Introduction to Windows: The Registry and NTFS

What is the windows registry?

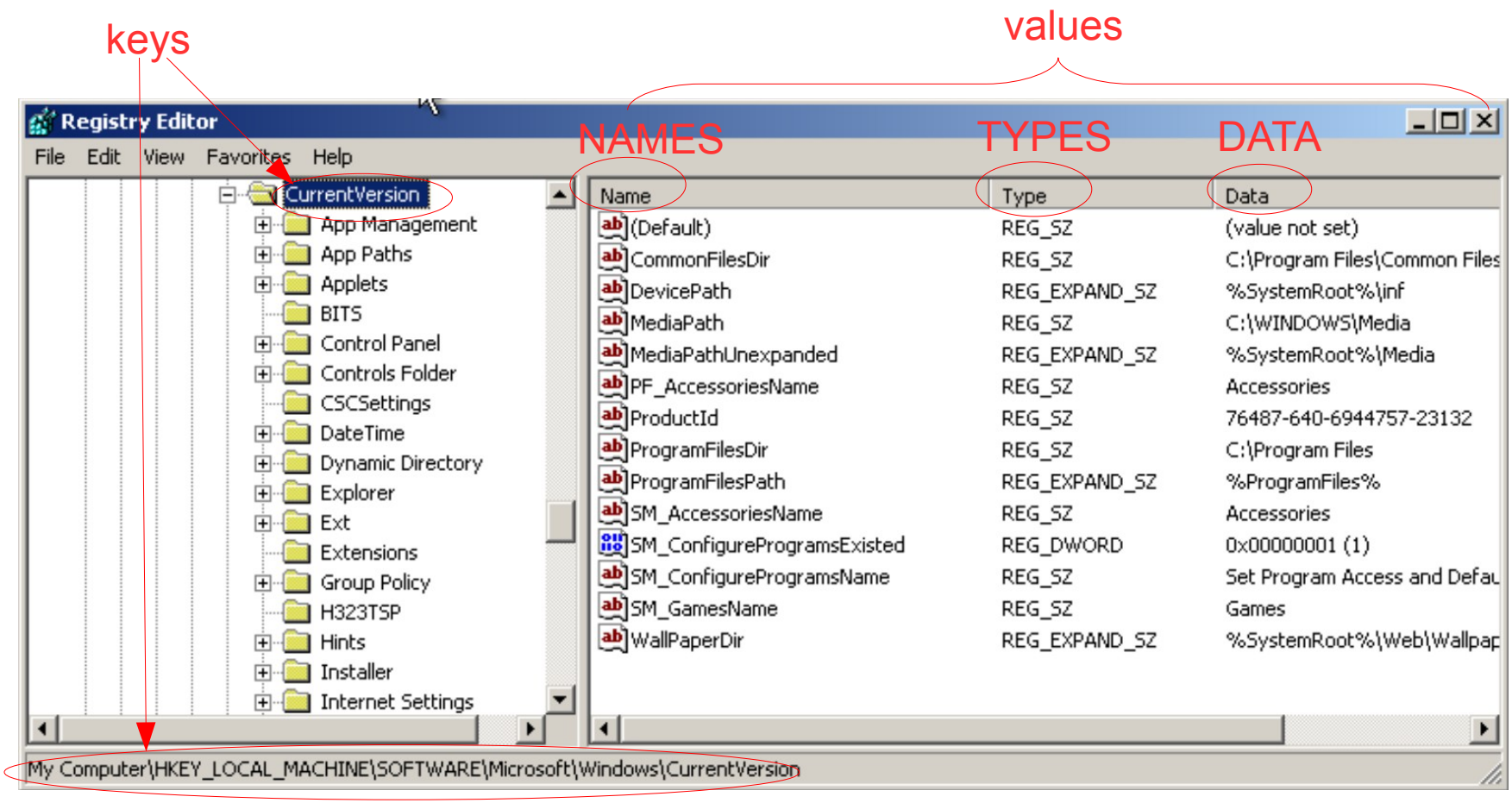
- Database of settings and options
  - Can be low-level (OS settings)
  - Can be used by applications (optionally)
- Replaced “.INI” text files for storing runtime configuration
  - Binary format for faster parsing
  - Type checking (is “size” a number?)
  - Roughly parallel to Linux /etc/ directories and hidden files/directories in the home folder

Windows Registry Structure

- Basic Elements: keys and values
  - A key can hold other keys or values
  - A value is a name/data pair:

    ProgramFilesDir C:\Program Files
- All registry value data has an associated type:
  - REG_BINARY
  - REG_SZ
  - REG_DWORD
  - others

