CSN11121/CSN11122
System Administration and Forensics

File System

28/10/2011

r.ludwiniak@napier.ac.uk

Lecture Objectives

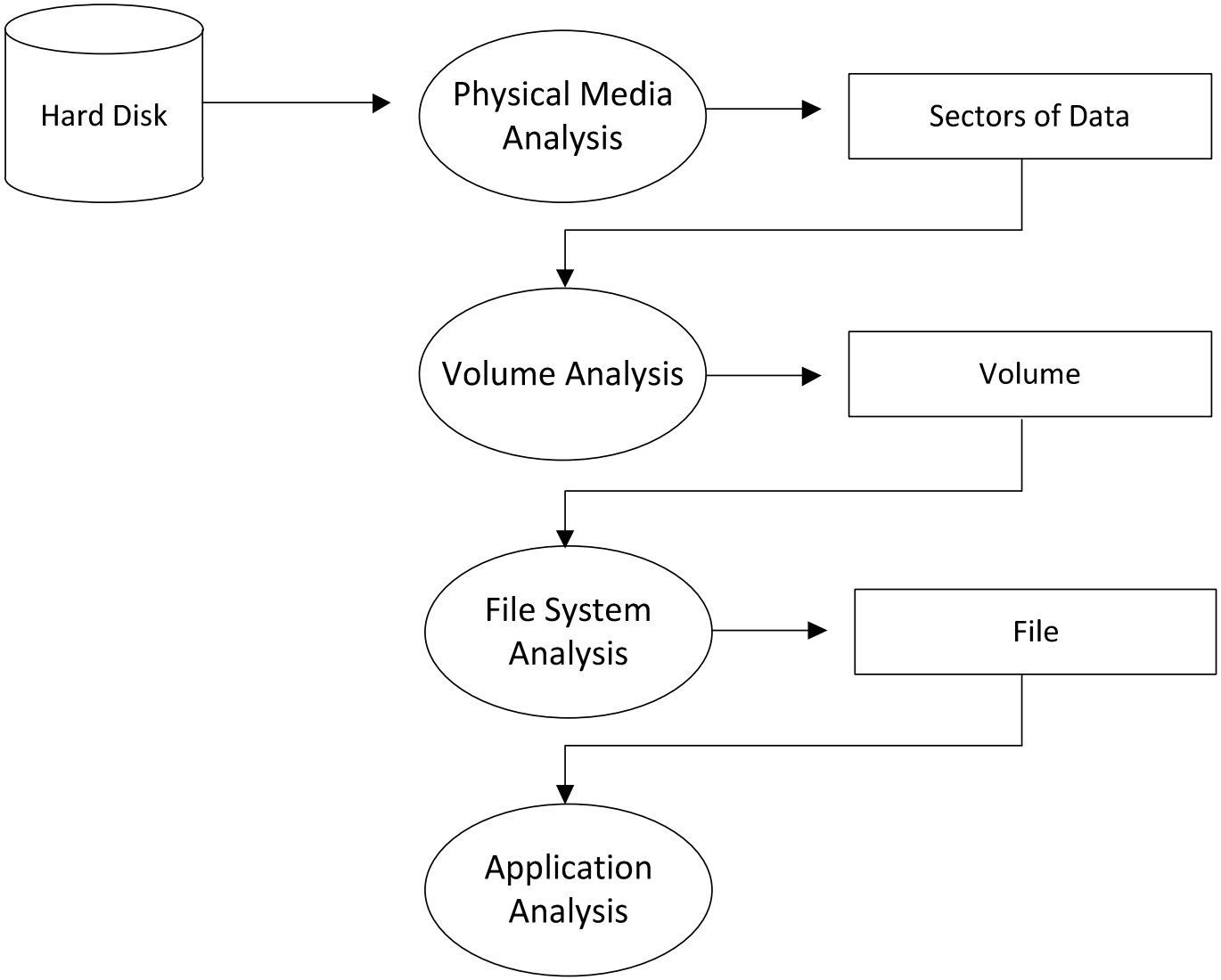
1. Investigative Process
   - Analysis Framework
2. File Systems
   - FAT
   - NTFS

Required Reading

- G.H. Fellow, "*The Joys and Complexity of the deleted file*" Digital Investigation, vol. 1, no. 5, pp. 89-93, 2005
- F. Bucholz, E. Spafford, *"On the role of file system metadata in digital forensics"*, Digital Investigation, vol 1, no 1, pp. 298-309.

