NTFS

New Technology File System

Some History

HPFS was developed when Microsoft and IBM were working on OS/2 as a joint venture

NTFS evolved out of HPFS after Microsoft and IBM parted ways

Introduced with Windows NT 3.1 in early 1990s

NT could use either NTFS or HPFS

Win 2K and beyond could no longer use HPFS

NTFS Overview

FAT (even FAT32) was too limiting

HPFS had features beyond those of FAT

NTFS had features beyond HPFS

But all this made NTFS more complex

NTFS Overview
Some NTFS Goals

Large disks (terabytes and beyond)

Security

- *Ownership*
- *Limit access on a file-by-file basis using ACLs*
- *Encryption*

Robust and reliable

Scaleable

Minimize fragmentation

Smaller cluster sizes than FAT

- *Minimized "wasted" disk space*

Use Unicode (avoids ASCII constraints)

Compression

Encryption

NTFS Overview

Employed many ideas from VMS and UNIX

Containers or wrappers

Data structure inside a container is not defined and can be anything

Everything is a file

All bytes in NTFS are in files

A file is a container

NTFS Cluster Sizes

Drive size	Sectors per cluster	Cluster size
7–512 MB	8	4 KB
512 MB–1 GB	8	4 KB
1–2 GB	8	4 KB
2 GB–2 TB	8	4 KB
2–16 TB	8	4 KB
16–32 TB	16	8 KB
32–64 TB	32	16 KB
64–128 TB	64	32 KB
128–256 TB	128	64 KB

NTFS Specifications

There is no "official" specification

- *Microsoft has not published a detailed specification*
  - Only some high-level descriptions
- *No information on details of disk/drive layout*

Other organizations (not Microsoft) have published unofficial specs

- *Based upon investigations of NTFS disks/drive*
- *These specifications may not be quite correct*
- *As NTFS evolves, the unofficial specs may be in error*

Unofficial NTFS Specification
NTFS-3G

NTFS-3G

- *Runs on many OSs including Linux and MACs*
- *Uses the FUSE (**F**ilesystem in **USE**r interface) for Unix-like OSs*
- *Developer is Tuxera*
- *No longer open source*

Latest stable version released in 2017

- *2017.3.23*

NTFS Specifications

Partition Magic, Partition Commander, Red Hat, Novell/SUSI and others have software that claims to create NTFS partitions

But there's no guarantee that partitions created by these organizations are exactly right

But they seem to work

Microsoft has made "minor" changes in NTFS with almost every new version of Windows

Having said this, we'll now study the details of NTFS

What we will cover has been stable and verifiable

NTFS High Level Layout

A NTFS partition has two main areas

Partition Boot Sector (PBS)

Or VBS (Volume Boot Sector) if you wish

NTFS file system proper

Consists of a collection of files (everything is a file)

File-system ***files***

User-file-area ***files***

Partition Boot Sector (PBS)

Before we look at NTFS proper, it's instructive to consider the NTFS Partition Boot Sector (PBS)

Partition Boot Sector (PBS)

Starts at sector zero of the partition

16 sectors are reserved when formatting the partition with NTFS

Used to “start” NTFS

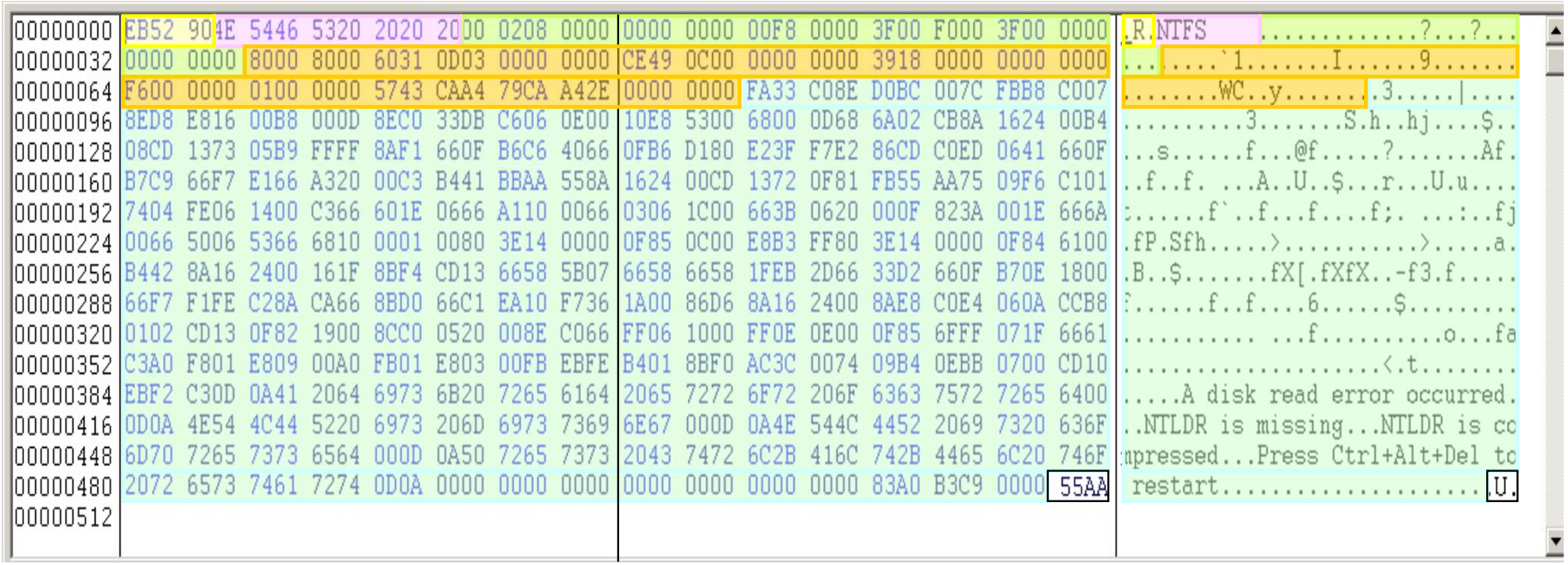
Duplicate PBS usually located near center of partition

PBS Format

Byte Offset	Field Length	Field Name
0x00	3 bytes	Jump Instruction
0x03	LONGLONG	OEM ID
0x0B (11_{10})	25 bytes	BPB (BIOS Parameter Block)
0x24 (36_{10})	48 bytes	Extended BPB
0x54 (84_{10})	426 bytes	Bootstrap Code
0x01FE (510_{10})	WORD	End of Sector Marker (0x55AA)

