This work is licensed under a Creative Commons license

NTFS File System

Giovanni Lagorio

giovanni.lagorio@unige.it
https://csec.it/people/giovanni_lagorio
X/GitHub/...: zxgio

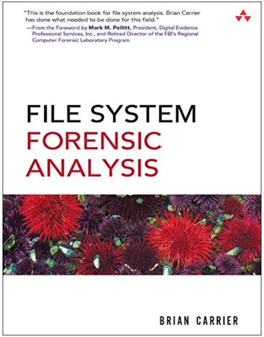
DIBRIS - Dipartimento di Informatica, Bioingegneria, Robotica e Ingegneria dei Sistemi
University of Genova, Italy

Outline

1. Introduction
2. Volume Organization
   - Fixups
3. MFT Entries
   - Directories
4. Eric Zimmerman's Tools
5. Recap

File System Forensics Analysis

The “bible” for this part of the course is Brian Carrier’s “File System Forensics Analsysis” [Car05], where you can find more details



“This is the foundation book for file system analysis. Brian Carrier has done what needed to be done for this field.”

—From the Foreword by **Mark M. Pollitt,** President, Digital Evidence Professional Services, Inc., and Retired Director of the FBI’s Regional Computer Forensic Laboratory Program

FILE SYSTEM
FORENSIC
ANALYSIS

BRIAN CARRIER

Introduction

New Technology File System (NTFS) is a proprietary journaling file system developed by Microsoft in 1993, supporting:

- access control lists (ACLs)
- encryption
- transparent compression
- sparse files
- journaling
- POSIX support (no, not WSL)
- multiple data streams
- . . .

Everything is a file

A nice feature is that everything is a file!

- except for the VBR, everything else is considered a data area
- any sector (except VBR) can be allocated to a file

A very scalable design where internal structures can change over time

- generic data structures embed specific content

Multiple data streams

- Each unit of information associated with a file is implemented as a file attribute (NTFS Object Attribute)
- Each attribute consists of a (byte) stream
- The contents of a file is “an attribute”, like its name or time stamps
- Each file has the special `$DATA` attribute with no name that corresponds to its content
- Application can create additional named streams, called Alternate Data Streams
  - E.g., the `$Zone.Identifier` is used by Windows for marking files downloaded from the web
  - You can list ADSs with `dir/r` or `streams`, and set/show their contents by redirecting `echo` and `more`

Mount options

When mounting NTFS file systems, you may want to specify:

- `show_sys_files` show the metafiles in directory listings
- `streams_interface=windows` to access ADS like in Windows

see `mount.ntfs(8)`

Links

Hard links allow multiple paths to refer to the same file (not directory)

- `mklink /h` *new-name existing-name*

As in Unix, reference-counted/limited to same FS.

Soft/Symbolic links are strings that are interpreted dynamically, can point to files/directories/non-existent-things

- `mklink` *new-name existing-name*

they are implemented as reparse points (=files or directory containing application specific reparse-data and a 32-bit reparse tag)

Junctions are a legacy concept and work almost identically to directory symbolic links, see [ARIS21] for more details

Explorer shortcuts

Shortcuts are `.lnk` files interpreted by Explorer

Outline

1. Introduction

2. Volume Organization
   - Fixups

3. MFT Entries
   - Directories

4. Eric Zimmerman's Tools

5. Recap

Clusters

Differently from FAT, NTFS maps the whole volume into clusters

- The default cluster factor varies with the size of the volume, but it is an integral number of physical sectors, always a power of 2
    - large clusters can reduce fragmentation and speed up allocation, at the cost of wasted space
- logical cluster numbers LCNs correspond to numbering clusters from the beginning of the volume
- data within a file is addressed using virtual cluster numbers VCNs
    - VCNs are not necessarily physically contiguous

Volume Layout

No layout except for the VBR, which guides us to the MFT:

0–2	Assembly instruction to jump to boot code	No (unless it is the bootable file system)
3–10	OEM Name	No
11–12	Bytes per sector	Yes
13–13	Sectors per cluster	Yes
14–15	Reserved sectors (Microsoft says it must be 0)	No
16–20	Unused (Microsoft says it must be 0)	No
21–21	Media descriptor	No
22–23	Unused (Microsoft says it must be 0)	No
24–31	Unused (Microsoft says it is not checked)	No
32–35	Unused (Microsoft says it must be 0)	No
36–39	Unused (Microsoft says it is not checked)	No
40–47	Total sectors in file system	Yes
48–55	Starting cluster address of MFT	Yes
56–63	Starting cluster address of MFT Mirror $DATA attribute	No
64–64	Size of file record (MFT entry)	Yes
65–67	Unused	No
68–68	Size of index record	Yes
69–71	Unused	No
72–79	Serial number	No
80–83	Unused	No
84–509	Boot code	No

8-bit sizes n should be interpreted as follows:

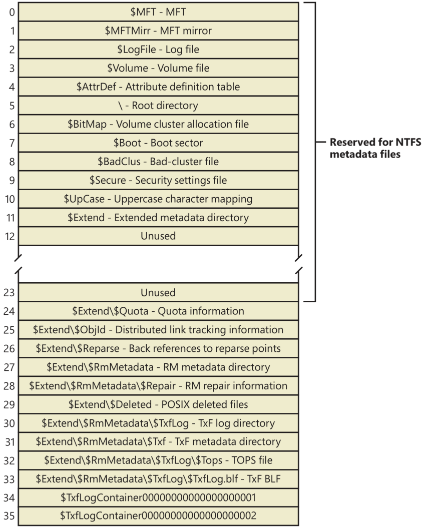
positive	# of clusters
negative	2^{-n}

Master File Table

The MFT is the heart of the NTFS volume structure

- Implemented as an array of file records ($\equiv$ inodes)
  - size of each record can be defined at format time (typically, 1K/4K)
    - default depends on the underlying physical medium: disks that have 4 KB sectors size generally use 4 KB file records, while older disks that have 512 bytes sectors size use 1 KB
    - The size does not depend on the clusters size
  - When a file needs more metadata space, the first one, the base file record, stores the location of the others
- Its location is specified in the BIOS Parameter Block, inside the VBR
- MFT contains one record for each file, including itself (the 1^{st} entry)
  - Can be fragmented; however, an MFT zone is typically reserved when formatting (about 12.5% of the entire volume)
  - In addition, there are other (system) metadata files
    - All these hidden files have a name that begins with a dollar sign $

File records for metadata files



0	$MFT - MFT
1	$MFTMirr - MFT mirror
2	$LogFile - Log file
3	$Volume - Volume file
4	$AttrDef - Attribute definition table
5	\ - Root directory
6	$BitMap - Volume cluster allocation file
7	$Boot - Boot sector
8	$BadClus - Bad-cluster file
9	$Secure - Security settings file
10	$UpCase - Uppercase character mapping
11	$Extend - Extended metadata directory
12	Unused
23	Unused
24	$Extend\$Quota - Quota information
25	$Extend\$ObjId - Distributed link tracking information
26	$Extend\$Reparse - Back references to reparse points
27	$Extend\$RmMetadata - RM metadata directory
28	$Extend\$RmMetadata\$Repair - RM repair information
29	$Extend\$Deleted - POSIX deleted files
30	$Extend\$RmMetadata\$TxfLog - TxF log directory
31	$Extend\$RmMetadata\$Txf - TxF metadata directory
32	$Extend\$RmMetadata\$TxfLog\$Tops - TOPS file
33	$Extend\$RmMetadata\$TxfLog\$TxfLog.blf - TxF BLF
34	$TxfLogContainer00000000000000000001
35	$TxfLogContainer00000000000000000002

From [ARIS21]

Fixups (1/2)

NTFS incorporates fixup values into data structures that are over one sector in length

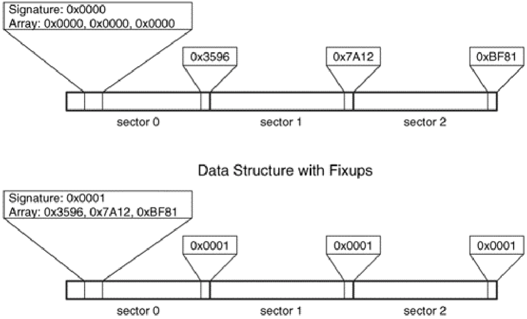
- the last two bytes of each sector are replaced with a “signature” value
- the signature is later used to verify the integrity of the data

Fixups are only in data structures, not in sectors with file content

For example,. . .