INVESTIGATIVE PROCESS

Investigative process

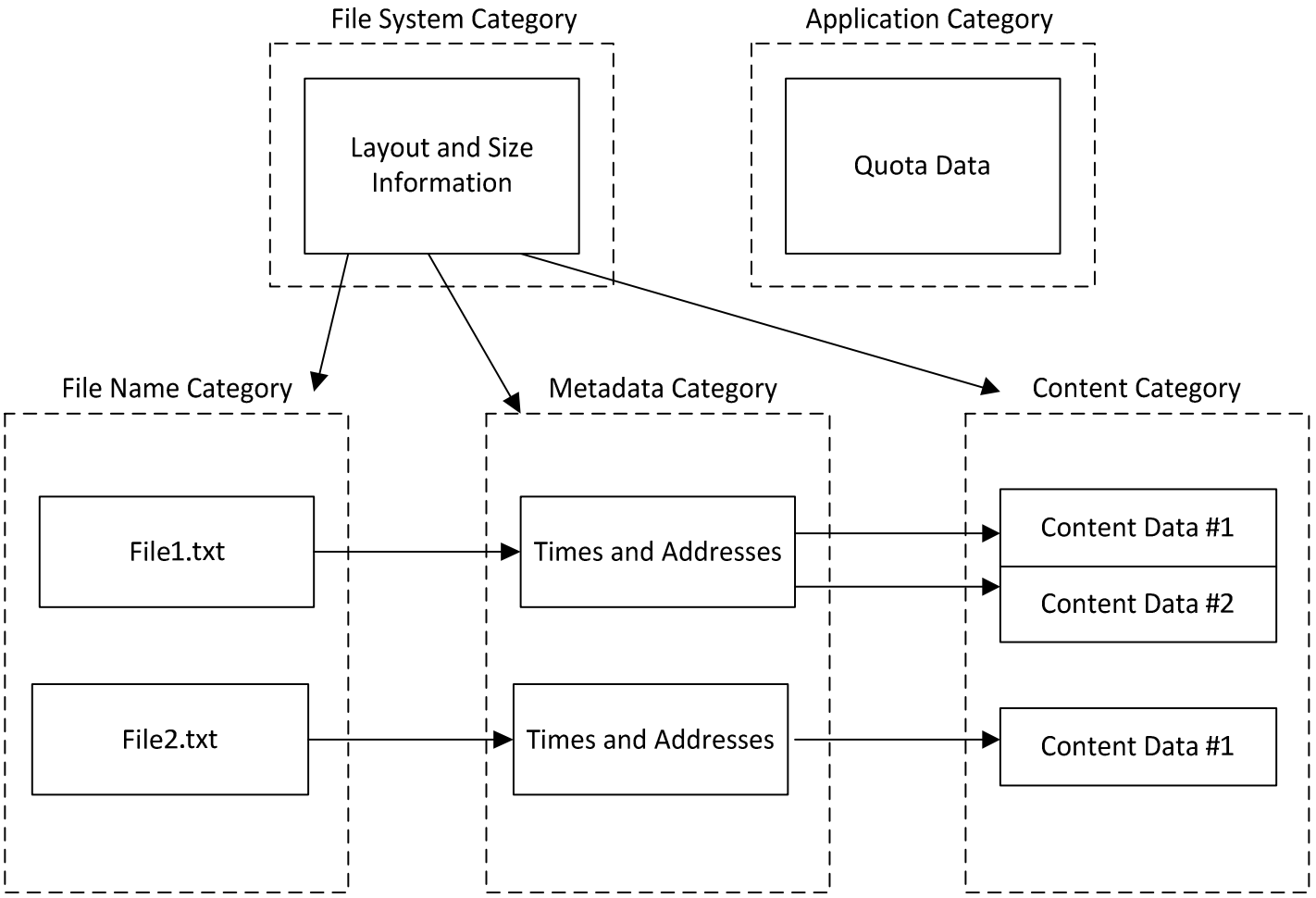


Analysis Framework

- B Carrier, **File System Forensic Analysis**
- Data Categories provide a basic reference model
  - Good for comparing different file system types
  - Also allows us to understand how to search using various tool types

Categories

- File System Category
  - Content Category
  - Metadata Category
  - File Name Category
  - Application Category


File System Category
Application Category
Layout and Size Information
Quota Data
File Name Category
Metadata Category
Content Category
File1.txt
Times and Addresses
Content Data #1
Content Data #2
File2.txt
Times and Addresses
Content Data #1

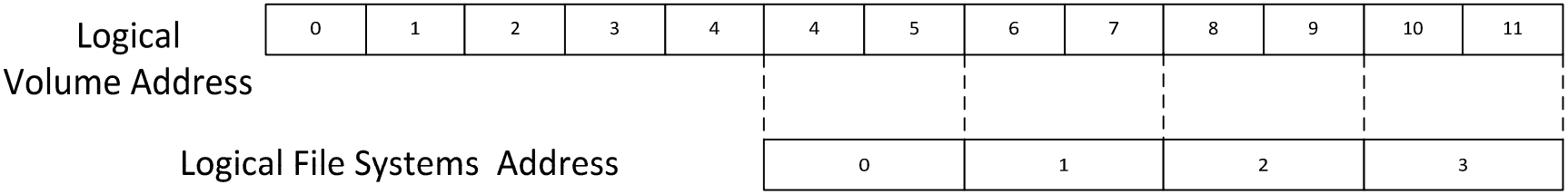
Analysis by Category

- Categories are important as they allow us to filter and search for files
- For example, if we want to search for all images
  - GIF, JPEG
  - Search for all files ending in .gif or .jpg, OR their file-header

File System Category

- All file systems have a general structure
- Informs of where to find data structures
  - Think of this as a map
- File system data resides on the first few sectors of the disk
- If corrupted, rebuilding by hand may need to occur
- Small amounts of data hiding can occur due to the sparse usage of the pre-allocated data structures

Disk and File System Layout

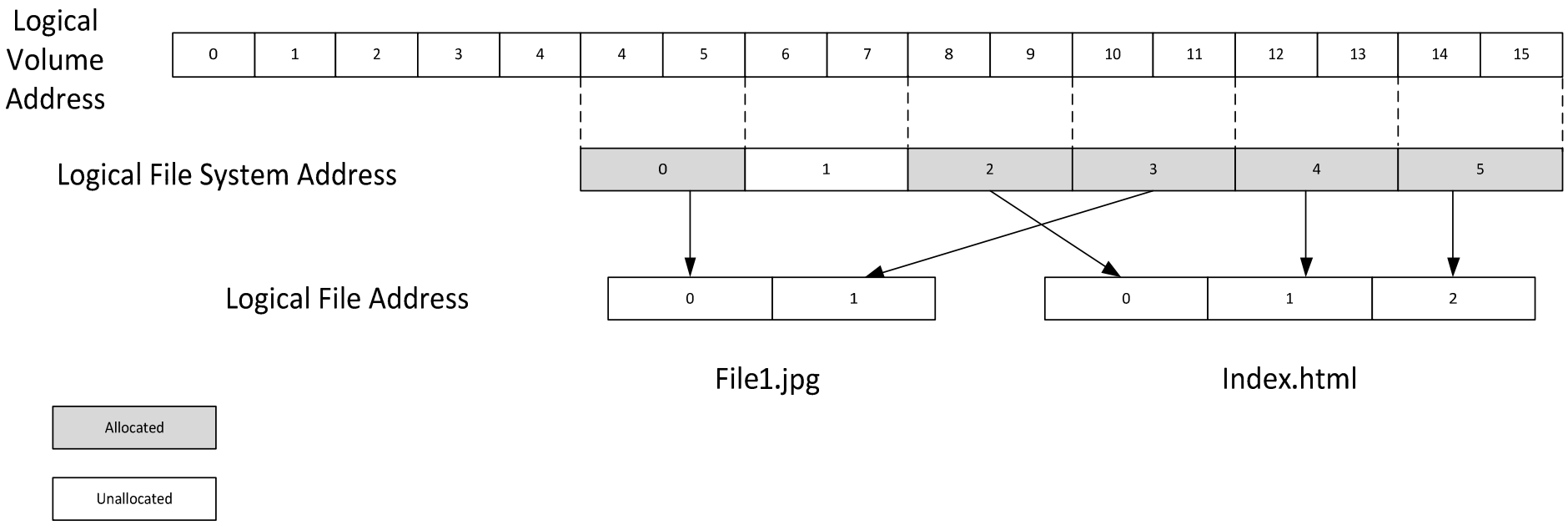


Content Category

- Actual content of the file
- Comprises the majority of the actual data
- Usually organised into standard-sized containers
  - Clusters
  - Blocks
  - ‘Data Unit’
- Allocation Strategy
  - On creating a new file
  - On Delete
  - Allocation order