Windows Registry Components



The registry is like a file system

- Hierarchy of keys forms a tree
- Branches of the tree are stored in “hives”
  - .reg files, mounted into the registry to form the whole structure
  - Some parts are virtualized (hardware)
- Keys are analogous to folders
- Values are analogous to files
- Value data is analogous to file content

Windows Registry Analysis

- Live System
  - RegEdit
  - Sysinternals autoruns tool
    - http://technet.microsoft.com/en-us/sysinternals/bb963902.aspx
- Offline
  - Fewer available tools
  - RegRipper
    - http://regripper.wordpress.com/

Reading the registry

- Key names can be unfriendly:
  - HKLM, HKCC,HKCR, HKCU, HKU
- Learn some abbreviations:
  - HKCU = HKEY_CURRENT_USER
    = Handle to registry key for the current user
  - “Handle” is a method of accessing something
- Google is helpful here.
- The full path can be very informative:
  - HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\IE4\Setup

Interesting Registry Keys

- Many Most Recently Used (MRU) lists
- USB Devices
  - HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Enum\USBSTOR
- Autostarts
  - Software that starts automatically
    - Use sysinternals autoruns utility
  - HKLM\SOFTWARE\Microsoft\Windows\ CurrentVersion\Run
- Wireless SSIDs
  - HKLM\SOFTWARE\Microsoft\Windows NT\CurrentVersion\NetworkList\Profiles

NTFS

NTFS Overview

- New Technology File System
- Developed by Microsoft
- Successor to FAT (File Allocation Table)
  - More metadata
  - Improved reliability
  - Many other features
    - Sparse files, Compression, etc.
- Proprietary—hard to get good information
- How MS does it vs. how it's defined

NTFS References

- File System Forensic Analysis, Brian Carrier
- Microsoft "TechNet":
  - http://technet.microsoft.com/en-us/library/cc781134(v=ws.10).aspx
- Wikipedia:
  - http://en.wikipedia.org/wiki/NTFS
- Much information in these slides is paraphrased from the above sources (especially Carrier)