Fixups (2/2)

E.g.,



The "signature" is incremented each time the structure is updated

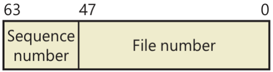
Outline

1. Introduction
2. Volume Organization
   - Fixups
3. MFT Entries
   - Directories
4. Eric Zimmerman's Tools
5. Recap

File identifiers

Files identified by 64-bit "file record numbers", which consist of:

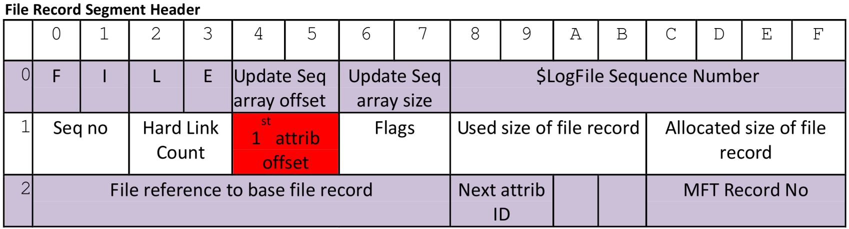
1. a file number, corresponding to the (0-based) position in the MFT
2. a sequence number, incremented when a file record is reused



From [ARIS21]

MFT Entries

File records start with a fixed header:

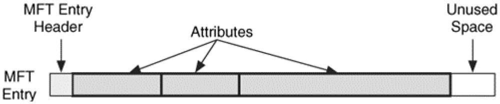


File Record Segment Header

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	F	I	L	E	Update Seq array offset		Update Seq array size		$LogFile Sequence Number							
1	Seq no		Hard Link Count		1st attrib offset		Flags		Used size of file record				Allocated size of file record			
2	File reference to base file record								Next attrib ID				MFT Record No			

(“Update Seq” ≡ fix-up) `https://www.writeblocked.org/resources/NTFS_CHEAT_SHEETS.pdf`

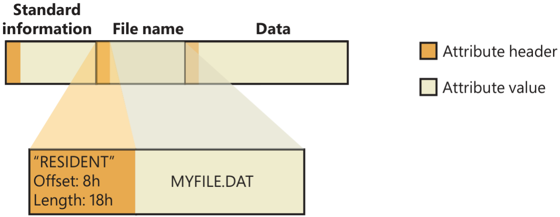
followed by attributes (and fixup values):



Resident attributes

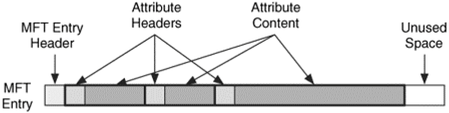
When the value of an attribute is stored in the MFT, the attribute is called a resident attribute

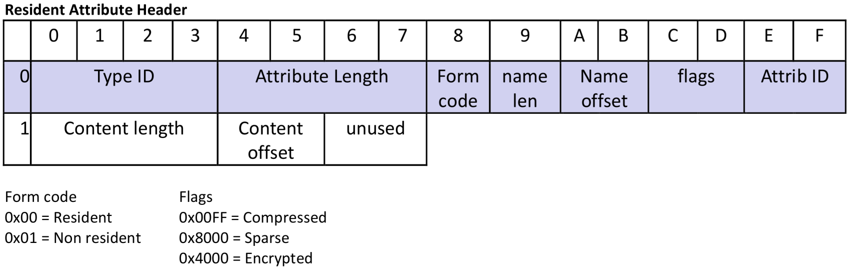
- Some attributes are always resident; e.g., $STANDARD_INFORMATION and $FILENAME are always resident
- Each attribute begins with a standard header, with a type id
    - headers are always resident



From [ARIS21]

Resident Attribute Header





Resident Attribute Header

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	Type ID				Attribute Length				Form code	name len	Name offset		flags		Attrib ID	
1	Content length				Content offset		unused									