Provides ***ntldr*** with starting cluster location of MFT, MFT2 (MFT mirror), and startup code

Sample PBS for NTFS



00000000	EB52 904E 5446 5320 2020 2000 0208 0000	0000 0000 00F8 0000 3F00 F000 3F00 0000	.R.NTFS	.....?...?...
00000032	0000 0000 8000 8000 6031 0D03 0000 0000	CE49 0C00 0000 0000 3918 0000 0000 0000	....`1.....I.....9.....	
00000064	F600 0000 0100 0000 5743 CAA4 79CA A42E	0000 0000 FA33 C08E D0BC 007C FBB8 C007	....WC..y.....3.....|....	
00000096	8ED8 E816 00B8 000D 8EC0 33DB C606 0E00	10E8 5300 6800 0D68 6A02 CB8A 1624 00B4	.....3.....S.h..hj....$..	
00000128	08CD 1373 05B9 FFFF 8AF1 660F B6C6 4066	0FB6 D180 E23F F7E2 86CD C0ED 0641 660F	...s.....f...@f.....?.....Af.	
00000160	B7C9 66F7 E166 A320 00C3 B441 BBAA 558A	1624 00CD 1372 0F81 FB55 AA75 09F6 C101	..f..f. ...A..U..$...r...U.u....	
00000192	7404 FE06 1400 C366 601E 0666 A110 0066	0306 1C00 663B 0620 000F 823A 001E 666A	t.....f`..f...f....f;. ...:..fj	
00000224	0066 5006 5366 6810 0001 0080 3E14 0000	0F85 0C00 E8B3 FF80 3E14 0000 0F84 6100	.fP.Sfh.....>.....>.....a.	
00000256	B442 8A16 2400 161F 8BF4 CD13 6658 5B07	6658 6658 1FEB 2D66 33D2 660F B70E 1800	.B..$.....fX[.fXfX..-f3.f.....	
00000288	66F7 F1FE C28A CA66 8BD0 66C1 EA10 F736	1A00 86D6 8A16 2400 8AE8 C0E4 060A CCB8	f.....f..f....6.....$.....	
00000320	0102 CD13 0F82 1900 8CC0 0520 008E C066	FF06 1000 FF0E 0E00 0F85 6FFF 071F 6661	....f.....o...fa	
00000352	C3A0 F801 E809 00A0 FB01 E803 00FB EBFE	B401 8BF0 AC3C 0074 09B4 0EBB 0700 CD10	.....<.t.....	
00000384	EBF2 C30D 0A41 2064 6973 6B20 7265 6164	2065 7272 6F72 206F 6363 7572 7265 6400	.....A disk read error occurred.	
00000416	0D0A 4E54 4C44 5220 6973 206D 6973 7369	6E67 000D 0A4E 544C 4452 2069 7320 636F	..NTLDR is missing...NTLDR is co	
00000448	6D70 7265 7373 6564 000D 0A50 7265 7373	2043 7472 6C2B 416C 742B 4465 6C20 746F	mpressed...Press Ctrl+Alt+Del to	
00000480	2072 6573 7461 7274 0D0A 0000 0000 0000	0000 0000 0000 0000 83A0 B3C9 0000 55AA	restart.....U.	
00000512				

Jump instruction

NTFS identifier

BPB (BIOS Parameter Block)

Extended BPB

Boot code

End-of-sector mark

Contents of BPB & Extended BPB

BPB

Byte Offset	Field Length	Sample Value	Field Name
0x0B	WORD	0x0002 (0x0200 = 512 bytesPerSector)	Bytes Per Sector
0x0D	BYTE	0x08 (8 x 512 = 4096 bytesPerCluster)	Sectors Per Cluster
0x0E	WORD	0x0000	Reserved Sectors
0x10	3 BYTES	0x000000	*always 0*
0x13	WORD	0x0000	*not used by NTFS*
0x15	BYTE	0xF8	Media Descriptor
0x16	WORD	0x0000	*always 0*
0x18	WORD	0x3F00	Sectors Per Track
0x1A	WORD	0xFF00	Number Of Heads
0x1C	DWORD	0x3F000000	Hidden Sectors
0x20	DWORD	0x00000000	*not used by NTFS*
0x24	DWORD	0x80008000	*not used by NTFS*
0x28	LONGLONG	0x4AF57F0000000000	Total Sectors
0x30	LONGLONG	0x0400000000000000	Logical Cluster Number for the file $MFT
0x38	LONGLONG	0x54FF070000000000	Logical Cluster Number for the file $MFTMirr
0x40	DWORD	0xF6000000 (Discussed later)	Clusters Per File Record Segment
0x44	DWORD	0x01000000 (Discussed later)	Clusters Per Index Block
0x48	LONGLONG	0x14A51B74C91B741C	Volume Serial Number
0x50	DWORD	0x00000000	Checksum

Extended BPB

Contents of BPB & Extended BPB

Byte Offset	Field Length	Sample Value	Field Name
0x0B	WORD	0x0002 (0x0200: 512 bytesPerSector)	Bytes Per Sector
0x0D	BYTE	0x08 (8 x 512 = 4096 bytesPerCluster)	Sectors Per Cluster
0x0E	WORD	0x0000	Reserved Sectors
0x10	3 BYTES	0x000000	*always 0*
0x13	WORD	0x0000	*not used by NTFS*
0x15	BYTE	0xF8	Media Descriptor
0x16	WORD	0x0000	*always 0*
0x18	WORD	0x3F00	Sectors Per Track
0x1A	WORD	0xFF00	Number Of Heads
0x1C	DWORD	0x3F000000	Hidden Sectors
0x20	DWORD	0x00000000	*not used by NTFS*
0x24	DWORD	0x80008000	*not used by NTFS*
0x28	LONGLONG	0x4AF57F0000000000	Total Sectors
0x30	LONGLONG	0x0400000000000000	Logical Cluster Number for the file $MFT
0x38	LONGLONG	0x54FF070000000000	Logical Cluster Number for the file $MFTMirr
0x40	DWORD	0xF6000000	Clusters Per File Record Segment
0x44	DWORD	0x01000000	Clusters Per Index Block
0x48	LONGLONG	0x14A51B74C91B741C	Volume Serial Number
0x50	DWORD	0x00000000	Checksum

Extended BPB

Further Discussion
"Clusters per File Record Segment"

What does *"Clusters per File Record Segment"* mean?

