ARIZONA STATE UNIVERSITY

CSE 469: Computer and Network Forensics

Topic 3: Drives, Volumes, and Files

Dr. Mike Mabey | Spring 2019

Review: Base Conversion, Endianness, and Data Structures

Converting Between Bases

- Decimal Number: 35,812

10,000 (10^4)	1,000 (10^3)	100 (10^2)	10 (10^1)	1 (10^0)
3	5	8	1	2

- Binary Number: 1001 0011

128 (2^7)	64 (2^6)	32 (2^5)	16 (2^4)	8 (2^3)	4 (2^2)	2 (2^1)	1 (2^0)
1	0	0	1	0	0	1	1

Converting Between Bases

- Hexadecimal Number: 0x<u>8BE4</u>

4,096 (16^3)	256 (16^2)	16 (16^1)	1 (16^0)
8	11	14	4

- 0xB = 11
- 0xE = 14

Binary and Hexadecimal

- 1001 0100 to Hexadecimal

1001	0100

↓

0x9	0x4

=148

- 0x98 to binary

0x9	0x8

↓

1001	1000

=152

Analog Example: Data Structure

- Paper form

SUN Card Application

Please fill out the following form

Name:

Address:

...

Data Structures: Considerations

- Data Size
  - Need to **allocate** a location on a storage device.
  - A **byte** can hold only **256** values.
    - Byte = 8 bits = 2^8 = 256
    - The smallest amount of data we'll work with.
- Organizing multiple-byte values:
  - Big-endian ordering.
  - Little-endian ordering.

Endianness refers to the sequential order in which bytes are arranged into larger numerical values when stored in memory or when transmitted over digital links.

Big- and Little-Endian

- Big-endian ordering:
  - Puts the **most significant byte** of the number in the **first** storage byte.
  - Sun SPARC, Motorola Power PC, ARM, MISP.
- Little-endian ordering:
  - Puts the **least significant byte** of the number in the **first** storage byte.
  - IA32-based systems.

Endianness: Example

Actual Value: 0x12345678 (4 Bytes)

- Big-endian ordering

23	24	25	26	27	28
00	12	34	56	78	00

- Little-endian ordering

23	24	25	26	27	28
00	78	56	34	12	00

Endianness and Strings

- Does Endianness affect letters and sentences?
  - The most common techniques is to encode the characters using ASCII and Unicode.
  - ASCII:
    - In Hexadecimal, 0x00 Through 0x7F.
    - Including control characters (0x07 – Bell Sound).
    - 1 byte per character.
    - The endian ordering does not play a role since each byte stores the value of a character.
    - Many times, the string ends with the NULL character (0x00).

ASCII Example

String: 1 Main St.

23	24	25	26	27	28	29	30	31	32	33
31	**20**	**4D**	**61**	**69**	**6E**	**20**	**53**	**74**	**2E**	**00**
1		M	a	i	n		S	t	.	

Unicode

- Version 11.0 (June 2018) supports 137,439 characters.
  - Covers 146 modern and historic scripts, as well as multiple symbol sets and emoji.
- 4-bytes per character.
- Three methods:
  - UTF-32 – uses a 4-byte value for each character.
  - UTF-16 – stores the most heavily used characters in a 2-byte value and the lesser-used characters in a 4-byte value.
  - UTF-8 – uses 1, 2, or 4 bytes to store a character and the most frequently used bytes use only 1 byte.
- Different methods make different tradeoffs between processing overhead and usability.

ARIZONA STATE UNIVERSITY

Data Structures

- Describes the layout of the data...
  - broken up into **fields** and
  - each field has **size** and **name**.
- Write operation:
  - Refer to the appropriate data structure to determine **where** each value should be written.
- Read operation
  - Need to determine **where the data starts** and then refer to its data structure to find out **where the needed values are** (offset from the start).

Data Structure: Example

Byte Range	Description
0-1	2-byte house number
2-31	30-byte ASCII street name

```
0000000: 0100 4d61 696e 2053 742e 0000 0000 0000   ..Main St....
0000016: 0000 0000 0000 0000 0000 0000 0000 0000   ..............
0000032: bb02 536f 7574 6820 4d69 6c6c 4176 652e   ??
0000048: 0000 0000 0000 0000 0000 0000 0000 0000
```

