

New Technologies File System (NTFS) Priscilla Oppenheimer

NTFS

- Default file system for Windows NT, 2000, XP, and Windows Server 2003
- No published spec from Microsoft that describes the on-disk layout
- Good sources for NTFS information
 - [Linux NTFS Project](#)
 - [www.ntfs.com](#)

Microsoft NTFS Goals

- Provide a reliable, secure, scalable, and efficient file system
- Get a foothold in the lucrative business and corporate markets
- Some concepts borrowed from OS/2 High Performance File System (HPFS)

NTFS Features

- Logging
- Transaction-based
- File and folder permissions
- Disk quotas
- Reparse points (used to link files)
- Sparse file support
- Compression
- Encryption
- Alternate data streams

Sparse Files

- Clusters that contain all zeros aren't written to disk
- Analysis considerations
 - A deleted sparse file is hard to recover
 - If file system metadata is deleted or corrupted, a sparse file might not be recoverable

File Compression

- Data is broken into equal-sized compression units (e.g. 16 clusters)
- An attempt is made to compress each unit
- Parts of a file may be compressed while other parts aren't

File Compression Analysis Considerations

- A single file can use different compression methods (e.g. none, sparse, or variant of LZ77)
- Recovery tools need to support decompression
- A deleted compressed file is hard to recover
- If file system metadata is deleted or corrupted, a compressed file might not be recoverable

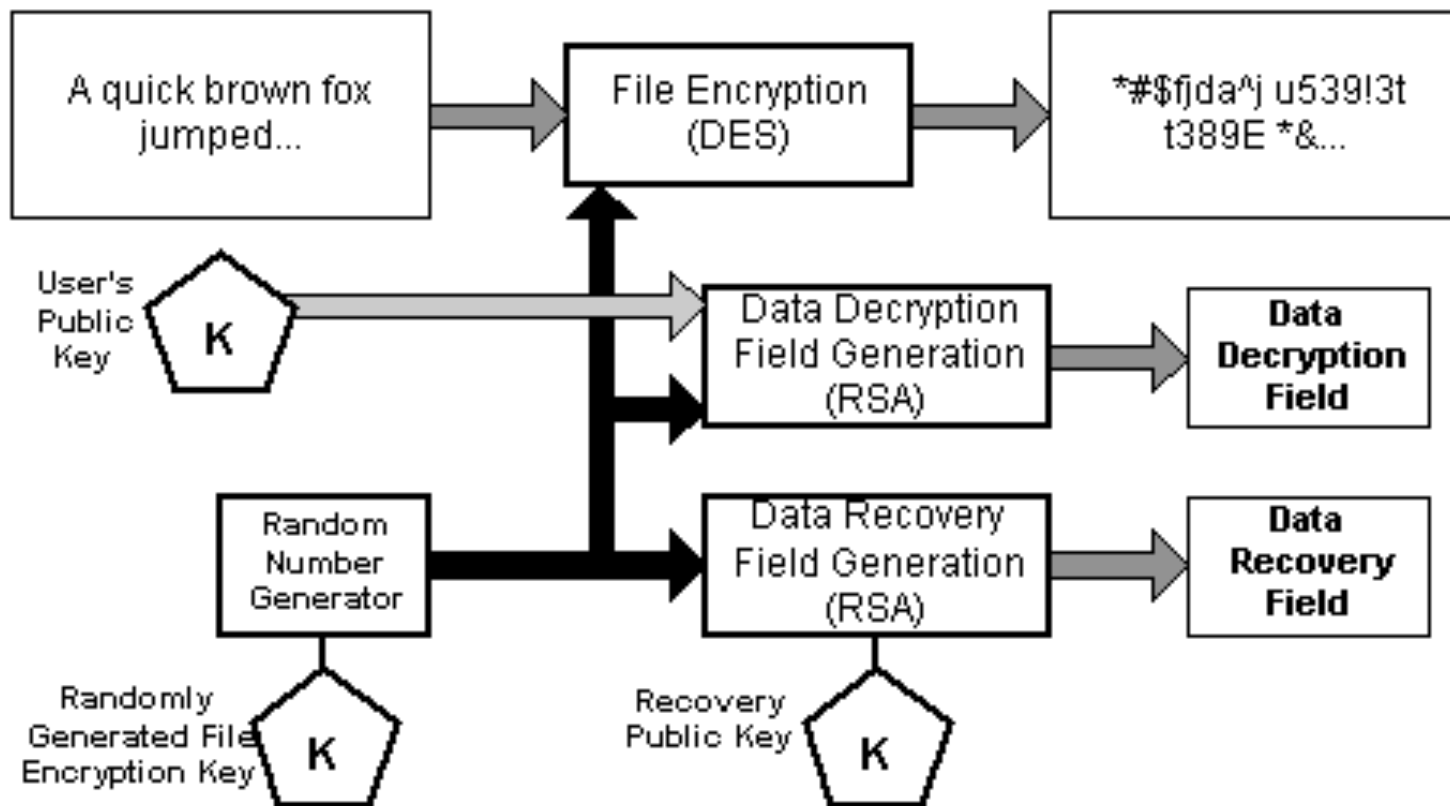
Encrypting File System (EFS)