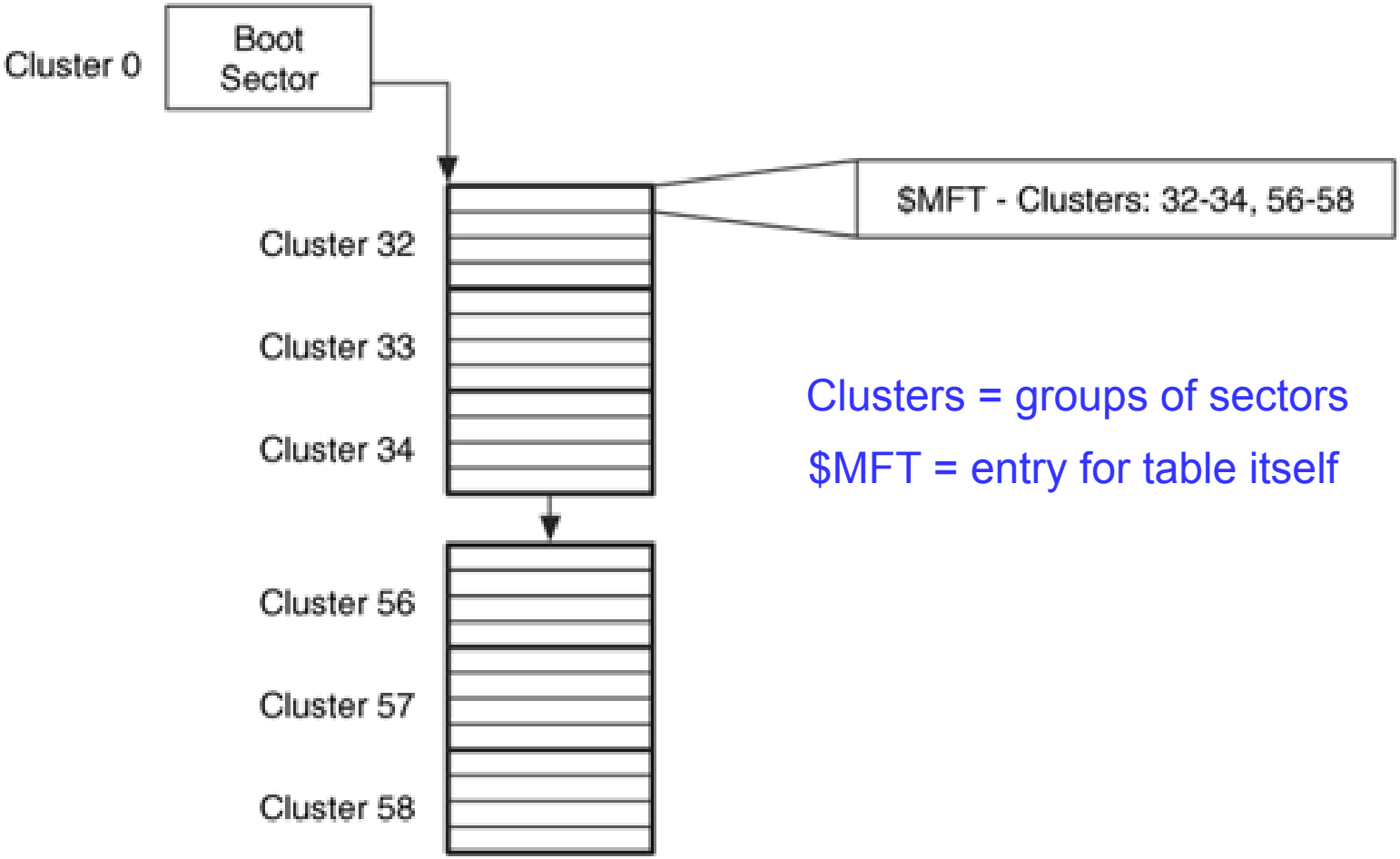
NTFS Concepts: Everything is a file

- No data “outside” files
  - Not just referring to user / application data:
  - Filesystem bookkeeping data is also stored in files
- Files can go anywhere
- Therefore, filesystem data could be anywhere
  - Exception: boot sector has to be the first sector of the file system
- In practice, there may be more structure

NTFS Concepts: Master File Table

- The MFT keeps track of every file (and therefore all data)
  - At least one entry for every file/directory
  - *Including itself* (since it's a file)
- Starts small and adds entries as needed
  - In Windows, entries are not deleted

Master File Table Layout



Clusters = groups of sectors

$MFT = entry for table itself

Image from *File System Forensic Analysis* by Brian Carrier, p. 275