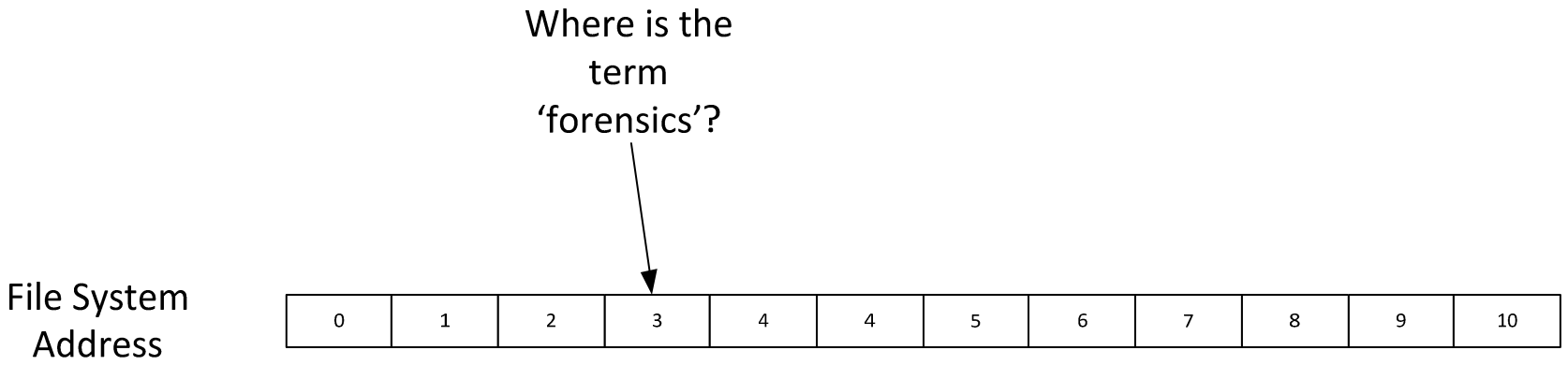
Metadata Category

- Where the descriptive data resides
  - Last access time
  - Data units allocated to the file
- Provides pointers to the address of the Logical File Address
- MAC times
  - Modification, Access, Change


Logical Volume Address
0
1
2
3
4
4
5
6
7
8
9
10
11
12
13
14
15
Logical File System Address
0
1
2
3
4
5
Logical File Address
0
1
0
1
2
File1.jpg
Index.html
Allocated
Unallocated

File-System Level Searching

- Logical File System-Level Search
  - E.g. looking for phrase ‘forensics’
  - Searches Every single allocated unit individually
  - Useful for both allocated and unallocated units

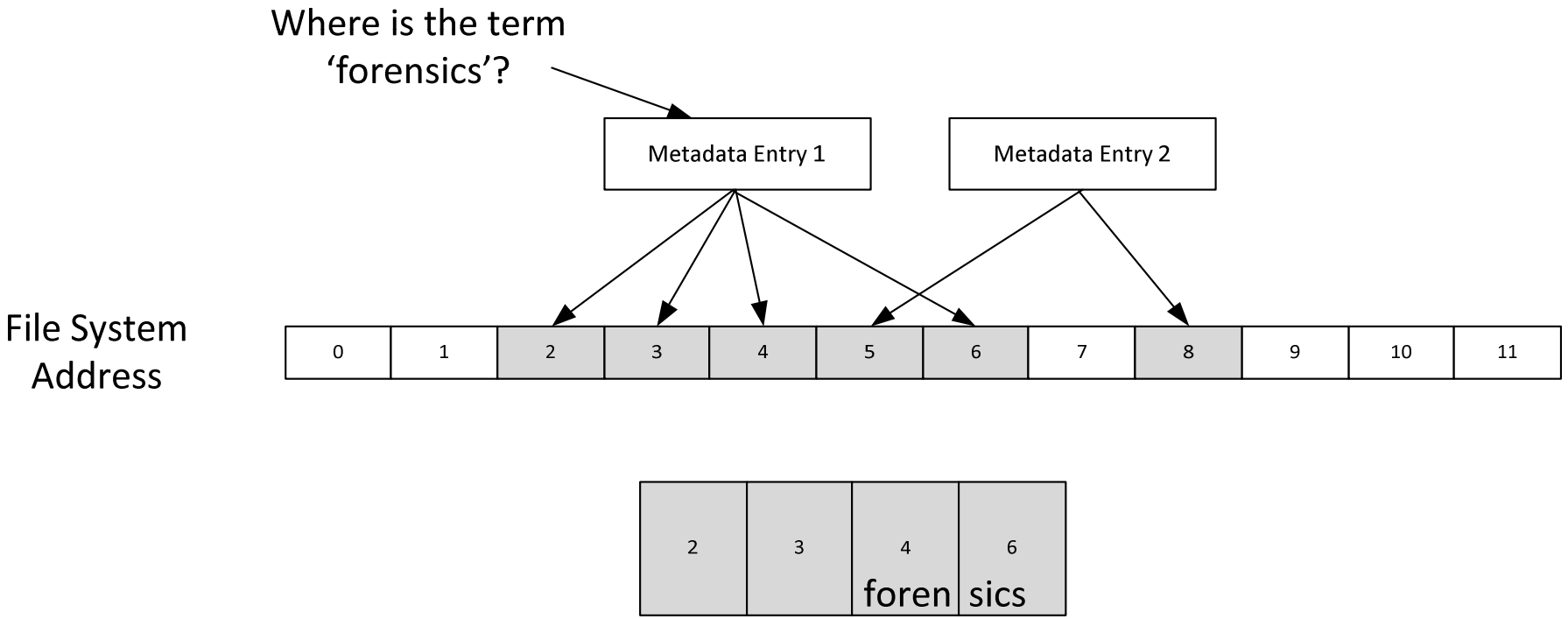


Linking Data Units to Files

- `ifind` allows you to discover the rest of the data units allocated to a file
- E.g. if we find something interesting in data unit 33, and want to look at the rest of the file