- Uses both symmetric key encryption (DESX) and asymmetric key encryption (RSA)
- Generates a single file encryption key (FEK) and encrypts file with FEK using DESX
- Stores FEK with file

File Encryption Key Encryption

- FEK is encrypted with user's public key
- FEK is decrypted with user's private key
- If policy allows it, FEK is also encrypted with public key of recovery agent (and decrypted with private key of recovery agent)

File Encryption



Source: NTFS.com

EFS Analysis Considerations

- By default a user's private key is stored in the Windows registry, encrypted with login password as key
 - Login password is susceptible to brute force attack and private key might be compromised
- EFS creates a temporary file (EFS0.TMP) with plaintext data
 - Marks it as deleted when finished but doesn't actually erase contents

Alternate Data Streams

- Data added to a file
- Introduced to support Macintosh files that have a data and resource fork
- Almost impossible to detect with normal file browsing techniques
- A favorite of hackers and criminals

Easy to Create an ADS

```
Directory of C:\adstest
02/14/2004  04:47p      <DIR>          .
02/14/2004  04:47p      <DIR>          ..
07/26/2000  09:00a                91,408 calc.exe
              1 File(s)                91,408 bytes
              2 Dir(s)           684,425,216 bytes free

C:\adstest>type c:\winnt\system32\notepad.exe>calc.exe:notepad.exe
█

C:\adstest>dir
Volume in drive C has no label.
Volume Serial Number is 8C3F-115B

Directory of C:\adstest
02/14/2004  04:47p      <DIR>          .
02/14/2004  04:47p      <DIR>          ..
02/14/2004  04:51p                91,408 calc.exe
              1 File(s)                91,408 bytes
              2 Dir(s)           684,371,968 bytes free

C:\adstest>
```

