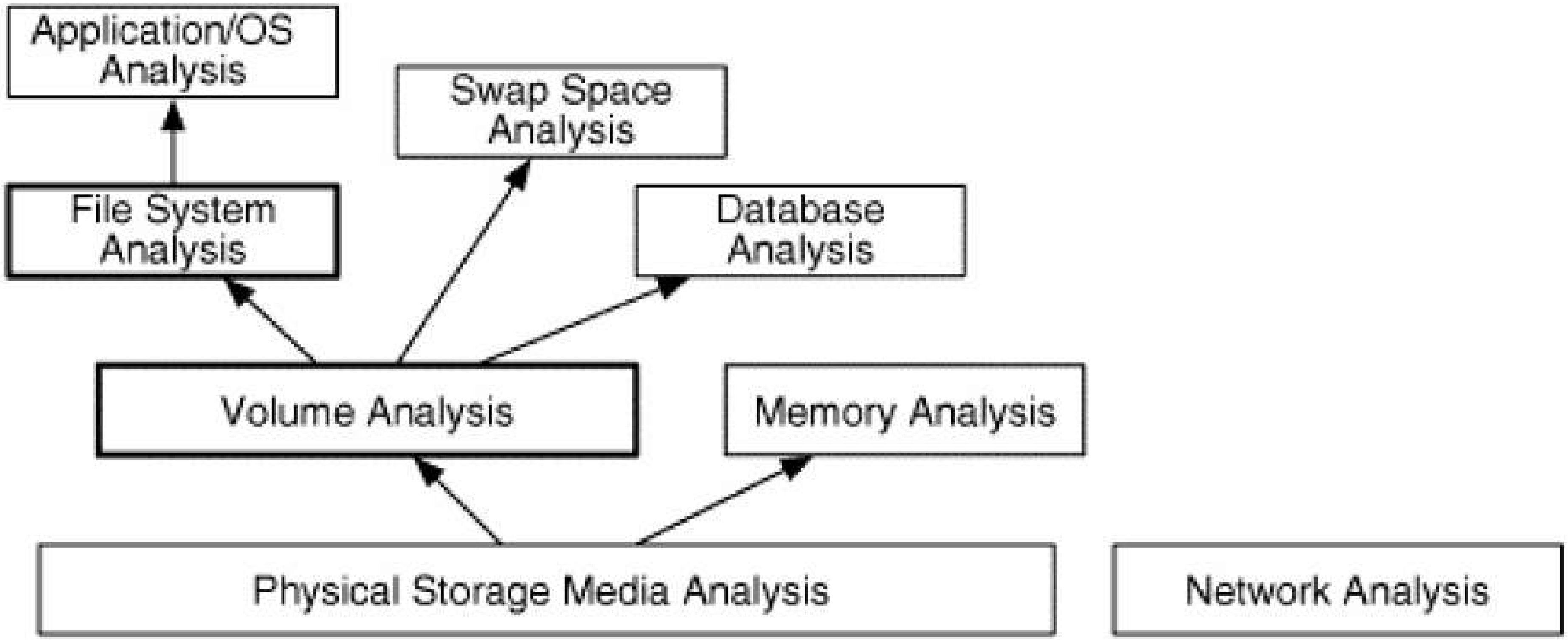
Ungraded Pre-Lecture Quiz

- To sum up our last 2 lectures on **live/volatile** and **static/non-volatile** memory acquisitions: *what are the differences between them?*
- Do mention the differences in terms of:
  - Acquisition **source or target**
  - Acquisition **time** (e.g. before or after a target machine is turned off)
  - **Machine** to perform the acquisition
  - **Usable tools** and **equipment**
  - Potential **issues**

IFS4102: Digital Forensics

Lecture 4: Storage Media, Disk and File Analyses

Layers of Disk & File Analysis



From: Brian Carrier, “File System Forensic Analysis”

Outline

- Introduction to **storage media**
- **Hard disk drive**: geometry, components, interfaces, capacity
- **Disk analysis**: addressing, hidden areas, boot sequence
- **Volume analysis**: volume, partition, partition management
- **File system (FS) analysis**: FS, metadata, FS management categories, TSK tool
- **Application-file analysis**: file signature, hash lookup, application metadata, file carving
- **Lab 4 exercises**

Questions to Answer in This Lecture

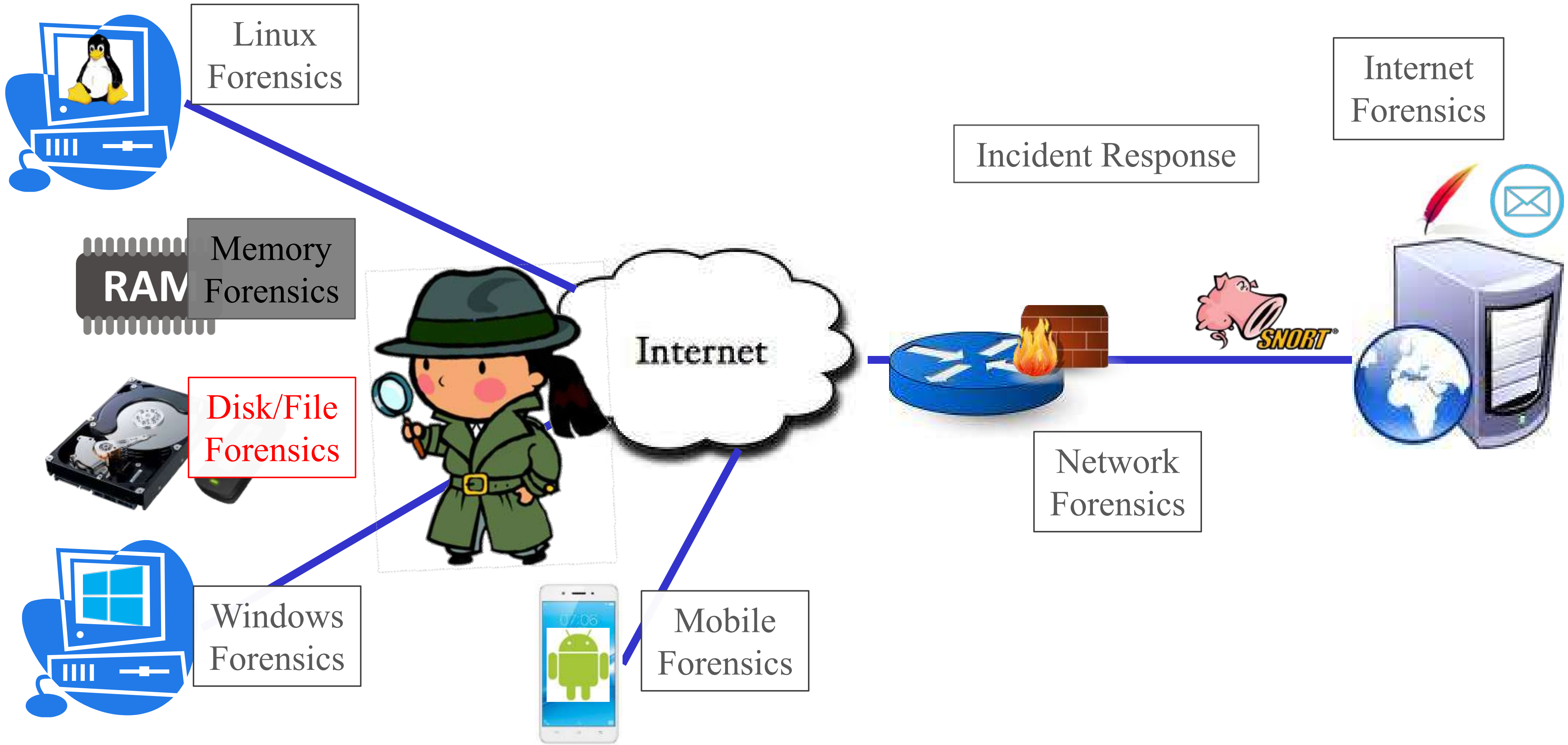
Source: Wikipedia

***Question**: Given an acquired **disk image file**:*

- Are there any **hidden areas** in the disk? Can we acquire them too?
- How to find out any **partitions/volumes** contained?
- How to analyze the files, including **deleted ones**, in its file system?

- How to find its files based on **known bad hash values**?
- How to inspect the **metadata** of application files?
- How to determine the **actual file type** of a file?
- How to perform a **file carving** on deleted files?

This Lecture's Focus



Introduction to Storage Media

Storage Media

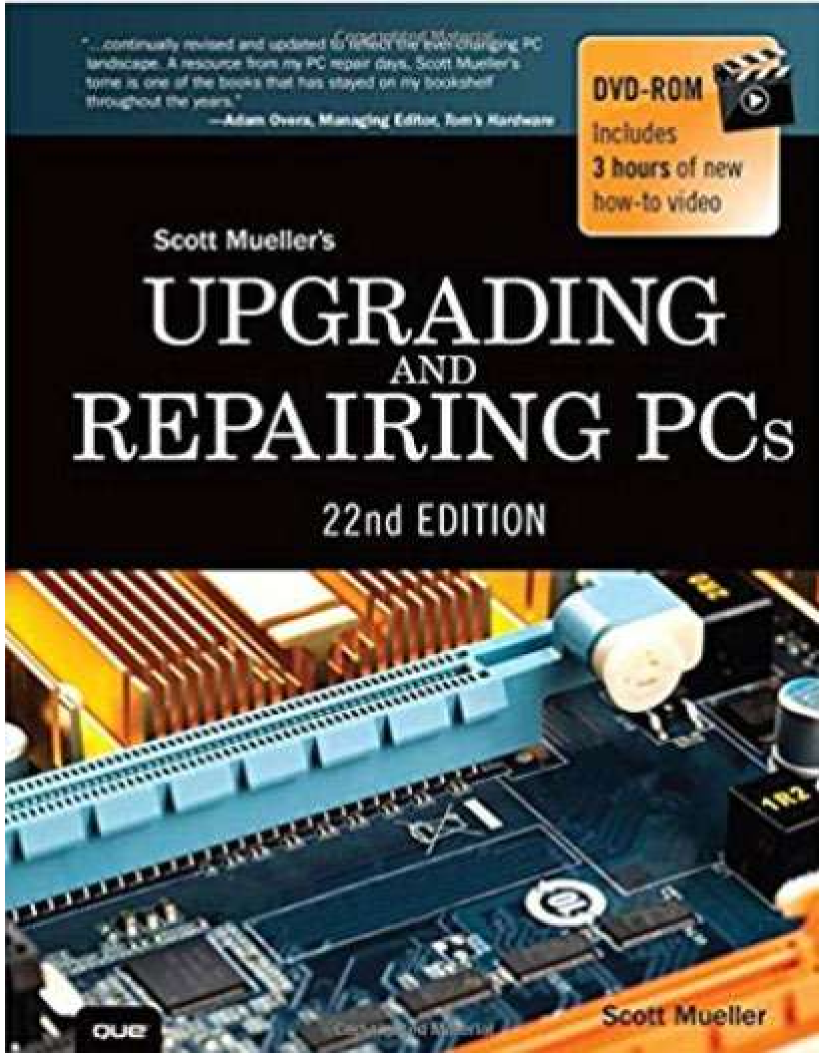
- In computer forensics we are often concerned with **storage media**
- Two **types** of storage media (review):
  - Volatile
  - Non-volatile
- ***Volatile*** storage media:
  - Preserving contents/read/write **requires power**
  - CPU: registers & cache
  - Memory/RAM: system portion vs user portion

Storage Media

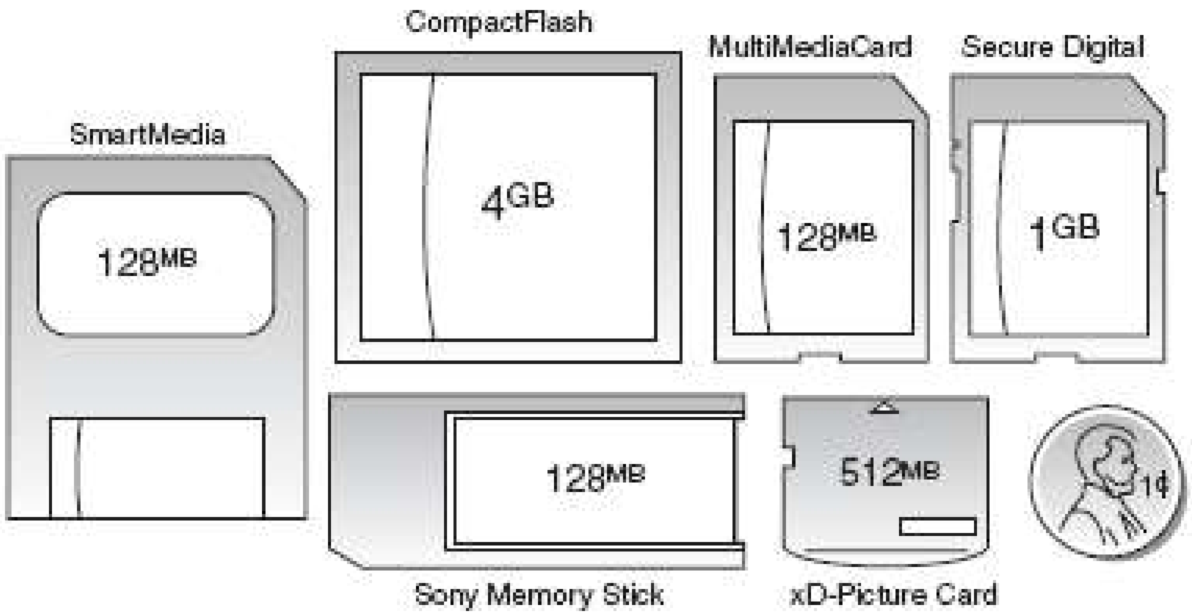
- ***Non-volatile*** storage media:
  - **Magnetic** media: **Hard disk drive**
  - **SSD**: NVRAM, flash memory (nowadays)
  - **USB flash/thumb drive**: can also be used as a boot device
  - **Small storage media**: flash memory cards, e.g. SD, XD, Mini SD, Micro SD, Memory Stick, Compact Flash
  - **Optical media**: CD, CD-RW, DVD, Blu-ray Disc (BD)
  - *Less common nowadays*: **Floppy disk**, **Zip disk**, **Magnetic tape**

Hardware Reference

- Scott Mueller, "***Upgrading and Repairing PCs*"**, 22nd edition, Que Publishing, 2015



Small Storage Media (Mueller 2009)



Shown in relative scale to a U.S. penny

Device	Capacity	Notes
Compact Flash	16MB – 100GB	Highest Capacity, commonly used for SLR
SmartMedia	16MB – 512 MB	Older Fujifilm and Olympus Digital Cameras
Multi Media Card (MMC)	16MB – 4 GB	MMC cards can work in most SD card slots
RS MMC	128MB – 2GB	Use adapter to plug into MMC closts
SD	16MB – 64GB	Used by most brands of digital cameras
MiniSD / MicroSD	128MB – 16GB	Used in mobile phones. Requires SD card adapter to be read in PC
Memory Stick	16MB – 128MB	Developed by Sony
USB	16MB – 128GB	Some include password protection / write protect features

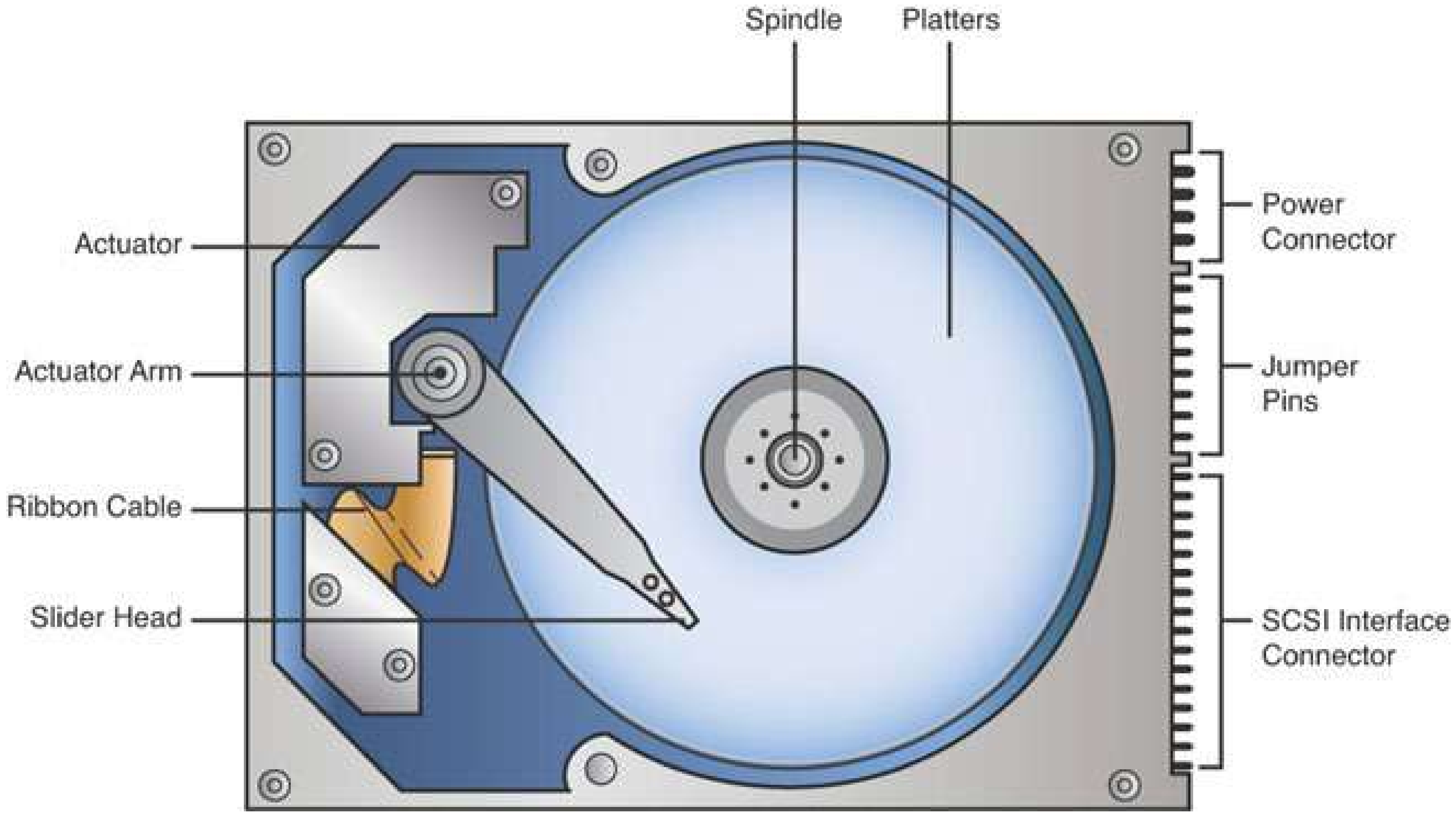
Hard Disk Drive

Hard Disk Drive: Geometry

- Cheap and **widely used**: *very important to digital forensics!*
- Disk **geometry** & **components**:
  - **Platters**
  - Read/write **heads**: one head on one platter side
  - Each platter is divided into **track** and **sector**:
    track no & sector no for data addressing and retrieval purposes
  - **Cylinder**: concentric tracks of all the platters
- Total ***capacity*** of a disk (a.k.a. **CHS formula**):
  #cylinders * #heads * #sectors * #bytes/sector

What an Operating System Is (cont.)