MFT Entries

- An MFT entry is 1024 bytes
  - Only the first 42 bytes are reserved
    - 12 fields
    - Signature, link count, used size of MFT entry, etc.
  - Remaining 982 bytes are flexible
    - Contain arbitrary “attributes”
    - We'll come back to these
  - Makes the system extensible
  - Also makes life a bit hard for us

MFT Entry Structure

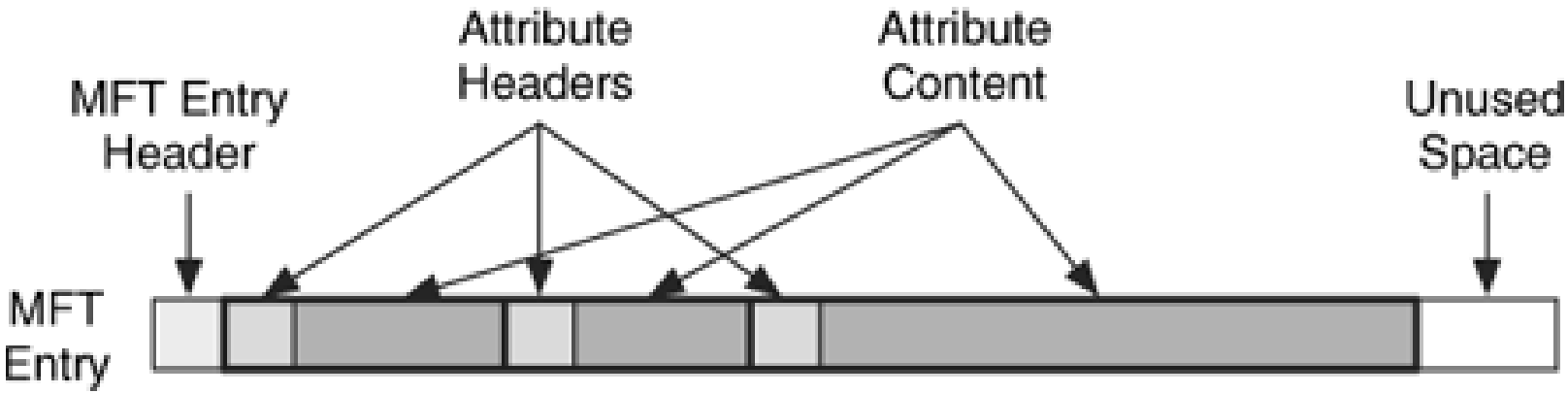


 Image from *File System Forensic Analysis* by Brian Carrier, p. 279

MFT Entry Layout

- Every MFT Entry has a 48-bit address
  - Called the "file number"
  - Starts with 0; one for each entry, most are unused
- Entry also contains a 16-bit "sequence number"
  - How many times the entry has been reallocated.
- Sequence number is *in* the entry, file number is the *location* of the entry
- Sequence number + file number

