Introduction to Computer Forensics and Security

6G7Z1009

Binary Numbers

- Computers store information using binary numbers, this includes (numeric information, textual information and Pictures).
- The number could be of any length.
- The following are all examples of binary numbers

0

1

10

01

111000

10101

Converting from Binary to Decimal

- Each position for a binary number has a value.
- For each digit, multiply the digit by its position value
- Add up all of the products to get the final result
- **The value of binary 1001 is decimal 9. This is worked out below:**

8	4	2	1
1	**0**	**0**	**1**

Answer: 9

Another example

- **The value of binary 10001010 is decimal 138. This is worked out below:**

128	64	32	16	8	4	2	1
1	0	0	0	1	0	1	0

0 X 1= 0

1 X 2 = 2

0 X 4 = 0

1 X 8 = 8

0 X 16 = 0

0 X 32 = 0

0 X 64 = 0

1 X 128 = 128

Answer: 138

Hexadecimal (AKA “Hex”) numbers

- A “hexadecimal” number is a number where each digit may be one of sixteen possible values.
- The possible values for a hexadecimal digit are:
  0 1 2 3 4 5 6 7 8 9 A B C D E F
- A digit of
  “A” stands for the number 10
  “B” stands for the number 11
  “C” stands for the number 12
  “D” stands for the number 13
  “E” stands for the number 14
  “F” stands for the number 15

Hexadecimal (AKA “Hex”) numbers

- The following are all valid hexadecimal nubmers
  - ❑ A
  - ❑ 9 (yes, a hexadecimal number does not HAVE TO contain letters)
  - ❑ 1001 (yes, a hexadecimal number does not HAVE TO contain letters)
  - ❑ 9C5
  - ❑ BFE
  - ❑ Etc.

Converting a Hexadecimal number to Decimal

- **The value of hexadecimal A12F is decimal 41,263. See below:**

4096 (i.e 16^3)	**256 (i.e 16^2)**	**16 (i.e 16^1)**	**1 (i.e 16^0)**	
A	**1**	**2**	**F**	
			15 X 1	**= 15**
		2 X 16		**= 32**
	1 X 256			**= 256**
10 X 4096				**= 40,960**

			Answer:	**41,263**

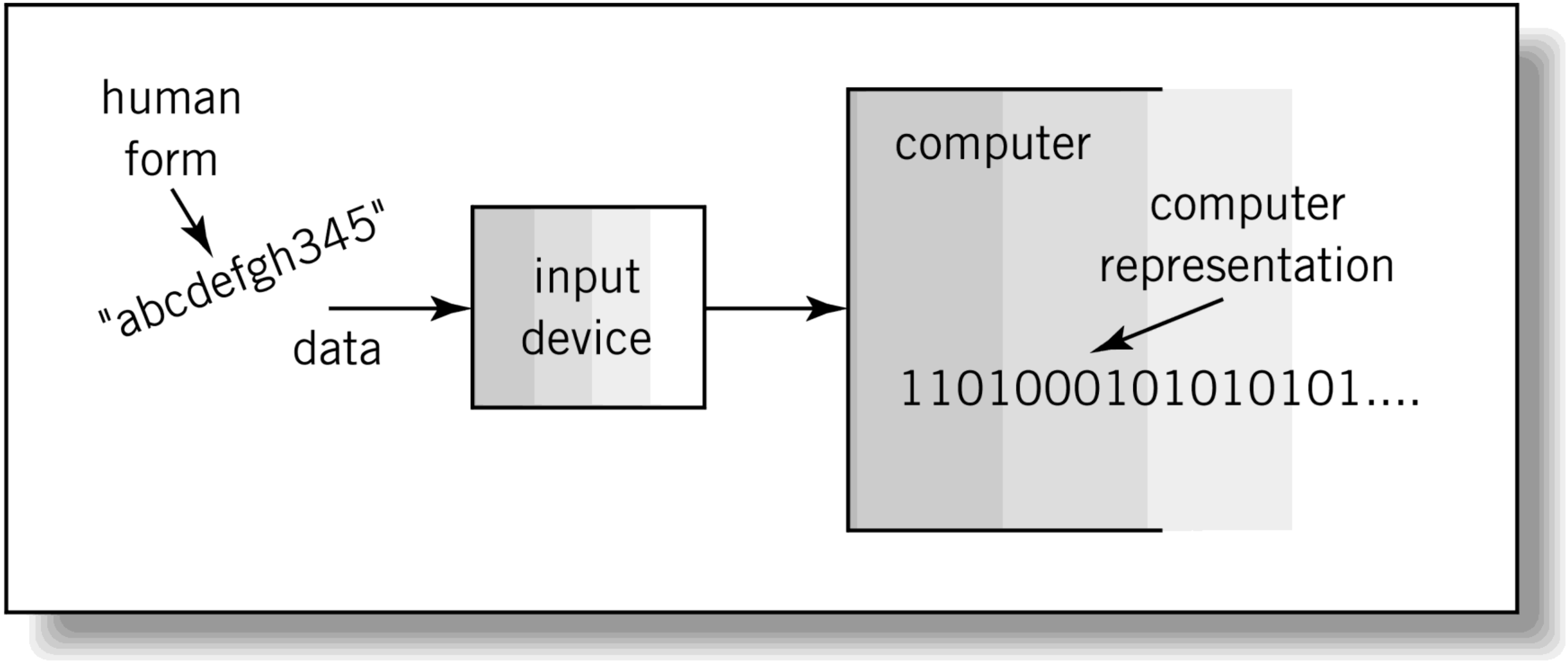
Terms (bit, byte, etc)

- BIT
  - definition: a single Binary digit
- BYTE
  - definition: 8 bits
- NIBBLE (NYBLE-US)
  - definition: 4 bits

Data Forms

- Human communication
  - Includes language, images and sounds
- Computers
  - Process and store all forms of data in binary format
- Conversion to computer-usable representation using data formats
  - Define the different ways human data may be represented, stored and processed by a computer