The Physical Layout of a Hard Disk



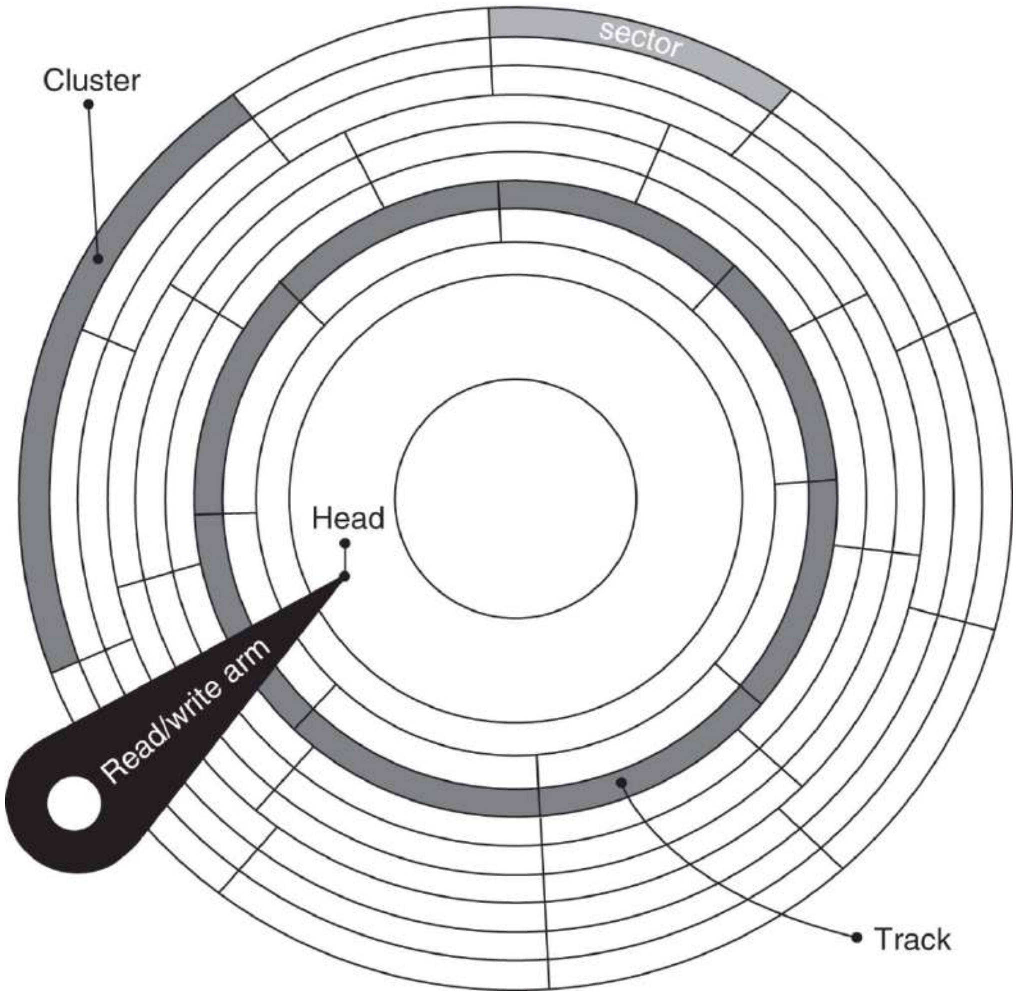
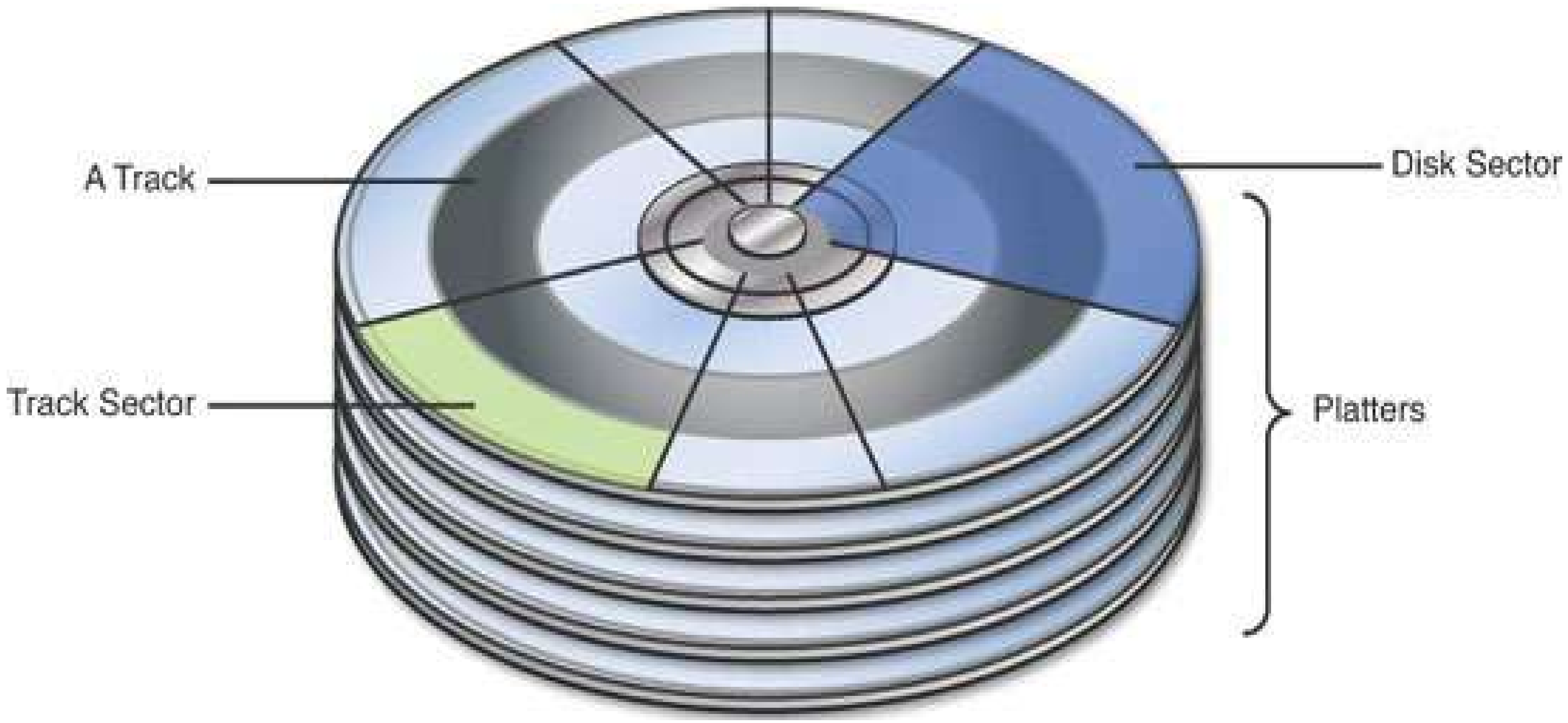


FIGURE 15.5 A depiction of platters, tracks, sectors, clusters, and heads on a computer disk.

What an Operating System Is (cont.)

The Layout of a Hard Drive

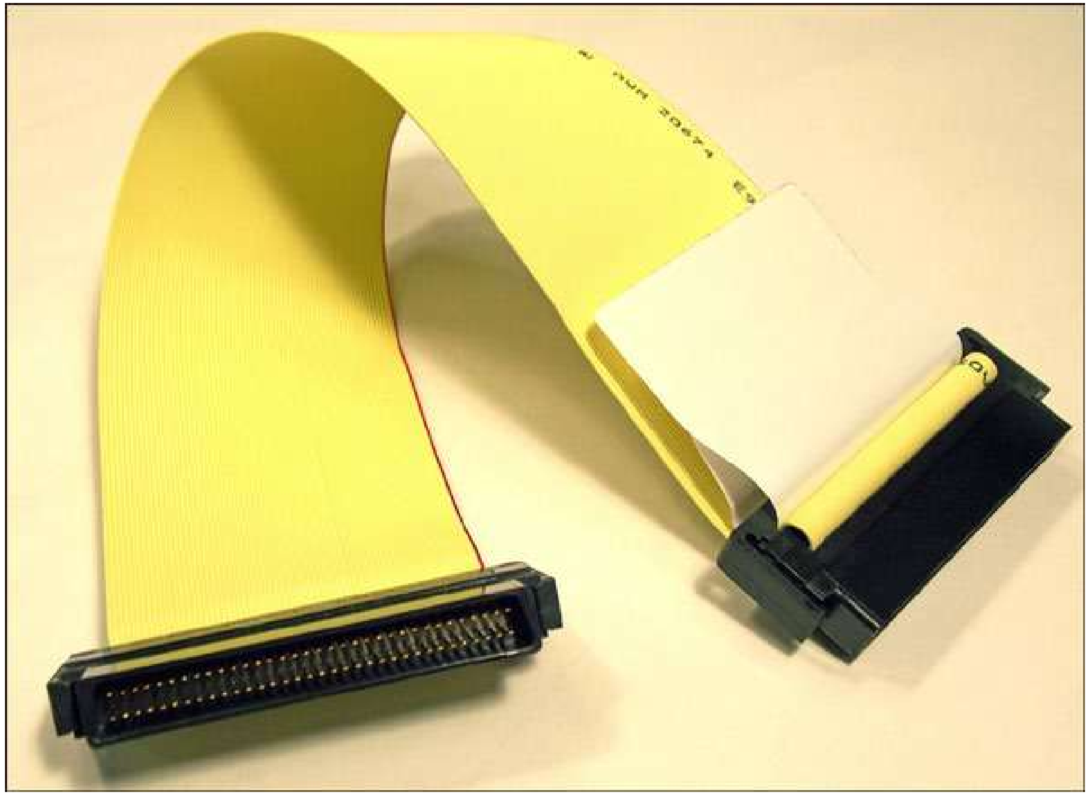


Hard Disk Drive: Interfaces

- Below are some hard drive's **common *interface types***
- **SCSI** ():
  - **Daisy-chain** up to 16 peripheral devices onto a single cable/controller
  - **Variants**: FastSCSI, Ultra SCSI, Ultra wide SCSI, Ultra2 SCSI, iSCI, …
  - Was popular before USB came along
  - See: https://www.youtube.com/watch?v=pR7SdrXdT4M
- **PATA**:
  - Parallel AT Attachment
  - Formerly also called **ATA**, **IDE**, **EIDE**,
  - Wide **ribbon-style** data cable, master & slave devices (with jumper setting)
  - See: https://www.youtube.com/watch?v=sOb4ur5EbTY

Handling Computer Hardware (cont.)

SCSI Connector

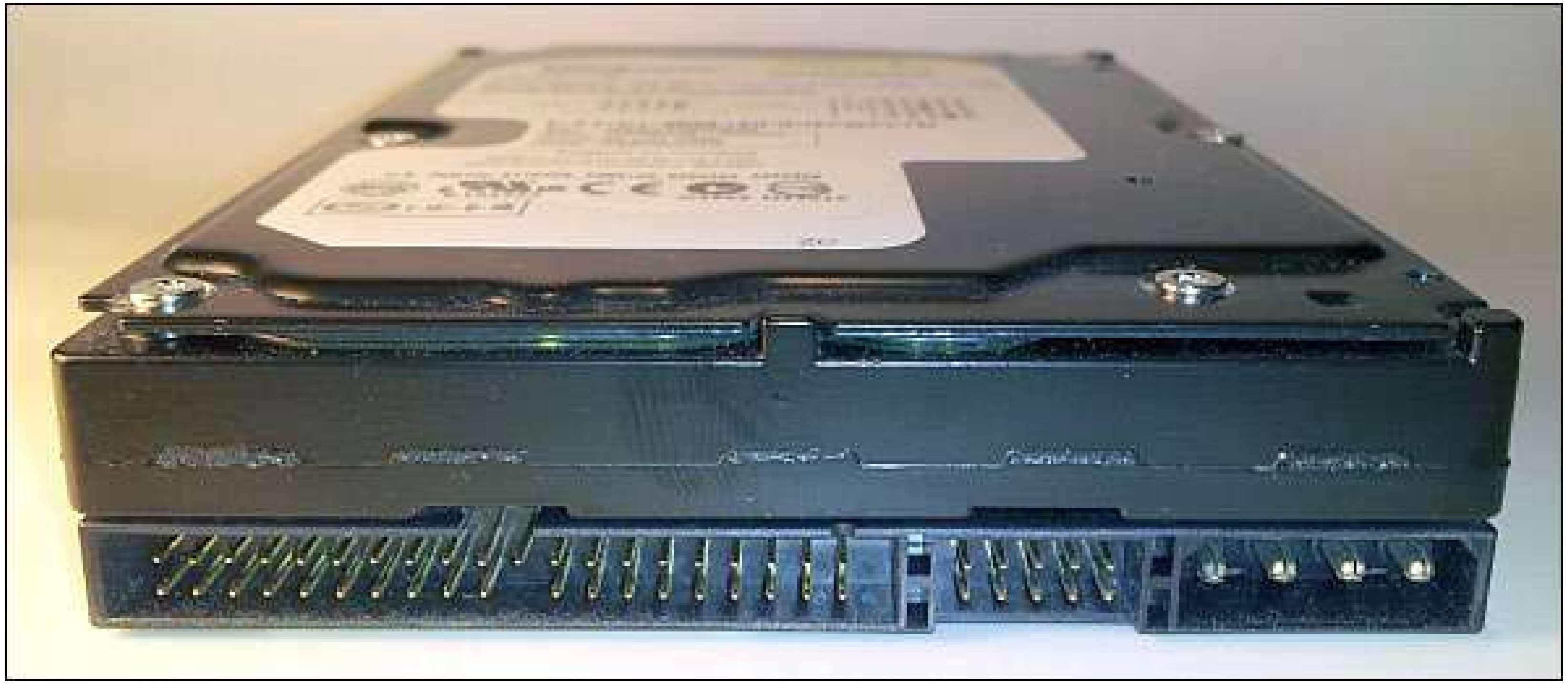


PATA

Source: Wikipedia

Handling Computer Hardware (cont.)

IDE Interface on a Hard Disk



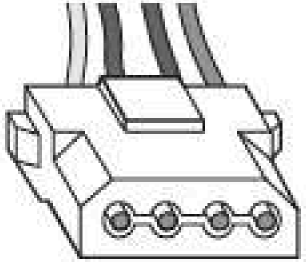
Hard Disk Drive: Interfaces

- **SATA** ():
  - Serial AT Attachment
  - **Newer** PATA: **succeeded** PATA to become the predominant interface
  - **1-to-1 connection**: a single drive on each connection
  - See: https://www.youtube.com/watch?v=N7FmrnUU4Y8
- **PATA vs SATA**:
  - See: https://www.youtube.com/watch?v=myU2x27FIIc
- **Fibre Channel (FC)**:
  - See: https://www.youtube.com/watch?v=prkPpAPm4lA

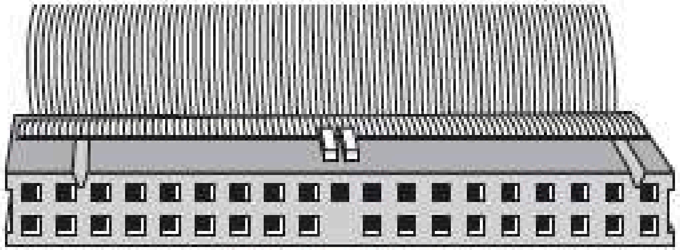
Handling Computer Hardware (cont.)

SATA Data Cable and SATA Power Cable

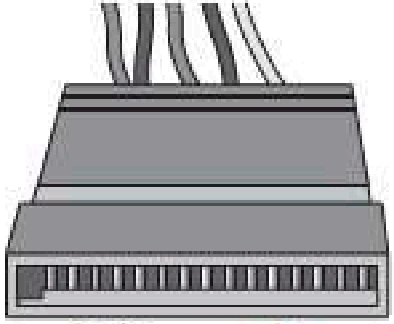
Parallel vs Serial ATA Cables (Mueller)



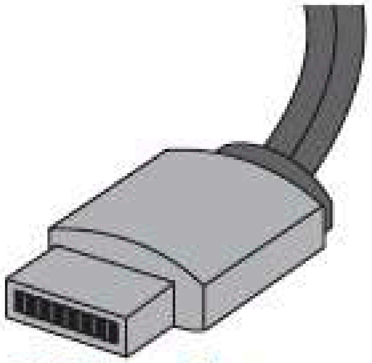
Parallel ATA power cable



Parallel ATA data cable



SATA power cable



SATA data cable

Evolution of Drive Interfaces (Mueller 2009)

Table 7.1 PC Drive Interfaces

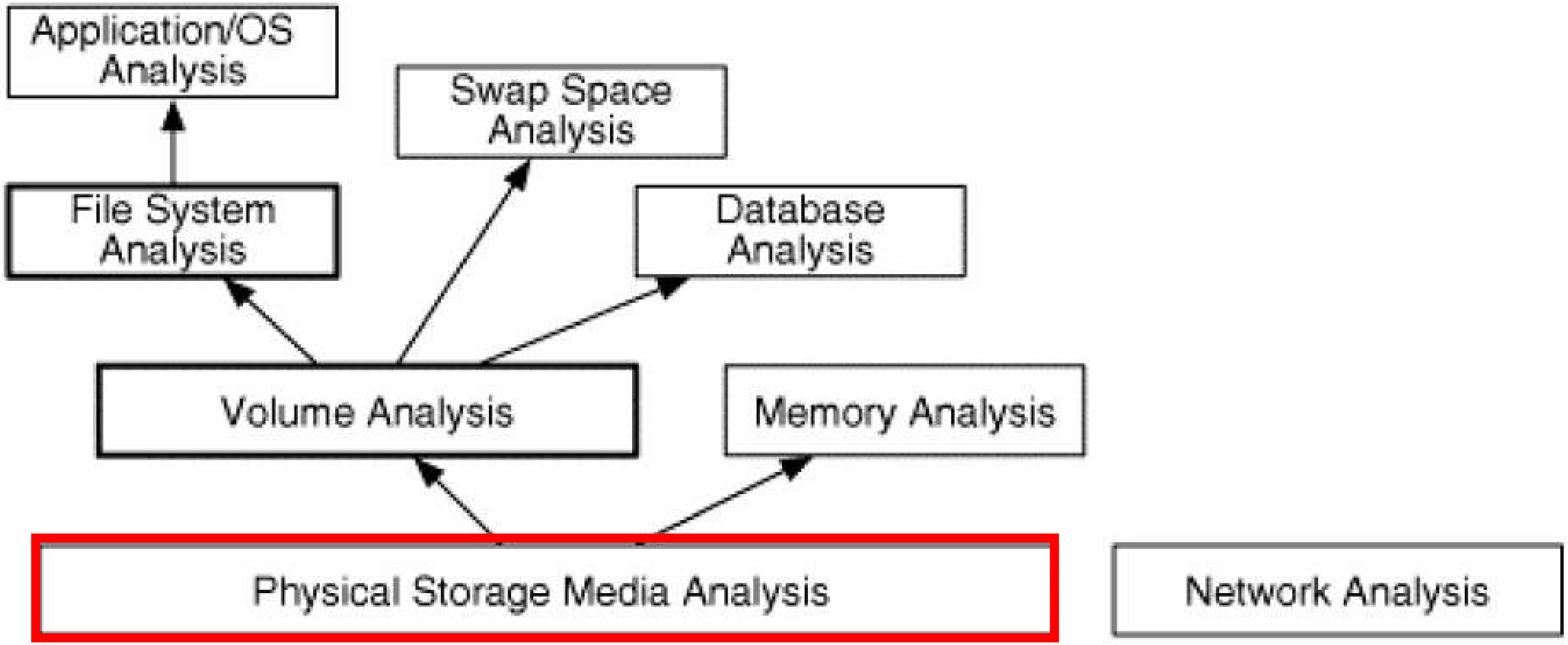
Interface	When Used
ST-506/412	1978–1989 (obsolete)
ESDI	1983–1991 (obsolete)
Non-ATA IDE	1987–1993 (obsolete)
SCSI	1986–present
Parallel ATA (IDE)	1986–present
Serial ATA	2003–present

Hard Disk Drive: A Note on Storage Capacity

- **Metric/SI prefixes** vs **binary prefixes** (standardized by International Electrotechnical Commission, IEC)
- Powers of **10** vs powers of **2** (1024 = 2^{10}):
  - 1000 = K (kilo) vs 2^{10} = **Ki** (kibi)
  - 1000^2 = M (mega) vs 2^{20} = **Mi** (mebi)
  - 1000^3 = G (giga) vs 2^{30} = **Gi** (gibi)
- *CS people love **powers of 2**: we count in binary!*
- Also, the JEDEC (Joint Electron Device Engineering Council) standards for semiconductor memory use the K, M, G and T in **the binary sense**
- See: https://en.wikipedia.org/wiki/Binary_prefix