Source: WindowSecurity.com

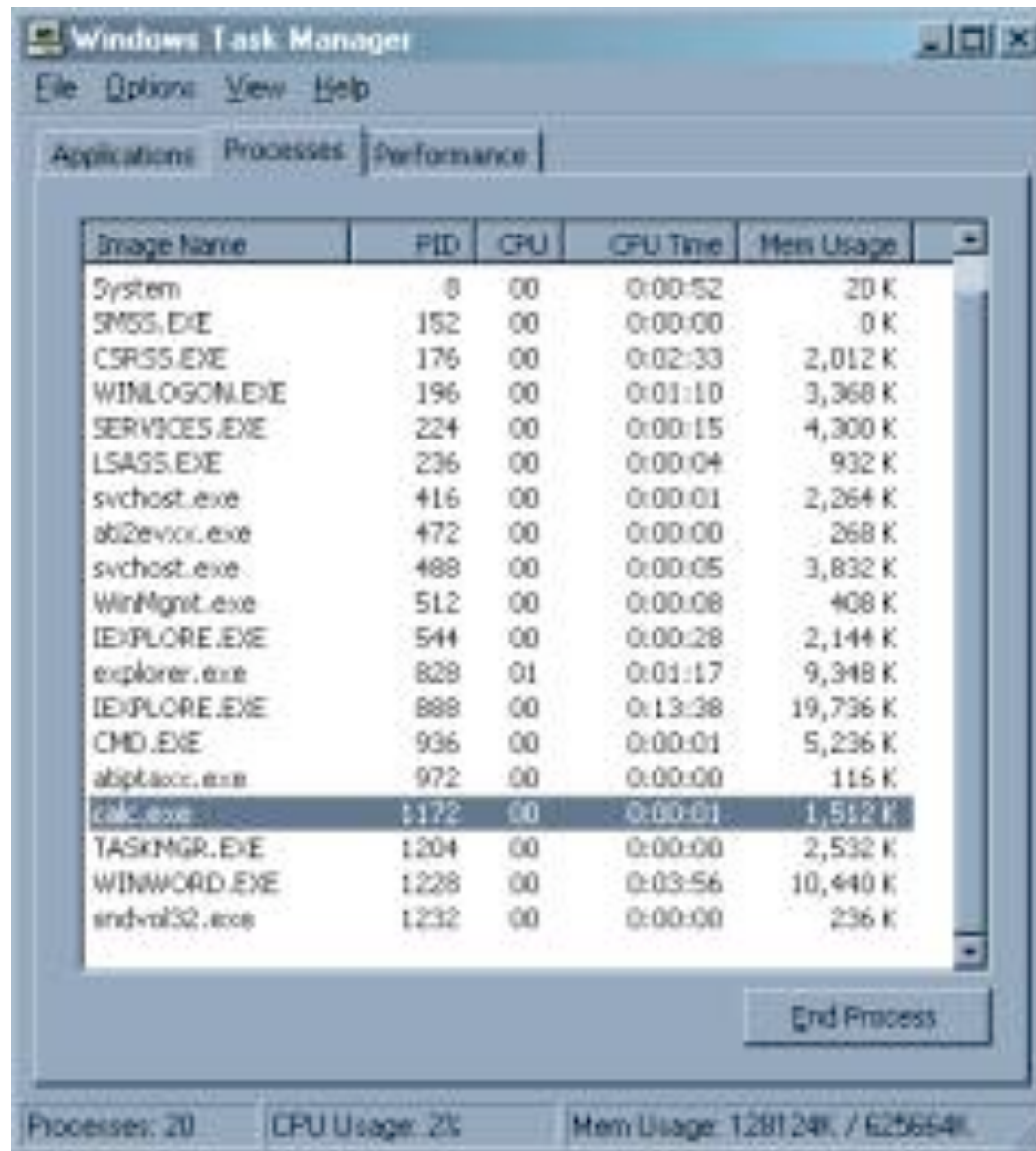
Start the Program

```
Directory of C:\adstest
02/14/2004  04:47p      <DIR>      .
02/14/2004  04:47p      <DIR>      ..
02/14/2004  04:51p                91,408 calc.exe
                1 File(s)                91,408 bytes
                2 Dir(s)          684,371,968 bytes free

C:\adstest>start c:\adstest\calc.exe:notepad.exe

C:\adstest>
```

What Program Is Running?



The image shows a screenshot of the Windows Task Manager Performance tab. The window title is "Windows Task Manager" and it has a menu bar with "File", "Options", "View", and "Help". The "Performance" tab is selected, showing a list of running processes. The list has columns for "Image Name", "PID", "CPU", "CPU Time", and "Mem Usage". The "calc.exe" process is highlighted in blue. At the bottom of the window, there is a status bar showing "Processes: 20", "CPU Usage: 2%", and "Mem Usage: 128124K / 625654K".

Image Name	PID	CPU	CPU Time	Mem Usage
System	0	00	0:00:52	20 K
SMSS.EXE	152	00	0:00:00	0 K
CSRSS.EXE	176	00	0:02:33	2,012 K
WINLOGON.EXE	196	00	0:01:10	3,368 K
SERVICES.EXE	224	00	0:00:15	4,300 K
LSASS.EXE	236	00	0:00:04	932 K
svchost.exe	416	00	0:00:01	2,264 K
ab2evxx.exe	472	00	0:00:00	268 K
svchost.exe	488	00	0:00:05	3,832 K
WinMgmt.exe	512	00	0:00:08	408 K
IEXPLORE.EXE	544	00	0:00:28	2,144 K
explorer.exe	828	01	0:01:17	9,348 K
IEXPLORE.EXE	888	00	0:13:38	19,736 K
CMD.EXE	936	00	0:00:01	5,236 K
abptaacc.exe	972	00	0:00:00	116 K
calc.exe	1172	00	0:00:01	1,512 K
TASKMGR.EXE	1204	00	0:00:00	2,532 K
WINWORD.EXE	1228	00	0:03:56	10,440 K
endvof32.exe	1232	00	0:00:00	236 K