Data Forms



Englander: The Architecture of Computer
Hardware and Systems Software, 2nd edition
Chapter 3, Figure 03-01

Data formats

- Proprietary formats
  - Unique to a product or company
  - E.g., Microsoft *Word*, *Word Perfect*
- Standards (evolve in two ways):
  - Proprietary formats become standards (e.g., Adobe *PostScript*)
  - Invented by an international standard organization (e.g., Motion Pictures Experts Group, *MPEG*)

Common Data Representations

Type of Data	Standard(s)
Alphanumeric	Unicode, ASCII, EBCDIC
Image (bitmapped)	■GIF (graphical image format) ■TIF (tagged image file format) ■PNG (portable network graphics)
Image (object)	PostScript, JPEG, SWF (Macromedia Flash), SVG
Outline graphics and fonts	PostScript, TrueType
Sound	WAV, AVI, MP3, MIDI, WMA
Page description	PDF (Adobe Portable Document Format), HTML, XML
Video	Quicktime, MPEG-2, RealVideo, WMV

Data formats

- Characters (r, T), number digits (0..9), punctuation (!, ;), special purpose characters (*$, &*)
- Four codes/standards to represent letters and numbers:
  - BCD (Binary-Coded Decimal)
  - **ASCII (American Standard Code for Information Interchange)**
  - Unicode
  - EBCDIC (Extended Binary Coded Decimal Interchange Code)

ASCII Features

- Developed by ANSI (American National Standards Institute)
- Defined in ANSI document X3.4-1977
- 7-bit code
- 8th bit is unused (or used for a parity bit or to indicate “extended” character set)
- 2^7 = 128 different codes
- Two general types of codes:
  - 95 are “Printing” codes (displayable on a console)
  - 32 are “Control” codes (control features of the console or communications channel)
- Represents
  - Latin alphabet, Arabic numerals, standard punctuation characters
  - Plus small set of accents and other European special characters (Latin-I ASCII)

ASCII Table – Control Codes

DEC	HEX	BIN	Symbol	Description
0	00	00000000	NUL	Null char
1	01	00000001	SOH	Start of Heading
2	02	00000010	STX	Start of Text
3	03	00000011	ETX	End of Text

ASCII Table – Printable Codes

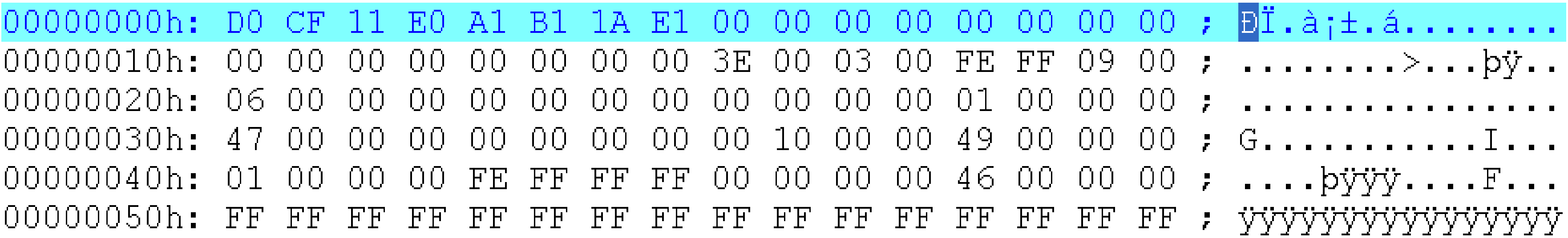
DEC	HEX	BIN	Symbol	Description
32	20	00100000		Space
33	21	00100001	!	Exclamation mark
34	22	00100010	"	Double quotes (or speech marks)
48	30	00110000	0	Zero
49	31	00110001	1	One
50	32	00110010	2	Two
51	33	00110011	3	Three
65	41	01000001	A	Uppercase A
66	42	01000010	B	Uppercase B
95	5F	01011111	_	Underscore
97	61	01100001	a	Lowercase a
98	62	01100010	b	Lowercase b
99	63	01100011	c	Lowercase c

Example – “Hello, world”

		Binary		Hexadecimal		Decimal
H	=	01001000	=	48	=	72
e	=	01100101	=	65	=	101
l	=	01101100	=	6C	=	108
l	=	01101100	=	6C	=	108
o	=	01101111	=	6F	=	111
,	=	00101100	=	2C	=	44
	=	00100000	=	20	=	32
w	=	01110111	=	77	=	119
o	=	01100111	=	67	=	103
r	=	01110010	=	72	=	114
l	=	01101100	=	6C	=	108
d	=	01100100	=	64	=	100

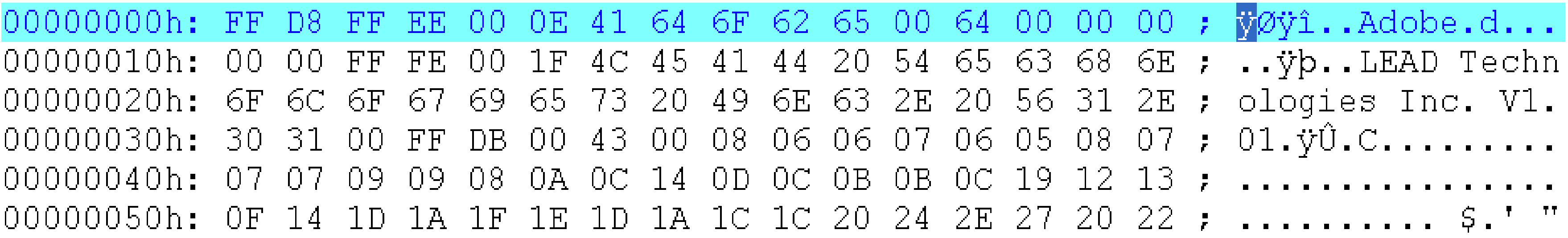
File Types – Some File Headers/Signatures