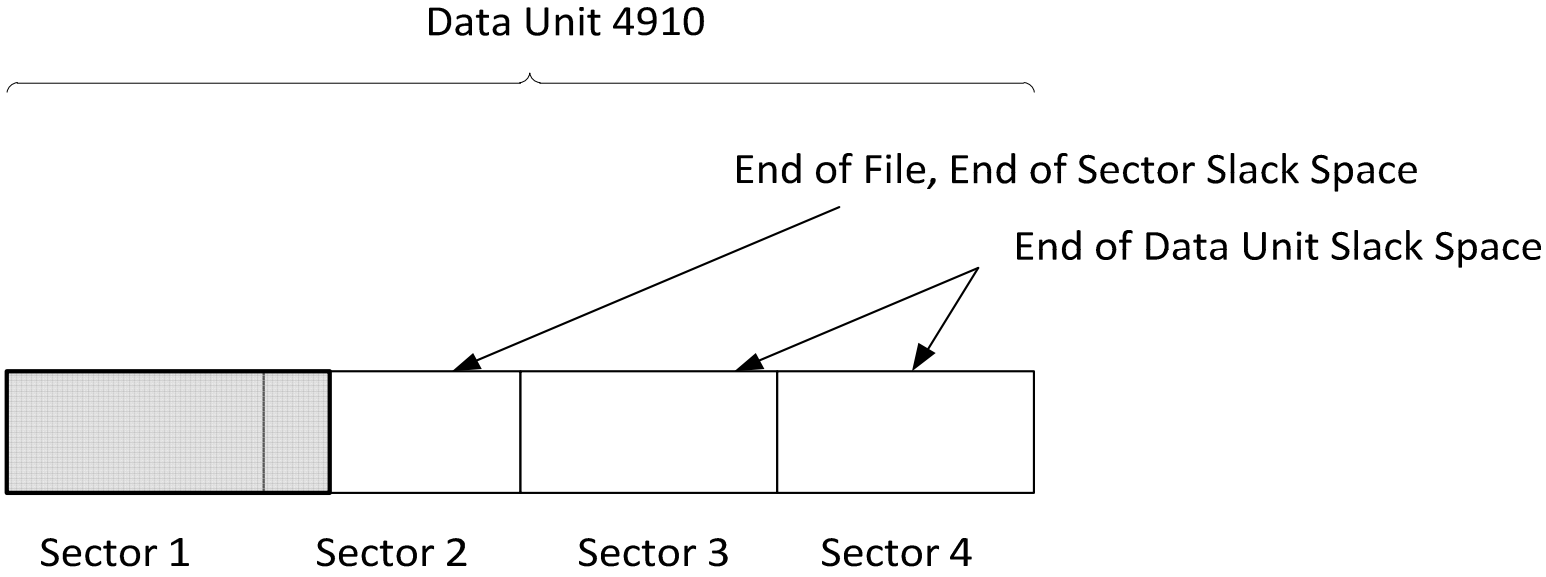
File Searching

- Logical File Search
  - Looks at the file-address level, takes into account fragmentation
  - Logical File Address allocated units only


Where is the term
‘forensics’?
Metadata Entry 1
Metadata Entry 2
File System
Address
0
1
2
3
4
5
6
7
8
9
10
11
2
3
4
foren
6
sics

Slack Space

- Disks are block-based
- Example:
  - 100 byte file
  - Needs to allocate full data unit (2,048 bytes)
  - The remaining 1,948 would be slack
- Two interesting areas:
  - Between end of file and end of sector
  - Allocated Data Unit Sectors that contain no file content


Data Unit 4910
End of File, End of Sector Slack Space
End of Data Unit Slack Space
Sector 1
Sector 2
Sector 3
Sector 4

Logical Metadata Viewing

- If we find a filename ‘illegalthings.txt’
- Want to view contents – need the metadata information to do so
- Most tools will allow you to look at the file by selecting the filename
- The tool `icat` in CAINE will show you the content data for a given metadata structure
  - *-s option gives slack space*
  - *-r attempts to recover deleted files*

Unallocated metadata

- File names may be deleted, but metadata may still exist
- Examine the metadata as this may contain evidence
- The tool `ils` in CAINE will list unallocated structures

Metadata Searching

- Allows time-based evaluation of activities
- Timelining events
- MAC
  - Modified
  - Access
  - Change
- Also owner ID and file permissions
- `mactime` tool in CAINE

File Name Category

- Includes names of files
- Allows a user to find a file by name, instead of its metadata entry
- When recovering files based on file names
  - We are still reliant on the metadata information
  - File names and metadata can get out of sync

File Name Analysis Techniques

- File Listings
- File Name Searching
- File Extension filtering and searching
- Can resolve metadata to file names using the `ffind` utility in CAINE
  - When evidence in a data unit is found, search for the metadata unit allocated, search for file name

Application Category

- Non-Essential Data
- Can be application specific data
- Search
  - `grep`
  - Data carving
  - File type sorting
    - Use `file` command to determine file types based on file signatures

FILE SYSTEMS

File Systems

- An operating system requires long term storage and retrieval
- A mechanism for storing files in hierarchy of files and directories
  - For example, a patient record filing system

File Systems

- Data
  - Files
  - Directories
- Metadata
  - Time stamps (modify, access, create/change, delete)
  - Owner
  - Security properties
- Structures
  - Superblock/Master File Table/File Access Table
  - inodes/clusters
  - data

File Systems

- More sophisticated data recovery requires deep knowledge of file system internals
- Structures that manage file system metadata
- Disk layout
- File deletion issues
- Many important file systems
  - DOS / Windows: FAT, FAT16, FAT32, NTFS
  - Unix: ext2, ext3, Reiser, JFS, … more
  - Mac: MFS, HFS, HFS+

File Systems: FAT

- FAT12, FAT16, FAT32
  - different size of addressable cluseter
- Common format for floppy disks (remember those?)
- Limited time/date information for FAT files
  - Last write date/time is always available
  - Creation date/time is optional and may not be available
  - Last access DATE ONLY is optional and may not be available
- Short file names (8.3) on FAT12 and FAT16
- No security features
- Long names for FAT32

FAT: Short Filename Storage

- `"foo.bar"       -> "FOO     BAR"`
- `"FOO.BAR"       -> "FOO     BAR"`
- `"Foo.Bar"       -> "FOO     BAR"`
- `"foo"           -> "FOO        "`
- `"foo."          -> "FOO        "`
- `"PICKLE.A"      -> "PICKLE  A  "`
- `"prettybg.big"  -> "PRETTYBGBIG"`

- Note case is not significant
- "." between primary filename and extension is implied (not actually stored)
- Further, everything is space-padded

FAT: More Dir Entry Details

- Date format:
  - Bits 0–4: Day of month, valid value range 1-31 inclusive.
  - Bits 5–8: Month of year, 1 = January, valid value range 1–12 inclusive.
  - Bits 9–15: Count of years from 1980, valid value range 0–127 inclusive (1980–2107).
- Time Format:
  - A FAT directory entry time stamp is a 16-bit field that has a granularity of 2 seconds
  - Bits 0–4: 2-second count, valid value range 0–29 inclusive (0 – 58 seconds).
  - Bits 5–10: Minutes, valid value range 0–59 inclusive
  - Bits 11–15: Hours, valid value range 0–23 inclusive

FAT: Long Filenames

- Summary: a kludge to add support without changing short-name handling
- Up to 255 characters in pathname component
- Total pathname no longer than 260
- More supported characters
- Leading/trailing spaces ignored
- Internal spaces allowed
- Leading/embedded “.” allowed
- Trailing “.” are ignored
- Stored case-sensitive
- Processed case-insensitive (for compatibility)
- File created with short name (uses “~1”, “~2”, etc. suffix)