This perhaps should read *Clusters per MFT Entry*

It means the size of the cluster allocated for each MFT entry

Further Discussion
"Clusters per Index Block"

What does *"Clusters per Index Block"* mean?

Perhaps it should be *"Clusters per Index Buffer"*

Index Blocks are used to store directory information

It means the number of clusters allocated for each each *Index Block* entry

NTFS
Master File Table (*MFT*)

Everything is a File

In a partition formatted with a NTFS file system

Every byte on the partition other than the PBS is allocated to a file container

Thus, NTFS doesn't have rigid locations for information on the partition

There exists a file called the ***Master File Table*** (***MFT***)

*Location (logical cluster number) of **MFT** is defined in the Extended BPB of the PBS*

The **MFT** consists entirely of a series of ***MFT entries***

MFT** entries often called **File Records

Master File Table (MFT)

Every file in an NTFS partition is referenced by at least one ***MFT entry*** in the master file table (**MFT**)

The **MFT** is itself a file (Everything is a file)

*Therefore the **MFT** has an entry for itself*

MFT Entries are usually 1KB (1024 bytes)

Actually the size is defined in the PBS

*Carrier says that all **MFT Entries** are 1KB up to now*

Nelson says 1.5K

I've typically seen 1KB

Conventions for the Next Few Slides

File names are in green

Abc

A Table within a green file is in orange

Ghi

Entries in an orange table within a file are in blue

Cdf

Some Needed Clarification*

In the NTFS file system there is always a NTFS file named *$MFT*

The *$MFT* file contains a table named *MFT*

Within the *MFT* table there is a *MFT* entry named *$MFT*

$MFT is not the same as *$MFT*

$MFT *is an entry while* $MFT *is a file*

$MFT* { *MFT* [*$MFT…] }

This reads: The file ***$MFT*** *contains the table* ***MFT*** *that consists of a number of* ***MFT*** *entries the 1st of which is* ***$MFT***

Unfortunately this is the terminology that is used

MFT Metadata Examination

To begin an examination of an NTFS file system, we need the metadata

The *$MFT* file is a good place to begin

It contains the MFT table

*In the MFT table there is a MFT entry for every file and folder in the file system (including the **$MFT** file)*

The starting sector address of the *$MFT* file is given in the *Partition Boot Sector*

The layout of the *MFT* table is determined by examining entry 0 in the *MFT* table (named *$MFT*)

MFT Metadata Examination

The first *MFT entry* in the *MFT* table is named *$MFT*

Some of the attributes always present in *$MFT*

$STANDARD_INFORMATION

Temporal information about the MFT

$FILE_NAME

$MFT

$DATA

Contains the clusters used by the *MFT*

$BITMAP

Handles the allocation status of *MFTentries* in the *MFT*

Analogous to the *FATat* in a *FATfs*, but only for the *MFT*

MFT Metadata

Metadata MFT Entries

The first 16 *MFT entries* (*file records*) are reserved for special information

These are called "metadata" entries

They are files stored in the root directory

Hidden from users

Name syntax: $<upperCaseFirstLetter><restOfName>

Entries (file records) in MFT

Record #0: MFT meta-data [*$MFT*]

MFT description of the MFT file

Record #1 [*$MftMirr*]

Mirror of 1st record (i.e., record #0)

Record #2 [*$LogFile*]

Log file (used for recovery)

Records #3 - #15

Various info about the file system

Records #16 and greater

Information on each file and folder in the volume

Both small and large files and folder

MFT MetaData (from Nelson)

Filename	System file	Record position	Description
$Mft	MFT	0	Base file record for each folder on the NTFS volume; other record positions in the MFT are allocated if more space is needed.
$MftMirr	MFT 2	1	The first four records of the MFT are saved in this position. If a single sector fails in the first MFT, the records can be restored, allowing recovery of the MFT.
$LogFile	Log file	2	Previous transactions are stored here to allow recovery after a system failure in the NTFS volume.
$Volume	Volume	3	Information specific to the volume, such as label and version, is stored here.
$AttrDef	Attribute definitions	4	A table listing attribute names, numbers, and definitions.
$	Root filename index	5	This is the root folder on the NTFS volume.
$Bitmap	Cluster bitmap	6	A map of the NTFS partition shows which clusters are in use and which are available.
$Boot	Boot sector	7	Used to mount the NTFS volume during the bootstrap process; additional code is listed here if it's the boot drive for the system.
$BadClus	Bad cluster file	8	For clusters that have unrecoverable errors, an entry of the cluster location is made in this file.
$Secure	Security file	9	Unique security descriptors for the volume are listed in this file. It's where the access control list (ACL) is maintained for all files and folders on the NTFS volume.
$Upcase	Upcase table	10	Converts all lowercase characters to uppercase Unicode characters for the NTFS volume.
$Extend	NTFS extension file	11	Optional extensions are listed here, such as quotas, object identifiers, and reparse point data.
		12–15	Reserved for future use.

MFT Meta-Data *(from Carrier)*

Entry	File Name	Description
0	$MFT	The entry for the MFT itself.
1	$MFTMirr	Contains a backup of the first entries in the MFT. See the "File System Category" section in Chapter 12.
2	$LogFile	Contains the journal that records the metadata transactions. See the "Application Category" section in Chapter 12.
3	$Volume	Contains the volume information such as the label, identifier, and version. See the "File System Category" section in Chapter 12.
4	$AttrDef	Contains the attribute information, such as the identifier values, name, and sizes. See the "File System Category" section in Chapter 12.
5	.	Contains the root directory of the file system. See the "File Name Category" section in Chapter 12.
6	$Bitmap	Contains the allocation status of each cluster in the file system. See the "Content Category" section in Chapter 12.
7	$Boot	Contains the boot sector and boot code for the file system. See the "File System Category" section in Chapter 12.
8	$BadClus	Contains the clusters that have bad sectors. See the "Content Category" section in Chapter 12.
9	$Secure	Contains information about the security and access control for the files (Windows 2000 and XP version only). See the "Metadata Category" section in Chapter 12.
10	$Upcase	Contains the uppercase version of every Unicode character.
11	$Extend	A directory that contains files for optional extensions. Microsoft does not typically place the files in this directory into the reserved MFT entries.