- Word



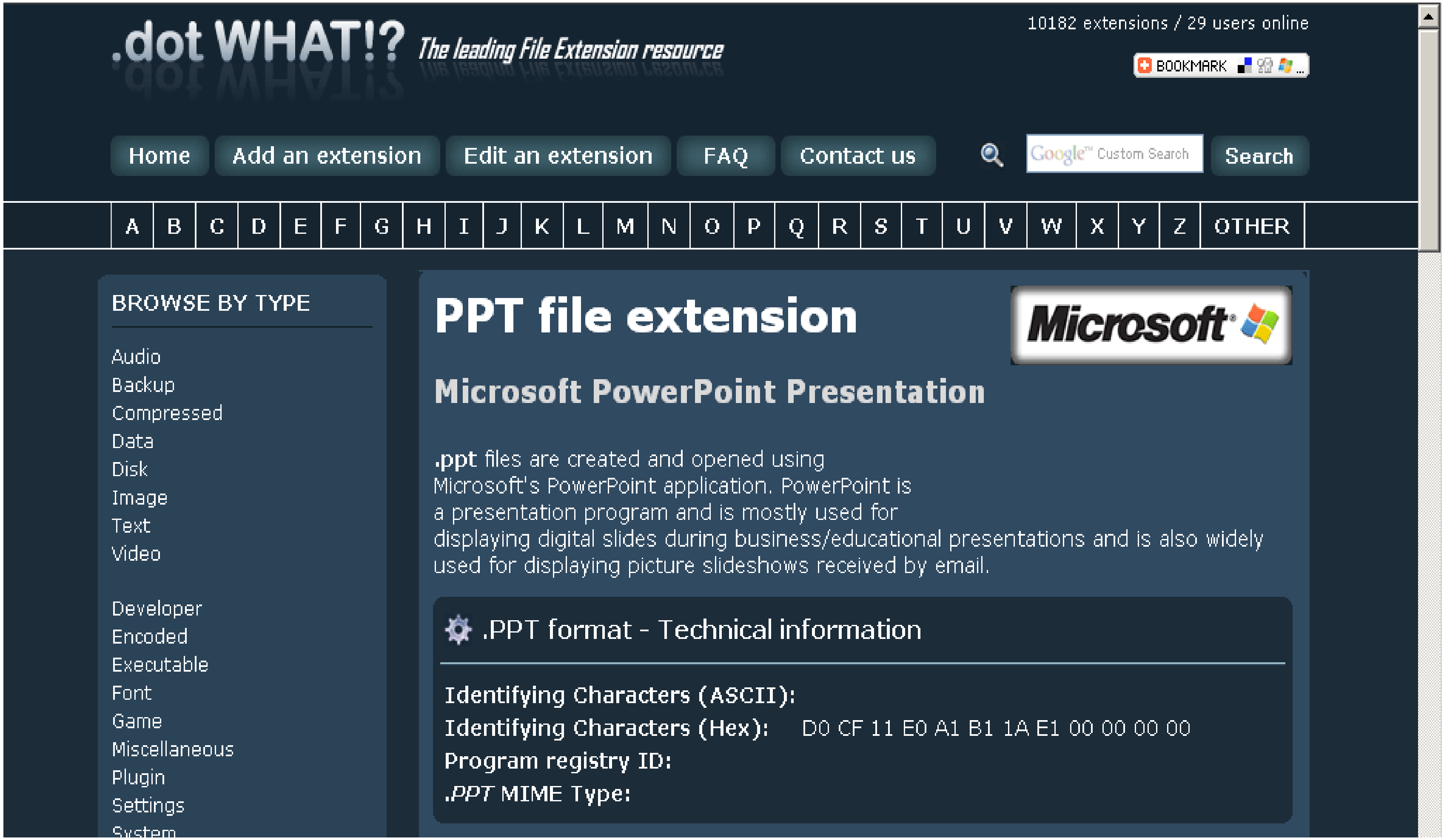
```
00000000h: D0 CF 11 E0 A1 B1 1A E1 00 00 00 00 00 00 00 00 ; ÐÏ.à¡±.á........
00000010h: 00 00 00 00 00 00 00 00 3E 00 03 00 FE FF 09 00 ; ........>...þÿ..
00000020h: 06 00 00 00 00 00 00 00 00 00 00 00 01 00 00 00 ; ................
00000030h: 47 00 00 00 00 00 00 00 00 10 00 00 49 00 00 00 ; G...........I...
00000040h: 01 00 00 00 FE FF FF FF 00 00 00 00 46 00 00 00 ; ....þÿÿÿ....F...
00000050h: FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF ; ÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿÿ
```

- JPEG



```
00000000h: FF D8 FF EE 00 0E 41 64 6F 62 65 00 64 00 00 00 ; ÿØÿî..Adobe.d...
00000010h: 00 00 FF FE 00 1F 4C 45 41 44 20 54 65 63 68 6E ; ..ÿþ..LEAD Techn
00000020h: 6F 6C 6F 67 69 65 73 20 49 6E 63 2E 20 56 31 2E ; ologies Inc. V1.
00000030h: 30 31 00 FF DB 00 43 00 08 06 06 07 06 05 08 07 ; 01.ÿÛ.C.........
00000040h: 07 07 09 09 08 0A 0C 14 0D 0C 0B 0B 0C 19 12 13 ; ................
00000050h: 0F 14 1D 1A 1F 1E 1D 1A 1C 1C 20 24 2E 27 20 22 ; .......... $.' "
```

File Types (http://dotwhat.net)



PPT file extension

Microsoft PowerPoint Presentation

.ppt files are created and opened using Microsoft's PowerPoint application. PowerPoint is a presentation program and is mostly used for displaying digital slides during business/educational presentations and is also widely used for displaying picture slideshows received by email.