The byte offset in decimal

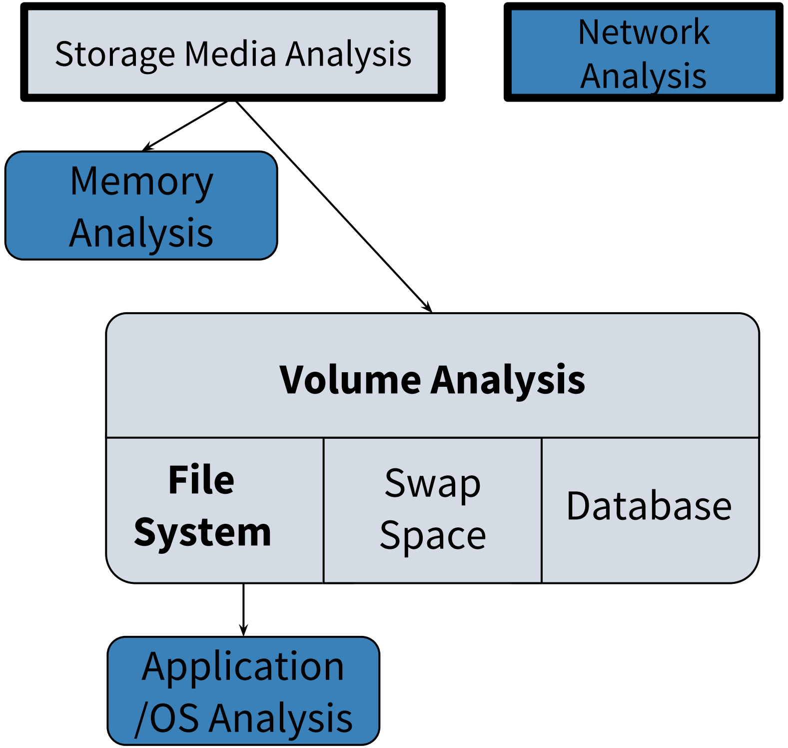
16 bytes of the data in hexadecimal

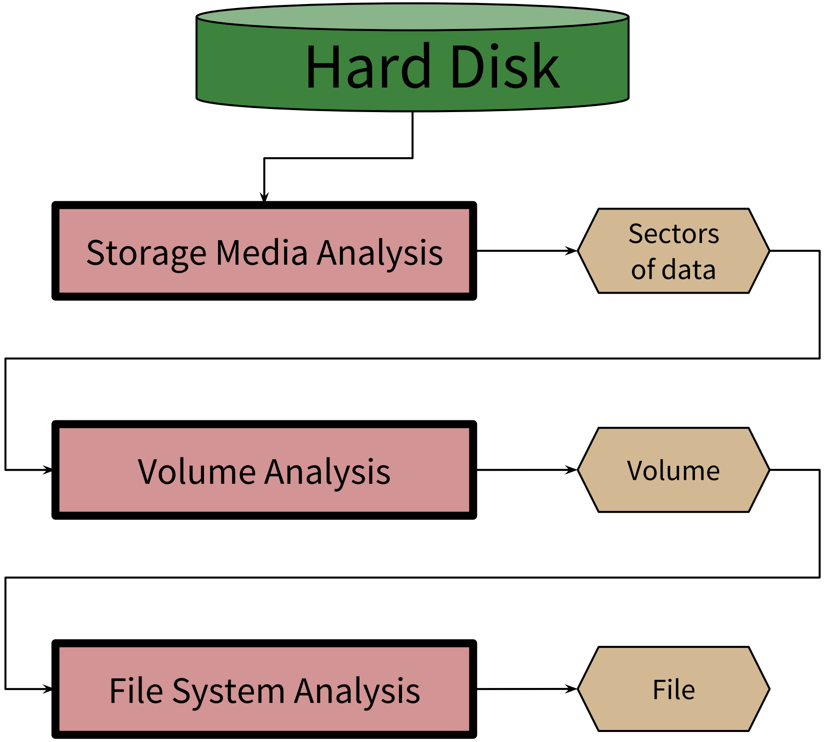
ASCII equivalent

Data structures are important!!

Layers of Forensic Analysis

Layers of Forensic Analysis





ARIZONA STATE UNIVERSITY

Layers of Analysis (1)

- Storage media analysis:
  - Non volatile storage such as hard disks and flash cards.
  - Organized into partitions / volumes:
    - Collection of ***storage locations*** that a user or application can write to and read from.
    - Contents are file system, a database, or a temporary swap space.

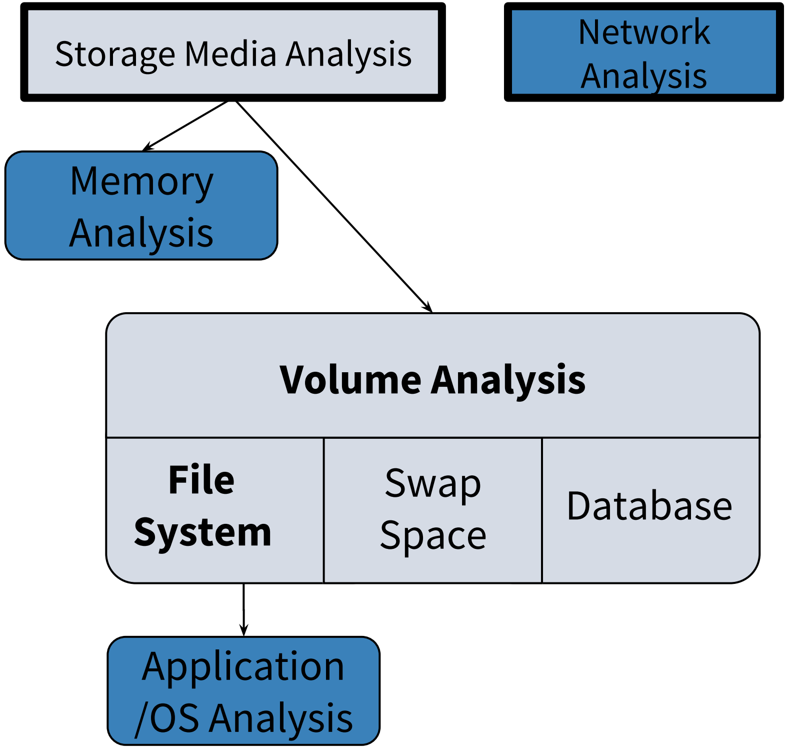
- Volume analysis:
  - Analyze data at the volume level.
  - Determine ***where*** the file system or other data are located.
  - Determine ***where*** we may find hidden data.

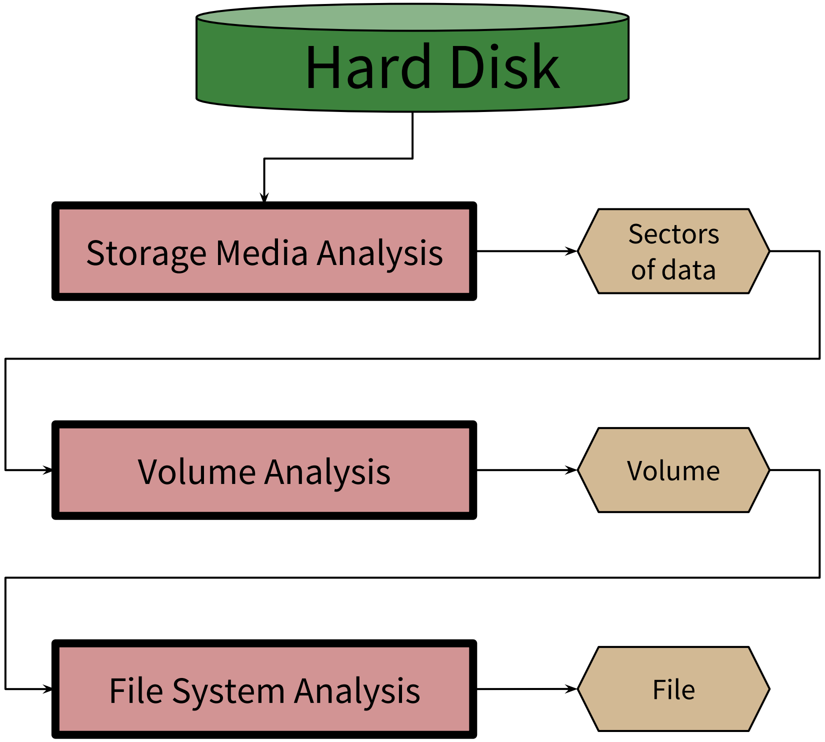
Layers of Analysis (2)

- File system analysis:
  - A collection of ***data structures*** that allow an application to create, read, and write files.
  - Purpose: To find files, to recover deleted files, and to find hidden data.
  - The result could be ***file content***, ***data fragments***, and ***metadata*** associated with files.