Processes: 20 CPU Usage: 2% Mem Usage: 128124K / 625654K

NTFS Basic Concepts

- Everything is a file
- Files have attributes
 - \$SOME_UPPER_CASE_THING
 - \$FILE_NAME
 - \$STANDARD_INFORMATION
 - \$DATA (the actual content)

File System Metadata Files

- Files that store file system administrative data
- Note that they are files (unlike FAT which was a separate data structure)
- Name begins with \$ and first letter is capitalized
 - \$MFT
 - \$LogFile

Master File Table

- Contains information about all files and directories
- Every file and directory has at least one entry in the table
- Each entry is simple
 - 1 KB in size
 - Entry header is first 42 bytes
 - Remaining bytes store attributes

File System Metadata Files

First 16 MFT Entries Are Reserved

Entry	File Name	Description
0	\$MFT	Entry for MFT itself
1	\$MFTMirr	Backup of MFT
2	\$LogFile	Journal
3	\$Volume	Volume label, etc.
4	\$AttrDef	IDs for attributes
5	/	Root directory
6	\$Bitmap	Allocation status of clusters
7	\$Boot	Boot sector
8	\$BadClus	Clusters with bad sectors

Resident and Non-Resident Attributes

- A resident attribute stores its content in the MFT entry
- A non-resident attribute stores its content in external clusters
- Non resident attributes are stored in cluster runs
- The attribute header gives the starting cluster address and its run length