.PPT format - Technical information

Identifying Characters (ASCII):
Identifying Characters (Hex): D0 CF 11 E0 A1 B1 1A E1 00 00 00 00
Program registry ID:
***.PPT* MIME Type:**

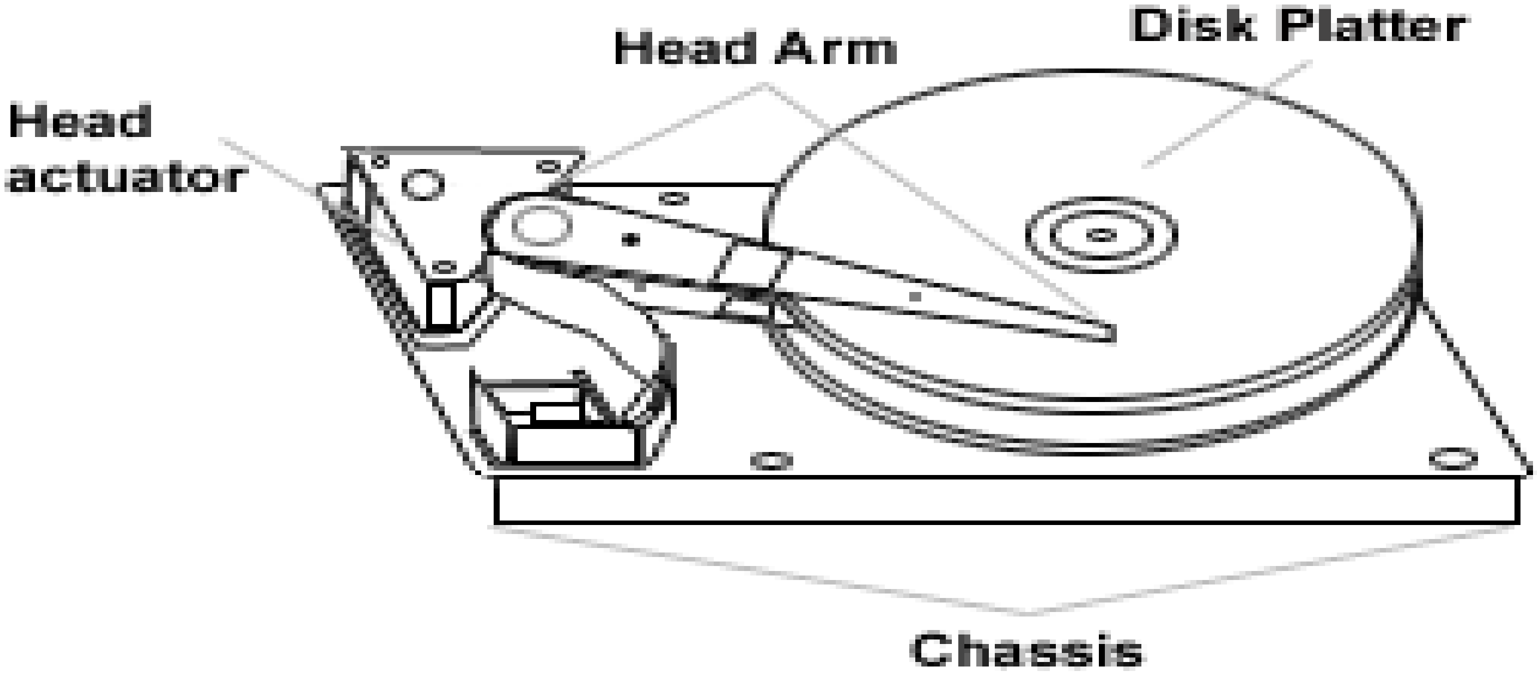
Hard disk drive

- Disk Structure
- Physical Disk Geometry
- Sector & Cluster/Block
- Lifecycle of Disk Drive

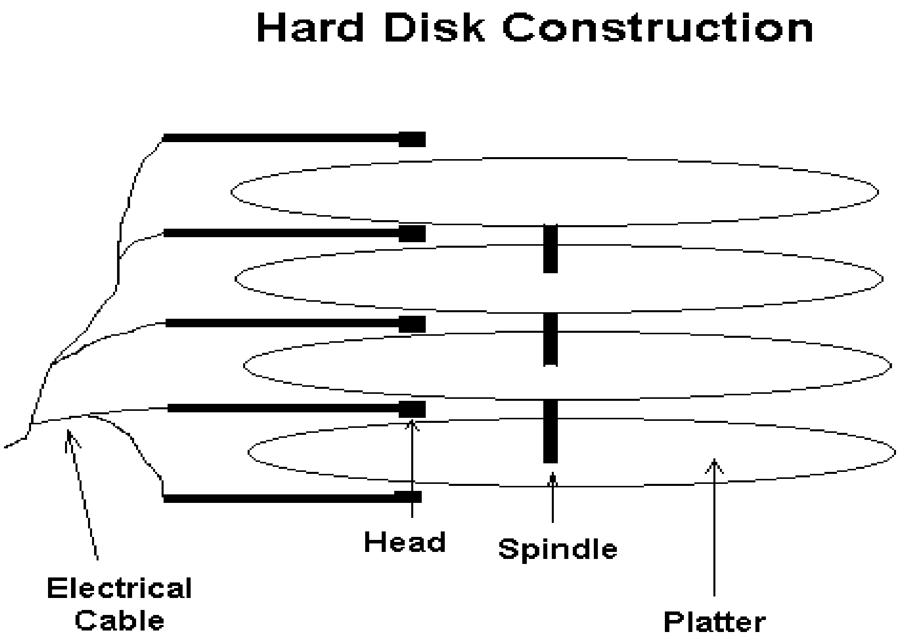
Understanding Disk Drives

- Composed of one or more platters
  - Coated with magnetic material
- Disk terminology:
  - Geometry (internal organization)
  - Head (reads and writes data to 1 platter)
  - Tracks (circular area for data)
  - Cylinders (column of tracks)
  - Sectors (section of a track, 512 bytes)