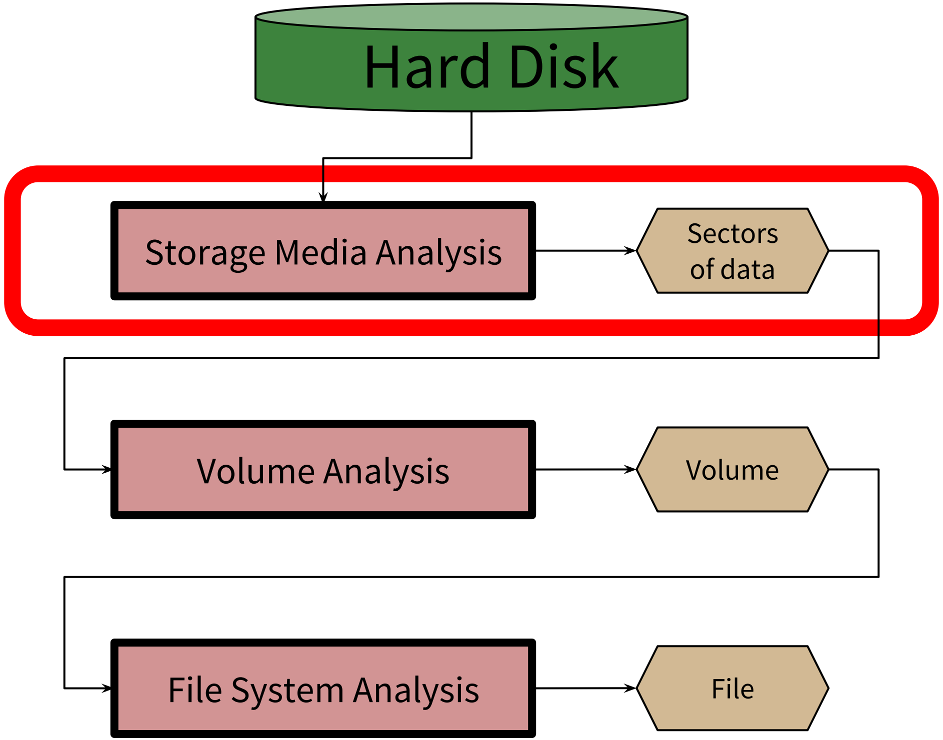
- Application layer analysis:
  - The structure of each file is based on the application or OS that created the file.
  - Purpose: To ***analyze files*** and to determine ***what program we should use***.