Form code
0x00 = Resident
0x01 = Non resident

Flags
0x00FF = Compressed
0x8000 = Sparse
0x4000 = Encrypted

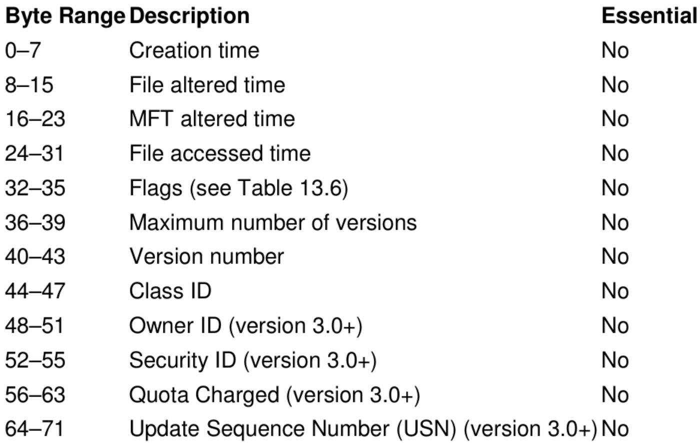
https://www.writeblocked.org/resources/NTFS_CHEAT_SHEETS.pdf

Note: most attributes don’t have names

Type Id

0x10/16	$STANDARD_INFORMATION
0x20/32	$ATTRIBUTE_LIST
0x30/48	$FILE_NAME
0x40/64	$OBJECT_ID
0x50/80	$SECURITY_DESCRIPTOR
0x60/96	$VOLUME_NAME
0x70/112	$VOLUME_INFORMATION
0x80/128	$DATA
0x90/144	$INDEX_ROOT — named $I30
0xA0/160	$INDEX_ALLOCATION — named $I30
0xB0/176	$BITMAP
0xC0/192	$SYMBOLIC_LINK
	...

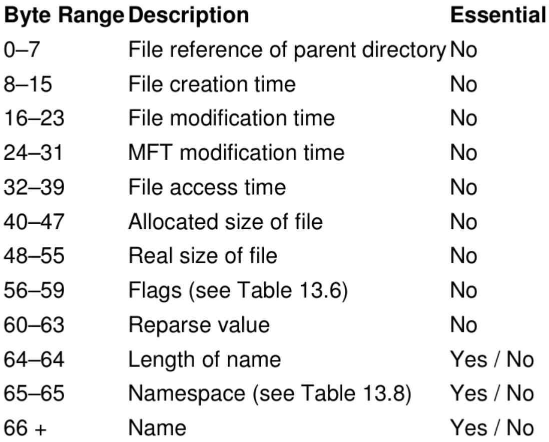
$STANDARD_INFORMATION



Byte Range	Description	Essential
0–7	Creation time	No
8–15	File altered time	No
16–23	MFT altered time	No
24–31	File accessed time	No
32–35	Flags (see Table 13.6)	No
36–39	Maximum number of versions	No
40–43	Version number	No
44–47	Class ID	No
48–51	Owner ID (version 3.0+)	No
52–55	Security ID (version 3.0+)	No
56–63	Quota Charged (version 3.0+)	No
64–71	Update Sequence Number (USN) (version 3.0+)	No

Only one SI attribute per file; timestamps available to Win32 APIs

$FILENAME



Byte Range	Description	Essential
0–7	File reference of parent directory	No
8–15	File creation time	No
16–23	File modification time	No
24–31	MFT modification time	No
32–39	File access time	No
40–47	Allocated size of file	No
48–55	Real size of file	No
56–59	Flags (see Table 13.6)	No
60–63	Reparse value	No
64–64	Length of name	Yes / No
65–65	Namespace (see Table 13.8)	Yes / No
66 +	Name	Yes / No

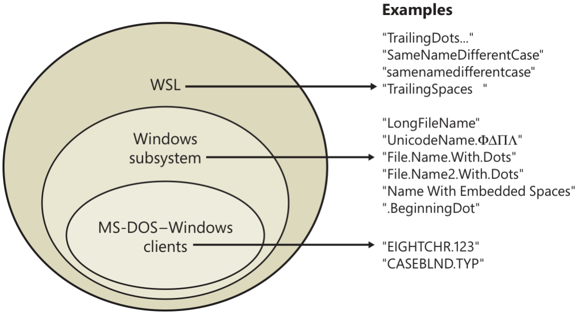
Possibly more file attributes, timestamps *not* available to Win32 APIs

Namespaces are POSIX, Win32, . . .