  = 64-bit "File Reference Address"

Building the File Reference Address

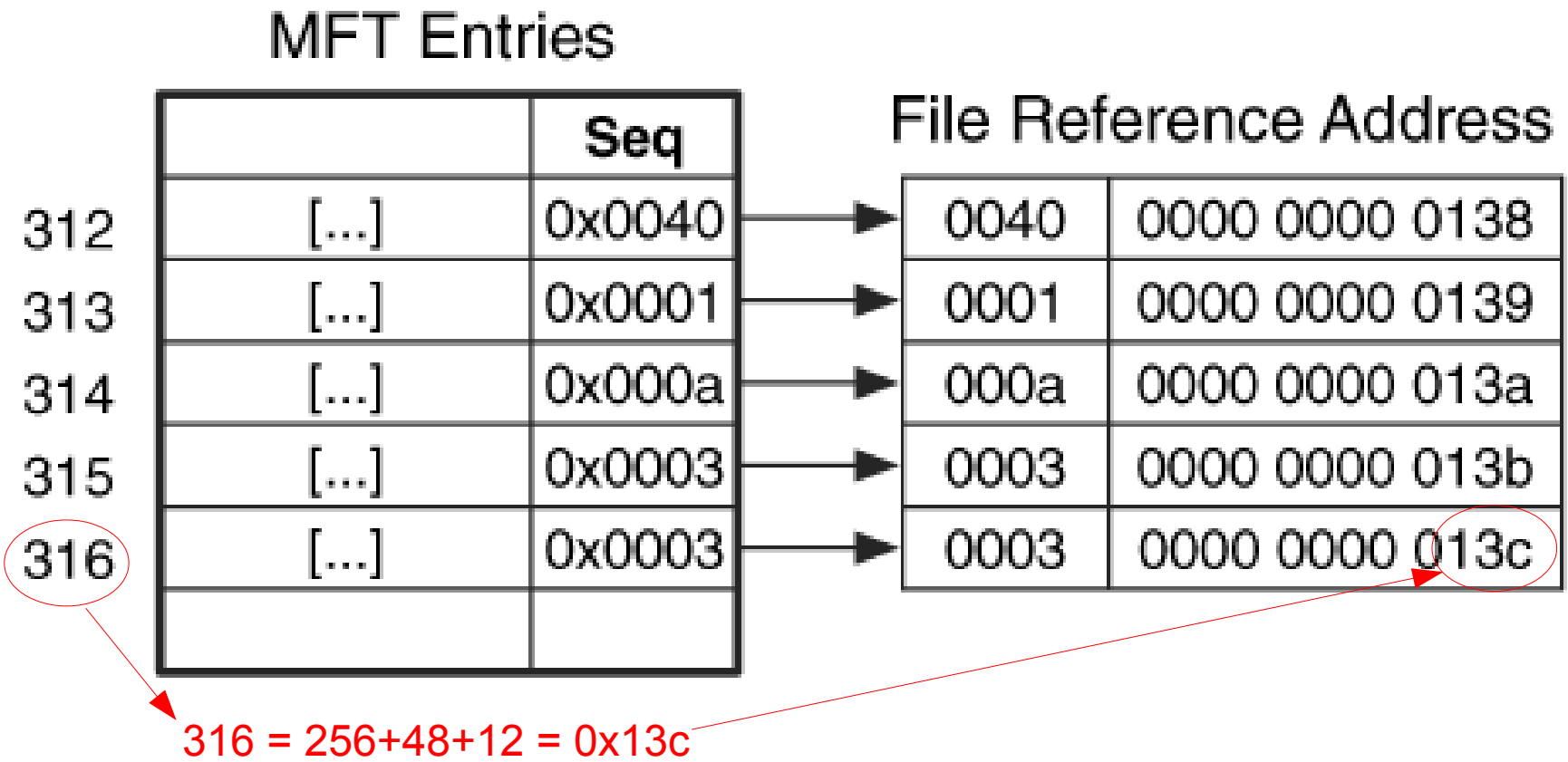


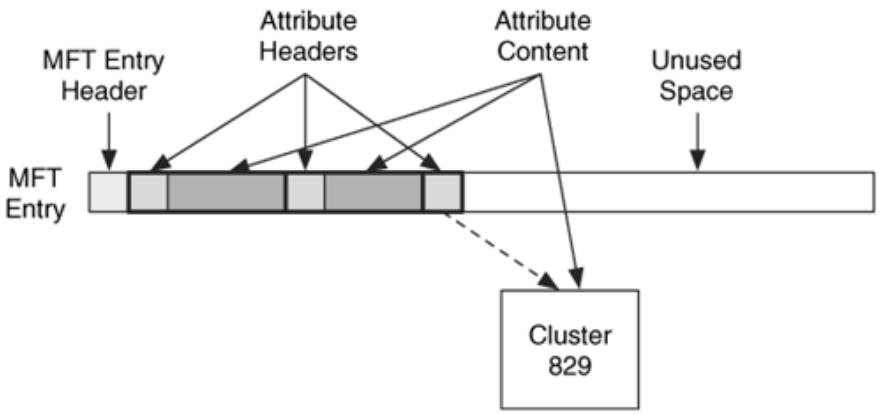
Image from *File System Forensic Analysis* by Brian Carrier, p. 277

MFT Entry Attributes

- Many shapes, sizes and functions
- Attributes contain data and metadata pertaining to the file—*everything* important is here
- Structure: headers and content
- Classified as either resident or non-resident
  - Resident: headers and content are completely contained in the MFT entry
  - Non-resident: header is in the MFT, but content is kept "elsewhere"

Attribute Overflow (1)

- If attribute content is too large, it becomes non-resident
  - Moved to a "cluster run" (like an "extent")

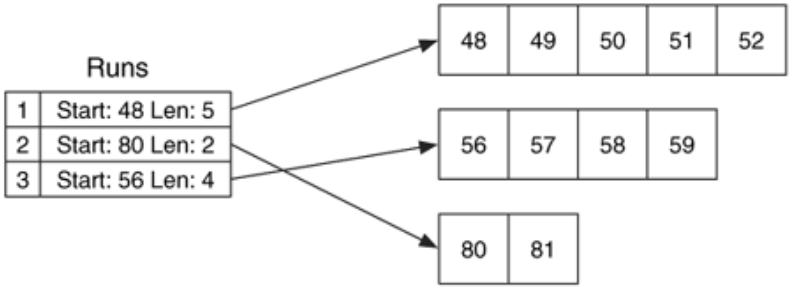


- Now we need a way of finding this data...

 Image from *File System Forensic Analysis* by Brian Carrier, p. 280

Cluster Runs and Run Lists

- The attribute header indicates whether it is resident or non-resident
- If non-resident, points to "runlist" structures
  - Located in the content
  - These tell you where the cluster runs are