Layers of Forensic Analysis





Disk Drive Geometry

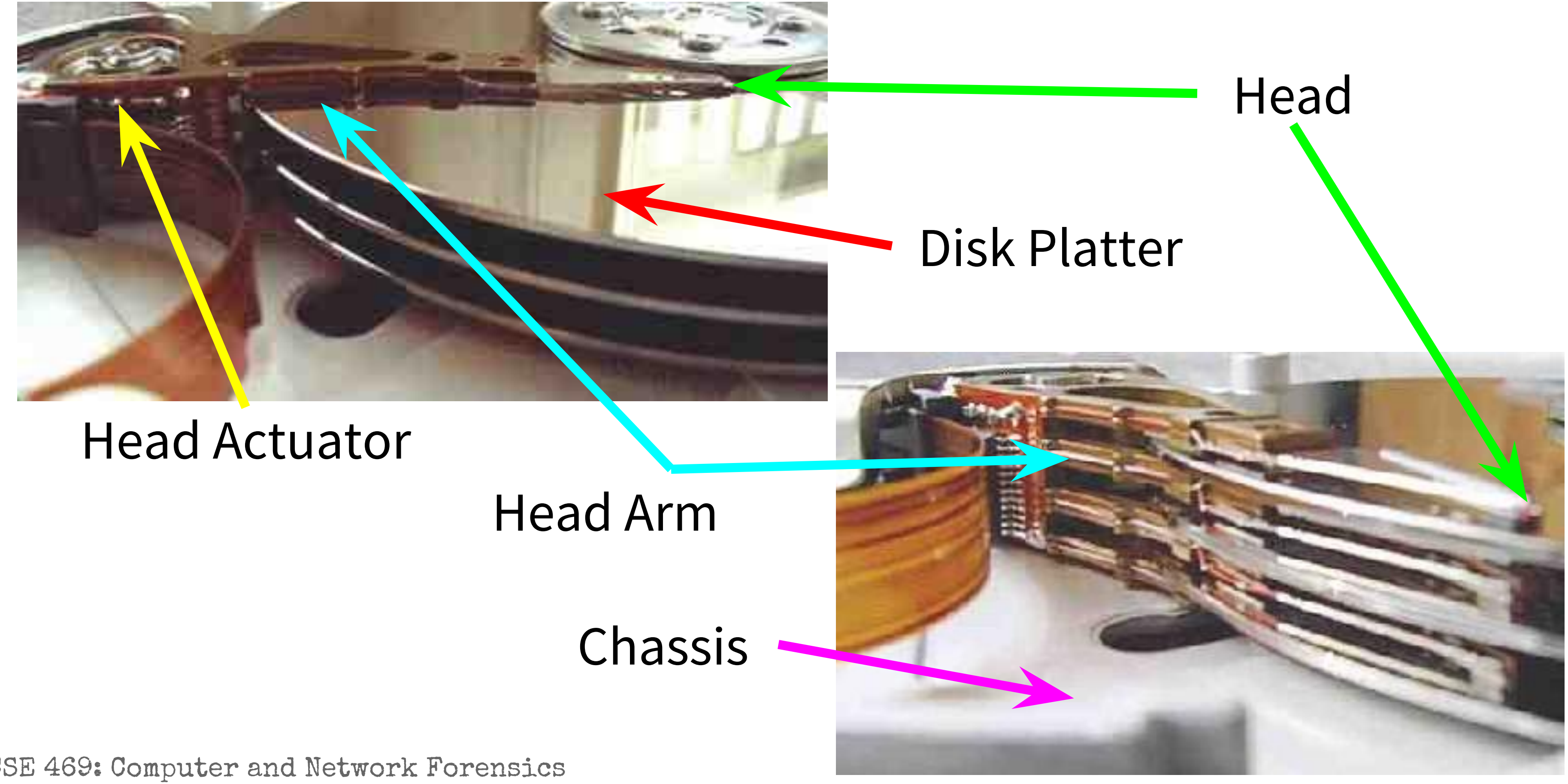

Hard Disk
Storage Media Analysis
Sectors of data
Volume Analysis
Volume
File System Analysis
File

ARIZONA STATE UNIVERSITY

Storage Media Analysis

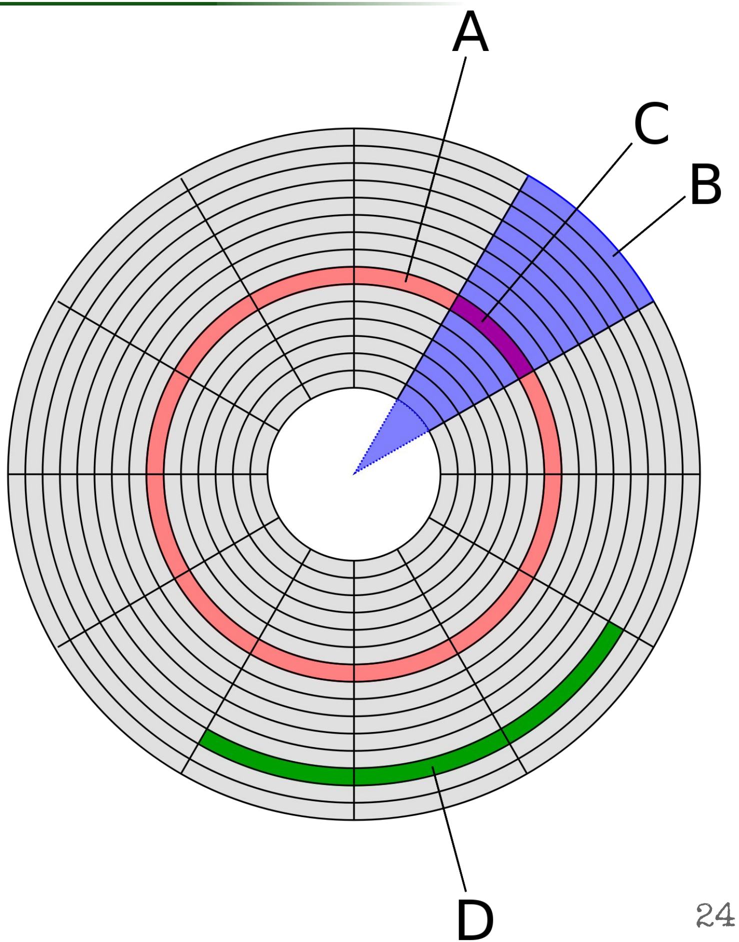
- Hard Disk Geometry
  - Head: The device that reads and writes data to a drive.
  - Track: Concentric circles on a disk platter.
  - Cylinder: A column of tracks on disk platters.
  - Sector: A section on a track.