MFT Entry Format & Attributes

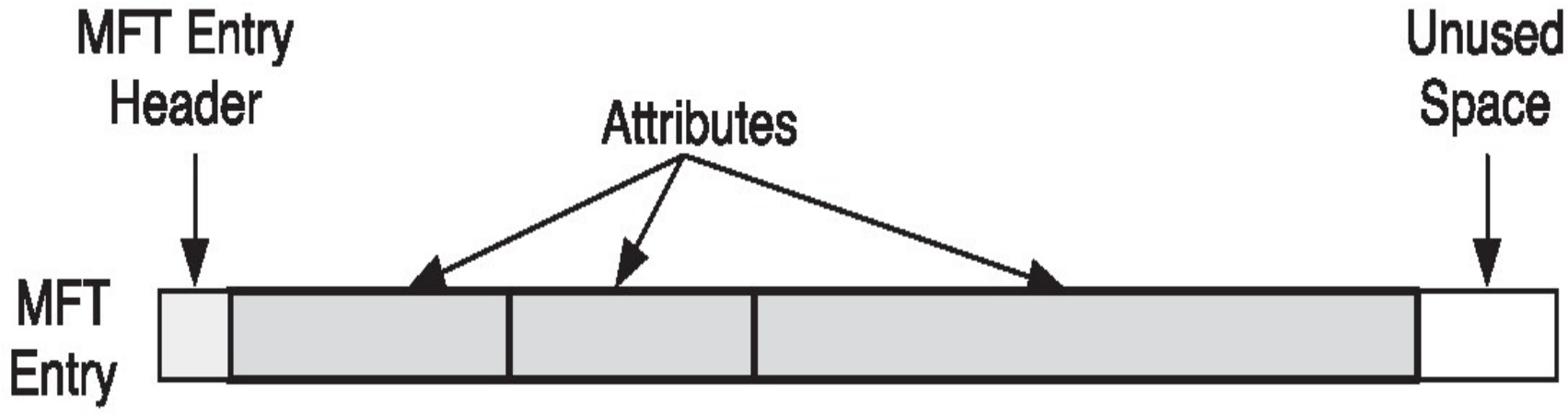
MFT Entry Format

Each MFT entry (usually 1KB) consists of

A MFT entry header

A number of attributes

Unused space



MFT Entry Header

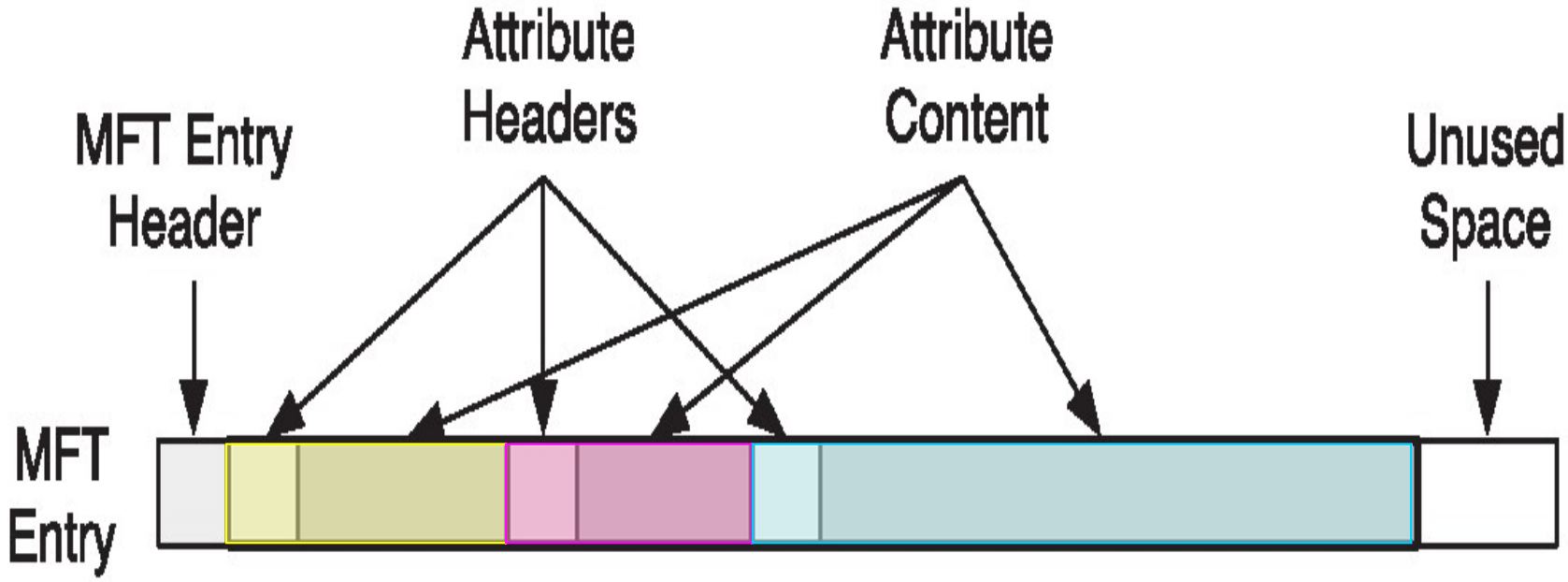
Field	Comment / Example
Signature:	*What entry is / FILE0, BAAD*
Sequence #	*Order of entry in MFT*
First attribute offset:	*Location of 1st attribute in entry.*
Flags:	*In use or not in use*
Base file record link:	*Address of base entry.* *If entry consumes > 1 attribute, after the 1st attribute, subsequent attributes point back to the 1st one.*

MFT Attribute Format

MFT attributes each have

Attribute Header

Attribute Content



MFT Attribute Header

The contents of the attribute header contains

Type of attribute

Size

Name

Compressed

Encrypted

MFT Attribute Content

No predefined structure

Two types of attributes

Resident attribute

Non-resident attribute

Attribute header identifies attribute as being resident or non-resident

Resident Attributes

Actual file or folder **content** is stored within the MFT entry along with the attribute header

Stored immediately after the attribute header

Can actually store the contents of a small file within the attribute of a MFT entry