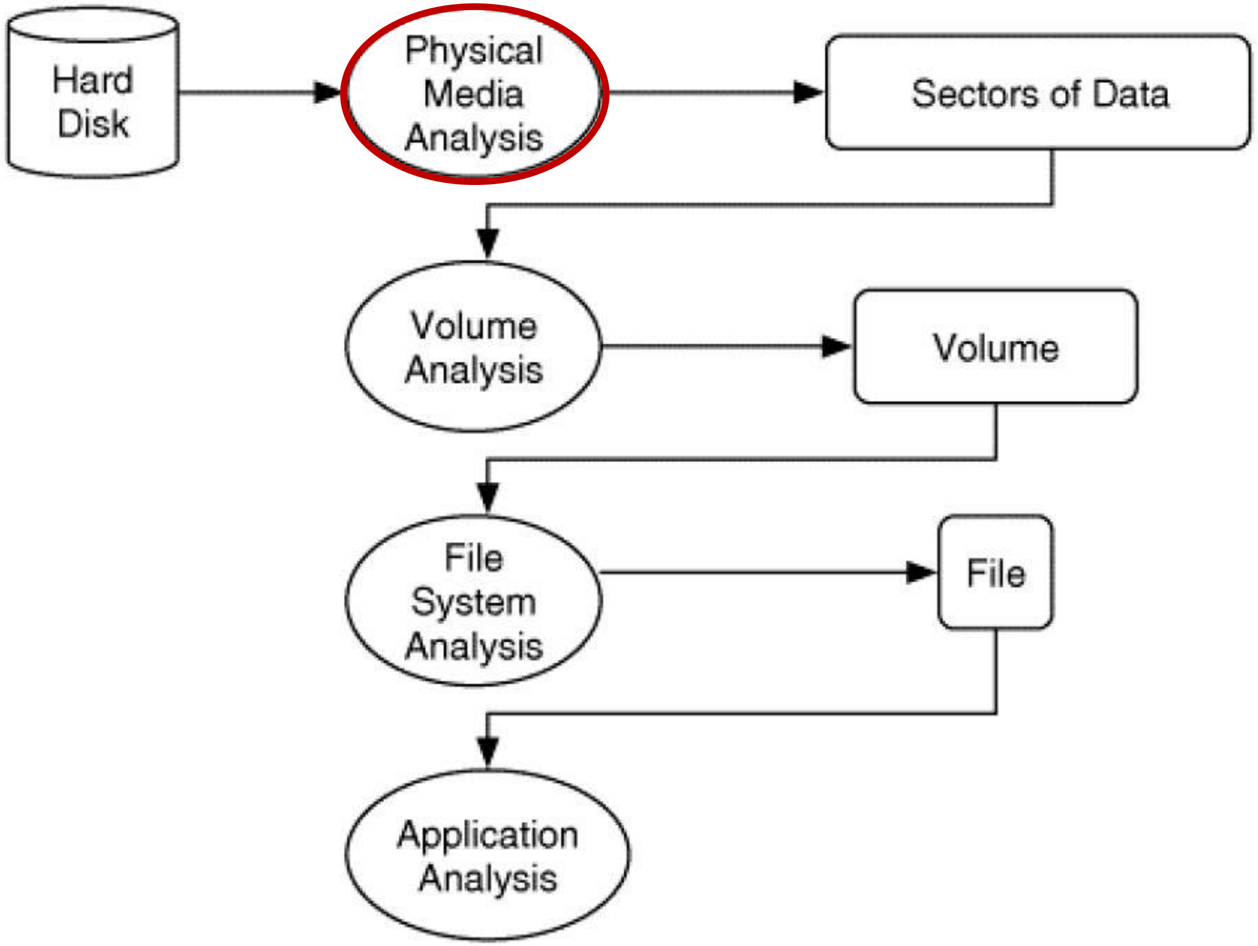
Disk Analysis

Layers of Disk & File Analysis



From: Brian Carrier, "File System Forensic Analysis"

Layers of Disk & File Analysis



From: Brian Carrier, “File System Forensic Analysis”

Disk Structure Analysis

Relevant **disk concepts**:

- Physical
- Partition
- Volume
- Sector
- ***Cluster*** *(more later)*
- File
- ***File slack*** *(more later)*
- ***Disk slack***: unallocated clusters & unused disk area

Notes on Reference Book on Disk/File Analysis

- A good reference book on disk and file analysis for forensics: **Brian Carrier**, ***"File System Forensic Analysis"***, 1 edition, Addison-Wesley Professional, 2005:
  - It is rather outdated
  - But it still a **standard reference book** on disk and file forensics
  - Written by **Brian Carrier**: the original author of TSK & Autopsy
- **TSK** is very useful to understand disk and file forensics better
  - Can allow for **powerful scripting**
  - Covered in this lecture and Lab 4
- Another reference book (with Linux): Bruce Nikkel, "*Practical Forensic Imaging: Securing Digital Evidence with Linux Tools*", 1st Edition, No Starch Press, 2016

Disk Addressing: Physical Addressing

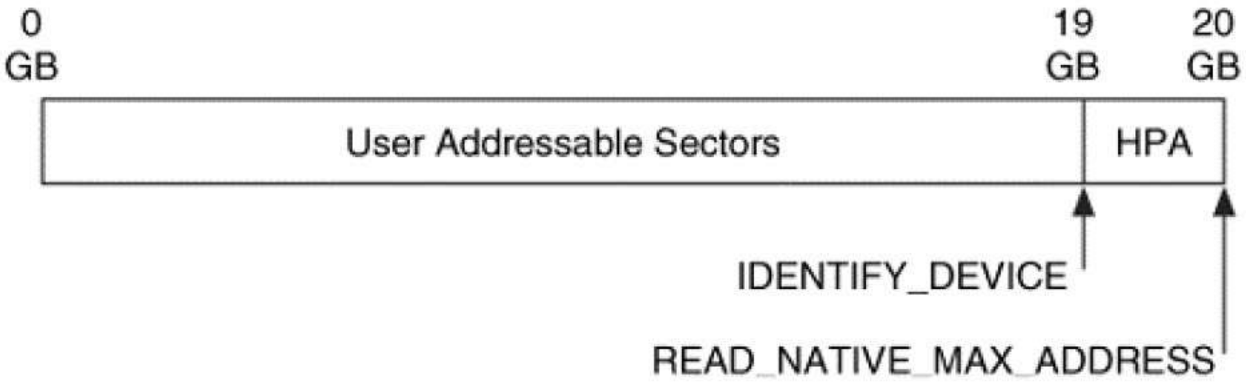
- General notion of ***offset***:
  - Use to refer to a location from a particular ***reference point*** (e.g. the beginning of a media, sector or file)
  - By ***how many bytes*** relative to the reference point (in hex, with 0x prefix)
- ***Physical address*** of a **sector**:
  its address **relative to** the start of the physical media
- Two ***physical addressing*** methods:
  - **CHS method:**
    - Cylinder, Head, Sector
    - The sector numbers always start at 1
    - Issue: too limiting (504MB - 8.1GB) → *considered outdated by now*

Disk Addressing: Physical Addressing

- **Logical Block Addresses (LBA) method**:
  - A ***linear numbering*** with a single number
  - Starting at **0**
  - Can derive LBA based on CHS:
    LBA = (((CYLINDER * heads_per_cylinder) + HEAD) * sectors_per_track) + SECTOR – 1
- Examples of CHS and LBA **conversion**:
  - CHS address 0, 0, 1 ↔ LBA address 0
  - CHS address 0, 0, 2 ↔ LBA address 1
  - …

Hidden Disk Area: HPA & DCO

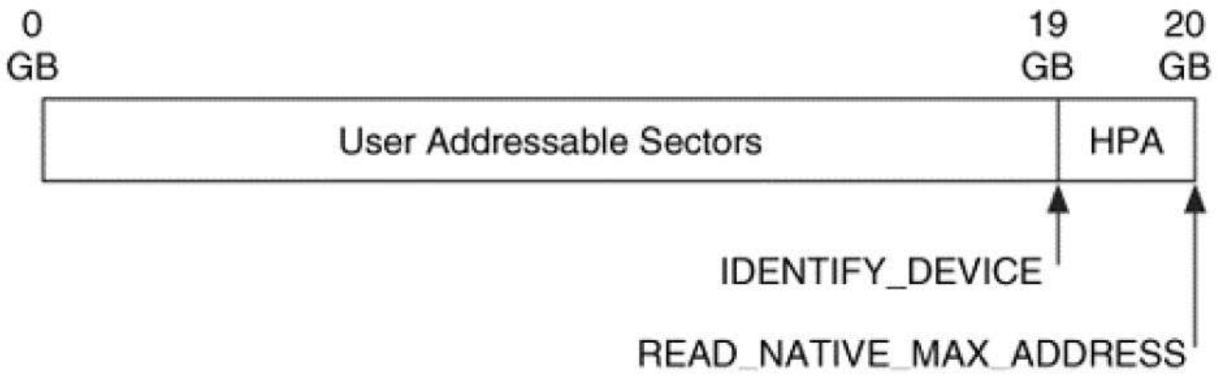
- HPA and DCO:
  - Disk areas that can be used to **hide data**
- **HPA (*Host Protected Area*):**
  - **Goal**: **vendors** could store data that would ***not* be erased** when a user formats & erases the disk contents: for system recovery data, diagnostic tools, etc.
  - **Size**: 0 (default), but is **configurable** using ATA (ATA-4) commands



From: Brian Carrier, "File System Forensic Analysis"

Hidden Disk Area: HPA

- To **check** an HPA, use two following commands returning the **max addressable sector** values:
  - READ_NATIVE_MAX_ADDRESS: returns the **maximum physical address**
  - IDENTIFY_DEVICE: returns only the no of sectors **accessible by user**
- To **create** an HPA: use SET_MAX_ADDRESS command to set the **max *user-accessible* sector**
- To **remove** an HPA: also use SET_MAX_ADDRESS command to the READ_NATIVE_MAX_ADDRESS value



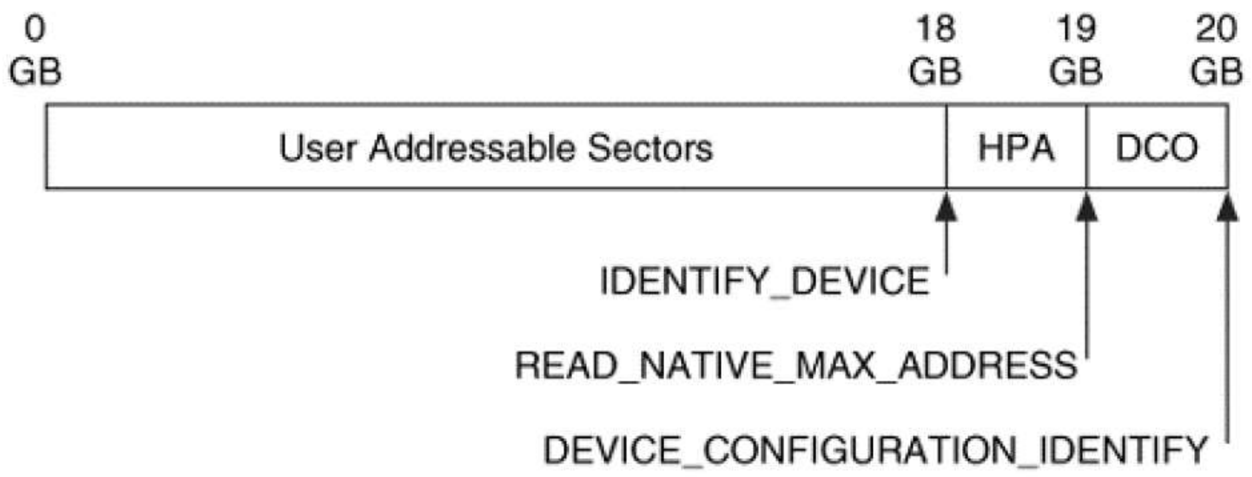
From: Brian Carrier, "File System Forensic Analysis"

Hidden Disk Area: HPA

- Notes on a **HPA removal action**:
  - SET_MAX_ADDRESS command has a ***volatility bit***:
    if **set**, it causes the HPA to exist *after* the HDD is reset/power-cycled
    $\rightarrow$ HPA removal becomes ***temporary***
  - Some ATA **locking mechanisms** can prevent max-address modification until the next reset:
    BIOS can perform some operations when the system is powering up
    $\rightarrow$ still *limited disk access* afterwards

Hidden Disk Area: DCO

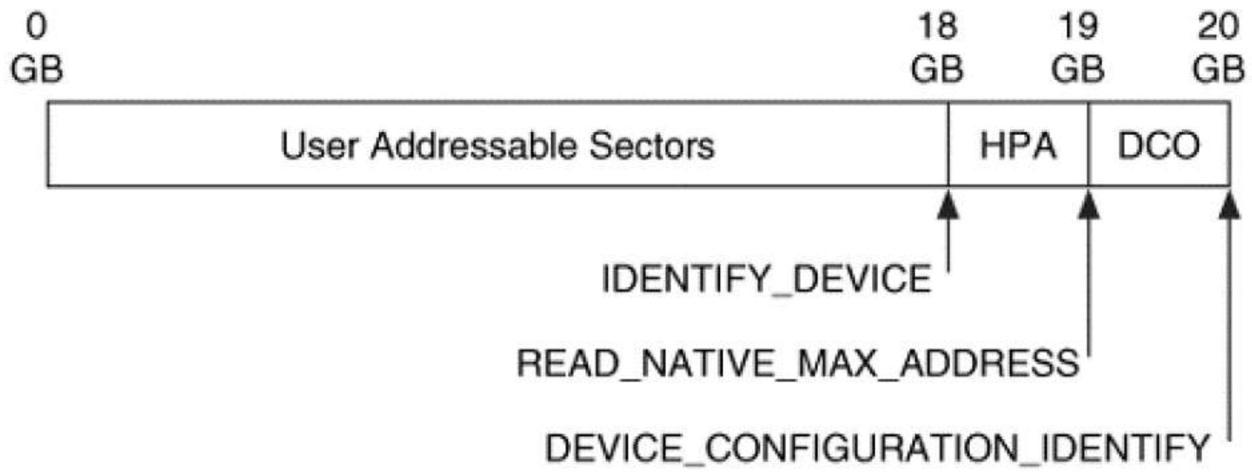
- **DCO (*Device Configuration Overlay*):**
  - Another hidden disk area:
    - To allow **configuration changes** that would limit **the capabilities of disk**
    - E.g. to make different drive models appear to have the same features
  - Can coexist with HPA, but **DCO must be created first**
  - To **check** DCO: use the DEVICE_CONFIGURATION_IDENTIFY (ATA-6) command returning the **actual** features & size of a disk



From: Brian Carrier, "File System Forensic Analysis"

Hidden Disk Area: DCO

- To **create/change** a DCO:
  use the DEVICE_CONFIGURATION_SET command
- To **remove** a DCO:
  use the DEVICE_CONFIGURATION_RESET command
- Changes are **permanent** across reset and power cycles:
  won't be revoked at the next reset



From: Brian Carrier, "File System Forensic Analysis"

Detecting HPA & DCO

- A command-line tool to **reveal** HPA and DCO: **`hdparm`** (on Linux)
- Showing disk information:
  `hdparm -I` <*disk-pathname*>

```
# hdparm -I /dev/sdl

/dev/sdl:

ATA device, with non-removable media
        Model Number:       WDC WD5003AZEX-00MK2A0
...
        LBA48  user addressable sectors:  926773168
        Logical  Sector size:                   512 bytes
        Physical Sector size:                  4096 bytes
        device size with M = 1024*1024:      452525 MBytes
        device size with M = 1000*1000:      474507 MBytes (474 GB)
...
           *    Device Configuration Overlay feature set
...
```

From: Bruce Nikkel, *"Practical Forensic Imaging: Securing Digital Evidence with Linux Tools"*

Detecting HPA & DCO

```
# hdparm -I /dev/hdb
[REMOVED]
        CHS current addressable sectors: 11088
        LBA     user addressable sectors: 12000
        LBA48   user addressable sectors: 12000
[REMOVED]
Commands/features:
        Enabled Supported:
           *    Host Protected Area feature set
```

From: Brian Carrier, "File System Forensic Analysis"

```
# dmesg | less
[REMOVED]
hdb: Host Protected Area detected.
        current capacity is 12000 sectors (6 MB)
        native capacity is 120103200 sectors (61492 MB)
```

```
# diskstat /dev/hdb
Maximum Disk Sector: 120103199
Maximum User Sector: 11999

** HPA Detected (Sectors 12000 - 120103199) **
```

From: Brian Carrier, "File System Forensic Analysis"

Detecting & Removing HPA

- Checking the existence of **HPA**:
  `hdparm -N` <*disk-pathname*>

```
# hdparm -N /dev/sdl

/dev/sdl:
 max sectors   = 879095852/976773168, HPA is enabled
```

From: Bruce Nikkel, "*Practical Forensic Imaging: Securing Digital Evidence with Linux Tools*"

- Disabling HPA (**temporarily**, the default):
  `hdparm -N` <*total-visible-sectors*> <*disk-pathname*>

```
# hdparm --yes-i-know-what-i-am-doing -N 976773168 /dev/sdl

/dev/sdl:
 setting max visible sectors to 976773168 (temporary)
 max sectors   = 976773168/976773168, HPA is disabled
```

From: Bruce Nikkel, "*Practical Forensic Imaging: Securing Digital Evidence with Linux Tools*"

Detecting & Removing HPA

- Disabling HPA (**permanently**):
  `hdparm -N` **p**<*total-visible-sectors*> <*disk-pathname*>

```
# hdparm --yes-i-know-what-i-am-doing -N p976773168 /dev/sdl

/dev/sdl:

setting max visible sectors to 976773168 (permanent)
max sectors   = 976773168/976773168, HPA is disabled
```

From: Bruce Nikkel, "*Practical Forensic Imaging: Securing Digital Evidence with Linux Tools*"

- Note: **testing** vs **actual** removal (**`--yes-i-know-what-i-am-doing`** option)
- Reference: https://forensicswiki.xyz/wiki/index.php?title=DCO_and_HPA

Detecting & Removing DCO

- Checking the existence of **DCO**:
  `hdparm --dco-identify` <*disk-pathname*>

```
# hdparm --dco-identify /dev/sdl

/dev/sdl:
DCO Revision: 0x0002
The following features can be selectively disabled via DCO:
        Transfer modes:
                udma0 udma1 udma2 udma3 udma4 udma5 udma6
        Real max sectors: 976773168
        ATA command/feature sets:
                security HPA
        SATA command/feature sets:
                NCQ interface_power_management SSP
```

From: Bruce Nikkel, *"Practical Forensic Imaging: Securing Digital Evidence with Linux Tools"*

Detecting & Removing DCO

- Removing DCO (**testing** mode):
  `hdparm --dco-restore` <*disk-pathname*>

```
# hdparm --dco-restore /dev/sdl

/dev/sdl:
Use of --dco-restore is VERY DANGEROUS.
You are trying to deliberately reset your drive configuration back to the factory
    defaults.
This may change the apparent capacity and feature set of the drive, making all data
    on it inaccessible.
You could lose *everything*.
Please supply the --yes-i-know-what-i-am-doing flag if you really want this.
Program aborted.
```

From: Bruce Nikkel, "*Practical Forensic Imaging: Securing Digital Evidence with Linux Tools*"

Detecting & Removing DCO

- Removing DCO (**actual** removal):
  `hdparm` **`--yes-i-know-what-i-am-doing`** `--dco-restore` <*disk-pathname*>

```
# hdparm --yes-i-know-what-i-am-doing --dco-restore /dev/sdl

/dev/sdl:
issuing DCO restore command
```

From: Bruce Nikkel, "*Practical Forensic Imaging: Securing Digital Evidence with Linux Tools*"

Detecting & Removing DCO

- Confirming the removal of DCO:
  `hdparm -I` <*disk-pathname*>

```
# hdparm -I /dev/sdl

/dev/sdl:

ATA device, with non-removable media
        Model Number:       WDC WD5003AZEX-00MK2A0
...
        LBA48  user addressable sectors:   976773168
        Logical  Sector size:                   512 bytes
        Physical Sector size:                  4096 bytes
        device size with M = 1024*1024:      476940 MBytes
        device size with M = 1000*1000:      500107 MBytes (500 GB)
...
```

From: Bruce Nikkel, *"Practical Forensic Imaging: Securing Digital Evidence with Linux Tools"*

Hidden Disk Areas: Image Acquisition

- Some imaging software* has the **capability to detect & remove** these hidden areas at acquisition time
- Yet, detection and removal of the HPA/DCO are commonly viewed as ***imaging preparation*** steps, not the actual imaging:
  - They're simply **disk sectors** protected by drive configuration parameters
  - Removing HPA/DCO modifies the **drive's configuration**, but does *not* modify its contents
  - No special technique to image these hidden areas **once they've been made accessible**
- Hence, **detect & remove HPA/DCO** in **your *preparatory steps*** to make them available for your subsequent imaging process