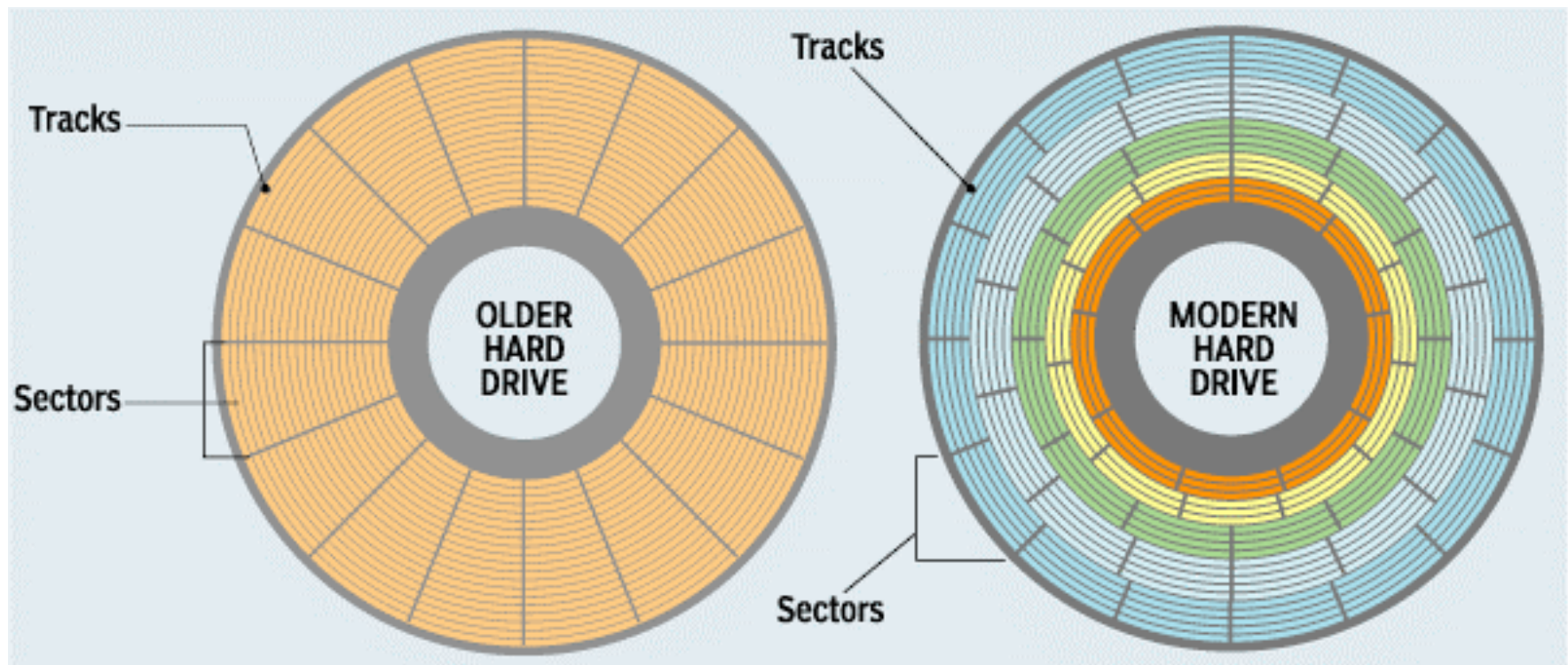
Non-Resident Attributes

- \$DATA attribute for files > 1 KB
- \$DATA attribute for \$Boot
- \$DATA attribute for \$MFTMirr
- \$DATA attribute for \$LogFile

A Step Backwards...

Hard Disk Drives Review

- Factory low-level formatting defines tracks and sectors on a blank disk
 - A track contains many sectors
 - A sector is typically 512 bytes
 - A sector is the minimum I/O unit



Clusters

- A cluster is a group of consecutive sectors
- A cluster is the minimum file allocation unit
- The number of sectors per cluster is a power of 2
 - The number is stored in the volume boot sector
 - Typical values are $2^1=2$, $2^2=4$, $2^3=8$, $2^4=16$