Logical Representation of a Runlist

Image from *File System Forensic Analysis* by Brian Carrier, p. 281

Run List Structure

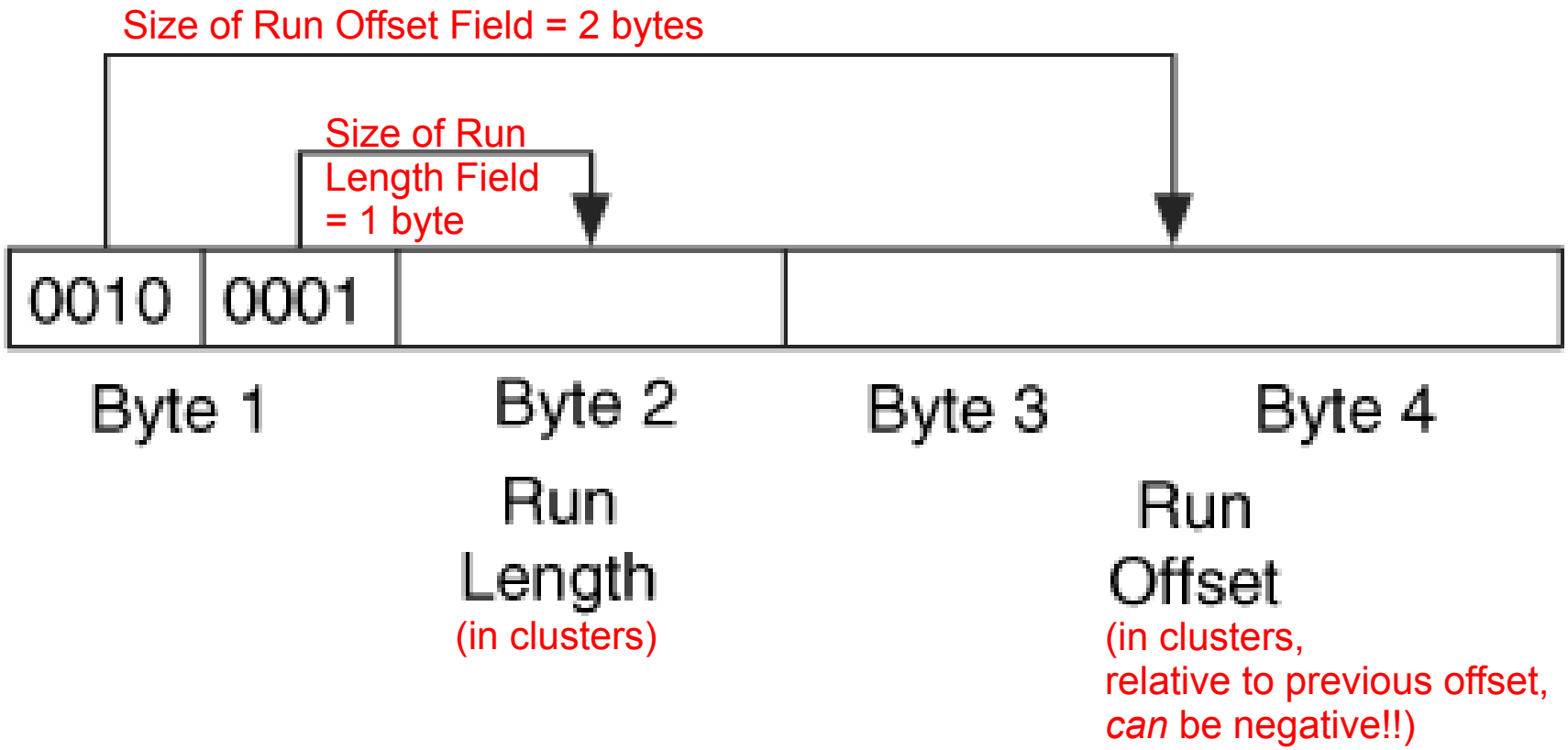


 Image from *File System Forensic Analysis* by Brian Carrier, p. 358

Attribute Overflow (2)

- What if the attribute headers don't fit in a single MFT entry?
  - They will spill into additional MFT entries
  - Now we need a way to keep track of these...
- Base MFT entry: the first entry for the file
- If there are more, it contains an $ATTRIBUTE_LIST attribute
  - Tells you where the non-base MFT entries are and what is in them.

Some Popular Attributes

Type Identifier	Name	Description
16	$STANDARD_ INFORMATION	General Info, Flags, Access Times, Owner, Security Info
32	$ATTRIBUTE_ LIST	Where the other attributes are
48	$FILE_NAME	File name (unicode) and access times
128	$DATA	The data!
144	$INDEX_ROOT	Root of an index tree
176	$BITMAP	For $MFT file itself and indexes

Table adapted from *File System Forensic Analysis* by Brian Carrier, p. 282

Where is the actual data?

- $DATA Attribute
  - Just like any other attribute
  - This means:
    - Small files may be held in MFT only (resident)
    - When non-resident, contains run lists that indicate the blocks in which data are stored and their ordering
- Additional $DATA attributes are allowed:
  - called alternate data streams (ADS)
  - Stream name is <filename>:<attribute name>

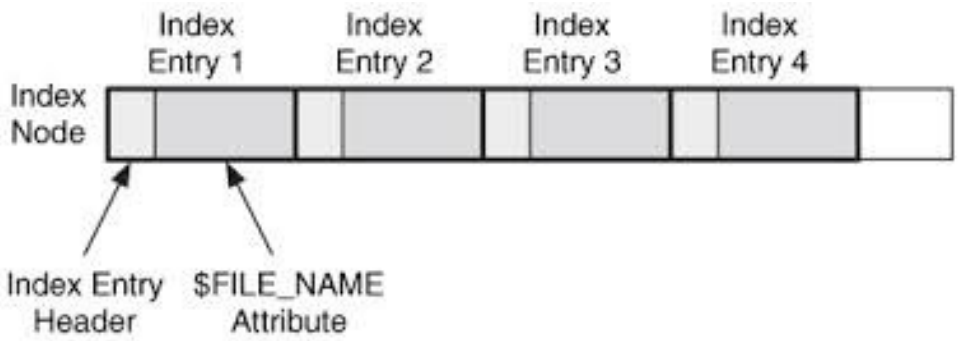
Where are the Files / Directories?