Only possible for small files or folders

e.g., A directory (folder) entry

A small text file that fits in the 1KB size

Comments

Minimizes slack

Dramatically improves performance

Non-Resident Attributes

Attribute content is stored in cluster(s) external to the MFT

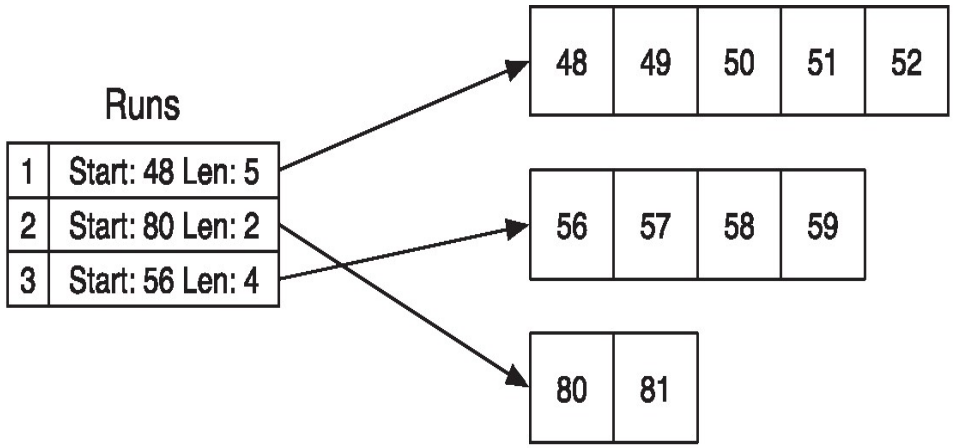
Content is too large to be stored in 1KB

Attribute header provides the external cluster address and size as "*cluster runs*"

Starting cluster address

Number of clusters

Here a single attribute header defines 3 cluster runs



Attribute Growth

Attribute can begin as resident & grow to be non-resident

If a resident attribute grows to exceed the capacity of its MFT entry

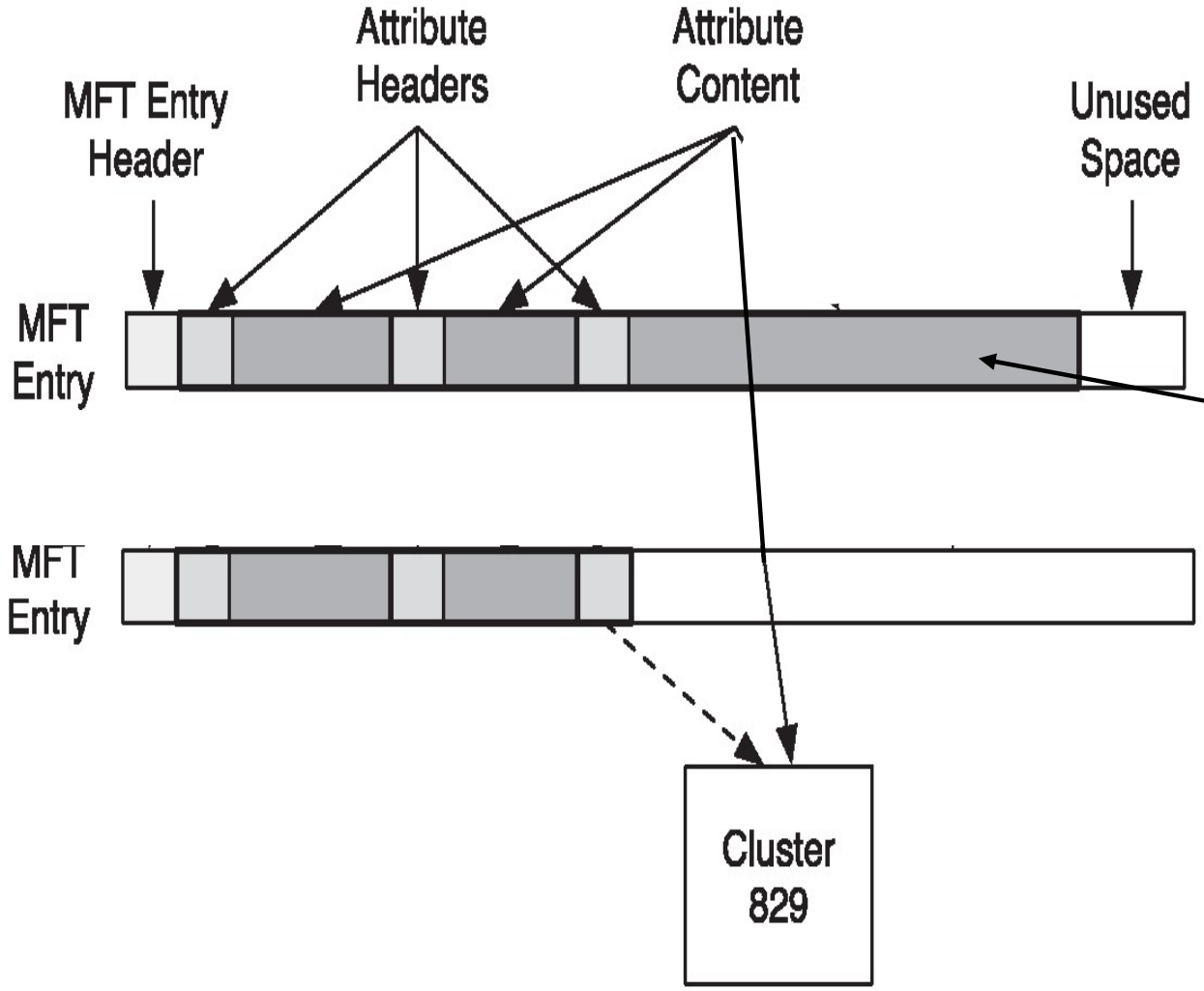
NTFS assigns non-resident cluster runs outside of the MFT