Namespaces

For compatibility reasons, when a file is created in the Windows namespace, NTFS can also generate an MS-DOS file name (if necessary)

- the mechanism is very similar to hard-links: the long-filename and the 8.3 are “aliases”, so users can access/delete the file using either names
- since Windows 8.1, by default all the NTFS nonbootable volumes have short name generation disabled



Examples

"TrailingDots..."
"SameNameDifferentCase"
"samenamedifferentcase"
"TrailingSpaces   "

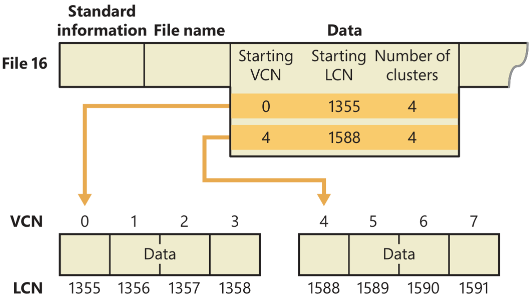
"LongFileName"
"UnicodeName.ΦΔΠΛ"
"File.Name.With.Dots"
"File.Name2.With.Dots"
"Name With Embedded Spaces"
".BeginningDot"

"EIGHTCHR.123"
"CASEBLND.TYP"

Runs/Extents

When a value is too large to be contained in an MFT file record

- NTFS allocates clusters outside the MFT
- Each contiguous group of clusters is called a run (or an extent)

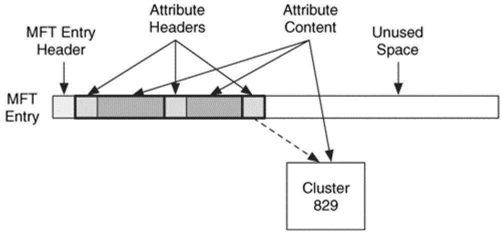


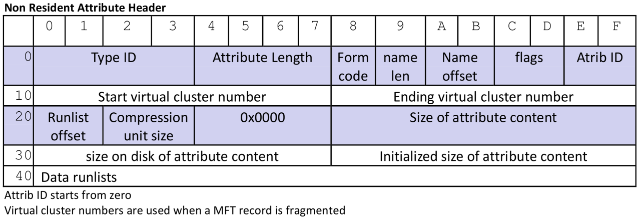
Starting VCN	Starting LCN	Number of clusters
0	1355	4
4	1588	4

From [ARIS21]

There can be “holes” in the VCN-to-LCN mappings for sparse files

Non-resident Attribute Header





Non Resident Attribute Header

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	Type ID				Attribute Length				Form code	name len	Name offset		flags		Atrib ID	
10	Start virtual cluster number								Ending virtual cluster number							
20	Runlist offset		Compression unit size		0x0000				Size of attribute content							
30	size on disk of attribute content								Initialized size of attribute content							
40	Data runlists															

Attrib ID starts from zero

Virtual cluster numbers are used when a MFT record is fragmented

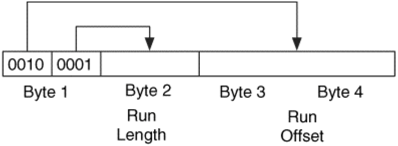
`https://www.writeblocked.org/resources/NTFS_CHEAT_SHEETS.pdf`

The offset of the run-list is given w.r.t. the start of the attribute

Run lists (1/2)

Efficient, yet rather confusing encoding

- variable length, but at least 1 byte
- the least-significant nibble of the 1st byte gives the # of bytes of the length field
- the most-significant nibble gives the # of bytes of the run offset, which follows the length field
    - 0 offset bytes means a “hole” (in a sparse file)



The values are in cluster-sized units, and the offset field is a signed value that is relative to the previous offset (the first one is relative to 0, AKA the beginning of the volume); e.g.,. . .