- Files and directories are described using one or more attributes contained in one or more MFT entries
- Required attributes
  - $STANDARD_INFORMATION
    - Time stamps and file permissions
  - $FILE_NAME
    - May have multiple to describe long and short names
    - Also contains timestamps

Directory Related Attributes

- Directories are implemented using *index nodes*
- Nodes point to MFT entries for each file contained in the directory
- Nodes are not attributes themselves; they live inside index attributes
- Namely:
  - $INDEX_ROOT
  - $INDEX_ALLOCATION

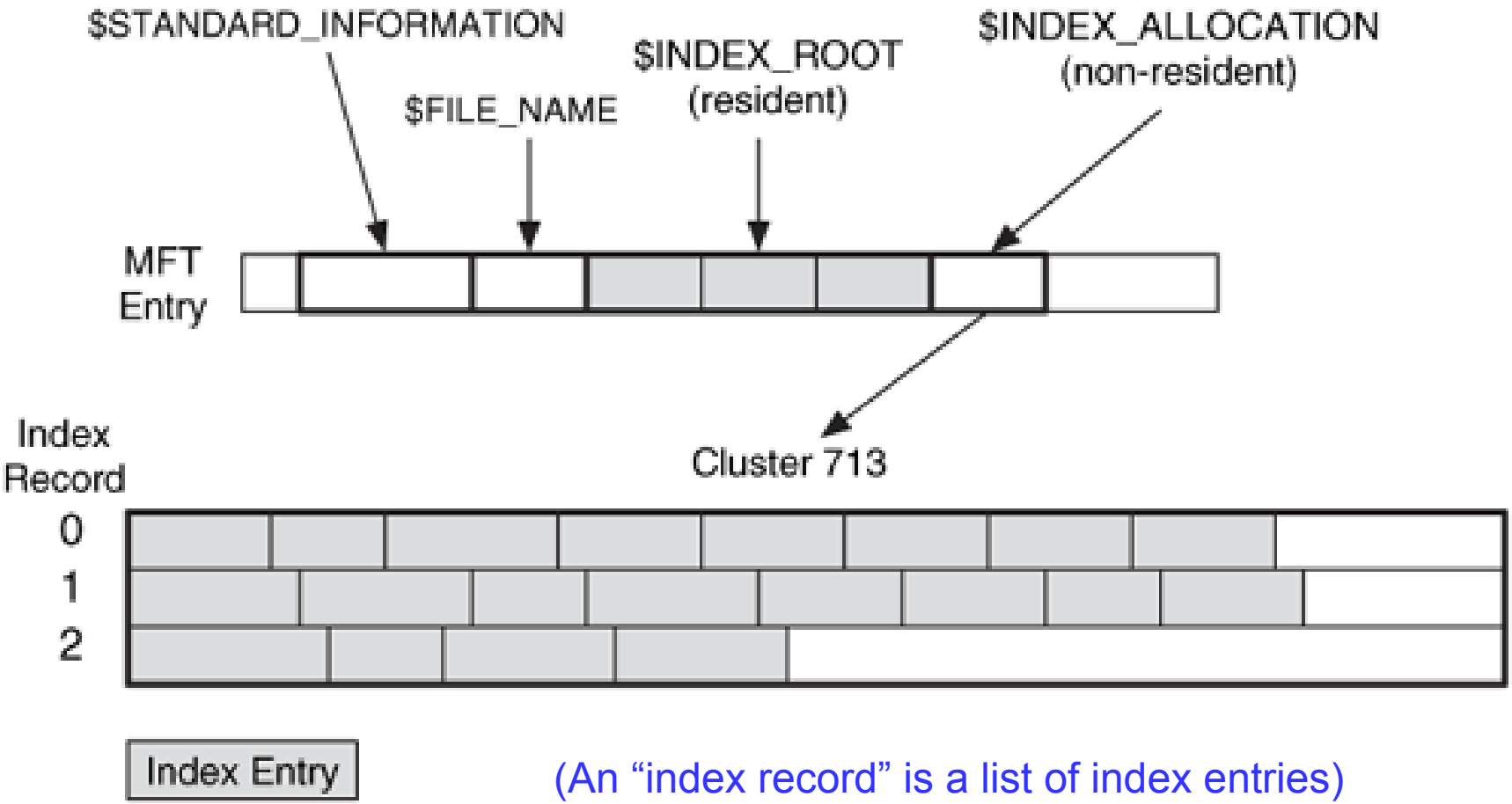
Index Nodes



For directories, this contains the MFT file reference address associated with the file name.