FAT Layout

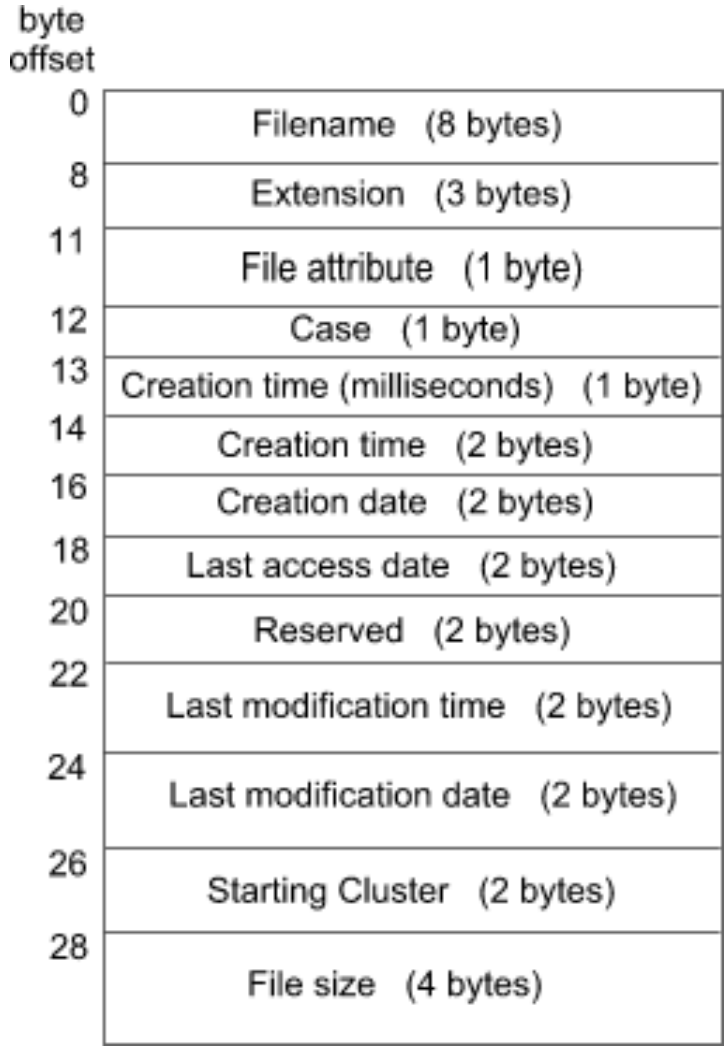

Boot Sector
Reserved Area
FAT Area
Root Directory
Data Area

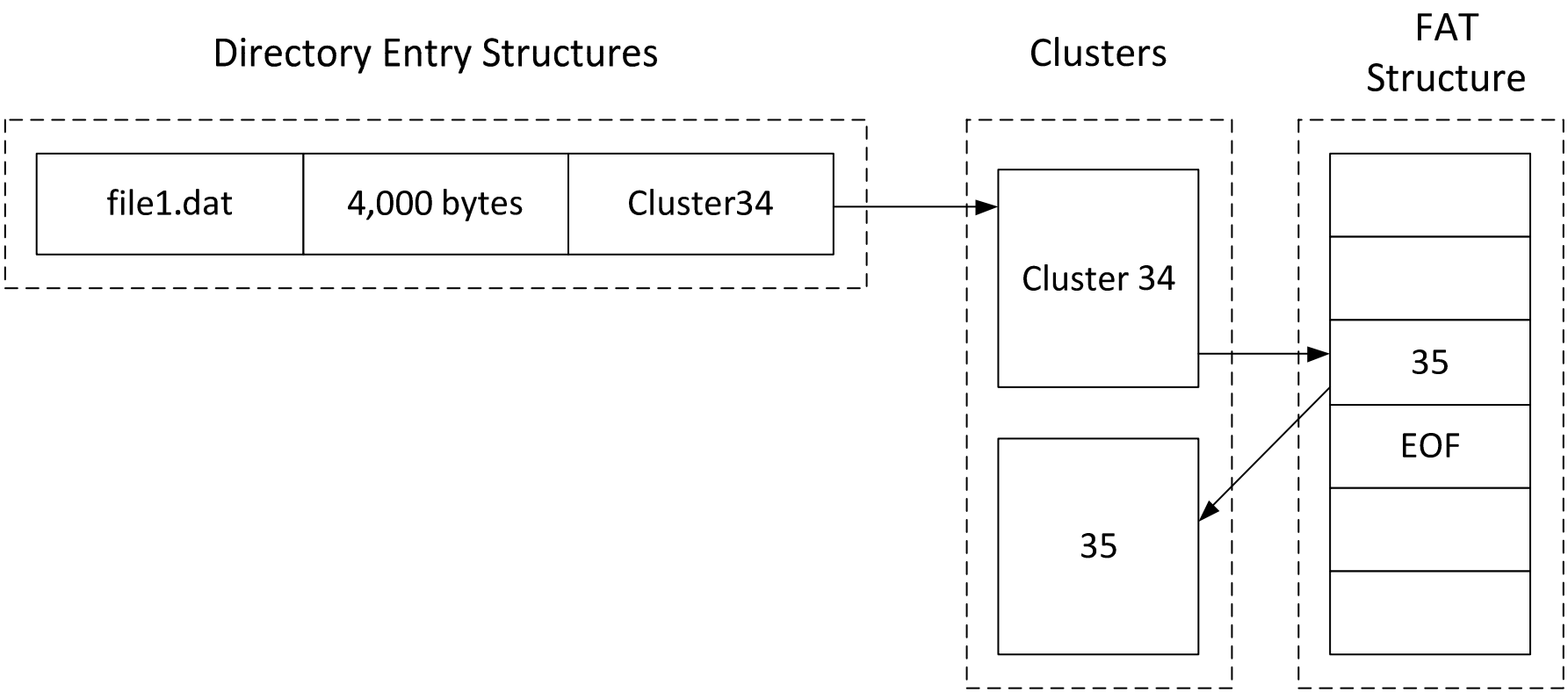
FAT File System Categories

	File System	Content	Metadata	File Name	Application
FAT	Boot Sector, FSINFO	Clusters, FAT	Directory Entries, FAT	Directory Entries	N/A