Note: For FTK Imager, see: https://www.dhs.gov/sites/default/files/publications/test_results_for_ftk_imager_version_4.3.0.18_with_coverjd1gd2.pdf

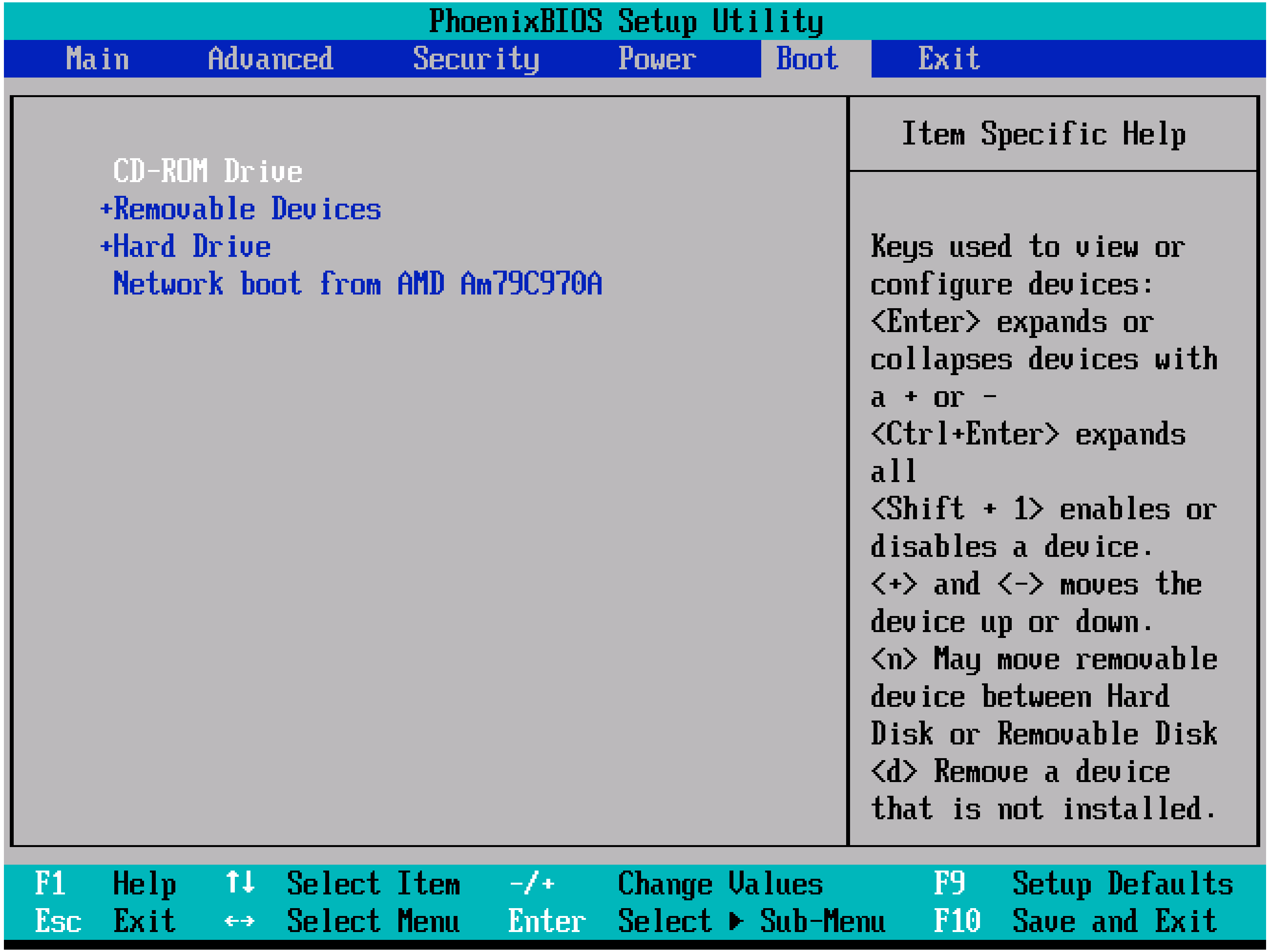
Hidden Disk Areas & Image Acquisition

- Caution: removing HPA/DCO is **very risky**!
- Below is the ***suggested workflow*** of a disk acquisition containing HPA/DCO
- First, do a **normal disk acquisition** *without* considering any HPA/DCO removal yet
- Then, **remove** HPA & DCO:
  - Do without a write blocker
  - Document all the steps and results!
- Lastly, do a **disk acquisition again** to capture HPA & DCO

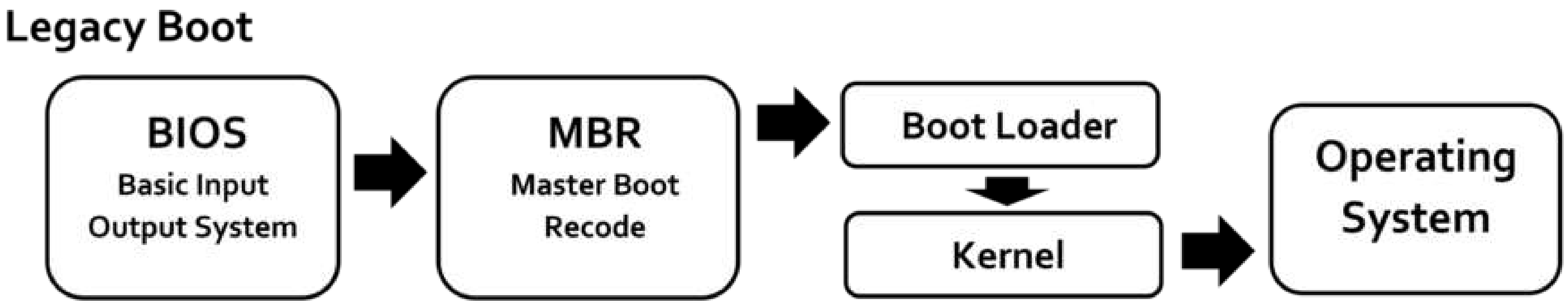
Boot Sequence

Boot Sequence

- **A software interface** between a computer's firmware and its OS
  - It is first run during the ***booting process***, and then it **loads the OS**
  - Two popular specifications: **BIOS** (older, outdated) and **UEFI** (newer)
- **Basic Input Output System (BIOS)**:
  - **Non-volatile firmware** used to perform **HW initialization** (Power-On Self-Test), to provide runtime services for OS and programs
  - Complementary Metal Oxide Semiconductor (**CMOS**): Non-volatile BIOS memory that **stores** time and date, hardware configuration info including **device-boot sequence**
  - Various **limitations**: 16-bit processor mode, 1MB addressable space, and PC AT hardware, etc.
  - See: https://en.wikipedia.org/wiki/BIOS

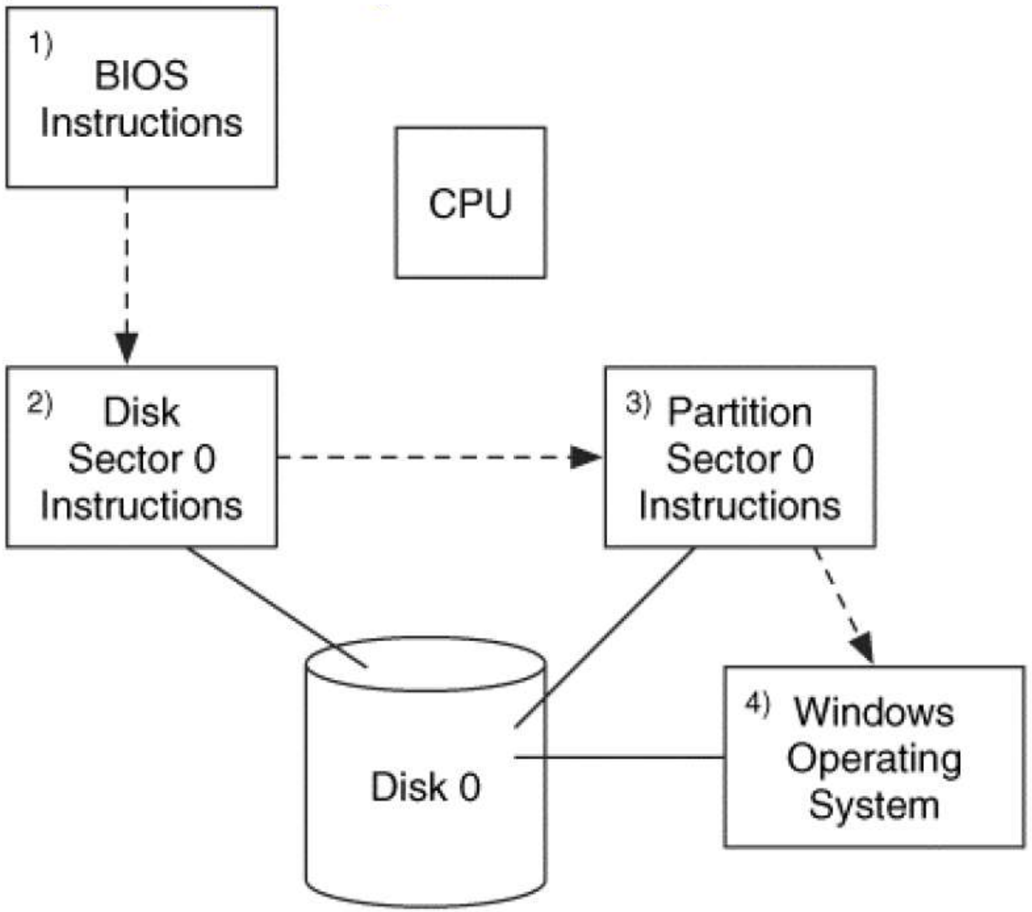

PhoenixBIOS Setup Utility
Main Advanced Security Power Boot Exit
CD-ROM Drive
+Removable Devices
+Hard Drive
Network boot from AMD Am79C970A
Item Specific Help
Keys used to view or configure devices:
<Enter> expands or collapses devices with a + or -
<Ctrl+Enter> expands all
<Shift + 1> enables or disables a device.
<+> and <-> moves the device up or down.
<n> May move removable device between Hard Disk or Removable Disk
<d> Remove a device that is not installed.
F1 Help ↑↓ Select Item -/+ Change Values F9 Setup Defaults
Esc Exit ↔ Select Menu Enter Select ▶ Sub-Menu F10 Save and Exit

Boot Sequence: BIOS



Source: Wikipedia

Sample Boot Sequence (Using BIOS & MBR)



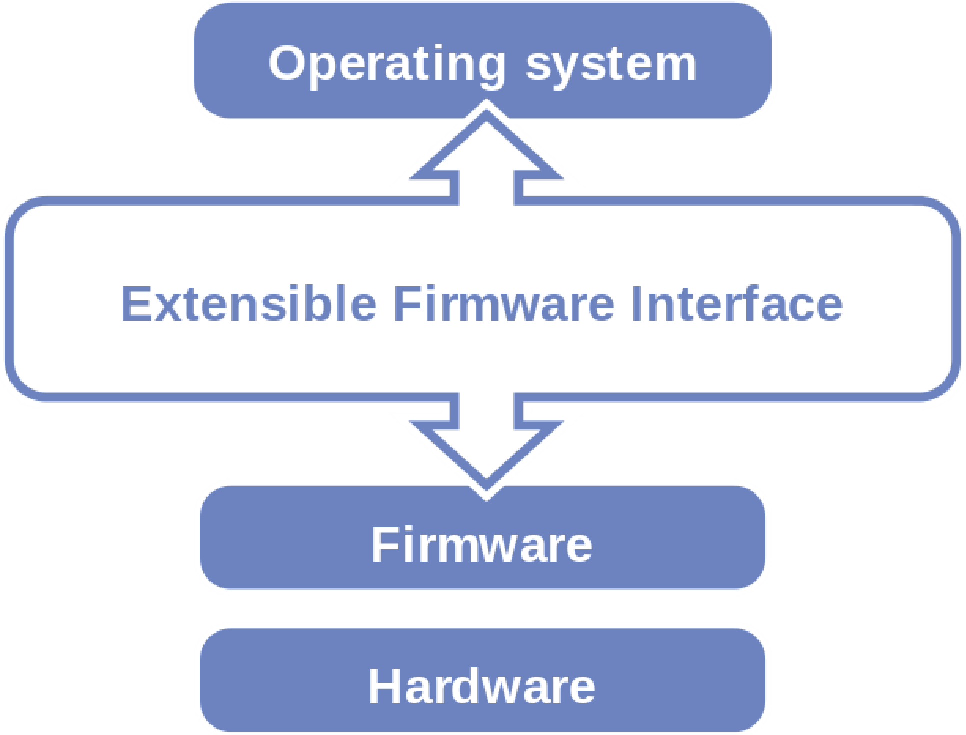
From: Brian Carrier, “File System Forensic Analysis”

Boot Code in Disk

- ***Boot code***: usually stored at **the start of a disk**, which contains code to start loading the OS (booting the OS)
- ***Master Boot Record* (*MBR*)**:
  - Stored in the ***first* 512-byte sector**
  - Contains the **initial *boot code*** (in *the first 446 bytes*), a partition table (including bootable status), signature value
- Boot loader, such as LILO, GRUB, can be installed on MBR: (http://www.src.wits.ac.za/groups/psi/linux/rhl-ig-x86-en-8.0/s1-upgrade-lilo.html)

Boot Sequence: UEFI

- **Unified Extensible Firmware Interface** (UEFI):
  - UEFI's **position** in the software stack



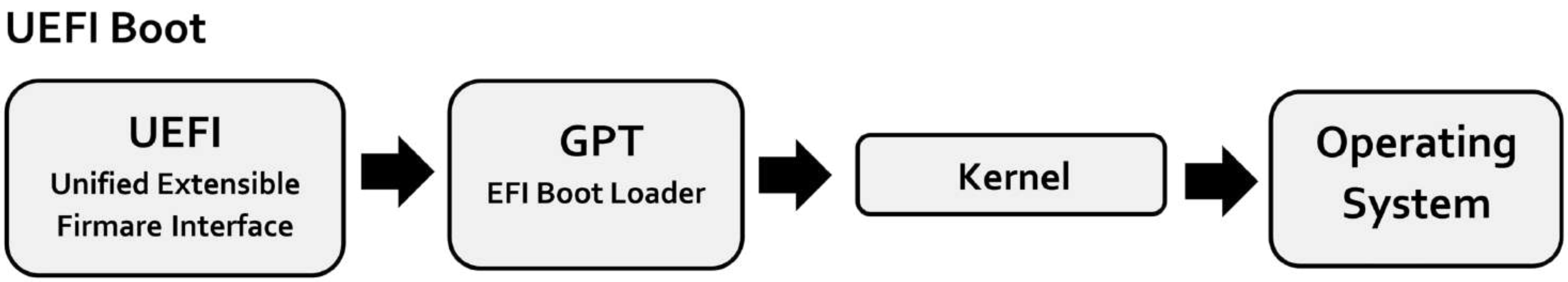
From: Wikipedia

Boot Sequence: UEFI

- Intel developed the **original EFI** specification in in the mid-1990s: some of the EFI's practices and data formats mirror those from Microsoft Windows
- In 2005, **UEFI** deprecated **EFI 1.10** (the final release of EFI)
- The ***Unified EFI Forum*** is the industry body that manages the UEFI specs
- Most UEFI implementations provides **legacy support for BIOS services**
- Various **new features** (*some are discussed more later*):
  can use **large disks (> 2 TB)** with a **GPT**, **CPU-independent** architecture & drivers, flexible pre-OS environment including network capability (for remote computer diagnostics & repair), backward & forward **compatibility**, etc.
- See: https://en.wikipedia.org/wiki/Unified_Extensible_Firmware_Interface

- BIOS vs UEFI: https://www.youtube.com/watch?v=SlzwMKcCoMI

Boot Sequence: UEFI

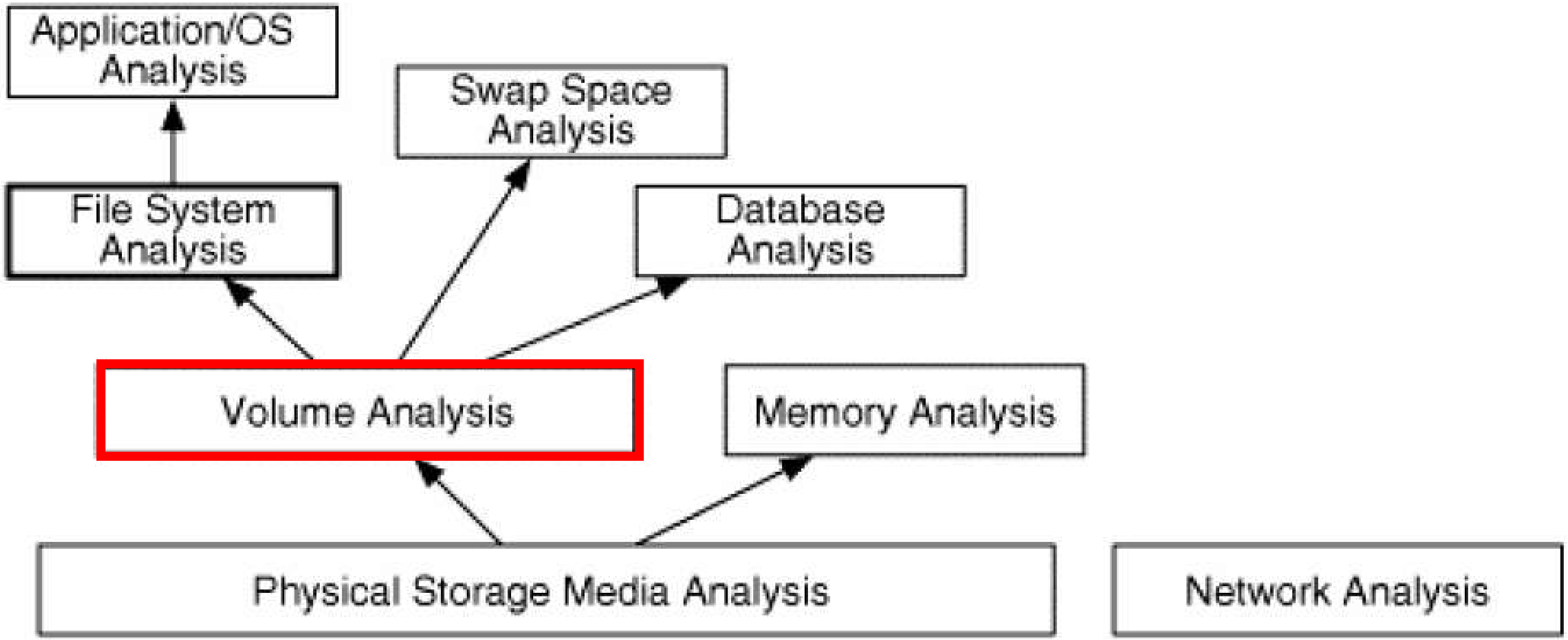


Source: Wikipedia

Break!