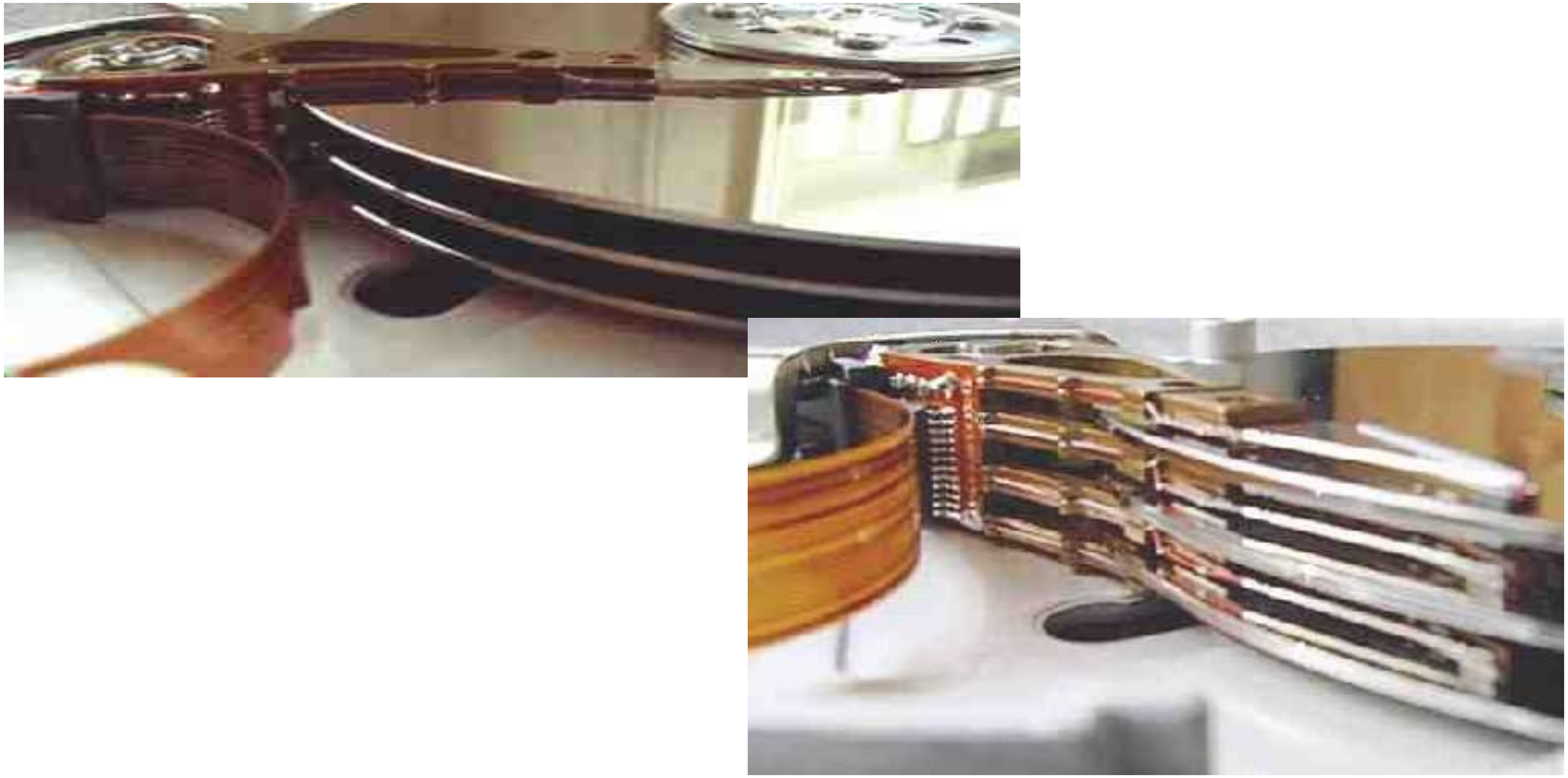
Understanding Disk Drives



Understanding Disk Drives



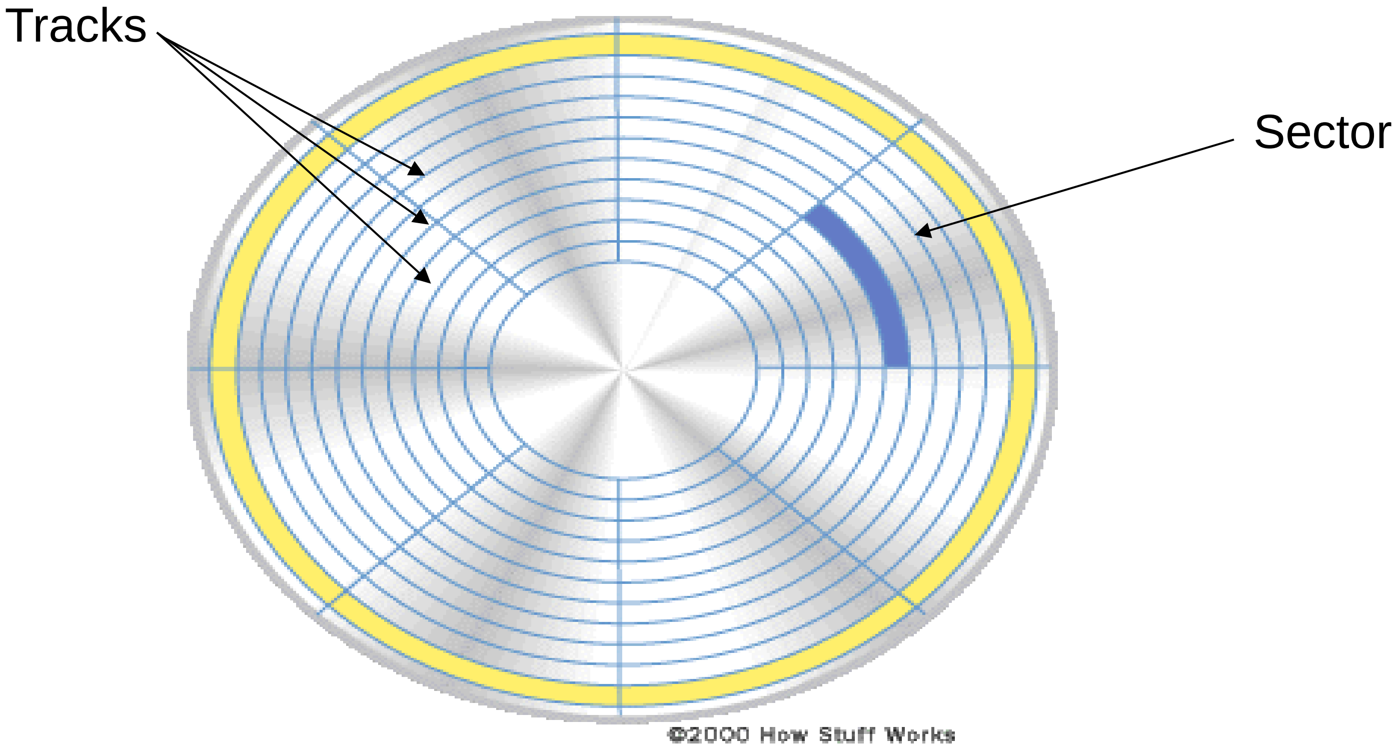
Understanding Disk Drives



Understanding Disk Drives