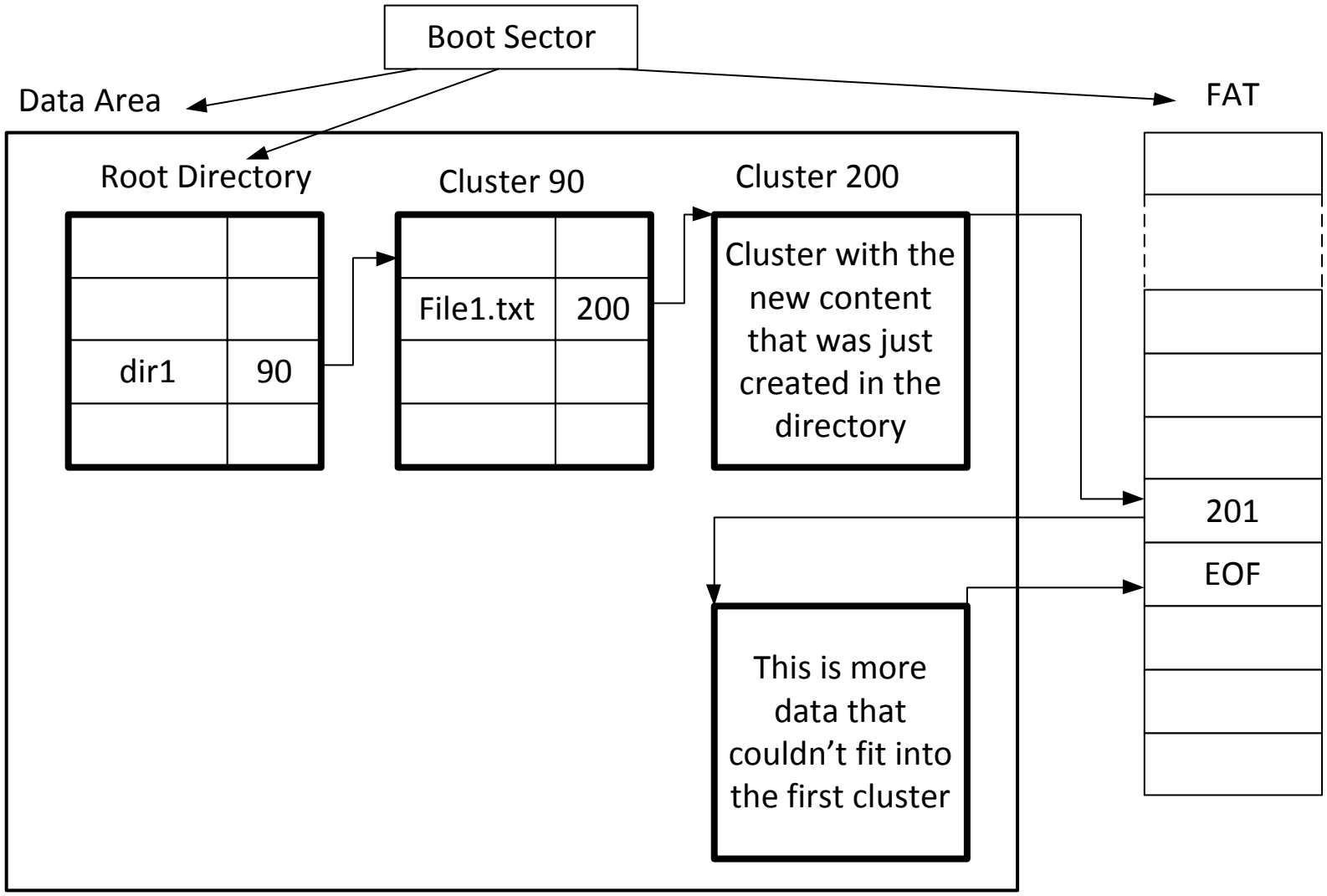
FAT32 Directory Structure

- An ordinary cluster chain
- Directory entry
  - 32 bytes for both files and directories
  - For deleted entries, first byte is set to 0xE5
  - First two entries in subdirectories are . and..
  - Uses more than one entry to implement long filenames

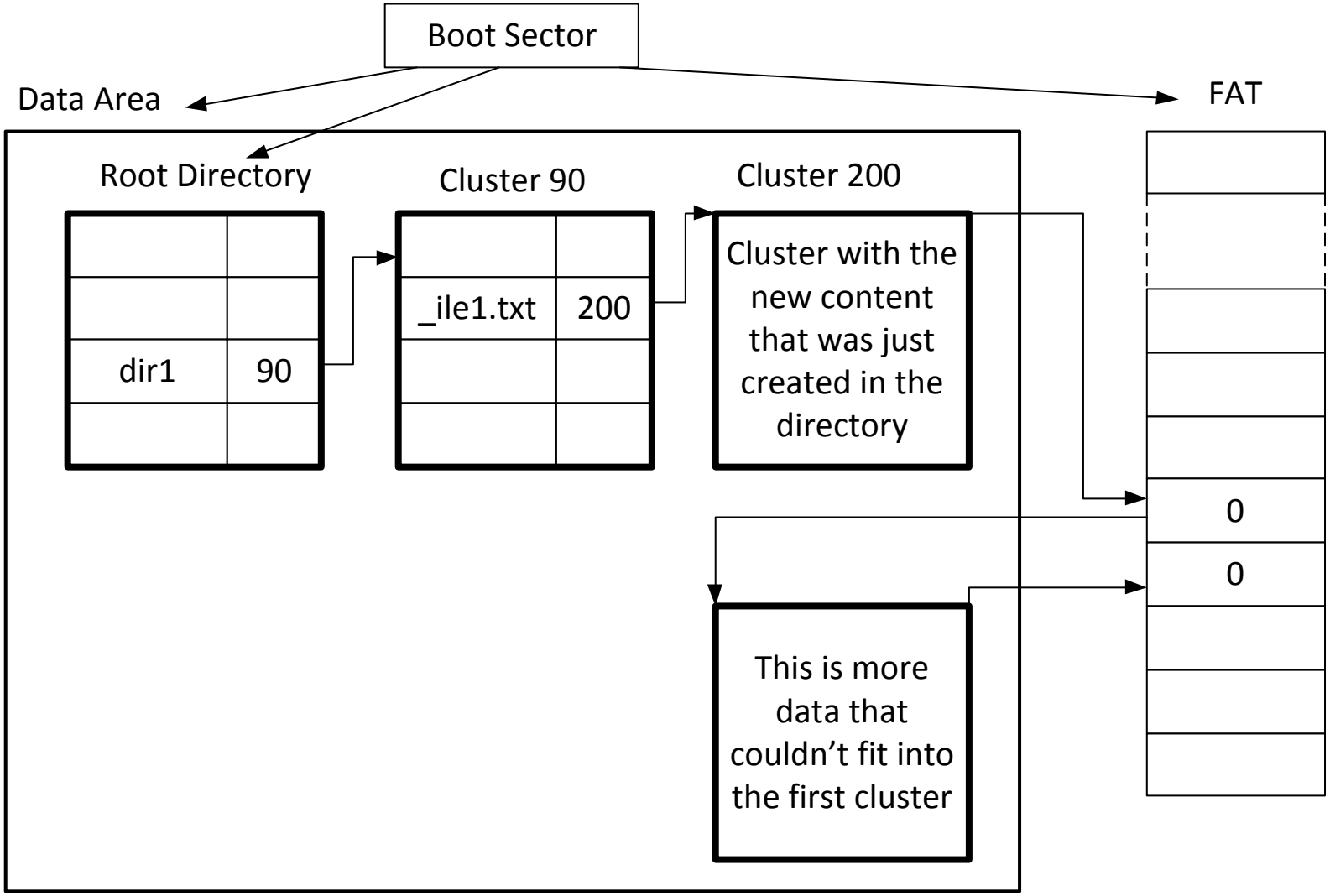



Directory Entry Structures
Clusters
FAT Structure
file1.dat
4,000 bytes
Cluster34
Cluster 34
35
35
EOF