Volume Analysis

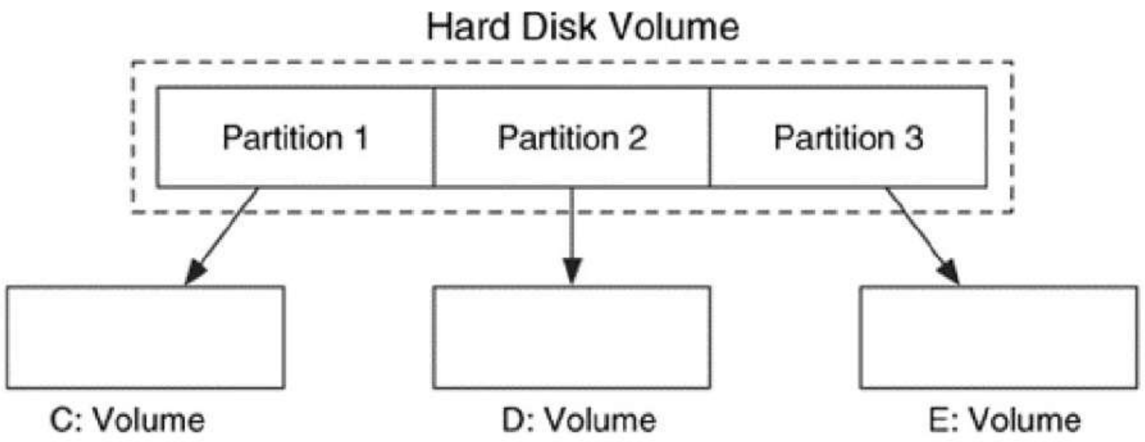
Layers of Disk & File Analysis



From: Brian Carrier, “File System Forensic Analysis”

Volume Analysis: Partition

- ***Partition***:
  - **Logical structure** that is overlaid on a **physical drive**
  - Contains **consecutive sectors** in **a** hard disk: *constrained within* that disk
- Partition is also a ***volume*** by definition*!* (*see the next few slides*)
- Example: a hard disk volume partitioned into 3 smaller partitions



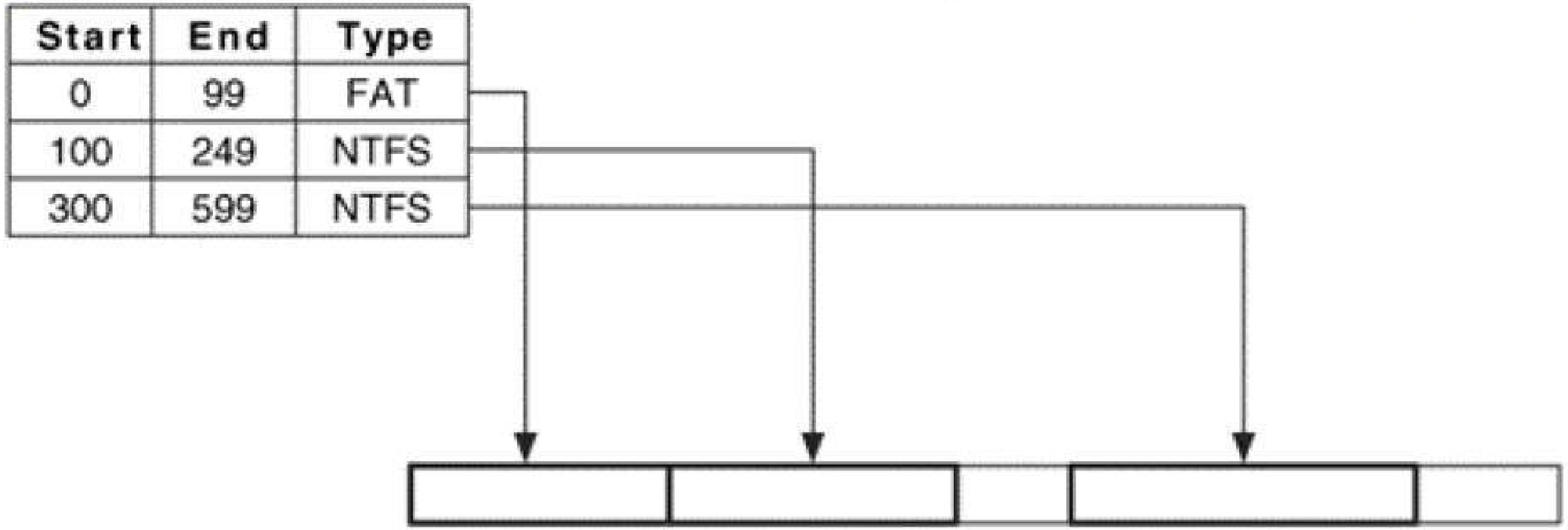
From: Brian Carrier, "File System Forensic Analysis"

Volume Analysis: Volume

- ***Volume***:
  - A collection of **addressable sectors** that an **OS or application can use** for data storage
  - Needs ***not*** be consecutive on a physical disk
- Note on terminology:
  - Partition = volume (taken by many people)
  - Partition ≠ volume (advocated by Brian Carrier):
    - Partition has all its sectors on *one* physical disk
    - Volume can span on *multiple* physical disks by means of multiple partitions
- In any case, **volume** refers to a logical disk structure that contains a ***file system***

Volume Analysis: Partition

- Usage of partitions: **different *file system types*** co-exist in a HDD
- Example:



Start	End	Type
0	99	FAT
100	249	NTFS
300	599	NTFS

From: Brian Carrier, “File System Forensic Analysis”

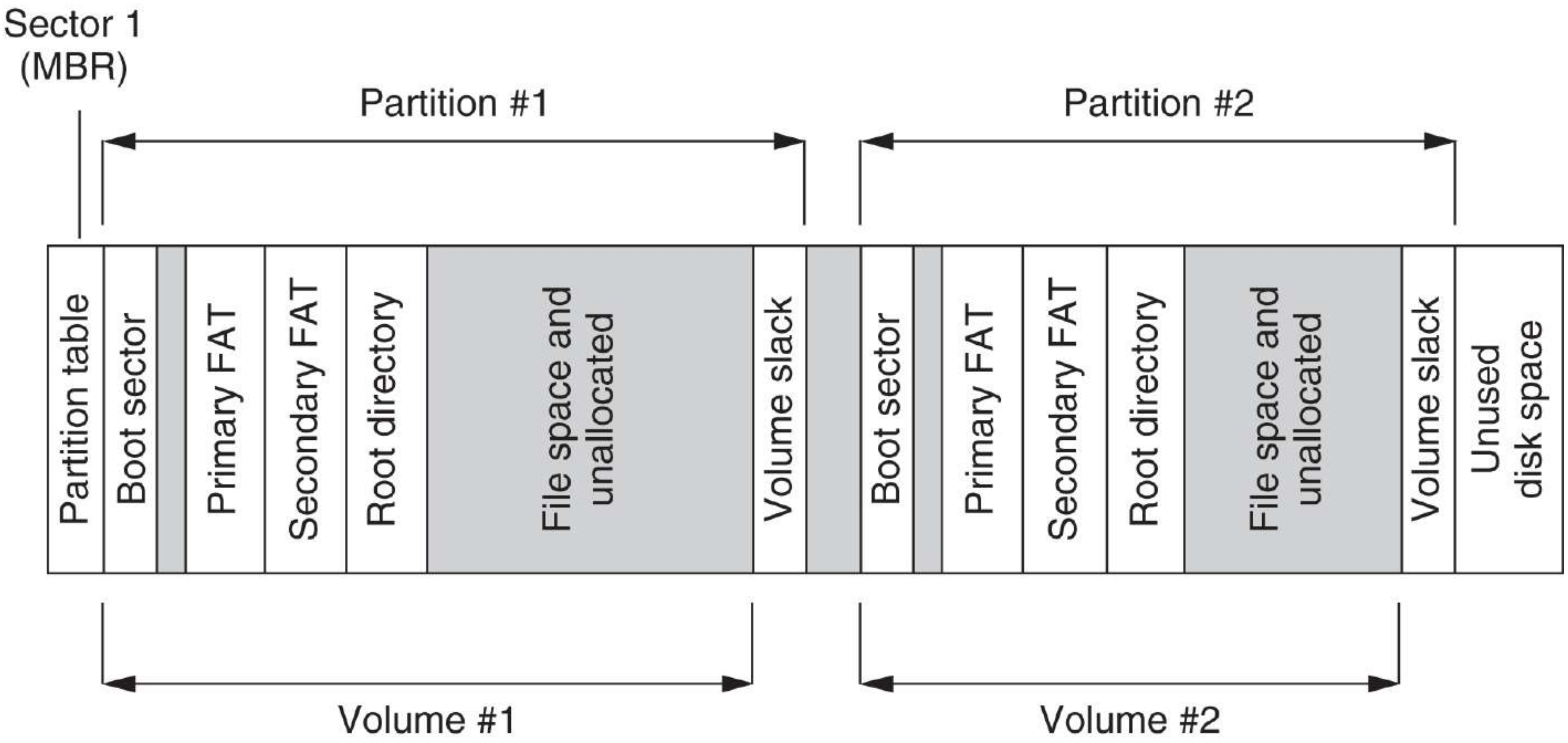
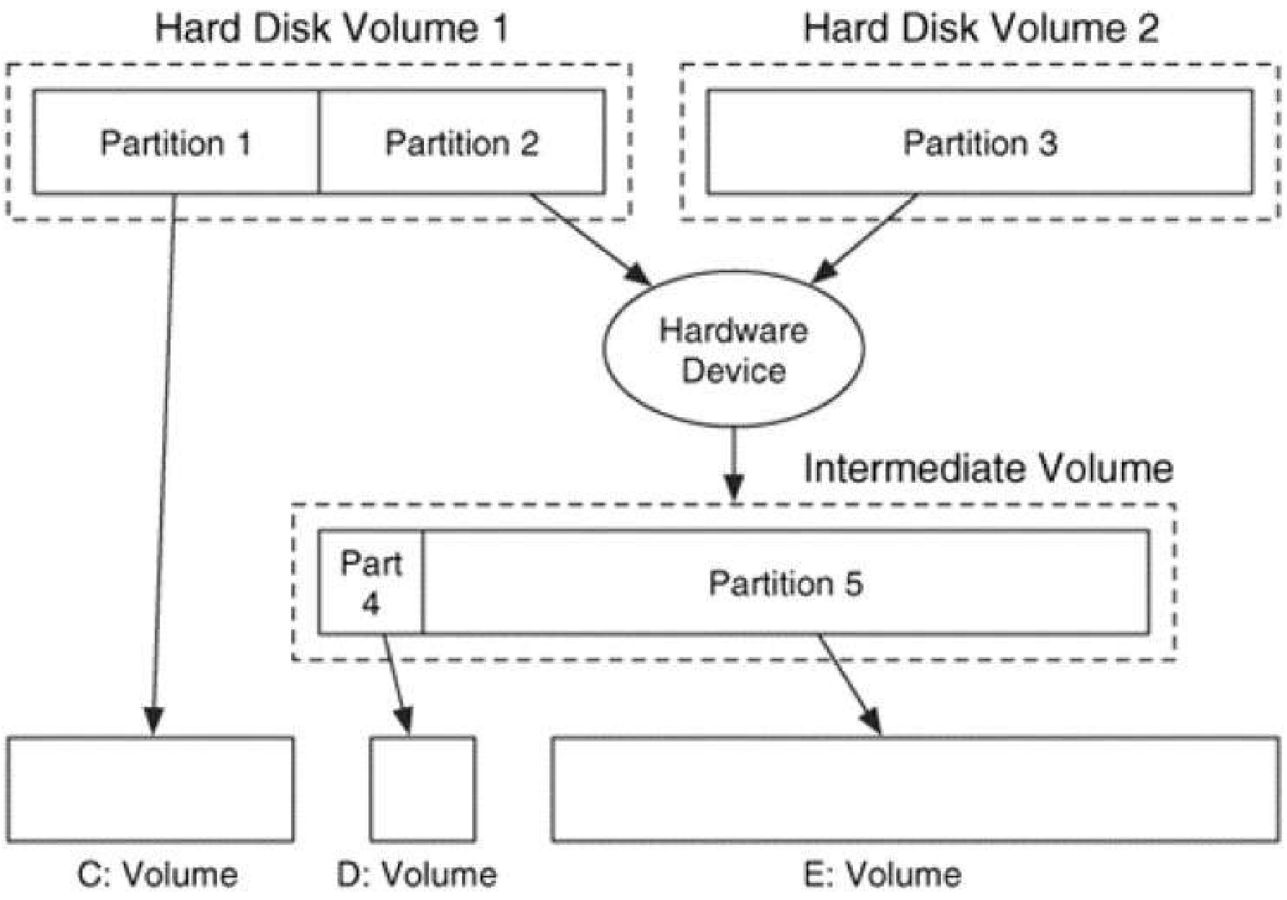


FIGURE 15.6 Simplified depiction of disk structure with two partitions, each containing a FAT formatted volume.

Partition vs Volume: An Example

- Example of ***volume assembly*** out of multiple partitions



From: Brian Carrier, "File System Forensic Analysis"

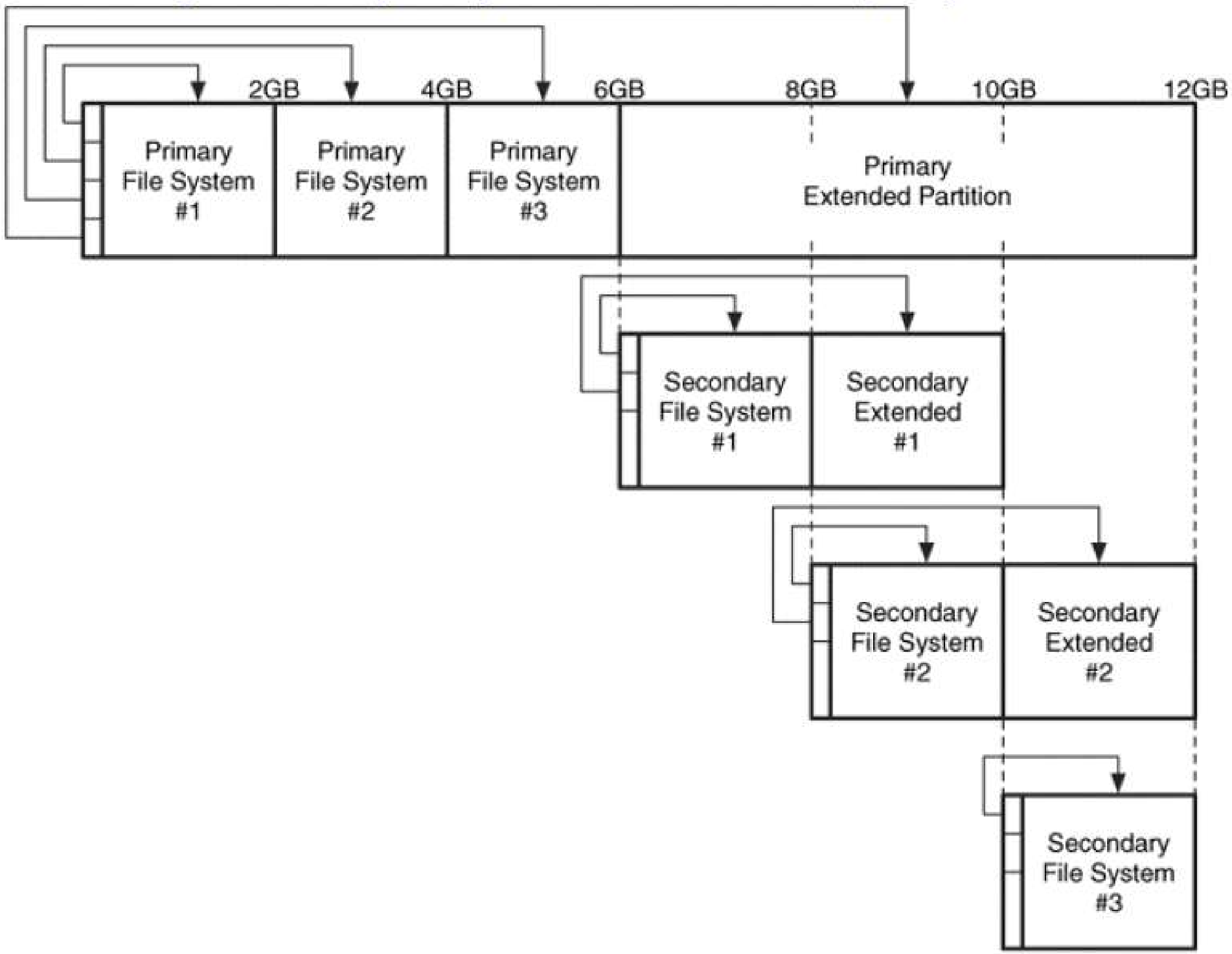
Disk Partitioning

- Disk **partitioning**:
  see https://en.wikipedia.org/wiki/Disk_partitioning,
  http://www.tldp.org/LDP/sag/html/partitions.html

- **Master Boot Record (MBR)**:
  - Also known as ***DOS-style partition*** `(-t dos` in TSK mm*)
  - Contains the **boot code** in the first 446 bytes of the first 512-byte sector
  - At most 4 **primary partitions**; or
    3 **primary partitions** + 1 **primary extended partition**
  - A ***primary*** partition: contains one file system
  - The ***extended*** partition: can be subdivided into
    multiple logical partitions

Disk Partitioning: MBR

- Example of MBR partitioning:



From: Brian Carrier, “File System Forensic Analysis”

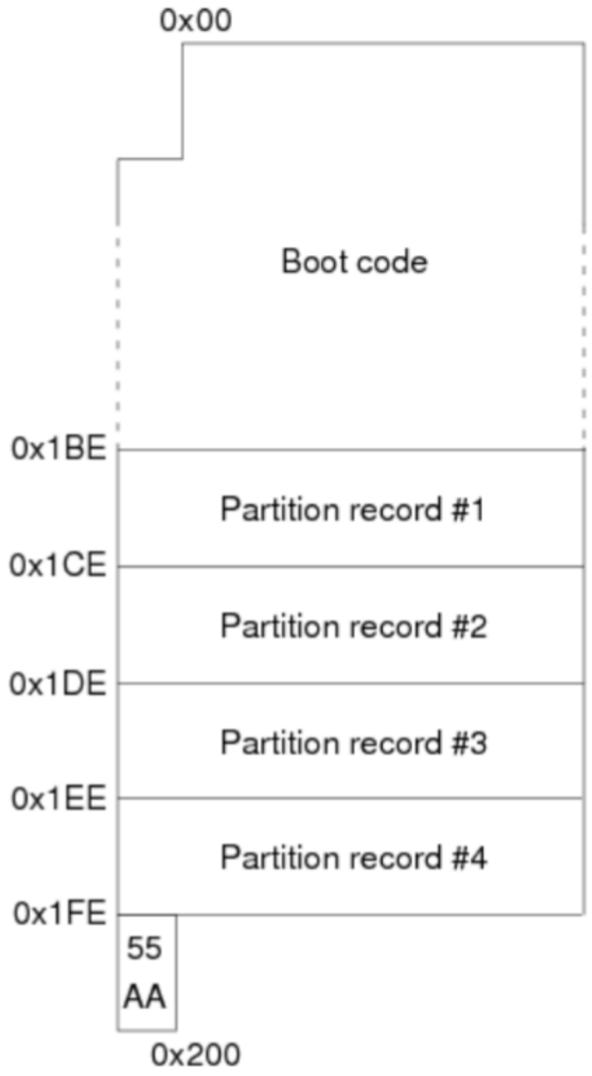
Disk Partitioning: MBR Partition Table

- Data structures for the MBR ***partition table***:

Byte Range	Description	Essential
0–445	Boot Code	No
446–461	Partition Table Entry #1 (see Table 5.2)	Yes
462–477	Partition Table Entry #2 (see Table 5.2)	Yes
478–493	Partition Table Entry #3 (see Table 5.2)	Yes
494–509	Partition Table Entry #4 (see Table 5.2)	Yes
510–511	Signature value (0xAA55)	No

From: Brian Carrier, "File System Forensic Analysis"

Disk Partitioning: MBR Partition Table



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

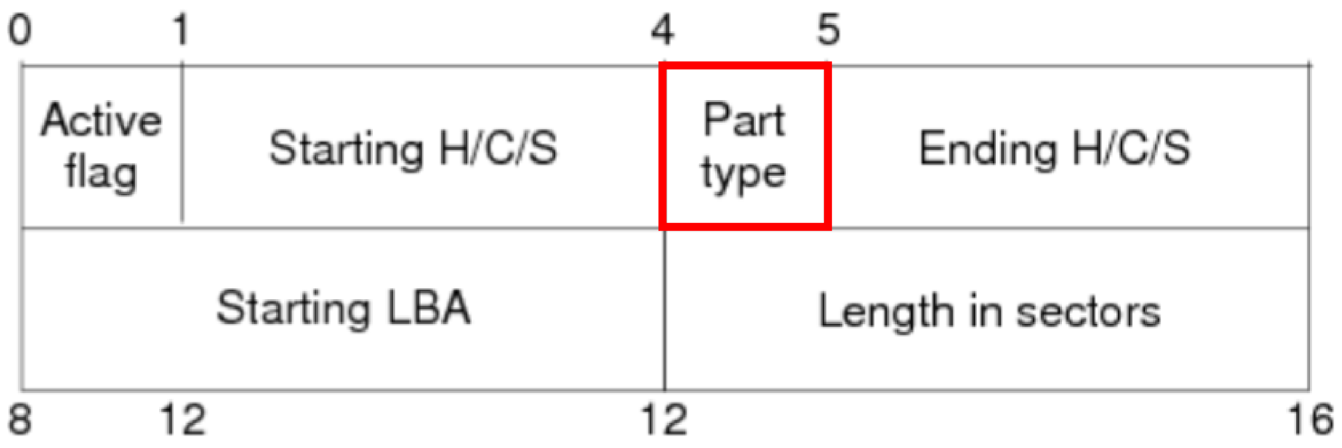
Disk Partitioning: MBR Partition Entry

- Data structures for **each *MBR partition entry***:

Byte Range	Description	Essential
0–0	Bootable Flag	No
1–3	Starting CHS Address	Yes
4–4	Partition Type (see Table 5.3)	No
5–7	Ending CHS Address	Yes
8–11	Starting LBA Address	Yes
12–15	Size in Sectors	Yes

From: Brian Carrier, “File System Forensic Analysis”

Disk Partitioning: MBR Partition Entry



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

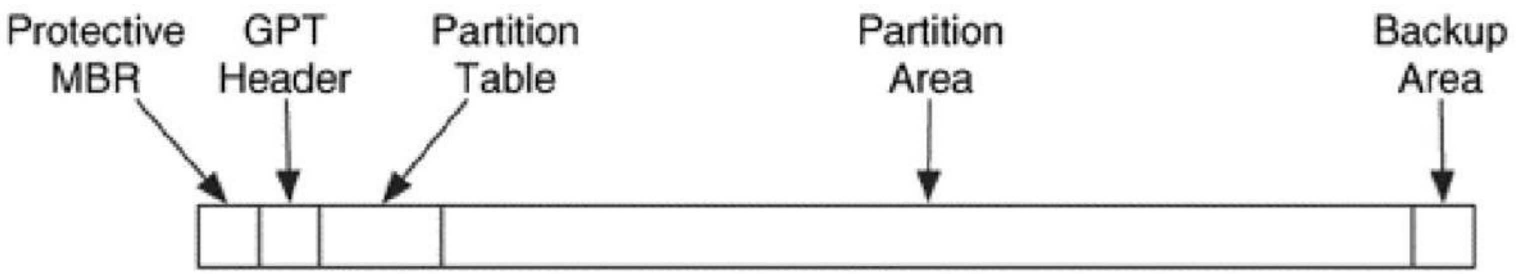
Disk Partitioning: MBR Partition Types

- Some ***partition types*** (see https://en.wikipedia.org/wiki/Partition_type):
  - 0x00: Empty
  - 0x01: FAT12, CHS
  - 0x04: FAT16, 16–32 MB, CHS
  - 0x05: Microsoft Extended, CHS → Extended partition with CHS addressing
  - 0x06: FAT16, 32 MB–2GB, CHS
  - **0x07**: **NTFS**
  - 0x0b: FAT32, CHS
  - 0x0c: FAT32, LBA
  - 0x0e: FAT16, 32 MB–2GB, LBA
  - 0x0f: Microsoft Extended, LBA → Extended partition with LBA addressing
  - **0x82**: **Linux Swap**
  - **0x83**: **Linux**
  - 0x85: Linux Extended
  - **0xee**: **EFI GPT Disk** → *to be discussed more*

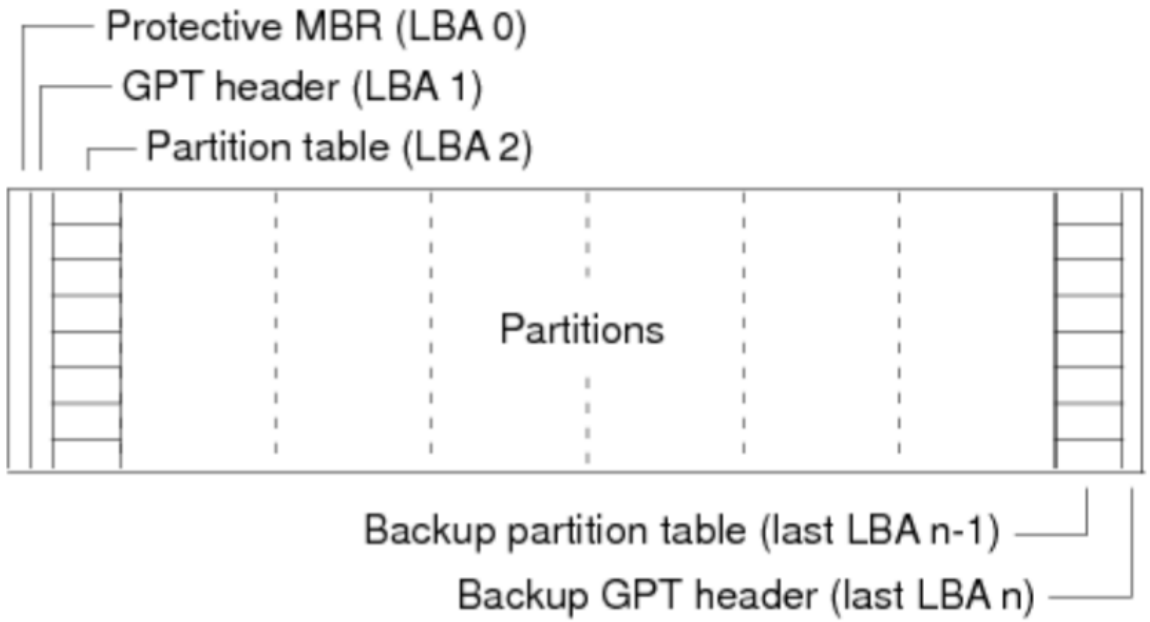
Disk Partitioning: GPT

- ***GUID Partition Table*** **(*GPT*)**:
  - Successor to MBR, for newer Intel systems
  - Is a part of the **UEFI standard**, but is also used on some BIOS systems
  - Uses ***Globally Unique ID*** **(*GUID*)/Universally Unique Identifier (UUID)**: a 128-bit random no used to identify information in computer systems
  - 64-bit **LBA** addresses: *no* more CHS addressing
  - *No* more primary, extended, or logical partitions: up to **128 primary partitions**
  - **Backup copies** of the important data structures are maintained in case of disk failure
  - TSK mm*: `-t gpt`

Disk Partitioning: GPT Layout



From: Brian Carrier, “File System Forensic Analysis”



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

Disk Partitioning: GPT Layout

- ***Protective MBR***:
  - In the **first sector** of the disk
  - Contains a **DOS/MBR partition table** with 1 partition entry of type **0xee**
  - Used to **prevent** legacy computers from formatting it
- Each ***partition table entry***:
  - **Contains** a GUID value, a starting and ending address, a name, attribute flags, and a type value (https://en.wikipedia.org/wiki/GUID_Partition_Table#Partition_type_GUIDs)
- For more on GPT: Bruce J. Nikkel, "*Forensic Analysis of GPT Disks and GUID Partition Tables*", https://www.digitalforensics.ch/nikkel09.pdf
- Others types of partitions (not discussed in our module): Apple (`-t mac`), BSD (`-t bsd`), Sun Solaris (`-t sun`)

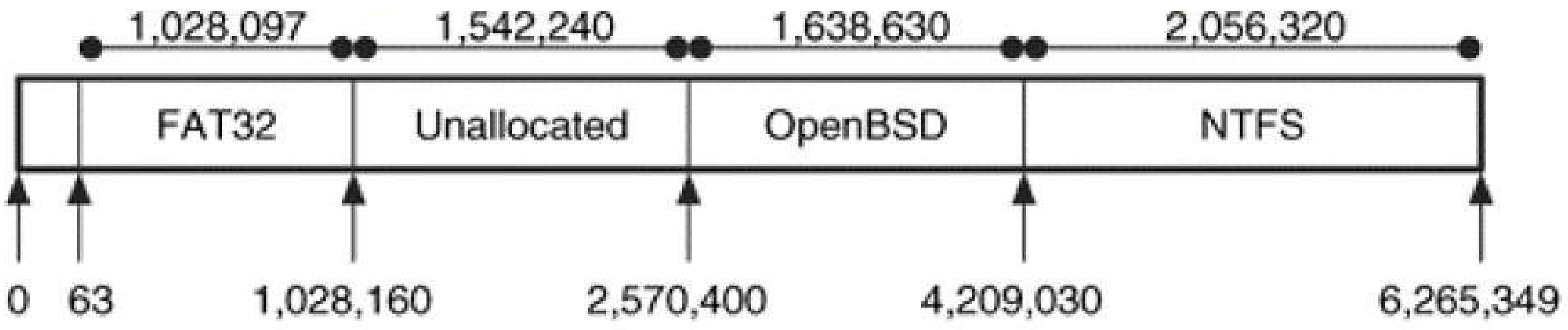
Extracting a Partition Content

- In Linux, just use the **dd** tool
- To extract a **partition data**, specify:
  - **`bs`**: the **block size** to read each time, 512 bytes is the default
  - **`skip`**: the **no of blocks to skip *before* reading**, each of size `bs`
  - **`count`**: the **no of blocks to copy** from the input to the output, each of size `bs`
- To know the content of a **partition table**, use the `mmls` command of TSK: *more on TSK later*

Extracting a Partition Content: Example

```
# mmls -t dos disk1.dd
Units are in 512-byte sectors
     Slot    Start        End          Length       Description
00:  -----   0000000000   0000000000   0000000001   Table #0
01:  -----   0000000001   0000000062   0000000062   Unallocated
02:  00:00   0000000063   0001028159   0001028097   Win95 FAT32 (0x0B)
03:  -----   0001028160   0002570399   0001542240   Unallocated
04:  00:03   0002570400   0004209029   0001638630   OpenBSD (0xA6)
05:  00:01   0004209030   0006265349   0002056320   NTFS (0x07)
```

From: Brian Carrier, "File System Forensic Analysis"



Extracting a Partition Content: Example

```
# dd if=disk1.dd of=part1.dd bs=512 skip=63 count=1028097
# dd if=disk1.dd of=part2.dd bs=512 skip=2570400 count=1638630
# dd if=disk1.dd of=part3.dd bs=512 skip=4209030 count=2056320
```

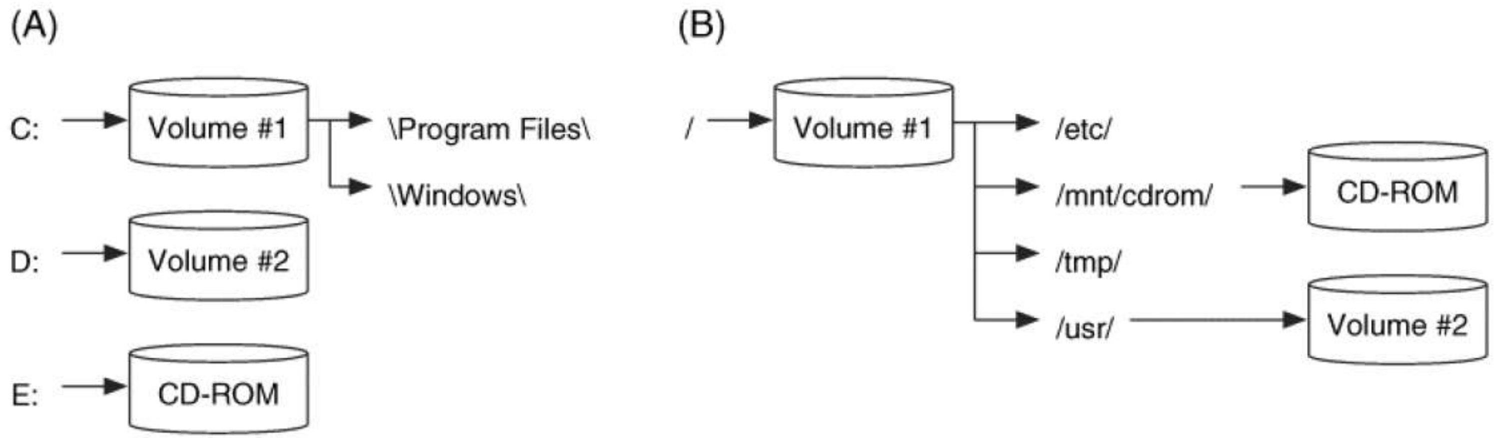
From: Brian Carrier, “File System Forensic Analysis”

Linux Disk & Partitions

- Storage device as a **file**: a well-known UNIX paradigm
- "***Universality of I/O***": memory and I/O devices are treated as file!
- **File-naming** of connected SCSI/SATA disks:
  - `/dev/sd`*xn*
  - *x* = a, b, c, … : (physical) drive letter
  - *n* = 1, 2, 3, … : partition number
- The file-naming: used by Linux commands as well as **forensic tools** that need to refer to disks

Partition Mounting

- ***Mounting***: make a partition **visible** to the OS
- ***Un-mounting***: make it **invisible** to the OS
- Linux:
  - Accessible via a ***mount point***: a path from /
  - Steps:
    - `mkdir` *<mount-point>*
    - `mount -t vfat` *<partition>* *<mount-point>*
- Difference between **Windows** (partition appears as a **logical drive**) and Linux:



From: Brian Carrier, “File System Forensic Analysis”

Linux Partition Management

- Managing Linux **disk partitions** using `fdisk` (fixed disk or format disk):
  - View, create, resize, delete, change, copy and move partitions
  - A root privilege is required: use sudo in Ubuntu
  - Various commands (see the next slide)
- **Format a partition** using `mkfs*`, such as:
  - Generic `mkfs`, which would call `mke2fs`:
    `# mkfs -t ext4 /dev/sda4`
  - Filesystem-specific commands that `mkfs` calls:
    `# mkfs.ext4 /dev/sda4`
- `mkfs` is similar to `format` in DOS/Windows:
  `C:> format D: /FS:ntfs`

Linux Partition Management Tool: `fdisk`

- CLI-based tool
- Can be used interactively
- Some command options:
  - `a` toggle a bootable flag
  - **`d`** delete a partition
  - **`l`** list known partition types
  - `m` print this menu
  - **`n`** add a new partition
  - **`p`** print the partition table
  - `q` quit without saving changes
  - `v` verify the partition table
  - `w` write table to disk and exit
  - `x` extra functionality (experts only)

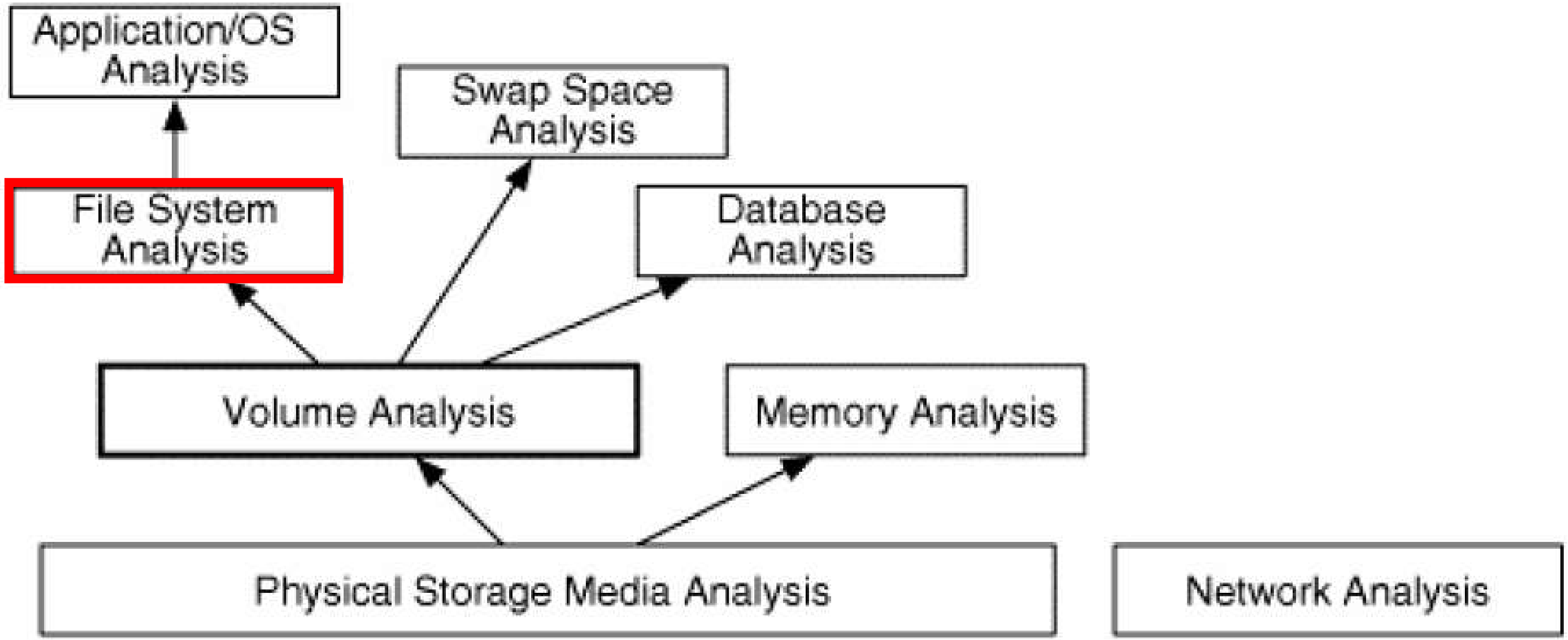
Linux Partition Management Tool: `Gparted`

- Another popular alternative tool for managing disk partitions
- **Gparted** (Gnome-based partition manager)
  - GUI-based
  - Live CD
  - “Easier” to use

From: https://gparted.org/

File System Analysis

Layers of Disk & File Analysis



From: Brian Carrier, “File System Forensic Analysis”

File System

- ***File system (FS)*** definitions:
  - An abstraction overlaid on a **volume**, which keeps track of **file system objects** (e.g. files, folders, inodes, …)
  - A collection of ***named files*** on a disk volume, including their organization + API + usage
  - A "file record-keeping"
- A file system also has **metadata** for file management:
  - It determines how to access the file on disk, i.e. file table of contents (TOC)
  - Generally, it is invisible to the users

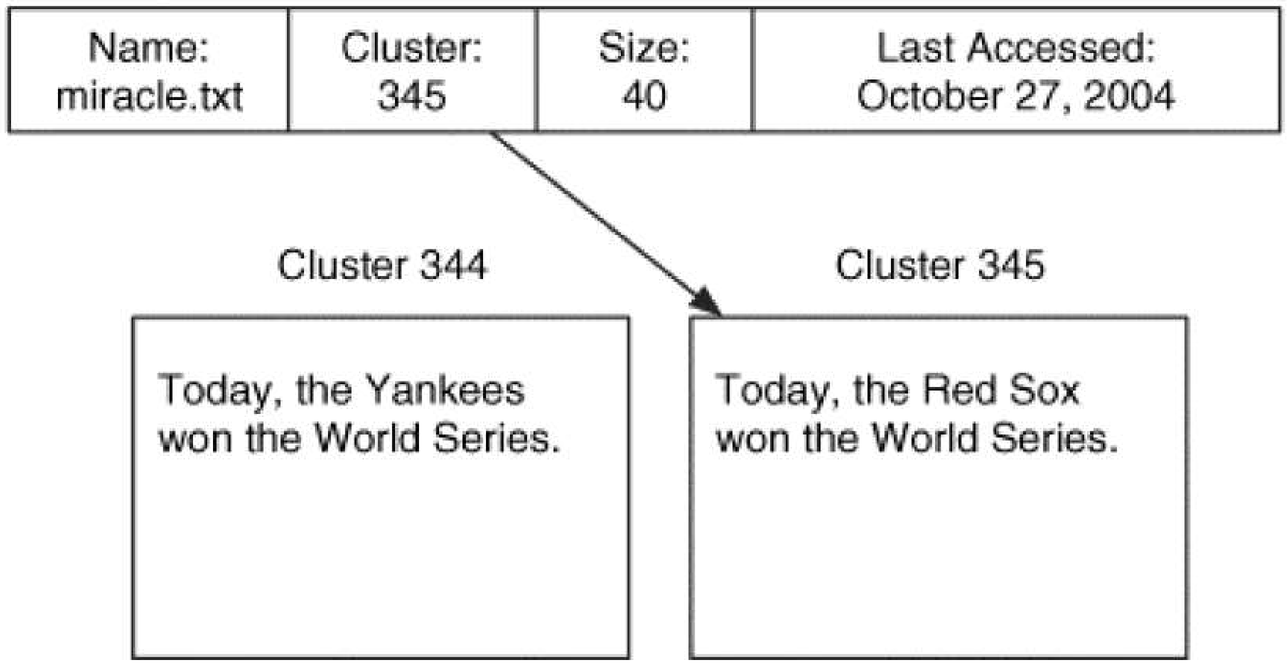
File System Data

- ***Essential*** file system data:
  - Data items that are **needed** to save and retrieve files
  - **Examples**: the addresses where the file content is stored, the name of a file, and the pointer from a name to a metadata structure
- ***Non-essential*** file system data:
  - Data items that are there ***for convenience*** but not needed **for the basic functionality** of saving & retrieving files
  - **Examples**: access times and permissions
- Why do we need to differentiate them?
  - We ***have* to trust** the essential data, but we ***do not* have to trust** the non-essential data, e.g. recorded access times

File System: File

- A ***file*** contains:
  - **Data**: its data content
  - **Metadata**: additional information about the file, which include:
    - **File name**: a way of referring to file, e.g. `secret.docx`
    - **File type**: special files (directories, devices, pipes, executables), application type which is sometimes encoded in file extension (i.e. DOS/Windows)
    - **Other information**: **ownership**, **dates** (creation, access, modification), **protection** (access-control attribute), **size**

File Data & Metadata: Illustration



From: Brian Carrier, "File System Forensic Analysis"

File Metadata: MAC+BD Times

- One important file metadata:
  - Last modification, access, change, creation/birth, and deletion times
- **MAC+BD times**:
  - Ref: https://en.wikipedia.org/wiki/MAC_times
  - *To be discussed in the next **2 lectures** (for Windows and UNIX/Linux)!*
- Caveats on **metadata reliability**:
  - We can only see the current state of a file system
  - We cannot inspect the operations which lead to it!
  - Adversary can **manipulate** metadata arbitrarily:
    do not need to use OS, and can use raw device!