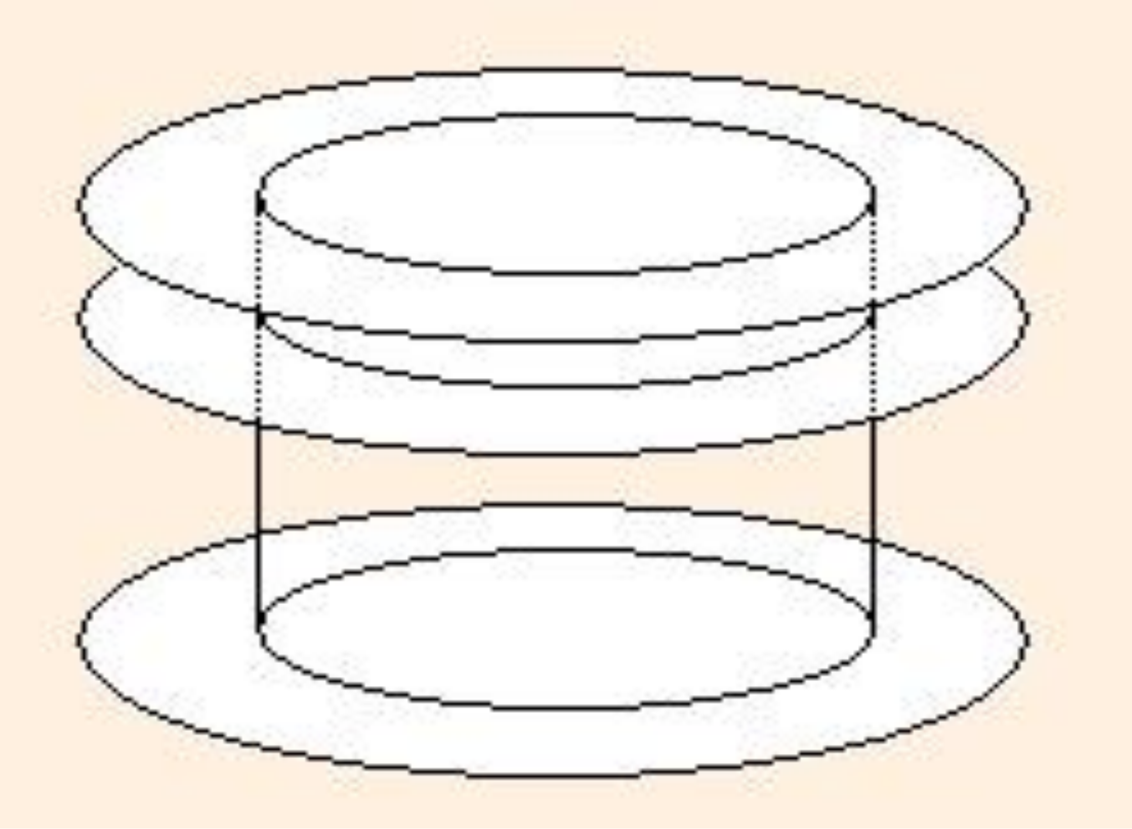
- Platters are divided into concentric rings called *tracks*
- Tracks are divide into wedge-shaped areas called *sectors*
  - A sector typically holds 512 bytes of data