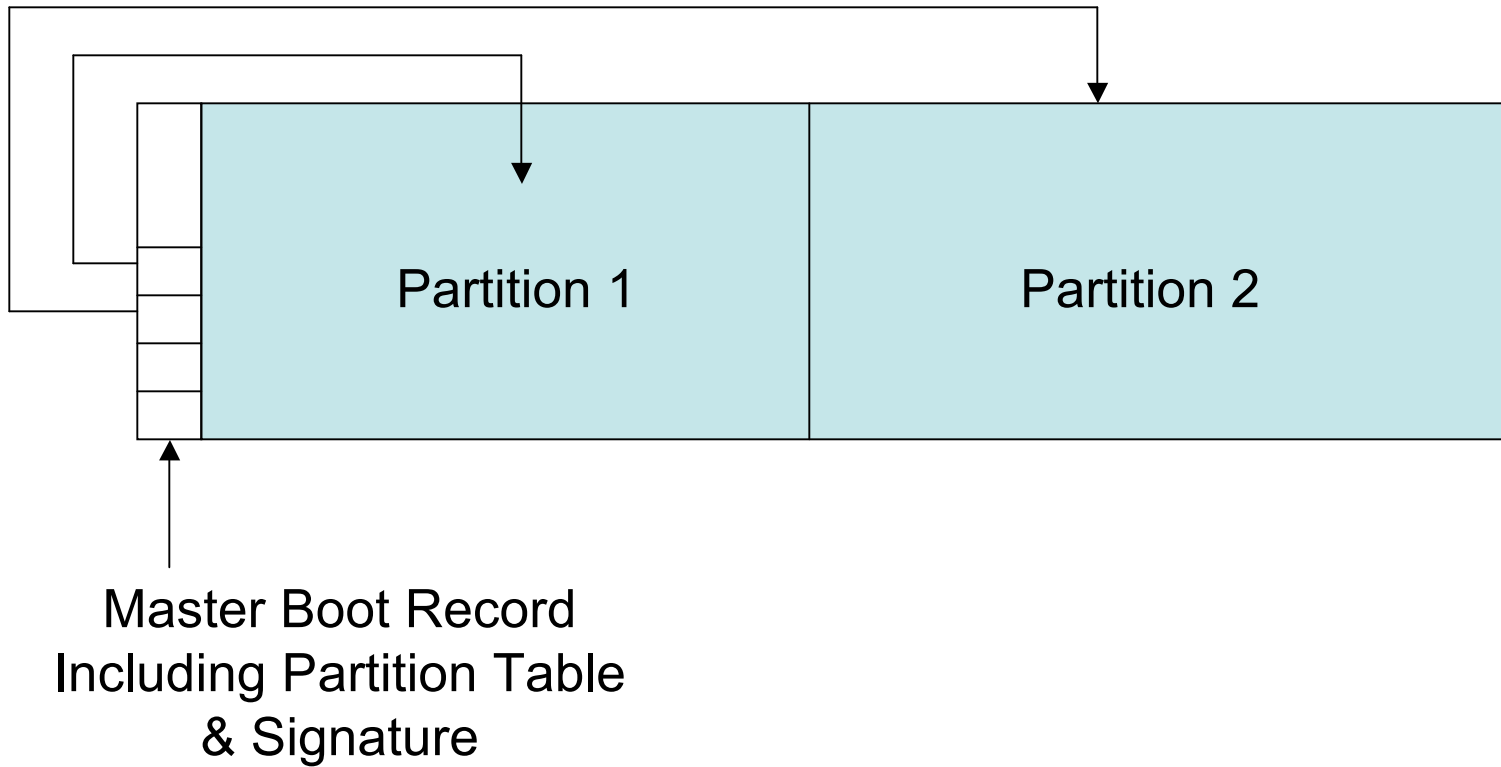
Partitions

- The user creates partitions (logical drives or volumes)
 - Each partition uses a file system
 - FAT12, FAT16, FAT32, NTFS on Windows systems
 - EXT2, EXT3, UFS1, UFS2 on Linux and UNIX systems