Disk & File Forensic Tool: The Sleuth Kit (TSK)

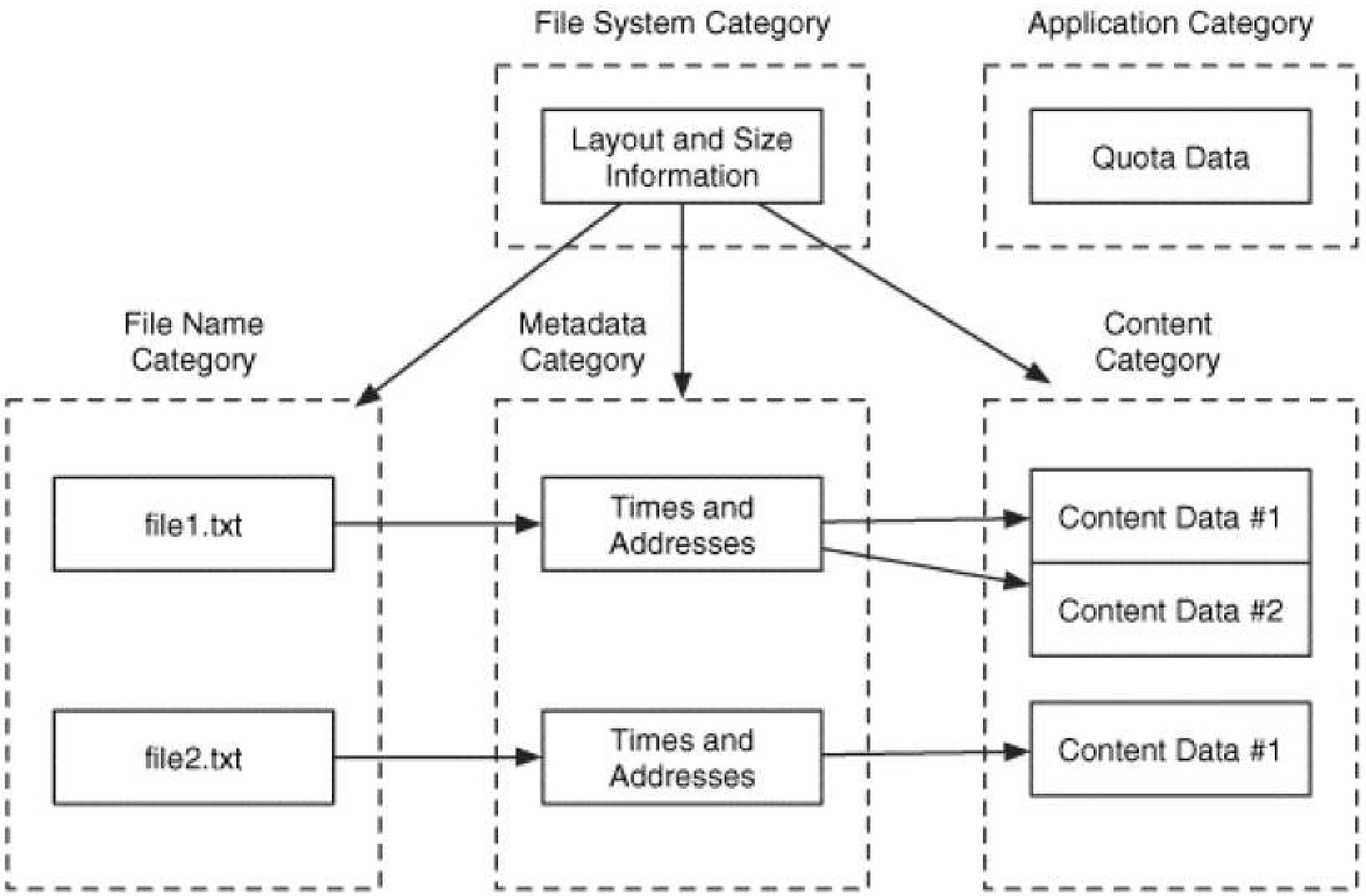
- A CLI-based file-system forensic tools developed by Brian Carrier
- Based on the Coroner's Toolkit (TCT)
- Tool names' **prefix** for **layer/category**:
  - `disk_*`: **disk**
  - `img_*`: **image** file
  - `mm*`: **volume** (media management) system

  - `fs*`: **file system**
  - `f*`: **file name**
  - `i*`: **metadata** (inode)
  - `blk*/d*`: **content** (data)

4 categories of "file system management" (*see the next slide*)

File System Management Categories

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



From: Brian Carrier, “File System Forensic Analysis”

File System Management Categories

- ***File system*** category:
  - Contains the **general file system information**,
    e.g. the layout, allocation structures, and boot blocks
  - TSK's **`fsstat:`** displays the file system details and statistics
- ***File name*** (human interface) category:
  - Contains the data that assign a **name** to each file
  - Located in the **contents of a directory**:
    a **list of file names** with the **corresponding metadata address**
  - TSK's **`fls`**: lists the allocated and deleted file names
  - TSK's **`ffind`**: finds allocated and unallocated file names
    that point to a given meta data structure (inode)

File System Management Categories

- ***Metadata* (*inode*)** category:
  - Contains the data that **describe a file**
  - Examples: FAT directory entries, NTFS Master File Table (MFT) entries, and UFS/Ext3 inode structures
  - **`istat`**: displays the statistics & details about a given metadata data structure
  - **`ils`**: lists the metadata structures and their contents
  - **`icat`**: extracts the data units of a file, which is specified by its meta data address (instead of the file name):
    - `-s` flag: the slack space is shown
    - `-r` flag: the `icat` attempts to recover deleted files
  - **`ifind`**: finds the **metadata structure** that has a given file name pointing to it or the metadata structure that points to a given data unit

File System Management Categories

- ***Content* (*data unit*)** category:
  - Contains the data that comprise **the actual content** of a file
  - Generic terms: ***data unit*** instead of ***data cluster*** or ***data block***
  - Different unit allocation strategies: first available, next available, best fit
  - Data unit **allocation status**: e.g. using bitmap
  - **`blkstat`**: displays the statistics about a given data unit
  - **`blkls`**: lists the details about data units, and can extract the unallocated space of the file system
  - **`blkcat`**: extracts the contents of a given data unit
  - **`blkcalc`**: calculates where data in the unallocated space image exists in the original image

File System Management Categories

- ***Application* category**:
  - Contains data that provide **special features**:
    not needed during the process of reading or writing a file,
    but it may be more efficient if implemented inside the file system
  - Examples: user quota statistics, file system journals

- Common tool names' **suffix**:
  - *`stat`: display statistics
  - *`ls`: list the content
  - *`cat`: dump/extract the content

Other TSK Commands

- **Disk** tools: to detect and remove an HPA
  - **`disk_stat`**: shows if an HPA exists
- **Image** file tools: to check the image file format
  - **`img_stat`**: shows the details of the image format
  - **`img_cat`**: shows the raw contents of an image file
- **Volume** (media management) tools:
  to analyze a disk's partition structures
  - **`mmstat`**: display details about a volume system (typically only the type)
  - **`mmls`**: displays the layout of a disk, including the unallocated spaces
  - **`mmcat`**: extracts the contents of a specific volume to STDOUT

Other TSK Commands

- **Miscellaneous** tools: which transcends the layer methodology
  - **`hfind`**: uses a binary sort algorithm to lookup hashes
  - **`mactime`**: takes input from the `fls` and `ils` to create a timeline of file activity
  - **`sorter`**: sorts files based on their file type and performs extension checking and hash database lookups
  - **`sigfind`**: searches for a binary value at a given offset

Disk & File Forensic Tool: The Sleuth Kit (TSK)

- Use **TSK commands** to analyse a disk image file:
  - Lab 4, Task 1
  - Assignment 1
- References:
  - https://wiki.sleuthkit.org/index.php?title=TSK_Tool_Overview
  - http://wiki.sleuthkit.org/index.php?title=FS_Analysis
  - Chris Marko, "*Introduction to The Sleuth Kit (TSK)*": on LumiNUS
- Videos:
  - https://www.youtube.com/watch?v=htAQ7EWeyv8,
  - https://www.youtube.com/watch?v=a4lSVOT4PeU,
  - https://www.youtube.com/watch?v=VQ1ni-jlbwE

File System: Types on Different OSes

- File systems on **Windows**: *discussed in Week 5*
  - File Allocation Table (**FAT**): e.g.: FAT, FAT16, VFAT, FAT32, FAT64
  - New Technology File System (**NTFS**):
    use Master File Table (MFT) to store file database
- File systems on **UNIX/Linux**: *discussed in Week 6*
  - Unix file system (**UFS**), also called the Berkeley Fast File System, the BSD Fast File System or FFS
  - Extended file system (**ext***)
  - Others: ReiserFS, ZFS, The Journaled File System (JFS) for Linux
- File systems on **Mac**: HFS, HFS+, Apple File System (APFS)

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

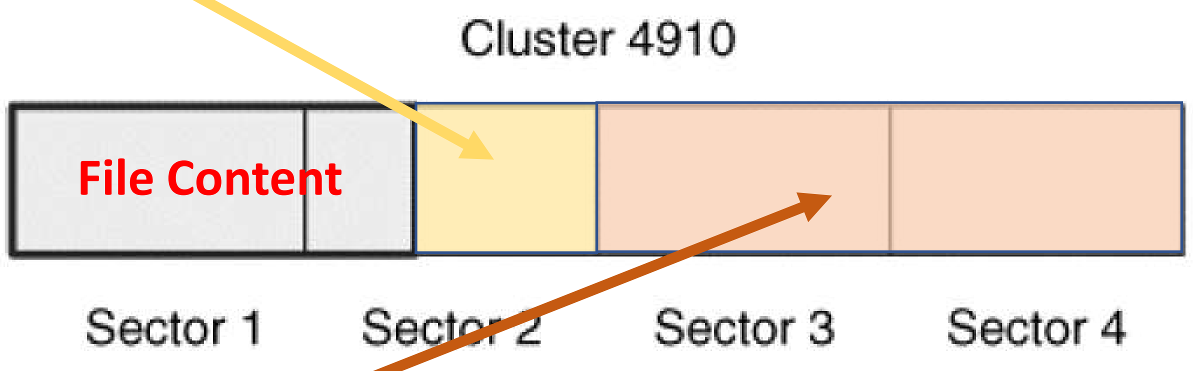
From: Brian Carrier, “File System Forensic Analysis”

File Data Allocation: Sectors vs Clusters

- ***Sector***:
  - Smallest ***physical*** unit of a file storage as seen by **a drive**, usually 512B
- ***Cluster***:
  - Smallest ***logical*** unit of a file storage as seen by the **OS**
  - Contains **1 or more** sectors: to minimise read/write overheads
- ***"File slack" space***:
  - Wasted space in a **file**
  - Smaller cluster size gives less slack spaces
  - **OSes/computers are "lazy"**:
    some do ***not*** wipe the unused bytes allocated to a file
  - The slack space contains ***residual data* from previous files**

File Slack Space: Example

- **Two types** of file slack space:
  - **First area**: located in between the end of a file and the end of the sector in which the file ends, usually filled by **OSes with zeros**



From: Brian Carrier, "File System Forensic Analysis"

  - **Second area**: located in the sectors that contain no file content in the allocated cluster, **can be ignored** by OSes

What Happened with Deleted Files?

- Typical file deletion **process**:
  - The file is typically ***not*** erased from the media
  - Instead, the information in the directory data structure that points to the location of the file is **marked as deleted**

- *What are the **implications**?*
  - This means that the ***file is still stored*** on the media, but is it no longer enumerated by the OS
  - The OS considers this to be **free space**, and can overwrite any portion of or the entire deleted file at any time

- ***Question**: can we recover/carve a deleted file?*

Data-Unit Wiping Technique

- ***Secure deletion* tools**: write **zeros** or **random data** to the data units that a file allocated or to all unused data units
- Some popular **tools**:
  - **Unix**: shred, wipe, secure-delete toolkit (srm, sfill), sswap, sdmem, dd (see https://www.tecmint.com/permanently-and-securely-delete-files-directories-linux/)
  - **Windows**: various available tools (see https://www.groovypost.com/howto/7-free-ways-securely-delete-files-windows/)
- If secure delete tools are used by users:
  - A slack space analysis ***won't*** give you good findings

SSDs: Revisited

- **Challenges** to acquisition were discussed last week
- ***Flash Translation Layer*** (***FTL***) in SSDs:
  - Takes read and write **requests on *logical blocks***
  - Turns them into **low-level commands** on the ***underlying physical blocks*** and **physical pages** (that comprise the actual flash device)
  - **Extra** disk capacity ("*spare pages*") invloved: *not* addressable by the OS
- References:
  - Bell and Boddington, "*Solid State Drives: The Beginning of the End for Current Practice in Digital Forensic Recovery?*", Journal of Digital Forensics, Security and Law, 2010

SSD's Flash Translation Layer

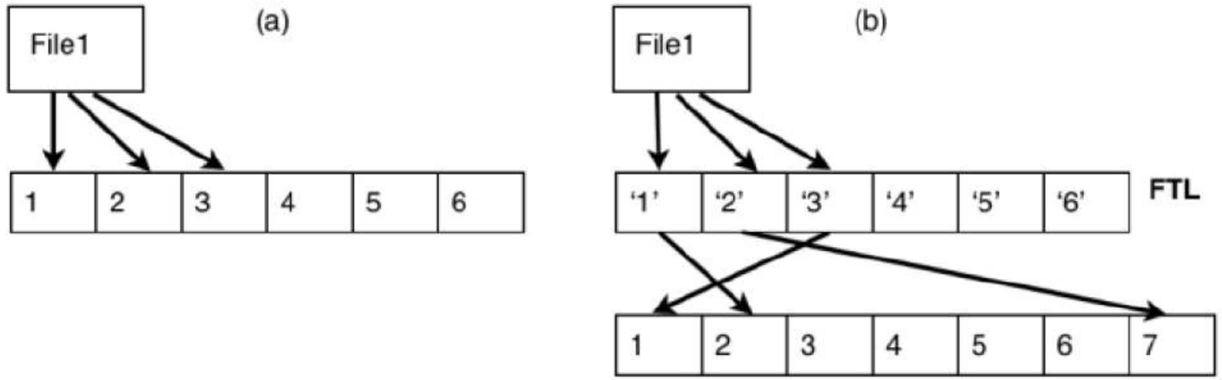
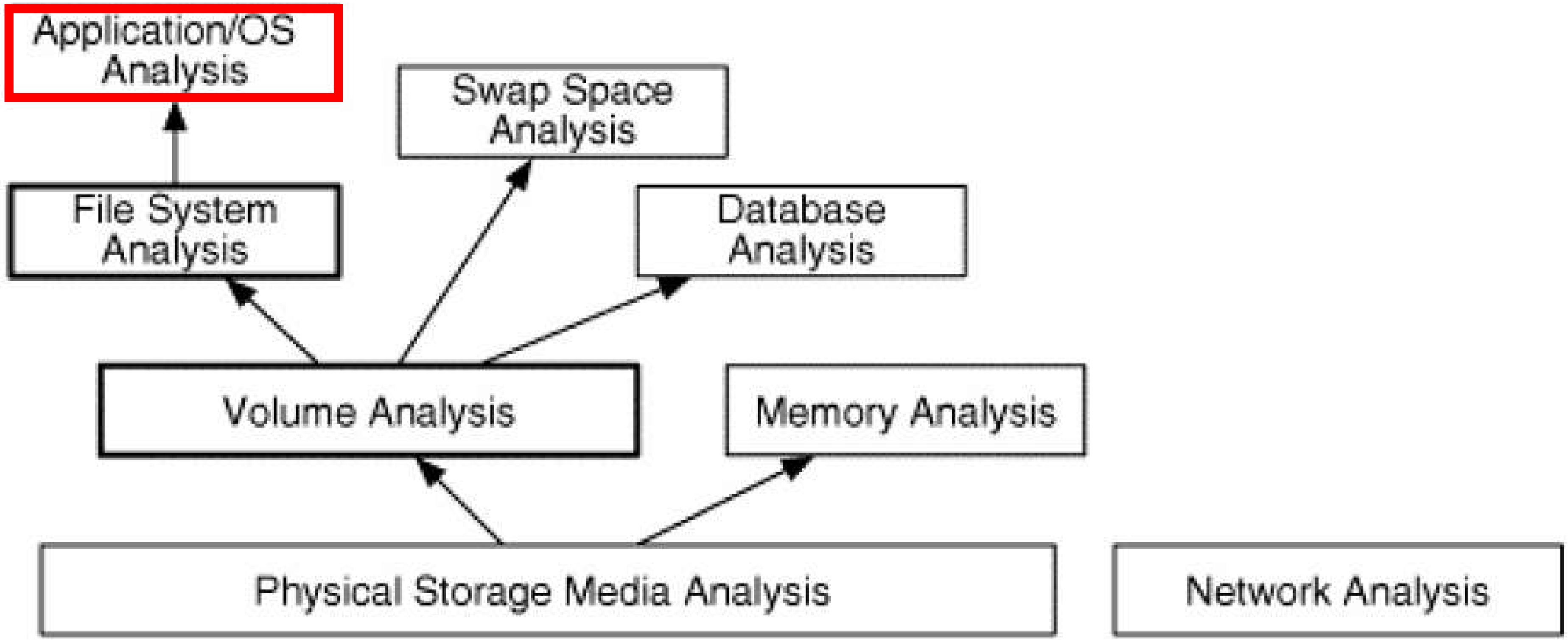


Figure 1: File1 is using blocks 1, 2, and 3 on the drive. (a) In the case of the hard drive, these blocks are generally used directly. (b) In an SSD, the FTL masks the real arrangement of data and hides the behaviour of the drive.

Bell and Boddington, "*Solid State Drives: The Beginning of the End for Current Practice in Digital Forensic Recovery?*", Journal of Digital Forensics, Security and Law, 2010

Application File Analysis

Layers of Disk & File Analysis



From: Brian Carrier, “File System Forensic Analysis”

File Signature Analysis

- ***File signature*** **(*magic number*)**:
  unique value at the first few bytes of a file that identifies the **content/type** of a file
- File signature **tables**:
  - https://en.wikipedia.org/wiki/List_of_file_signatures
  - https://www.garykessler.net/library/file_sigs.html
- Linux `file` command: determines the file type of a file
- File **signature** vs file **extension**:
  - Sometimes, a file's extension (docx, xlsx, dll, …) has been **changed** so that it can't be properly opened using the associated applications
  - A **hex editor** can be used to inspect the file signature, and help us correct the extension: Lab 4 Task 2

File Signature Analysis: Example

- Lab 4, Task 2: “SecretFile.docx”
  - File with extension missmatch:

- Also Lab 4, Tasks 4-B and 4-C (using Autopsy)

Application Metadata Analysis

- ***Metadata***: data about data
  - File metadata: from the file system (discussed earlie)
  - ***Application-level* metadata**:
    dates/times, creator, location information in pictures, …
- Possible files containing application metadata:
  - Microsoft Office files
  - Image files
  - …
- Sometime can also identify the computer/device used
- A valuable **source** of forensic evidence: *more later*!