Creating a File



File Deletion



FAT File Deletion

- First letter of the file is overwritten with xE5
- FAT pointers to allocation areas set to zero
  - Indicated that they are ready for re-use

File Systems: NTFS

- Master File table grows, never shrinks
- B-tree algorithm used for file tree
  - re-“balances” file system tree when tree changes
  - creating or deleting a file can cause entire tree to change and can overwrite nodes that were marked as free but still had information in them
- Lots of attributes on files, can be confusing (e.g., which access time is the “official” one to use)
  - most useful attributes are MAC times
- Master File Table (MFT)
  - Contains information about all files and directories
  - Each has at least one entry in the table

File System Metadata Files

- Do not confuse with file metadata
- First 16 MFT entries reserved for files that describe the file system
- Listed in the root directory
- Each file begins with ‘$’

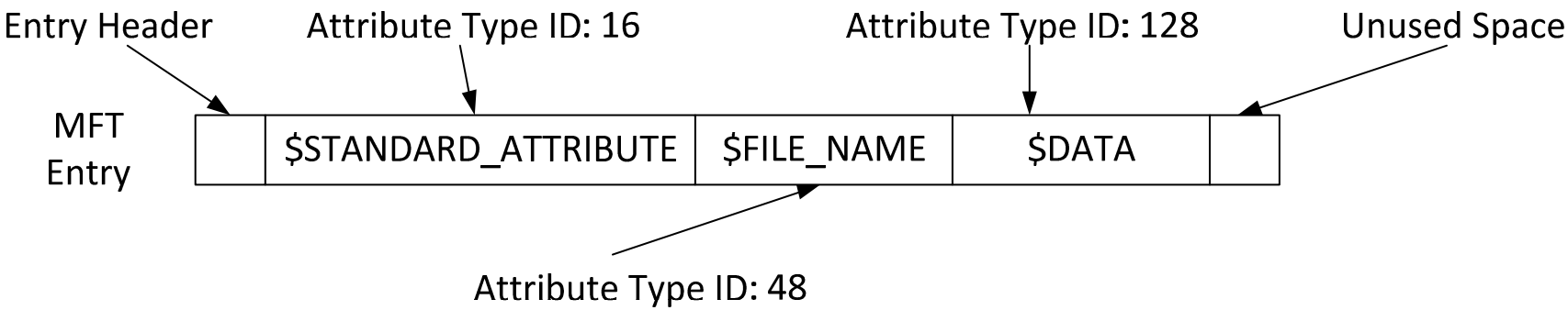
File System Metadata Files

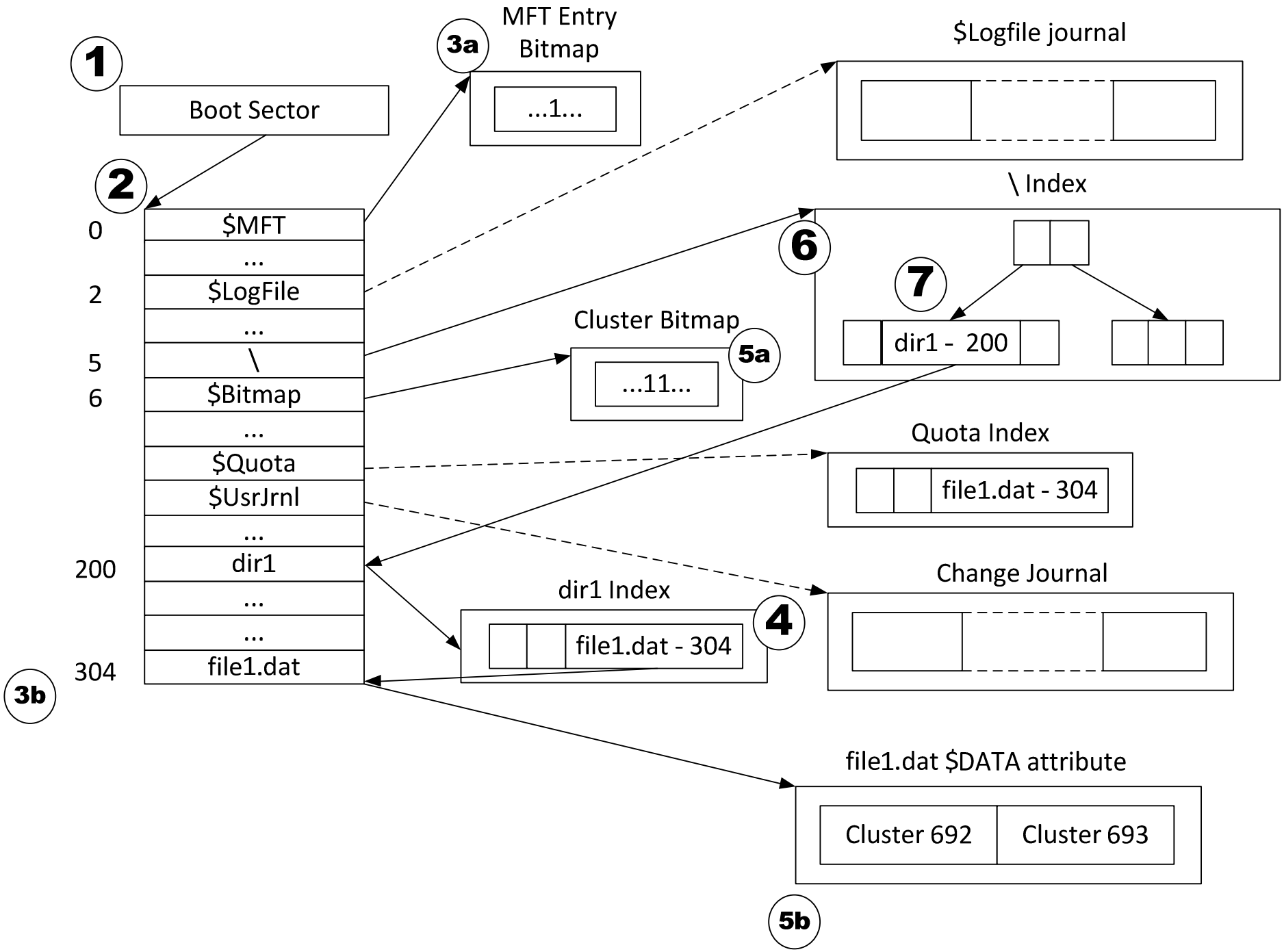
Entry	File Name	Description
0	$MFT	MFT entry
1	$MFTMirr	Backup of the MFT
2	$LogFile	Contains journal information for metadata transactions
3	$Volume	Volume Information: label, identifier, version
4	$AttrDef	Attribute information: identifier values, name, sizes
5	.	Root directory of the files system
6	$Bitmap	Contains allocation status for each cluster
7	$Boot	Contains the boot code
8	$BadClus	Contains clusters that have bad sectors