Inside a Hard Drive

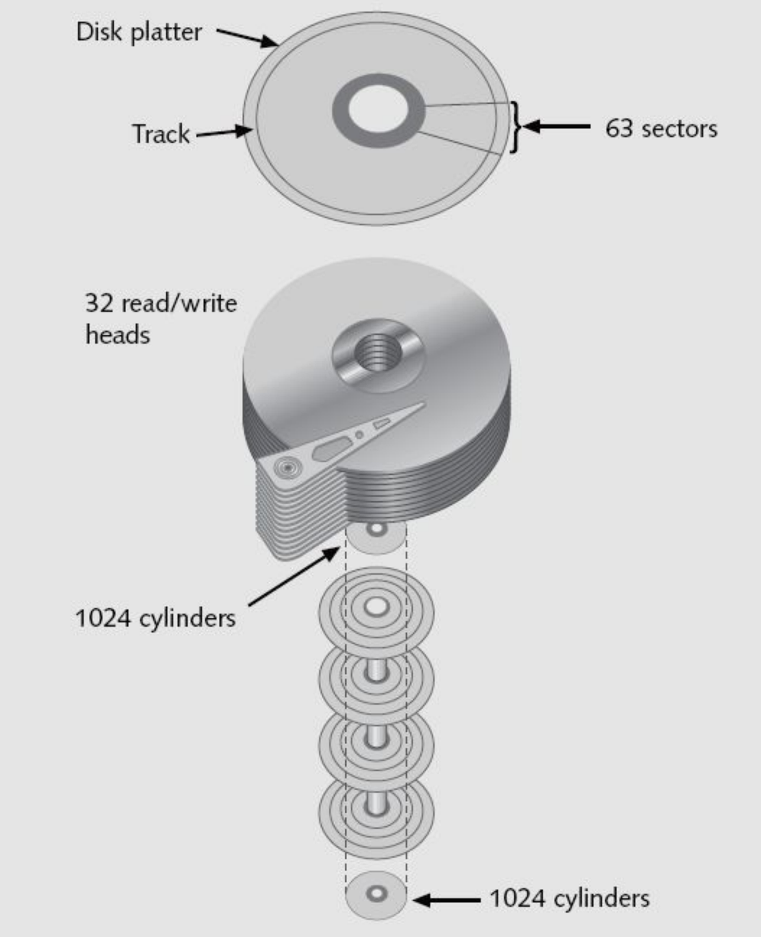


Tracks, Sectors, and Clusters

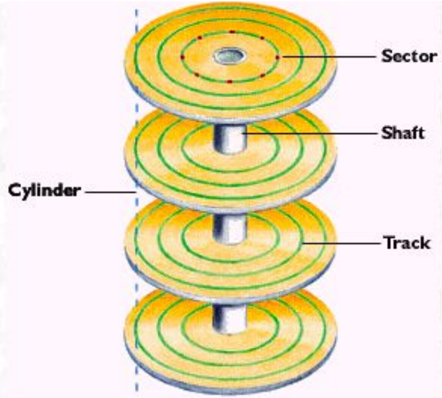
- Platters are divided into concentric rings called ***tracks*** (A).
- Tracks are divided into wedge-shaped areas called ***sectors*** (C).
  - A sector typically holds 512 bytes of data.
  - A collection of sectors is called a ***cluster*** or ***block*** (D).
- (B) is apparently called a *geometrical sector* (uncommon).



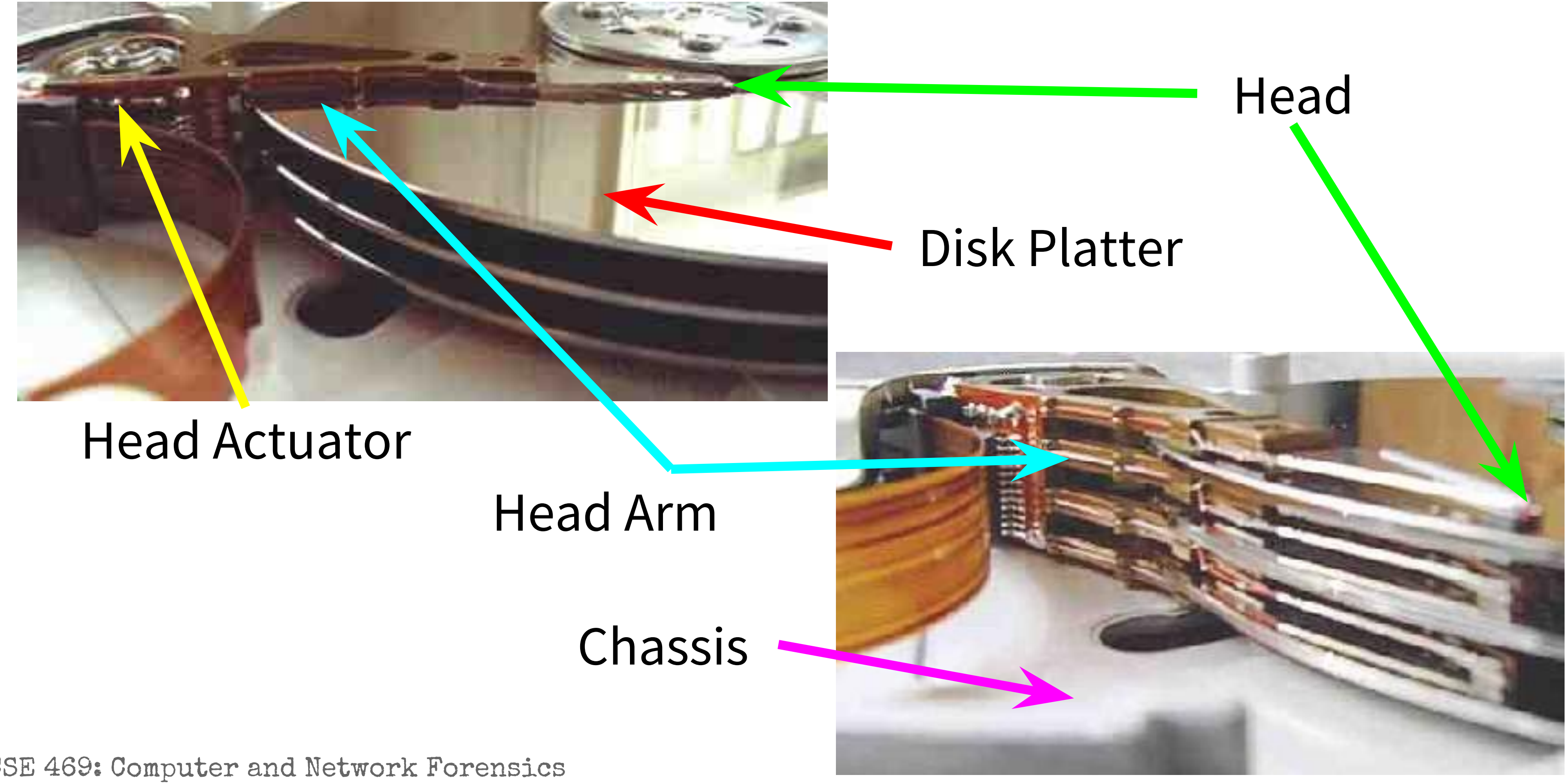
Cylinders

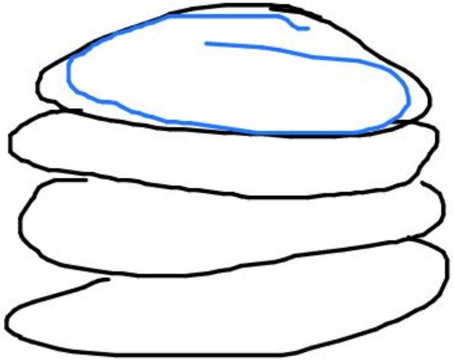


- A ***cylinder*** is a three-dimensional concept consisting of all *tracks* in the same position vertically