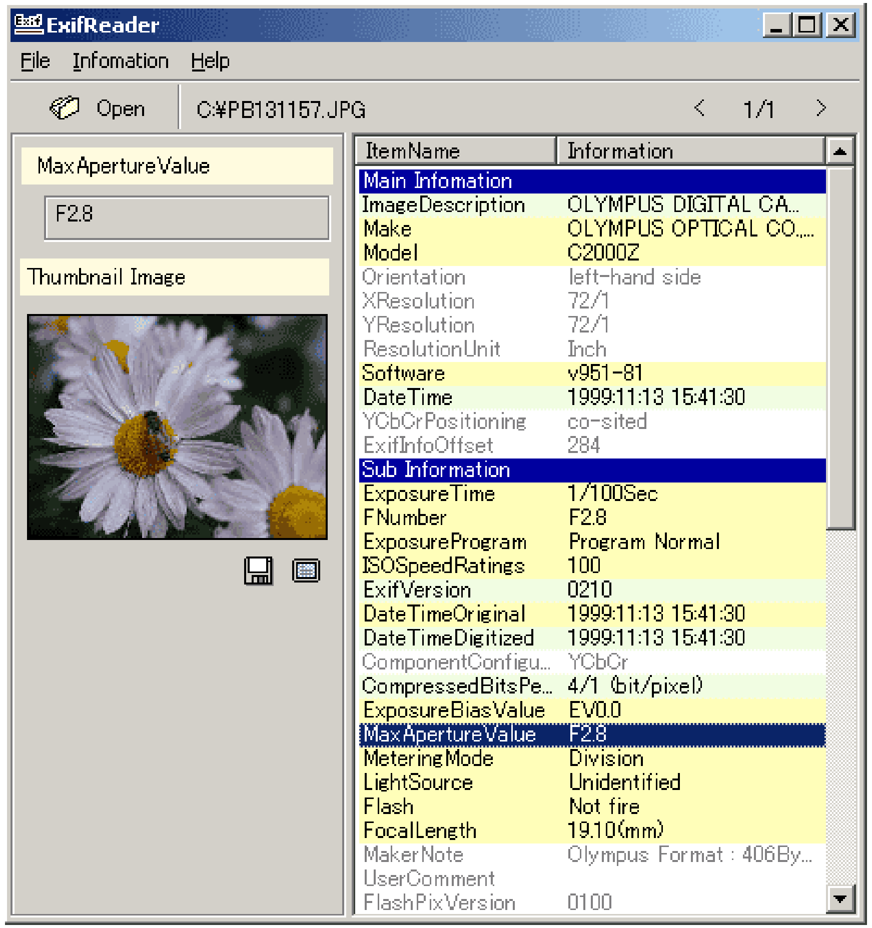
Application Metadata Analysis: Example

- **Microsoft Office** files: Lab 4, Task 3-A, Task 4-F (Autopsy)
- Access a file as **a zip file**!


```
<?xml version="1.0" encoding="UTF-8" standalone="true"?>
- <cp:coreProperties xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:dcmitype="http://purl.org/dc/dcmitype/"
  xmlns:dcterms="http://purl.org/dc/terms/" xmlns:dc="http://purl.org/dc/elements/1.1/"
  xmlns:cp="http://schemas.openxmlformats.org/package/2006/metadata/core-properties">
      <dc:title/>
      <dc:subject/>
      <dc:creator>Rio-Home</dc:creator>
      <cp:keywords/>
      <dc:description/>
      <cp:lastModifiedBy>Rio-Home</cp:lastModifiedBy>
      <cp:revision>1</cp:revision>
      <dcterms:created xsi:type="dcterms:W3CDTF">2018-03-14T13:55:00Z</dcterms:created>
      <dcterms:modified xsi:type="dcterms:W3CDTF">2018-03-14T13:56:00Z</dcterms:modified>
  </cp:coreProperties>
```

Application Metadata Analysis: Example

- Images files and **Exif data**: Lab 4, Task 3-B, Task 4-E (Autopsy)



Exif

- **Exif (Exchangeable image file format)**
  - A standard specifying the formats for images, sound, and ancillary tags used by digital cameras (including smartphones), scanners, …
- **Exif file format**:
  - The same as JPEG file format, but it inserts some of **image/digicam information data** and **thumbnail** image to JPEG
  - See https://www.media.mit.edu/pia/Research/deepview/exif.html
- **Importance** to forensics:
  - **Date and time** information, camera settings, thumbnail, **geolocation** information (from GPS-enabled camera),
  - A sample **case**: a photo of John McAfee with a reporter taken with a phone that had **geotagged** the image → *he was captured two days later*!
- Reference: https://en.wikipedia.org/wiki/Exif

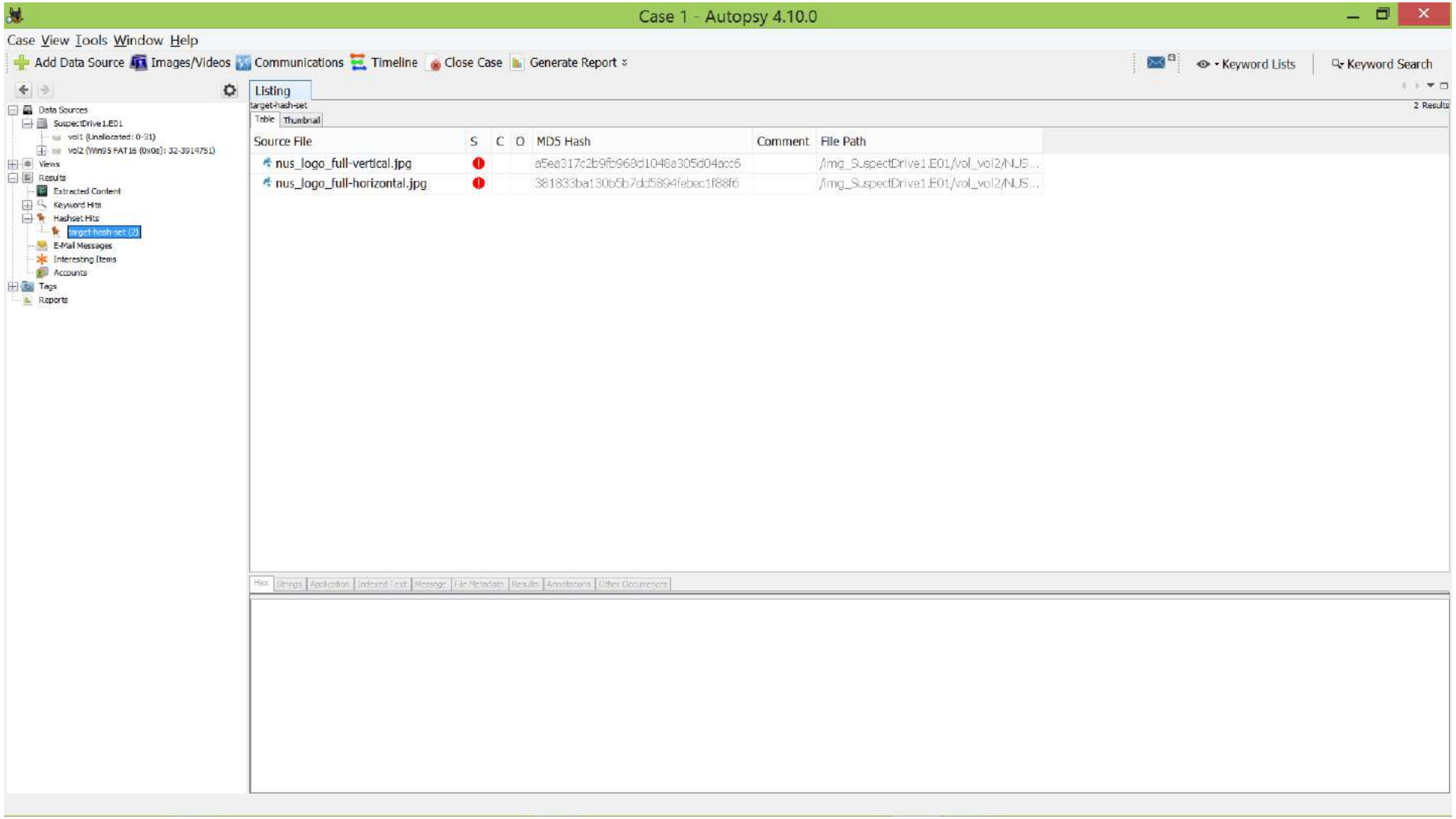
Hashing in Digital Forensics

- Different **roles** of a hashing
- **Evidence preservation**:
  - Ensures that an acquisition done derives an exact copy/replica
- **Evidence verification**:
  - Once an evidence file is generated, its hash value can be computed and then securely recorded
  - An **illegal modification** of the file will result into a different hash value
- ***Hash-lookup analysis*:**
  - Search for files with **known hash values**, e.g. bad files, files of interest
  - *Explained and illustrated next!*

Hash-Lookup Analysis

- A ***hash-lookup analysis*** of your target file system:
  - Search for files with known hash values
  - Based on the **supplied *hash-set files***
  - **Known software files**: *National Software Reference Library (NSRL)* collects **software** from various sources, and incorporate file profiles computed from this software into its Reference Data Set (RDS)
  - NSRL database helps automate the process of identifying known files on computers used in crimes
  - You can additionally define **your own** hash-set files
  - The hash-lookup can be done using **Autopsy**: Lab 4 Task 4-D

Hash-Lookup Analysis

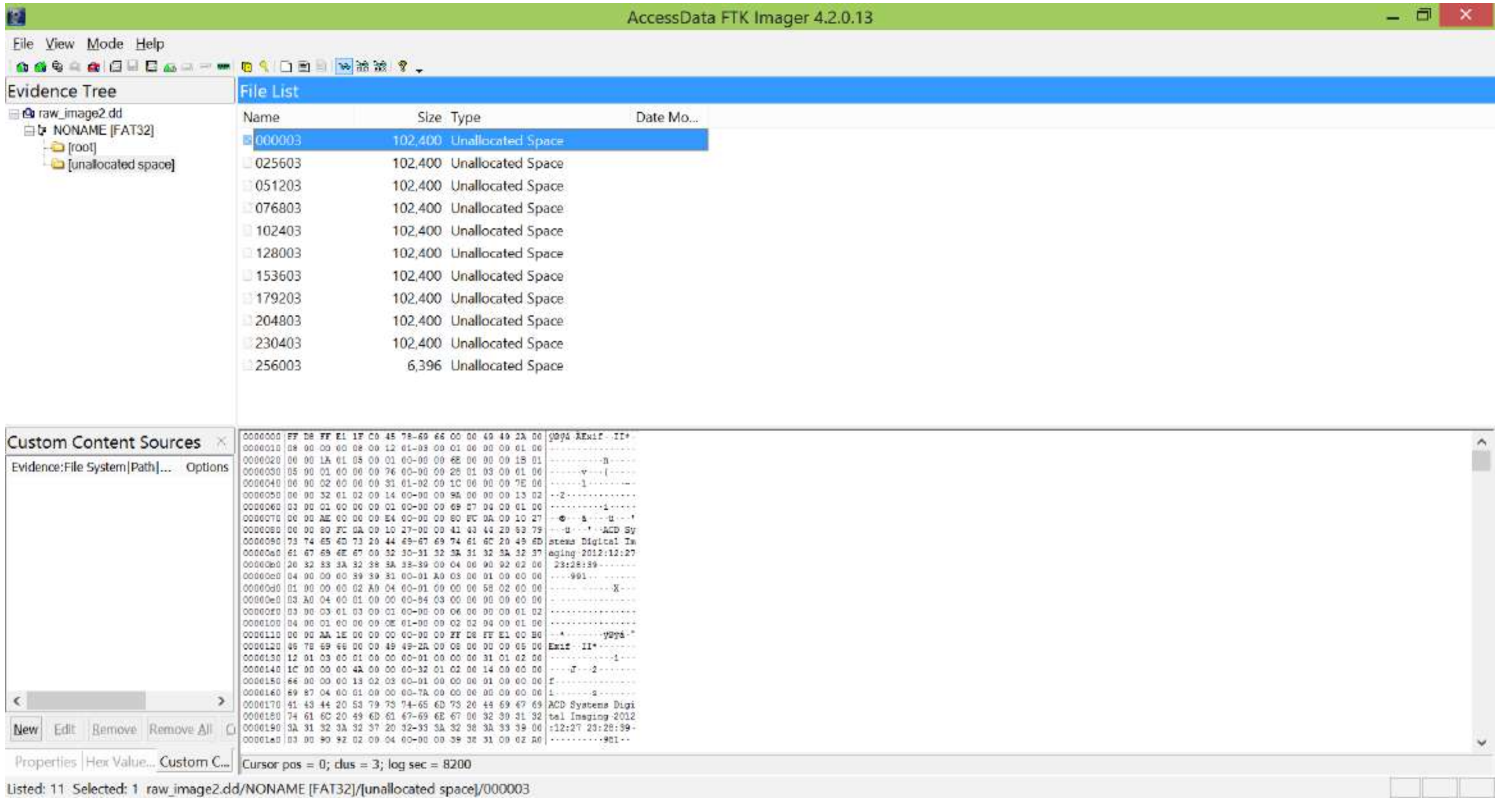


Application-based File/Data Carving

- ***File/data carving***: a process where a chunk of data is searched **for signatures** that correspond to the **start** and **end** of known file types
- Commonly performed on the **unallocated space** of a file system
- Allows the DF investigator to recover files that have **no metadata** structures pointing to them
- The **output**: a collection of files that contain one of the signatures
- Example (on deleted JPEG pictures):
  - A JPEG picture has standard **header** and **footer** values
  - **Extract** the unallocated space
  - Run a **carving tool** that looked for the JPEG header and extract the data in between the header and footer
- In Lab 5: both manual and automated; in Lab 4 Task 4-A (with Autopsy)

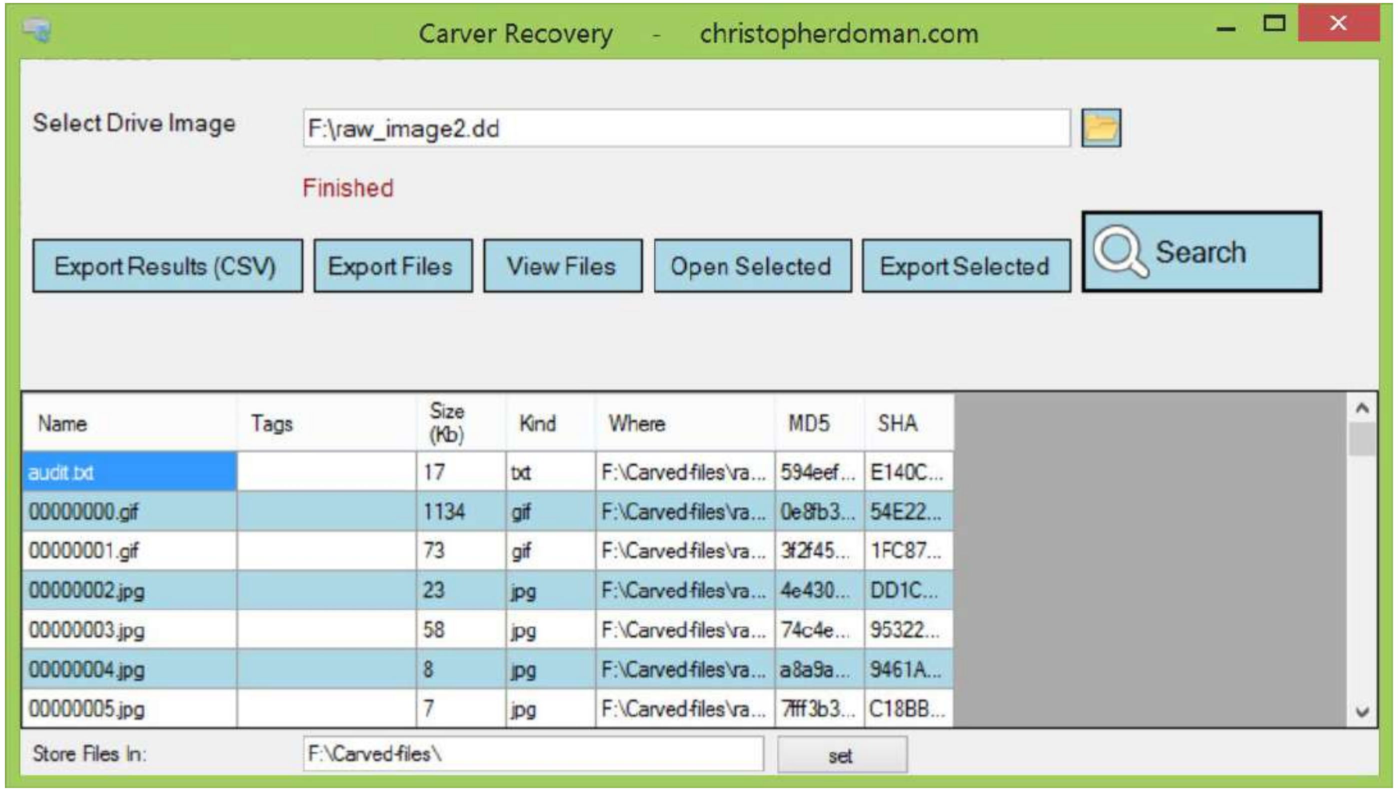
File/Data Carving: Example

- **Manual** extraction (e.g. using **FTK Imager**)
- Easier if there are ***no*** fragmentations; otherwise it is "*an art*"



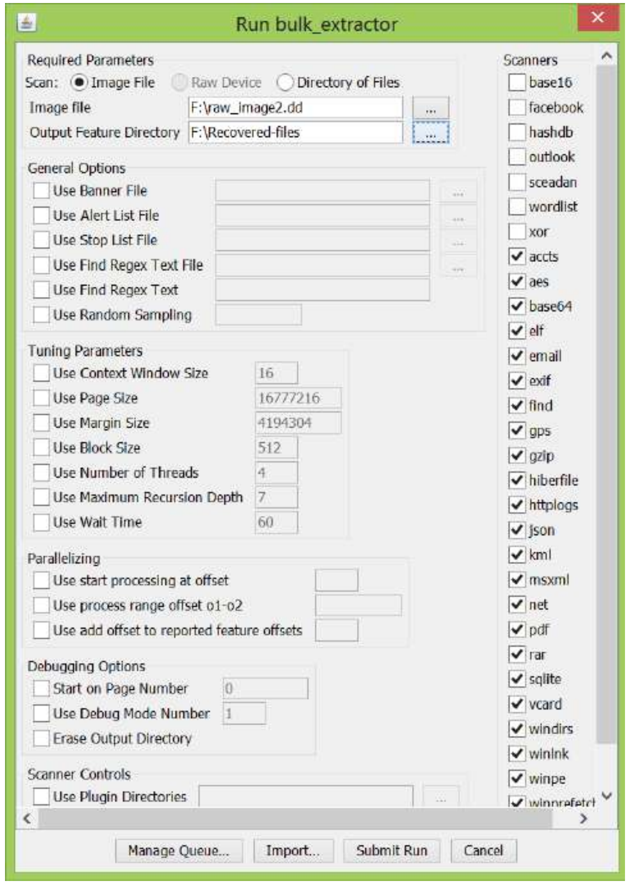
File/Data Carving: Example

- **Automated** extraction (e.g. using **Carver Recovery**)
- Carver Recovery internally runs **Scalpel**



File/Data Carving: Example

- **Automated** extraction (e.g. using **Bulk Extractor**)
- Bulk Extractor Viewer: launch Bulk Extractor and inspect results



Lab 4 Exercises

Lab 4 Exercises

- Task 1: Inspecting a disk image using **TSK**
- Task 2: Performing a **file signature analysis** to fix a concealed file with its altered file extension
- (*Optional*) Task 3: Viewing the **application metadata** of:
  - Microsoft Office files
  - Image files
- Task 4: Using **Autopsy** to:
  - Inspect/extract deleted files, perform file type identification, detect extension mismatch, perform a hash lookup, (*optional*) view Exif data, (*optional*) extract archive file formats

For Your Offline Discussion

- You are asked to describe to a non-technical judge/jury on how **the presented evidence files** are **stored on a suspect's hard disk drive**
- How would you go about describing this, and what **visual aids** and/or **analogies** would you use?

- ***Hint:*** Can you use some diagrams shown in this lecture?

Questions?
See you next week!