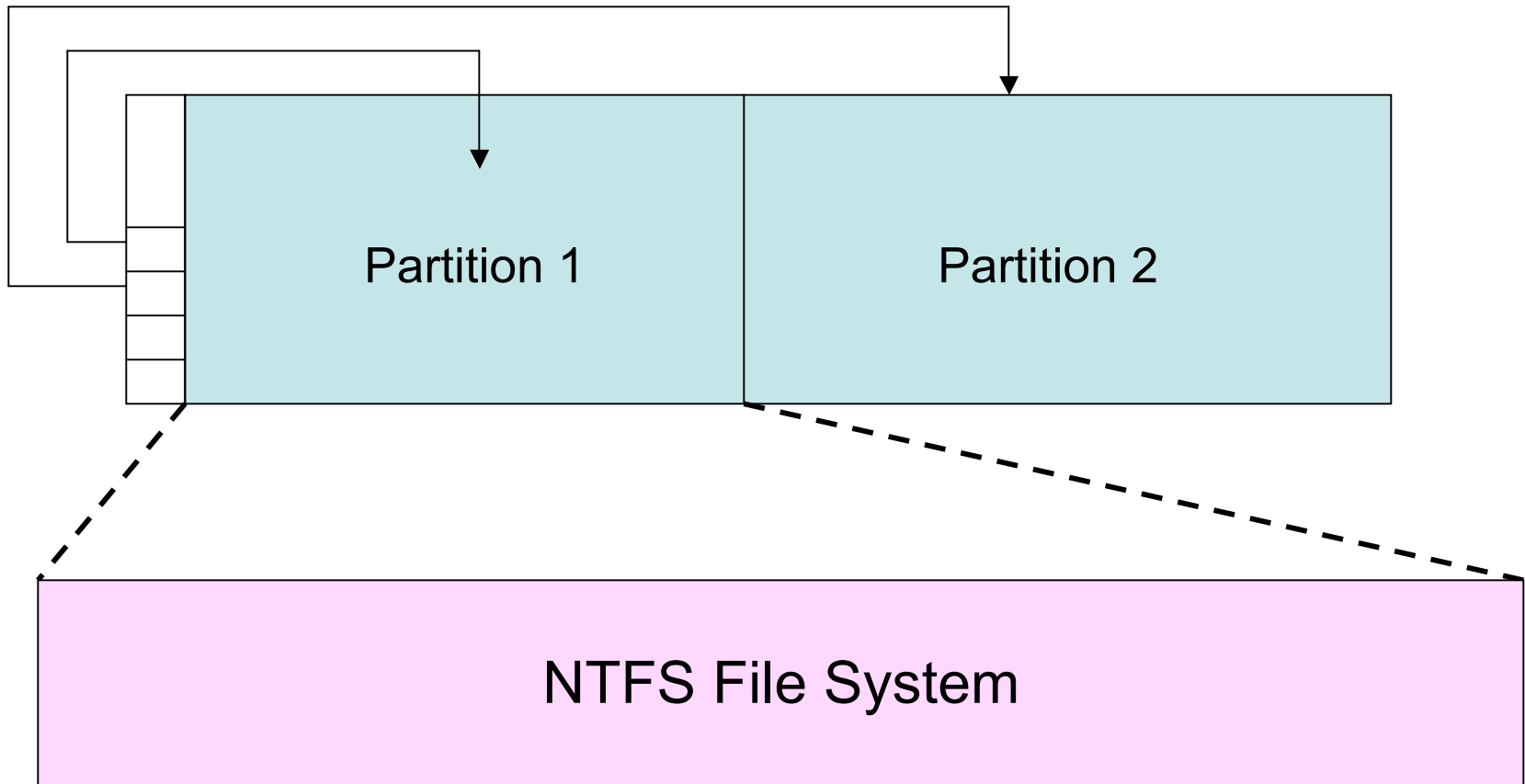
File Systems

- High-level formatting creates file system data structures
 - Root directory
 - Data that tracks which clusters are unused, allowing the OS to find available clusters quickly
 - File Allocation Table (FAT) on older Windows systems
 - \$Bitmap file in the Master File Table (MFT) on newer Windows systems