Puts the location in the attribute header

Changes attribute header to be non-resident

Previous resident content is usually moved to the cluster run

Attribute Growth



Suppose this attribute grows

Standard Attribute Types

Overview

There are many "standard" types of attributes

Have well known attribute formats

Have well known uses and content

Have default values

Each attribute type has a type number

Can be redefined in the $AttrDef metadata file

Each attribute type has a name

Syntax: $<allUpperCaseLetters>

Some Default Attribute Types *from Carrier*

Almost all MFT entries have these attributes

All files have a $DATA attribute

All directories have a $INDEX_ROOT attribute

Type Identifier	Name	Description
16 ✓	$STANDARD_ INFORMATION	General information, such as flags; the last accessed, written, and created times; and the owner and security ID.
32	$ATTRIBUTE_LIST	List where other attributes for file can be found.
48 ✓	$FILE_NAME	File name, in Unicode, and the last accessed, written, and created times.
64	$VOLUME_VERSION	Volume information. Exists only in version 1.2 (Windows NT).
64	$OBJECT_ID	A 16-byte unique identifier for the file or directory. Exists only in versions 3.0+ and after (Windows 2000+).
80	$SECURITY_ DESCRIPTOR	The access control and security properties of the file.
96	$VOLUME_NAME	Volume name.
112	$VOLUME_ INFORMATION	File system version and other flags.
128 ✓	$DATA	File contents.
144 ✓	$INDEX_ROOT	Root node of an index tree.
160	$INDEX_ ALLOCATION	Nodes of an index tree rooted in $INDEX_ROOT attribute.
176	$BITMAP	A bitmap for the $MFT file and for indexes.
192	$SYMBOLIC_LINK	Soft link information. Exists only in version 1.2 (Windows NT).
192	$REPARSE_POINT	Contains data about a reparse point, which is used as a soft link in version 3.0+ (Windows 2000+).
208	$EA_INFORMATION	Used for backward compatibility with OS/2 applications (HPFS).
224	$EA	Used for backward compatibility with OS/2 applications (HPFS).
256	$LOGGED_UTILITY_ STREAM	Contains keys and information about encrypted attributes in version 3.0+ (Windows 2000+).

Some Default Attribute Types* *from Nelson*

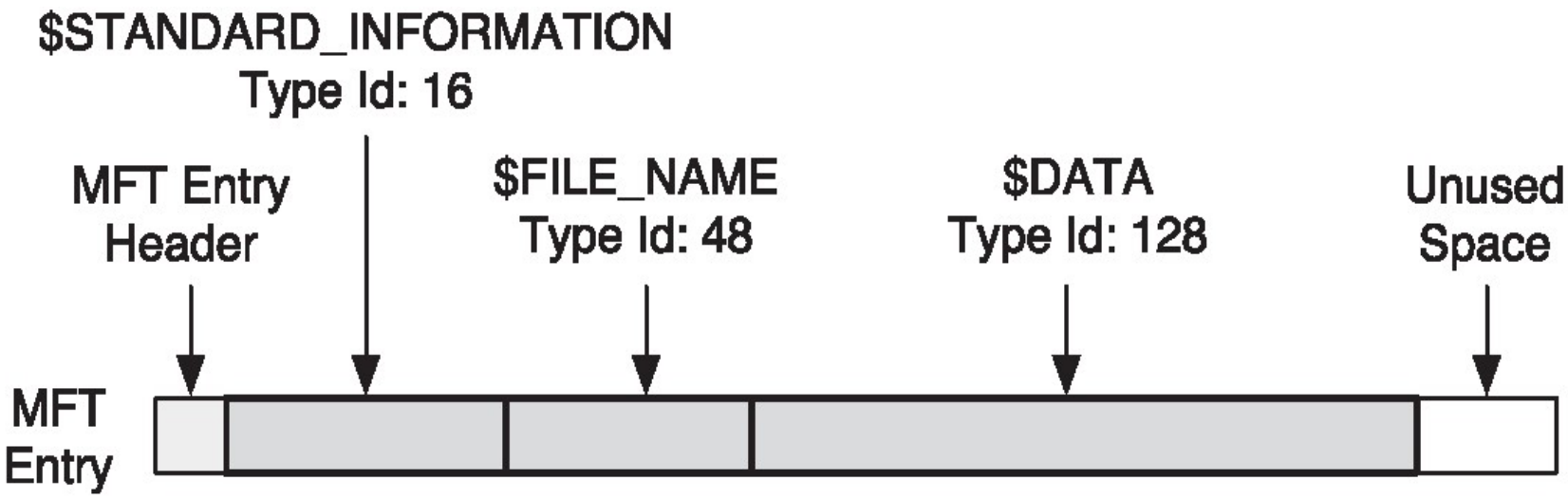
Almost all MFT entries have these attributes