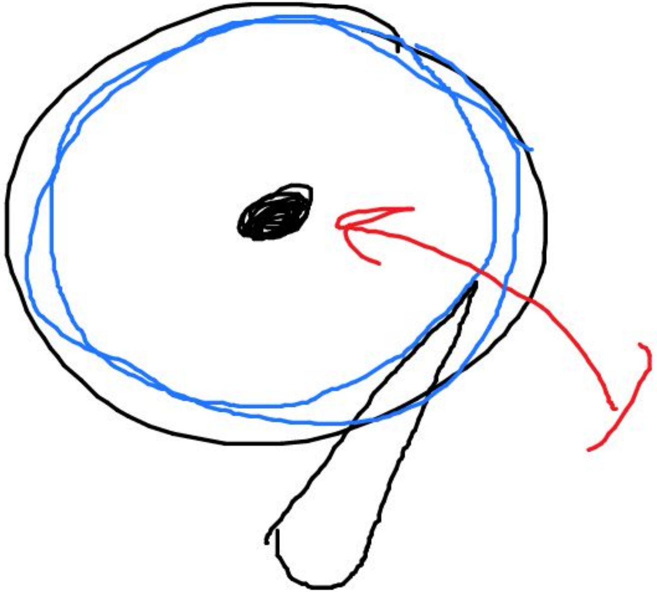


Inside a Hard Drive



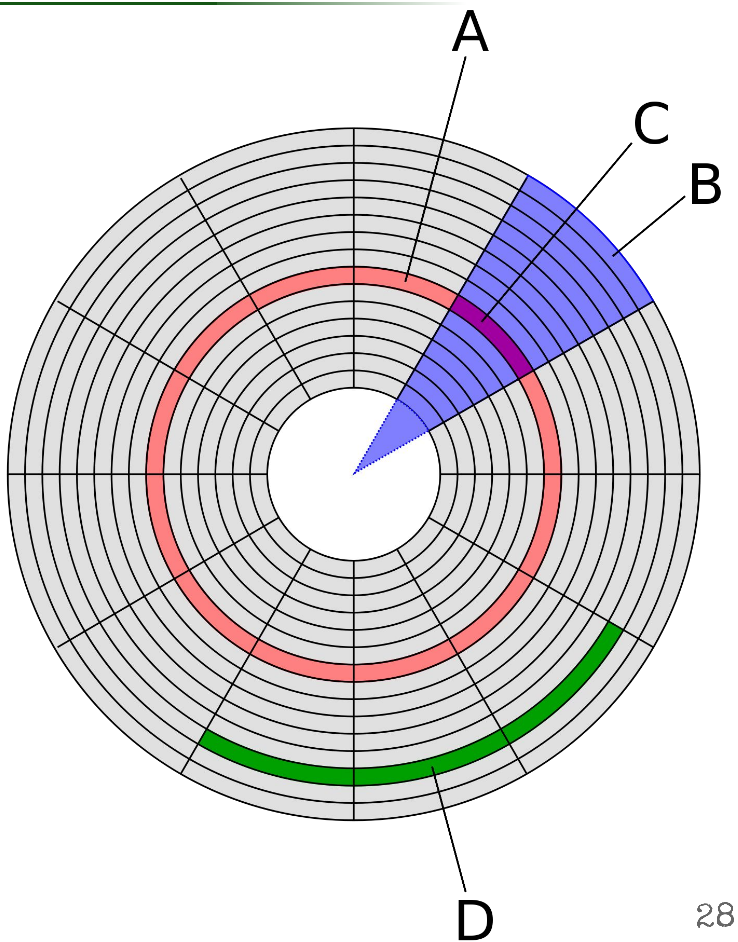


5.4K RPM
7 KRPM
10 KRPM



CHS Addresses

- ***Tracks/Cylinders***: Numbered from the outside in, **starting at 0**.
  - All sectors of all tracks in cylinder 0 will be filled up before using cylinder 1.
- ***Heads***: Numbered from the bottom up, **starting at 0**.
  - All platters are double-sided, one head per side.
- ***Sectors***: Each sector is numbered, **starting at 1**.
  - Typically holds 512 bytes of data.
- First sector has CHS address: **0,0,1**



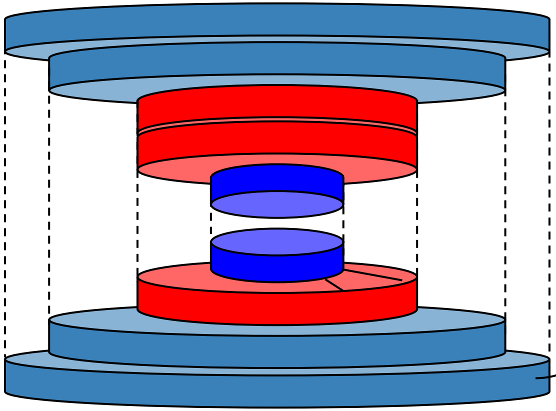
Logical Block Address (LBA)

- CHS addresses have a limit of 8.1 GB.
  - Not enough bits allocated to store values in the Master Boot Record of disks.
- Logical Block Addresses (LBA) overcome this:
  - Singe address instead of three.
  - **Starts at 0**, so LBA 0 == CHS 0,0,1.
  - To convert from CHS, need to know:
    - CHS address.
    - Number of heads per cylinder.
    - Number of sectors per track.

CHS to LBA Conversion

- LBA = (((**CYLINDER** * heads_per_cylinder) + **HEAD**) * sectors_per_track) + **SECTOR** -1

== num_platters * 2



- CHS (***x***,***y***,***z***)
  - Locate the ***x***-th cylinder and calculate the number of sectors
  - Locate the ***y***-th head and calculate the number of sectors
  - Add (***z***-1) sectors

