 10*", J. Forensic Sci, 2017, onlinelibrary.wiley.com

Lab 5 Exercises

Lab 5 Exercises

- Task 1: Performing an **automated file carving** on a target disk image using:
  - Carver Recovery (based on Scalpel)
  - Bulk Extractor
- *(Optional)* Task 2: Performing a **manual file carving** of a deleted file using FTK Imager
- Task 3-A: Finding **interesting files** using Autopsy
- Task 3-B: Performing **keyword search** in Autopsy
- Task 4-A: **Manual registry analysis** using **RegEdit**
- *(Optional)* Task 4-B: **Automated registry analysis** using **RegRipper**

Don't forget your Graded Lab Tasks #3!

For Your Offline Discussion

- Suppose you are given a disk image file of a target **Windows machine** belonging to a suspect
- You are asked to find out ***Windows OS,*** ***user configurations*** **of the machine**, and also **some files of interest** in the file system
- *Question:* What **pieces of evidence** should you check, and how would you inspect them?

Questions?
See you next week!