All files have a $DATA attribute

All directories have a $INDEX_ROOT attribute

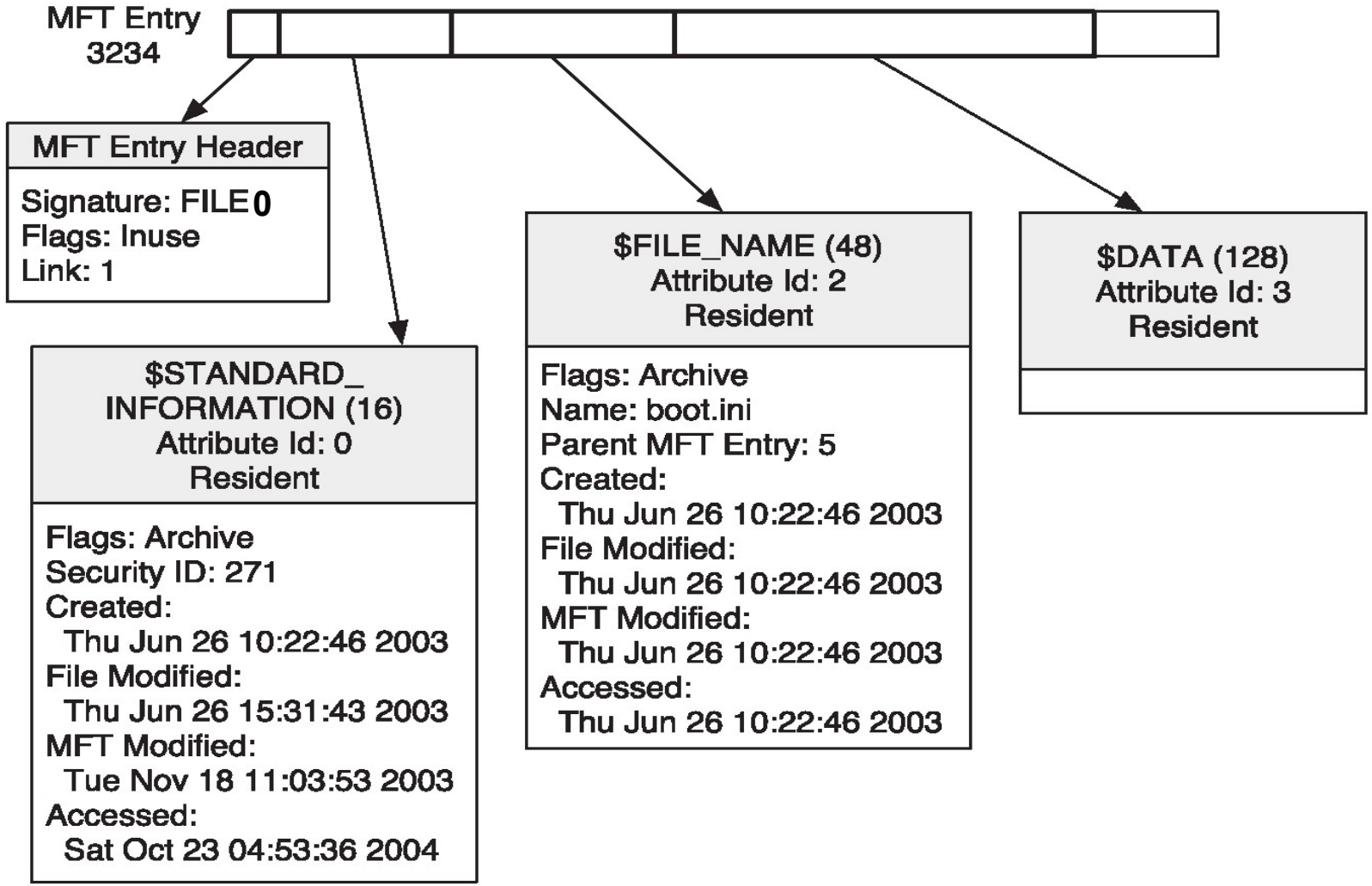
Attribute ID	Purpose
0x10 ✓	$Standard Information This field contains data on file creation, alterations, MFT changes, read dates and times, and DOS file permissions.
0x20	$Attribute_List Attributes that don't fit in the MFT (nonresident attributes) are listed here along with their locations.
0x30 ✓	$File_Name The long and short names for a file are contained here. Up to 255 Unicode bytes are available for long filenames. For POSIX requirements, additional names or hard links can also be listed. Files with short filenames have only one attribute ID 0x30. Long filenames have two attribute ID 0x30s in the MFT record: one for the short name and one for the long name.
0x40	$Object_ID ($Volume_Version in Windows NT) Ownership and who has access rights to the file or folder are listed here. Every MFT record is assigned a unique GUID. Depending on your NTFS setup, some file records might not contain this attribute ID.
0x50	$Security_Descriptor Contains the access control list (ACL) for the file.
0x60	$Volume_Name The volume-unique file identifier is listed here. Not all files need this unique identifier.
0x70	$Volume_Information This field indicates the version and state of the volume.
0x80 ✓	$Data File data for resident files or data runs for nonresident files.
0x90 ✓	$Index_Root Implemented for use of folders and indexes.
0xA0	$Index_Allocation Implemented for use of folders and indexes.
0xB0	$Bitmap A bitmap indicating cluster status, such as which clusters are in use and which are available.
0xC0	$Reparse_Point This field is used for volume mount points and Installable File System (IFS) filter drivers. For the IFS, it marks specific files used by drivers.
0xD0	$EA_Information For use with OS/2 HPFS.
0xE0	For use with OS/2 HPFS.
0x100	$Logged_Utility_Stream This field is used by Encrypting File System (EFS) in Windows 2000 and later

MFT Entry Containing 3 Standard Attributes



This could be a MFT entry for a simple text file.
All attributes are resident including $DATA

MFT Entry Containing 3 Standard Attributes



Allocation Algorithms

Allocation Algorithms

As we previously discussed, there are a number of different algorithms for allocation space on a disk

Let's review 3 allocation algorithms of interest

First available

Next available

Best fit

Allocation Algorithms*
First Available

Allocate disk space from low ordered sectors (or clusters) forward

Fills up the low ordered sectors first

Simple

Fragmentation likely during allocation because the data is not likely to be located as a contiguous whole

Reuses low numbered sectors or clusters

Overwrites previously deleted files

Harder to recover deleted files because high probability of overwriting

Beginning of disk is full; end of disk sparse or empty

Name a file system that uses this algorithm *FAT*

Allocation Algorithms
Next Available

Here the disk is searched for the first sectors (or clusters) available after the last allocation

Fragmentation likely during allocation because the data is not located as a contiguous whole

Relatively simple algorithm

Spreads data over the entire disk better than *First Available* algorithm

Does not overwrite deleted data as often as *First Available* algorithm

Allocation Algorithms
Best Fit

Data is placed on the disk in a location that will minimally accommodate the entire data unit without fragmentation

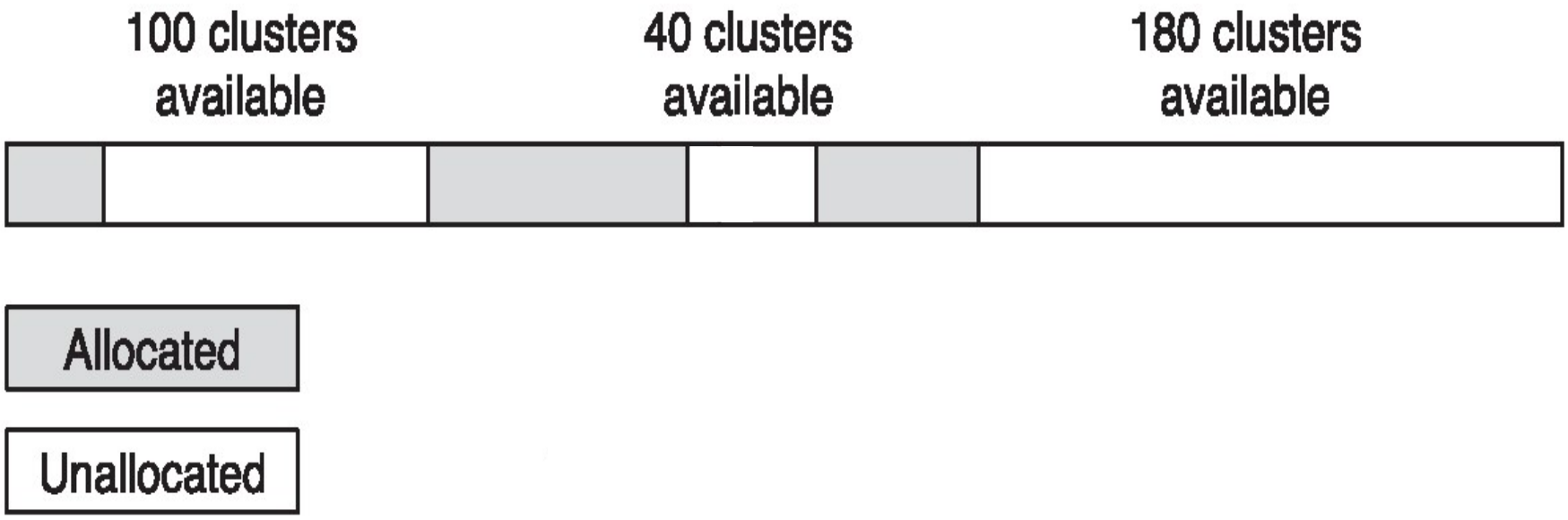
Ideally the contiguous space is just big enough