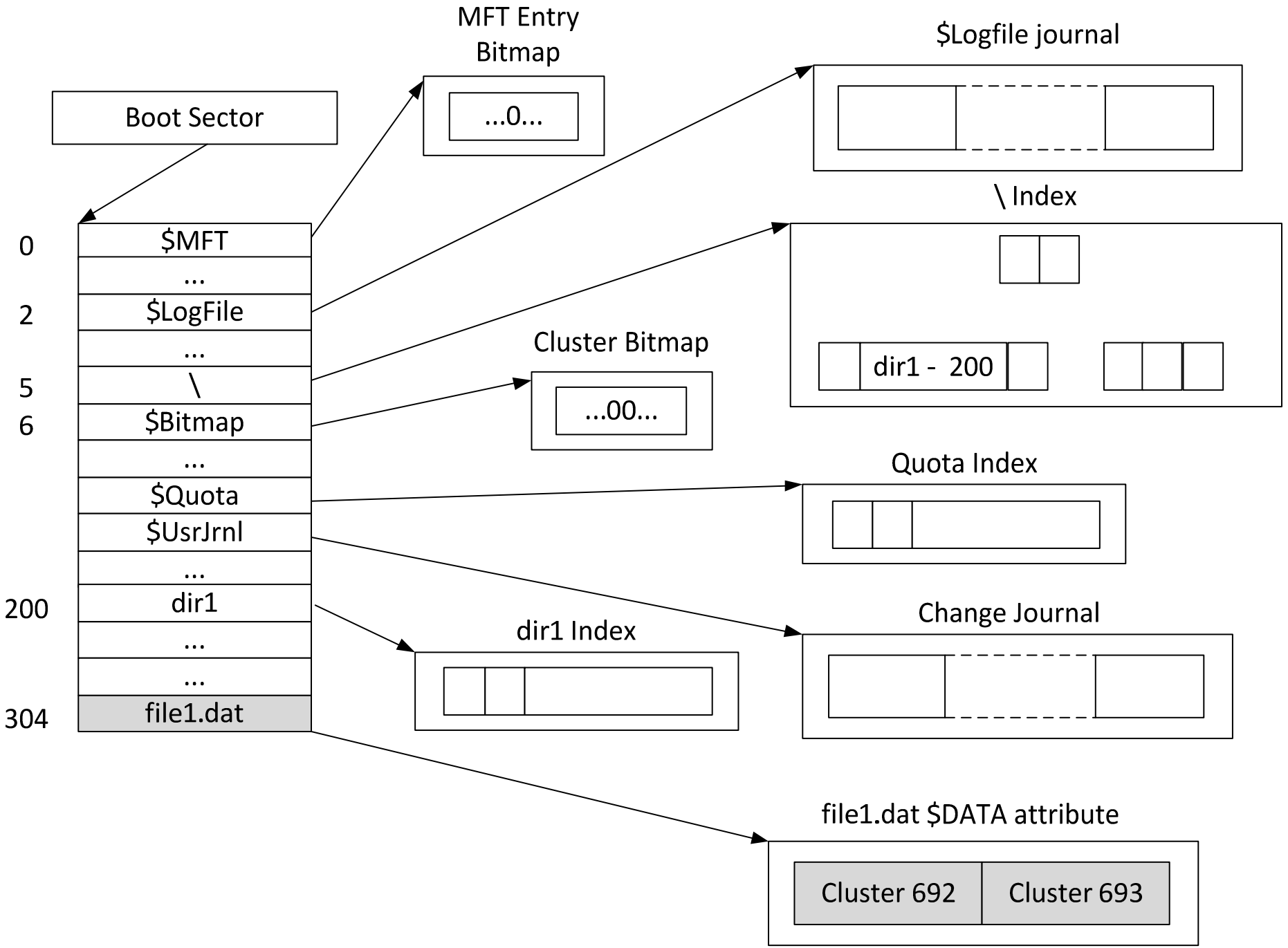
Data Structure Categories

	File System	Content	Metadata	File Name	Application
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

NTFS Layout


Entry Header
Attribute Type ID: 16
Attribute Type ID: 128
Unused Space
MFT Entry
$STANDARD_ATTRIBUTE
$FILE_NAME
$DATA
Attribute Type ID: 48


1
Boot Sector
2
0
$MFT
...
2
$LogFile
...
5
\
6
$Bitmap
...
$Quota
$UsrJrnl
...
200
dir1
...
...
304
file1.dat
3b
3a
MFT Entry
Bitmap
...1...
$Logfile journal
\ Index
6
7
dir1 - 200
Cluster Bitmap
5a
...11...
Quota Index
file1.dat - 304
Change Journal
dir1 Index
4
file1.dat - 304
file1.dat $DATA attribute
Cluster 692
Cluster 693
5b


MFT Entry
Bitmap
...0...
$Logfile journal
Boot Sector
\ Index
0
$MFT
...
2
$LogFile
...
Cluster Bitmap
...00...
dir1 - 200
5
\
6
$Bitmap
...
Quota Index
$Quota
$UsrJrnl
...
200
dir1
Change Journal
dir1 Index
...
...
304
file1.dat
file1.dat $DATA attribute
Cluster 692
Cluster 693

MFT List of possible attributes

- Defined in $AttrDef entry of MFT, but default is:
  - 0x10 STANDARD_INFORMATION
  - 0x20$ATTRIBUTE_LIST
  - 0x30$FILE_NAME0
  - X40 (NT) $VOLUME_VERSION (2K) $OBJECT_ID
  - 0x50 $SECURITY_DESCRIPTOR
  - 0x60$VOLUME_NAME
  - 0x70 $VOLUME_INFORMATION
  - 0x80$DATA
  - 0x90$INDEX_ROOT
  - 0xA0$INDEX_ALLOCATION
  - 0xB0$BITMAP
  - 0xC0 (NT) $SYMBOLIC_LINK, (2K) $REPARSE_POINT
  - 0xD0$EA_INFORMATION
  - 0xE0$EA0xF0NT$PROPERTY_SET
  - 0x100 (2K) $LOGGED_UTILITY_STREAM

MFT Attribute Layout

- Attributes can be resident or non-resident.
- Beginning is always the same:
  - 0x00 Attribute Type Identifier
  - 0x04 Length of Attribute
  - 0x08 non-resident flag
  - 0x09 length of name
  - 0x0a offset to name
  - 0x0c flags

MFT Attribute Example

Standard Info Attribute Layout

0x00	8		File Creation Time
0x08	8		File Alteration Time
0x10	8		MFT Change
0x18	8		File Read Time
0x20	4		DOS File Permissions
0x24	4		Maximum number of versions
0x28	4		Version number
0x2C	4		Class ID
0x30	4	2K	Owner ID