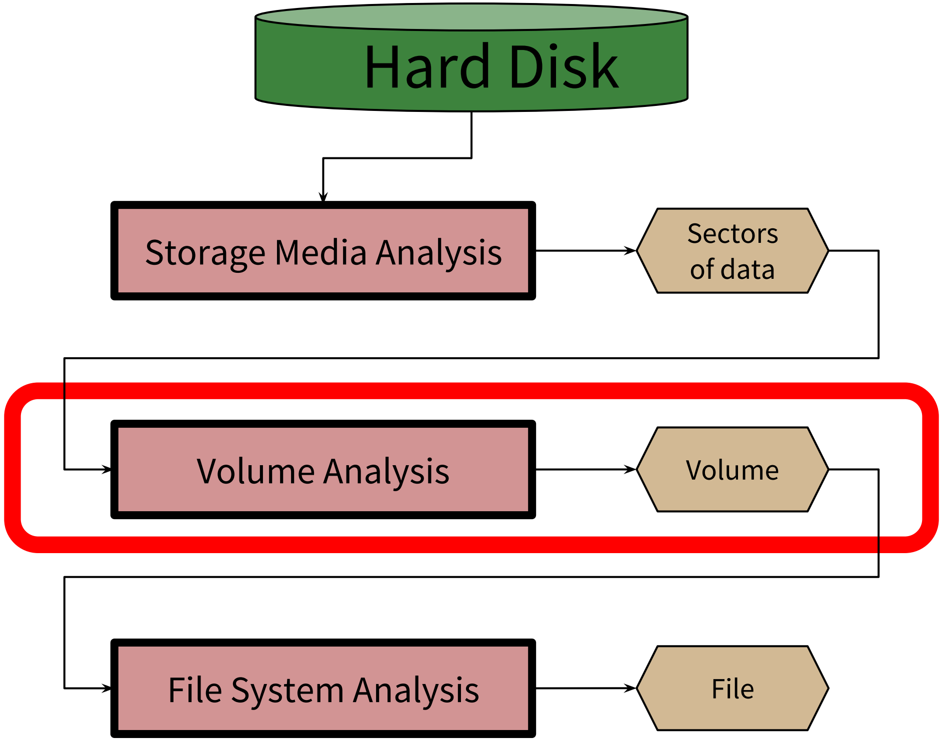
Address Conversion: Practice

- Given a disk with **16 heads** per cylinder and **63 sectors** per track, if we had a CHS address of **cylinder 2**, **head 3**, and **sector 4**, what would be the LBA (a.k.a CHS (2,3,4))?

LBA = (((**CYLINDER** * heads_per_cylinder) + **HEAD**) * sectors_per_track) + **SECTOR** -1

$$(((2*16)+3)*63)+4-1=2208$$

Volumes and Partitions


Hard Disk
Storage Media Analysis
Sectors of data
Volume Analysis
Volume
File System Analysis
File

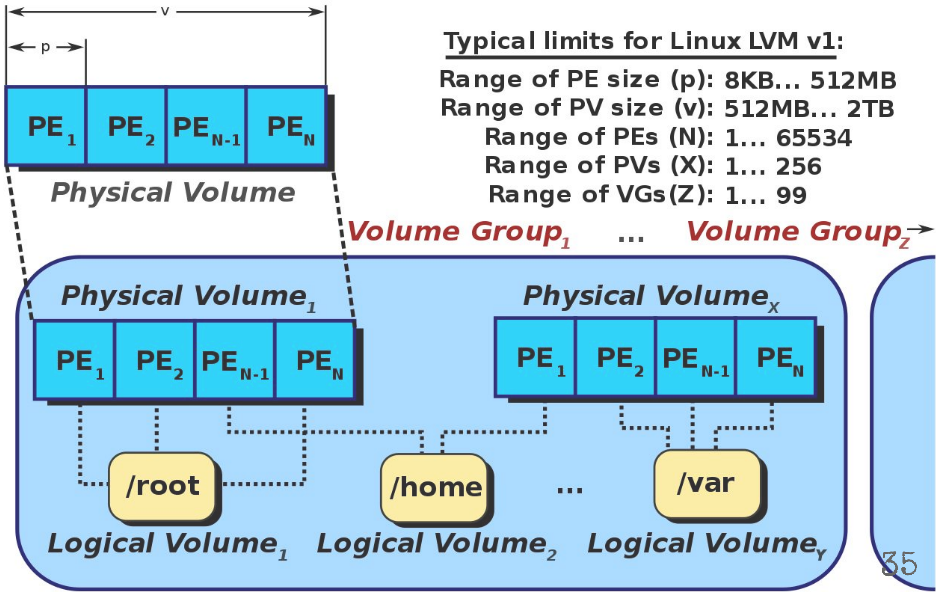
Volume Analysis

- Volume/Partition:
  - Collection of ***addressable sectors*** that an OS or application can use for data storage.
  - Used to store file system and other structured data.

- Purpose of Volume Analysis:
  - Involves looking at the data structures that are involved with partitioning and assembling the bytes in storage devices.

Logical Volume Management (LVM)

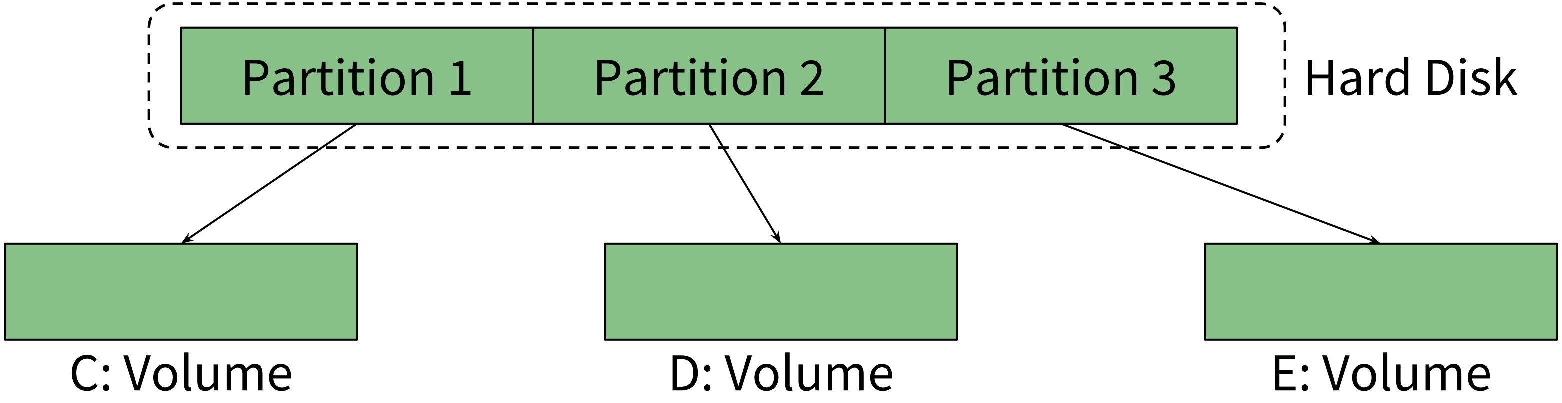
- LVM terms:
  - Physical volumes (PVs): Hard disks or hard disk partitions
  - Physical Extents (PEs): Basically clusters (groups of sectors)
  - Physical Volume Group (PVG): Pool of PEs
  - Logical Extents (LEs): Logical mapping to PEs
  - Volume Group (VG): Pool of LEs
  - Logical Volumes (LVs): Concatenation of LEs



Img source: https://en.wikipedia.org/wiki/File:LVM1.svg

Partitions

- Collection of ***consecutive*** sectors in a volume.
- Each OS and hardware platform use a different partitioning method.