Large unallocated space is not used if not needed

If a location of sufficient size cannot be found, it then separates the data into a small number of fragments

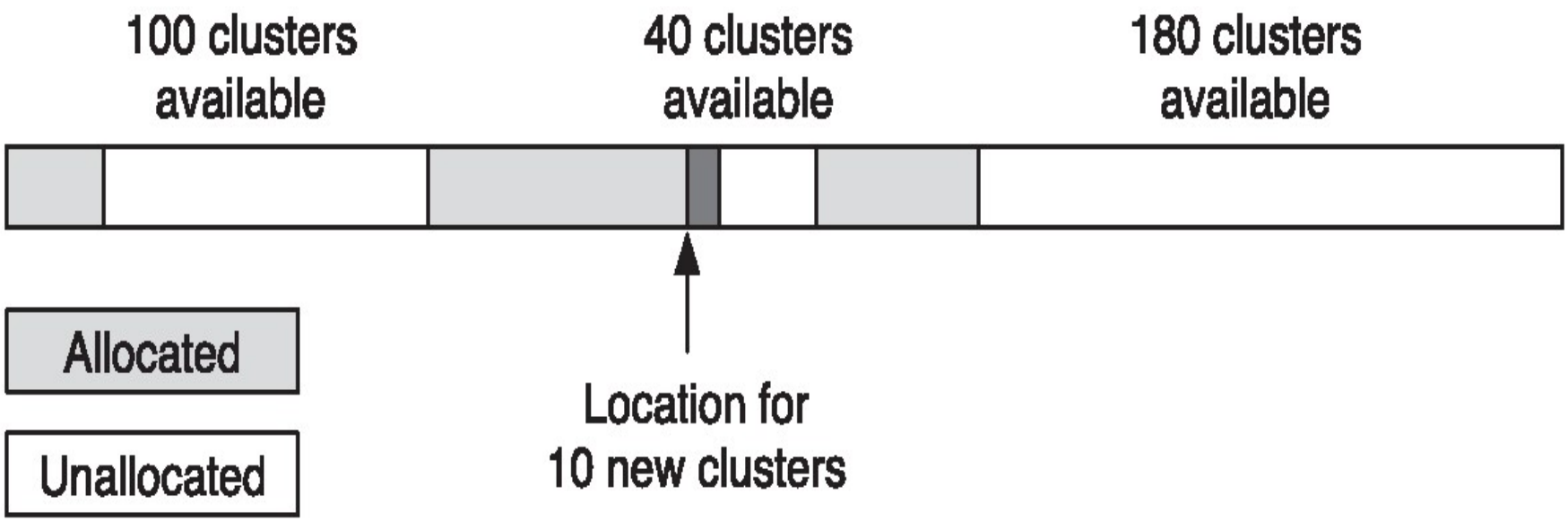
NTFS uses this strategy

Allocation Algorithms
Best Fit



Suppose we wanted to allocate 10 clusters
Where would we put them?

Allocation Algorithms
Best Fit



Suppose we wanted to allocate 10 clusters
Where would we put them?

NTFS File System Layout

NTFS File System Layout

There are no strict layout requirements

But there are some guidelines

Although layout is different for different OSs

MFT Zone

Windows creates an MFT that is as small as possible

It then expands when needed

This leads to fragmentation

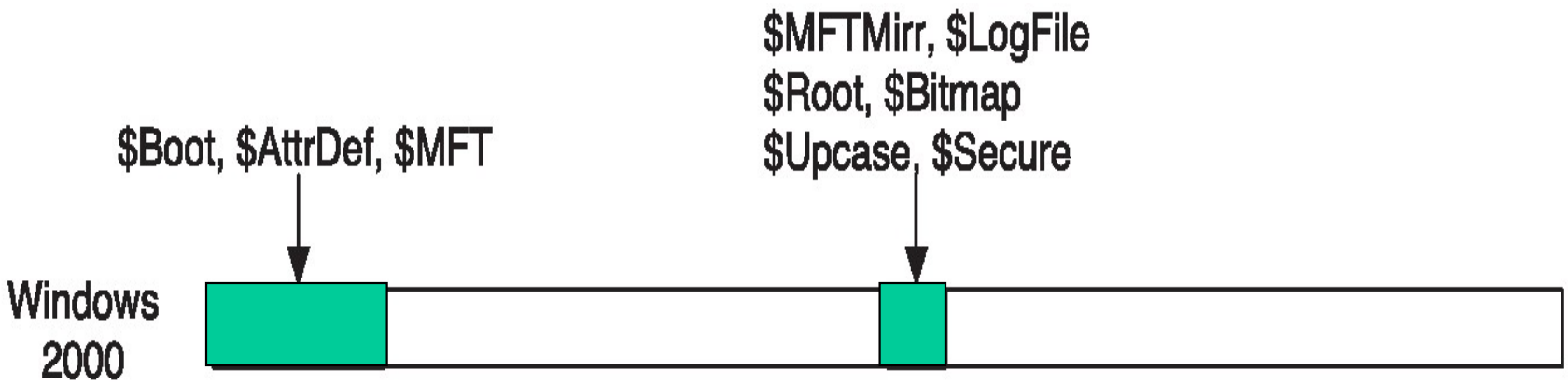
To prevent this OSs allocates a MFT Zone

Consecutive string of clusters not used until all other space has been used

MFT Zone is normally 12.5% of the file system partition

$Boot file is always located in the first set of clusters

Windows 2000 NTFS Metadata File Layout



Windows XP NTFS Metadata File Layout