Understanding Disk Drives



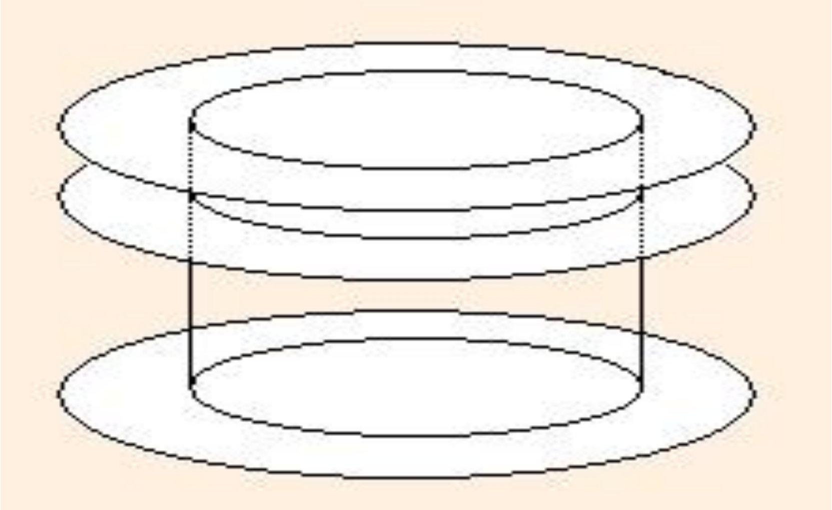
©2000 How Stuff Works

Understanding Disk Drives



Understanding Disk Drives

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



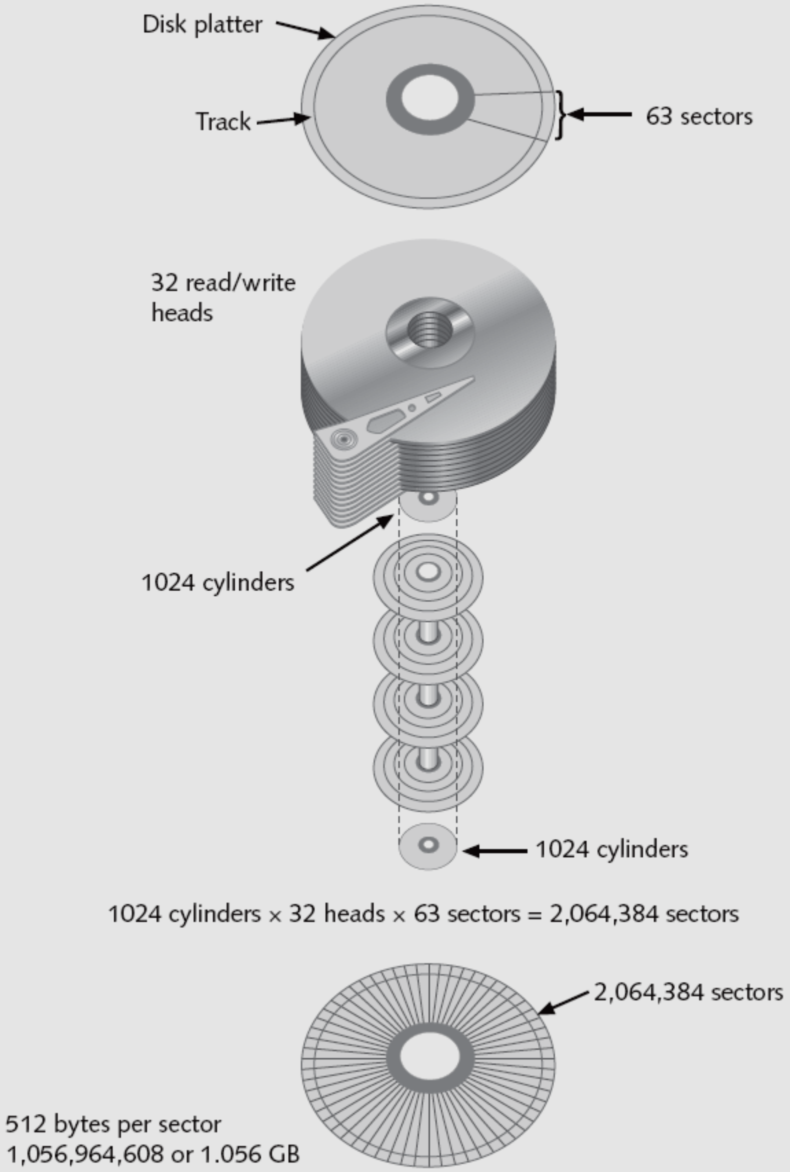


Figure 6-3 CHS calculation

Understanding Disk Drives

- Cylinder, head, sector (CHS) calculation
  - 512 bytes per sector
  - X sectors per track
  - Y tracks per cylinder
  - Number of bytes on a disk =
    - Cylinders (tracks) x Heads (platters) x sectors
- First track is track 0