Run lists (2/2)

E.g., `32 c0 1e b5 3a 05 21 70 1b 1f 00` means:

1. 32 → two bytes in the run-length, and three in the run-offset
   - `c0 1e` → `0x1ec0=7872` clusters
   - `b5 3a 05` → at offset `0x53ab5 [+0] = 342709`
2. 21 → one byte in the run-length, and two in the run-offset
   - `70` → `0x70=112` clusters
   - `1b 1f` → at offset `0x1f1b + 0x53ab5 = 350672`
3. 00 — the end

The attribute list attribute

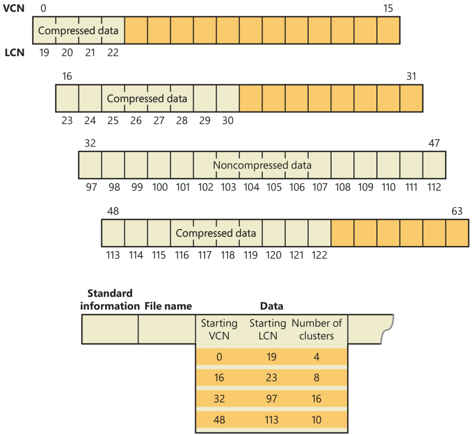
When a file metadata can't fit a single MFT record (e.g., the file has too many attributes or is too fragmented)

- other MFT records are used to contain the additional attributes/attribute-headers
- an attribute called the attribute list is added, to contain
  - the name and type code of each attribute, and
  - the file number of the MFT record where the attribute is located

Transparent compression

NTFS divides the file's data into compression units 16 clusters long

- This size represents a trade-off between producing smaller compressed files and slowing random-access read operations

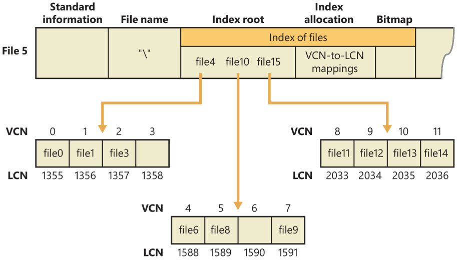


Starting VCN	Starting LCN	Number of clusters
0	19	4
16	23	8
32	97	16
48	113	10

Indexing/Directories (1/2)

A directory contains a sorted list of the files (in its index root attribute)

- For large directories the names are stored in 4 KB, fixed-size index buffers (the nonresident values of the index allocation attribute)
- Index buffers implement a B-tree data structure, which minimizes the number of disk accesses needed to find a particular file
- The index root attribute contains the first level of the B-tree, and points to index buffers containing the next level



From [ARIS21]

Indexing/Directories (2/2)

Previous figure shows only file names, but each entry also contains

- the record number in the MFT where the file is described
- time stamp information
- file size

NTFS duplicates these information to speed up directory browsing, at the cost of updating those information in two places

- From a DF point of view, indexes could contain traces of previously allocated files that are no longer present in the MFT

Outline

1. Introduction
2. Volume Organization
   - Fixups
3. MFT Entries
   - Directories
4. Eric Zimmerman's Tools
5. Recap

MFTECmd

You can export the `$MFT` with TSK's `icat`, and then parse/explore with EZ's tools

E.g.,

```
MFTECmd.exe -f input-mft --csv out-dir --csvf out-filename
```

Then use MFTExplorer, TimelineExplorer, . . .

https://ericzimmerman.github.io/

Outline

1. Introduction
2. Volume Organization
   - Fixups
3. MFT Entries
   - Directories
4. Eric Zimmerman's Tools
5. Recap

GnG

Demo/Exercise

Explore, with `mount`, ImHex and TSK, `gng-ntfs.dd` (SHA256: `f55e7253c4326...`)

You should find:

- a very small file (content is in MFT)
- a file with two names
- a symbolic link
- a file with alternate data streams (which/where are the contents?)

References

[ARIS21] Andrea Allievi, Mark Russinovich, Alex Ionescu, and David Solomon.
Windows Internals, Part 2, 7th Edition.
Microsoft Press, 2021.

[Car05] Brian Carrier.
File System Forensic Analysis.
Addison-Wesley Professional, 2005.