 Image from *File System Forensic Analysis* by Brian Carrier, p. 294

$INDEX_ROOT, $INDEX_ALLOCATION



(An “index record” is a list of index entries)

 Image from *File System Forensic Analysis* by Brian Carrier, p. 295

B-Trees

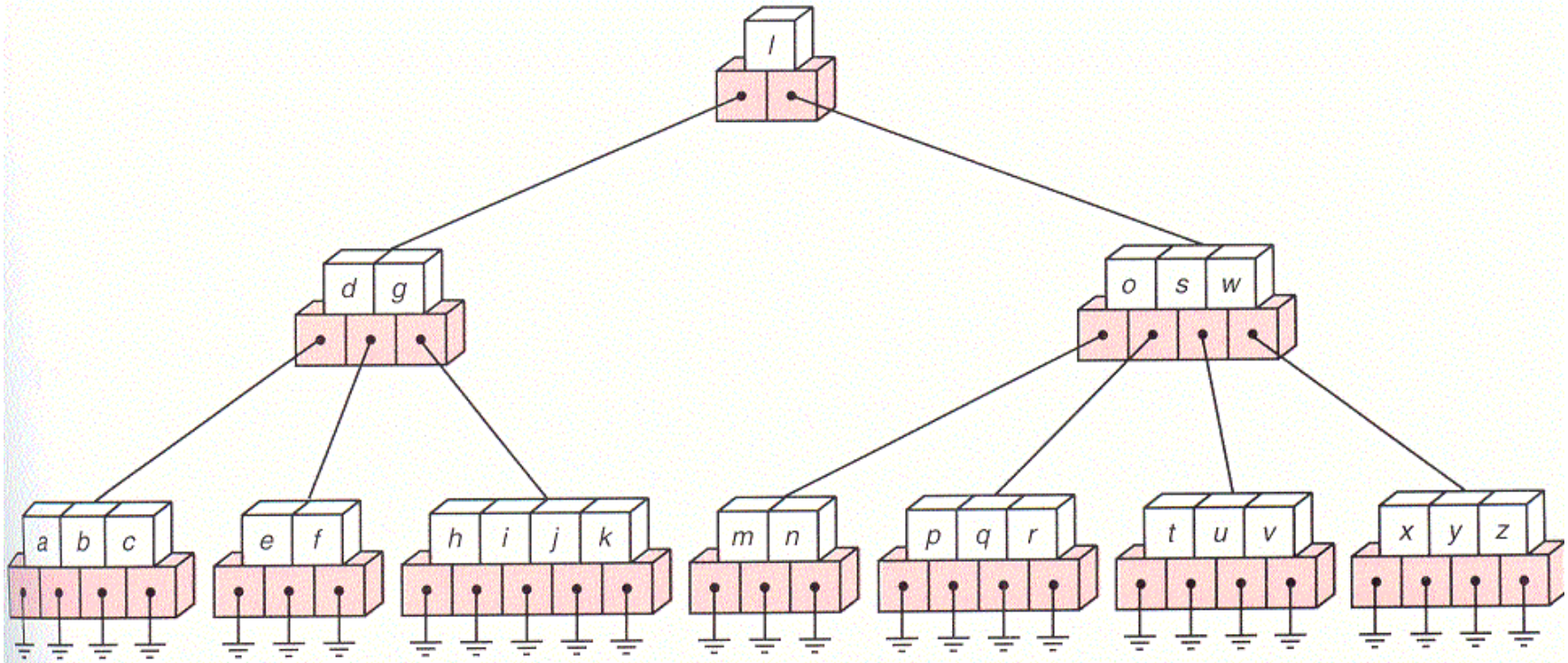


Figure 11.9. A B-tree

B-Trees, in brief

- A data structure for keeping data sorted
- Often used for databases and file systems
- To implement, you need:
  - Children
  - Parents
  - Some other things
- We're not going to go into depth

What manages the file system?

- MFT does not do this all alone:
  - Help from File System Metadata Files
  - (Info about the file system has to live in files)
- First 16-24 MFT entries are for these files
  - Listed in the root directory
  - Typically hidden from users
  - Begin with $ and a capital letter
  - like the superblock, etc., in EXT
  - Entry 0 is $MFT

Other Special File System Files

Entry	File Name	Description
0	$MFT	The entry for the MFT itself
1	$MFTMirr	Backup of first entries in the MFT
2	$LogFile	Journal for metadata transactions
...		
5	.	The root directory of the partition
6	$Bitmap	Cluster allocation status for each cluster.
7	$Boot	Boot sector and boot code for the file system
8	$BadClus	List of bad clusters
...		

Table adapted from *File System Forensic Analysis* by Brian Carrier, p. 278

Other Notable Complications

- Sparse Attributes
  - No need to store all the zeros
- Compressed attributes
  - File system level compression
- Encrypted Attributes
  - An exciting can of worms
  - Implementation details demonstrate how hard it is to get encryption right

NTFS in Practice

- Extremely flexible—data *could* be anywhere
- In practice, it's probably not
  - Allocation algorithms
  - “MFT zone”
  - Other patterns
- All “observed” in practice
- Could (and do) change from version to version