Understanding Disk Drives

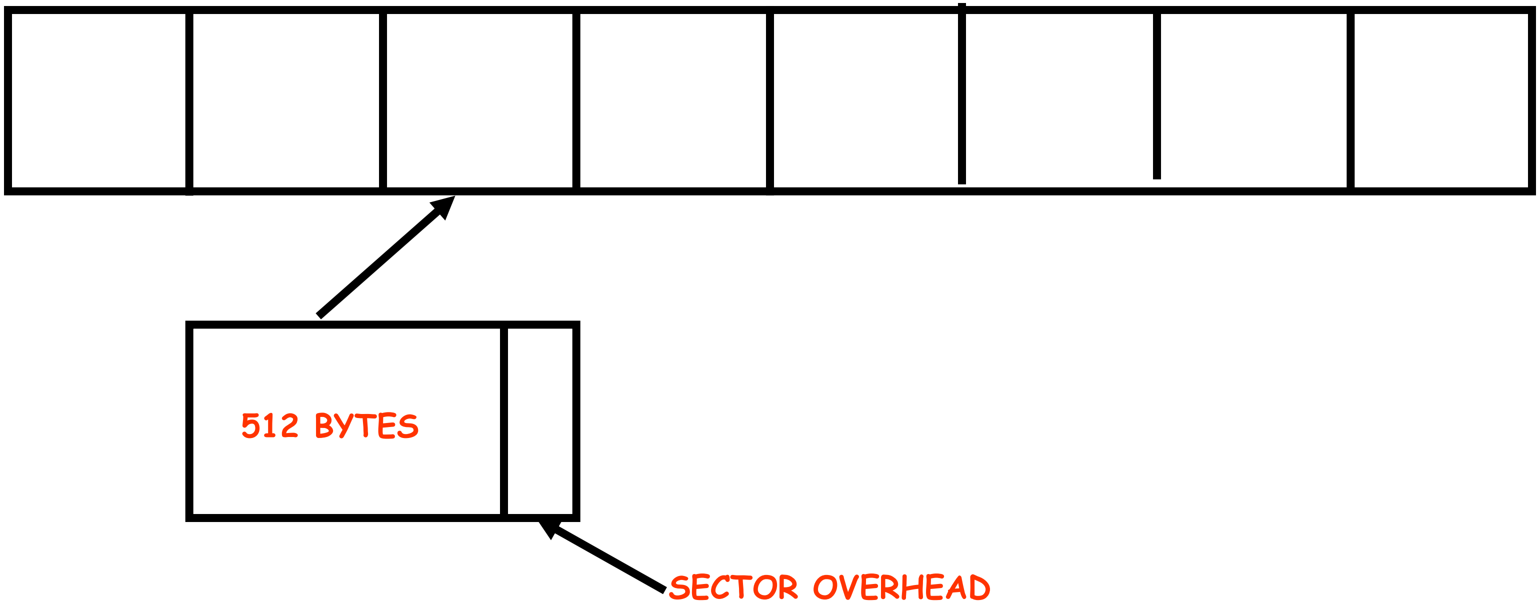
Lifecycle of Disk Drive

- Blank media
- Low level format
  - Performed at the factory
- Partition
- High level file system format
- Operating system install
- System operations

Low Level Format

- Low level formatting creates sectors
- Each sector holds 512 bytes + overhead bytes
- Overhead provides error correction and timing recovery
- Bad sectors remapped to redundant sectors by the HDD controller.

Low Level Format – continue

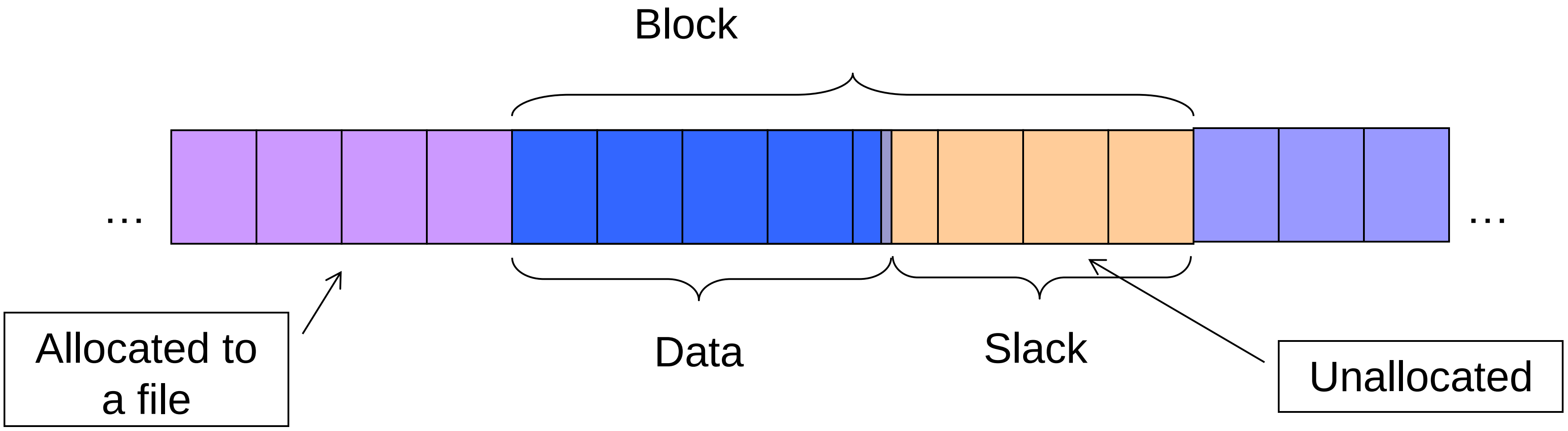


Slack Space

- Microsoft allocates disk space based on clusters
- Results in *drive slack*
  - Unused space between
  - End of file and
  - End of cluster the file is stored in

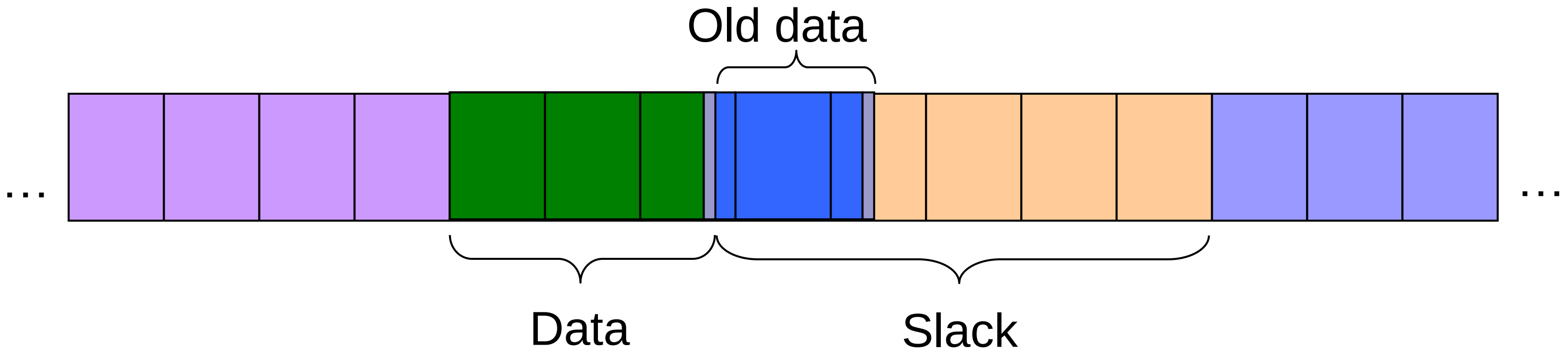
Examining FAT Disks

- Allocate 1 cluster (8 sectors)
- Record data
- Write End-of-File marker



Examining FAT Disks

- Delete the file (nothing happens to data itself)
- Create a new file



Questions?

m.owda@mmu.ac.uk