Arizona State University

Partitions: Purpose

- Partitions organize the layout of a volume.
- Essential data are the ***starting*** and ***ending*** location for each partition.
- Common partition systems have one or more tables and each table describes a partition:
  - Starting sector of the partition.
  - Ending sector of the partition (or the length).
  - Type of partition.

Master Boot Record (MBR)

- First sector (CHS 0,0,1) stores the disk layout.
- Each **partition entry** has the structure shown on the next slide.

Offset	**Description**	**Size**
0x0000	Executable Code (Boots Computer)	446 Bytes
0x01BE	1st Partition Entry	16 Bytes
0x01CE	2nd Partition Entry	16 Bytes
0x01DE	3rd Partition Entry	16 Bytes
0x01EE	4th Partition Entry	16 Bytes
0x01FE	**Boot Record Signature (0x55 0xAA)**	2 Bytes

ARIZONA STATE UNIVERSITY

MBR Partition Entry

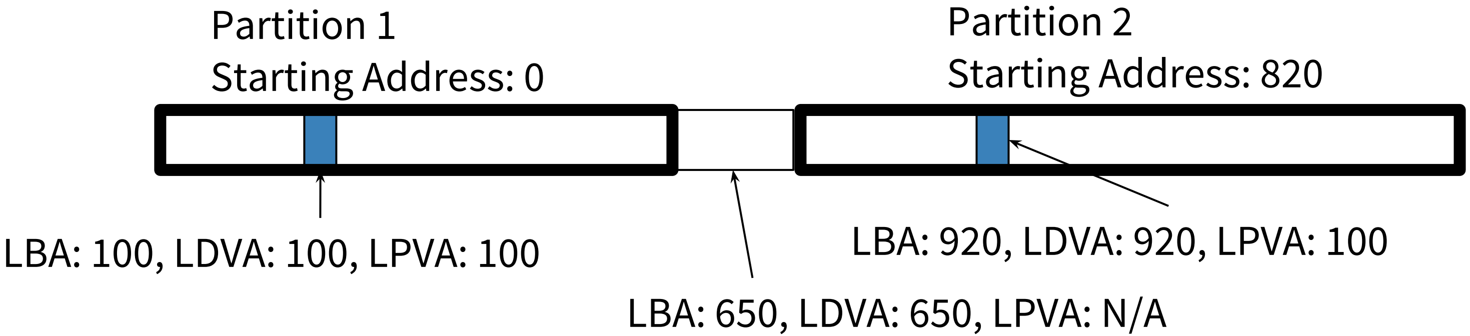
Offset	Description	Size
0x00	Current State of Partition (0x00=Inactive, 0x80=Active)	1 byte
0x01	Beginning of Partition - Head	1 byte
0x02	Beginning of Partition - Cylinder/Sector	1 word (2 bytes)
0x04	Type of Partition	1 byte
0x05	End of Partition - Head	1 byte
0x06	End of Partition - Cylinder/Sector	1 word (2 bytes)
0x08	LBA of First Sector in the Partition	1 double word (4 bytes)
0x0C	Number of Sectors in the Partition	1 double word

Note on MBRs

- Maximum addressable storage space: 2 TiB.
  - 2^{40} bytes.
- In the process of being superseded by the GUID Partition Table (GPT) scheme.
  - A little more complicated, not going to explain here.
  - GPTs offer limited backwards compatibility.
- See Wikipedia for more info:
  - https://en.wikipedia.org/wiki/Master_boot_record
  - https://en.wikipedia.org/wiki/GUID_Partition_Table
- Tons of supported partition types (offset 0x04):
  - https://en.wikipedia.org/wiki/Partition_type

Sector Addressing

- Logical Volume Address:
  - Logical “Disk” Volume Address (LDVA)
    - Relative to the start of the volume.
  - Logical “Partition” Volume Address (LPVA)
    - Relative to the start of the partition.



Arizona State University

Partition Analysis Steps

1. Locate the partition tables.
2. Process the data structures to identify the layout since we need to know the offset of a partition.
   - It is important to discover the **partition layout** of the volume because not all sectors need to be assigned to a partition and they **may contain data from a previous file system or that the suspect was trying to hide**.
3. Conduct the consistency checks:
   - Looks at the last partition and compares its starting location with the end of its parent partition.
   - To determine where else evidence could be located besides in each partition.

Note: To analyze the data inside a partition, we need to consider what type of data it is—normally it's a file system.

Extraction of Partition Contents

- Need to extract the data in or in between partitions to a separate file.
- Tools:
  - dd tool:
    - `if`, `of`, `bs` (512 bytes), `skip` (blocks to skip), `count` (blocks to copy)
  - `mmls` tool from the Sleuth Kit.
  - Any hex editor.

Volume Analysis

```
# 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)
```


```
# 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
```

Volume Analysis (MBR)

```
0000432: 0000 0000 0000 0000 0000 0000 0000 0001
0000448: 0100 07fe 3f7f 3f00 0000 4160 1f00 8000
0000464: 0180 0bfe 3f8c 8060 1f00 cd2f 0300 0000
```

The first 446 bytes contain boot code

The byte offset in decimal

16 bytes of the data in hexadecimal

#	Flag	Type	Starting Sector	Size
1	0x00	0x07	0x0000003f (63)	0x001f6041 (2,056,257)
2	?	?	?	?