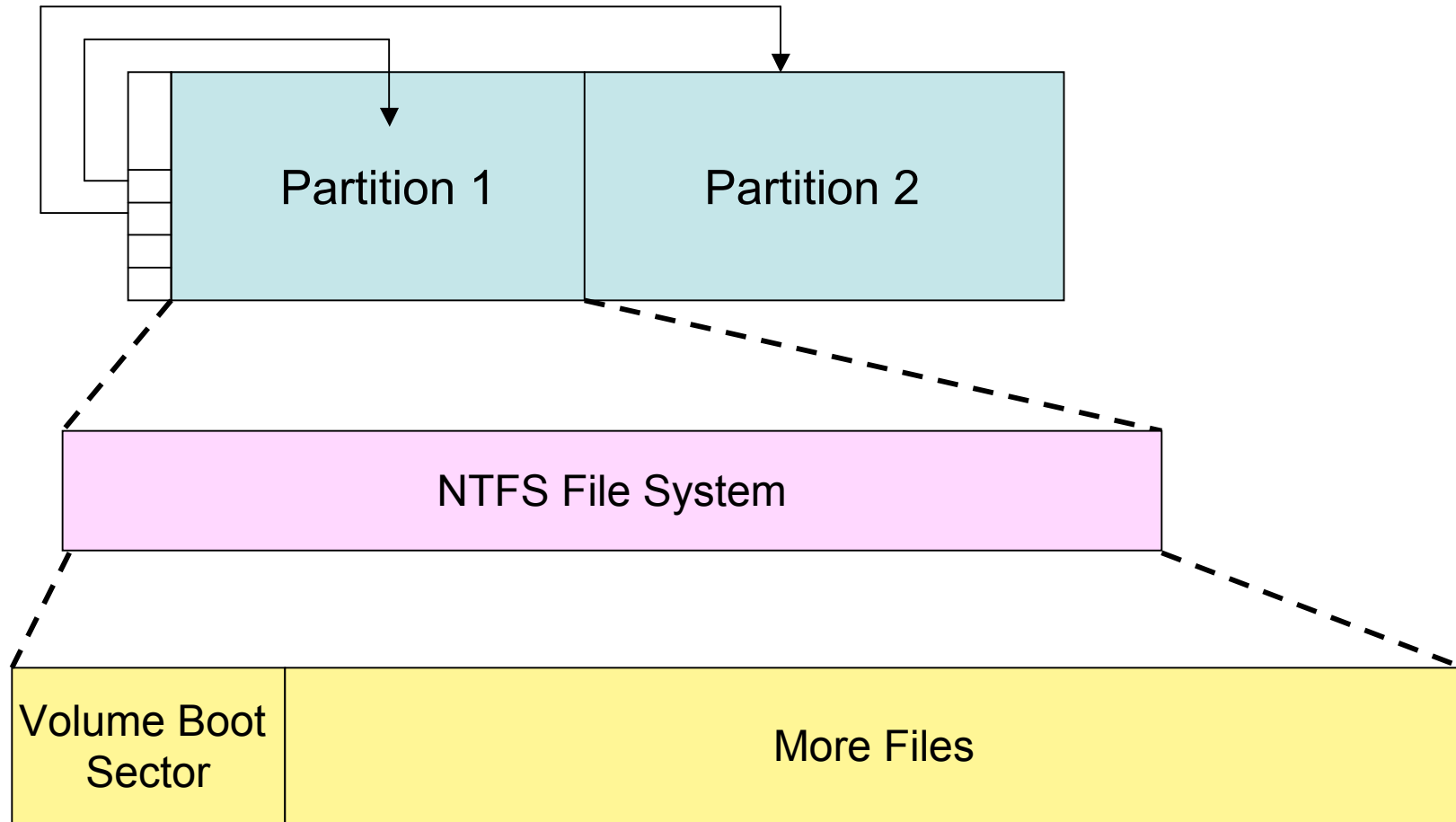
DOS Disk Review



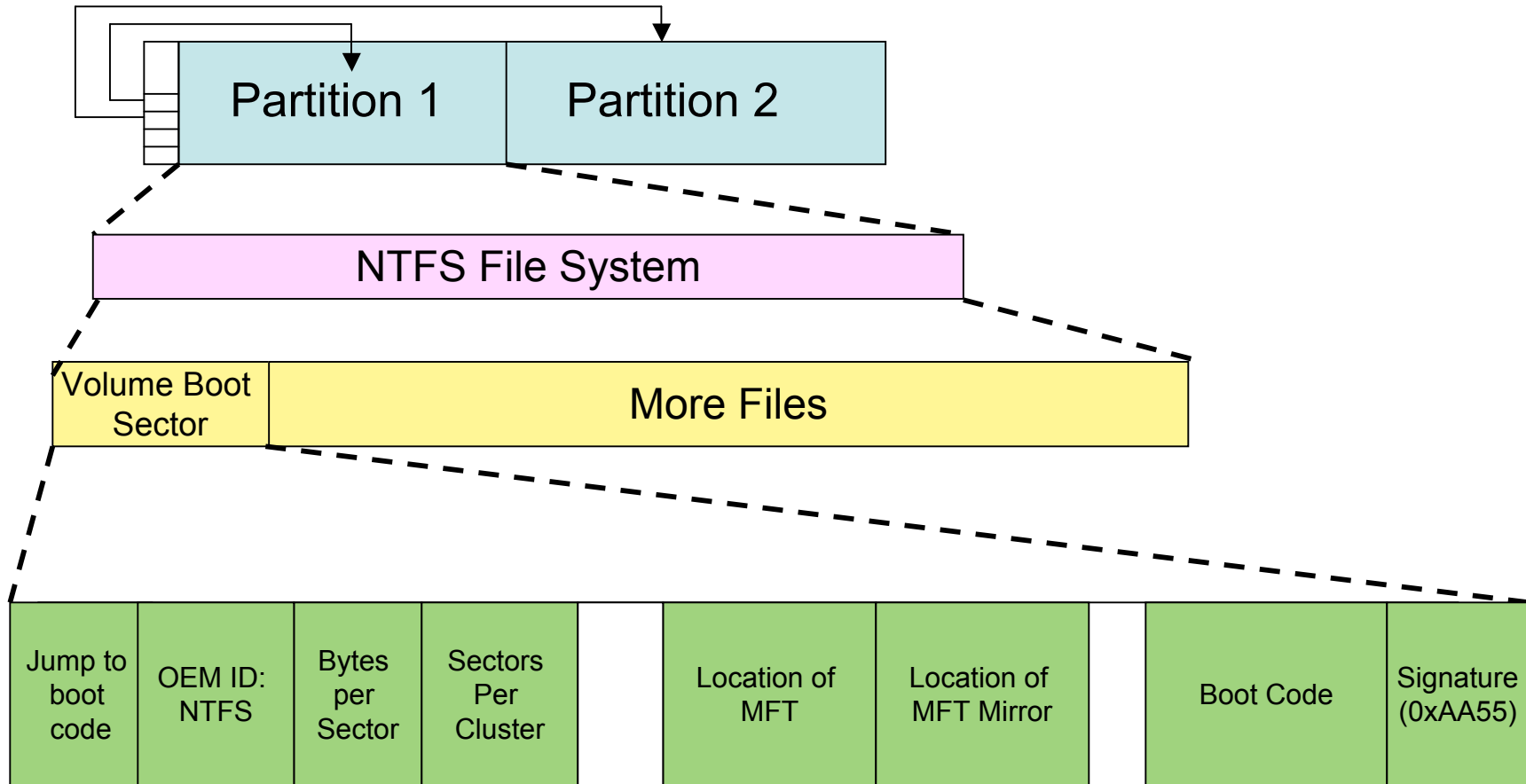
Partition Holds an NTFS File System



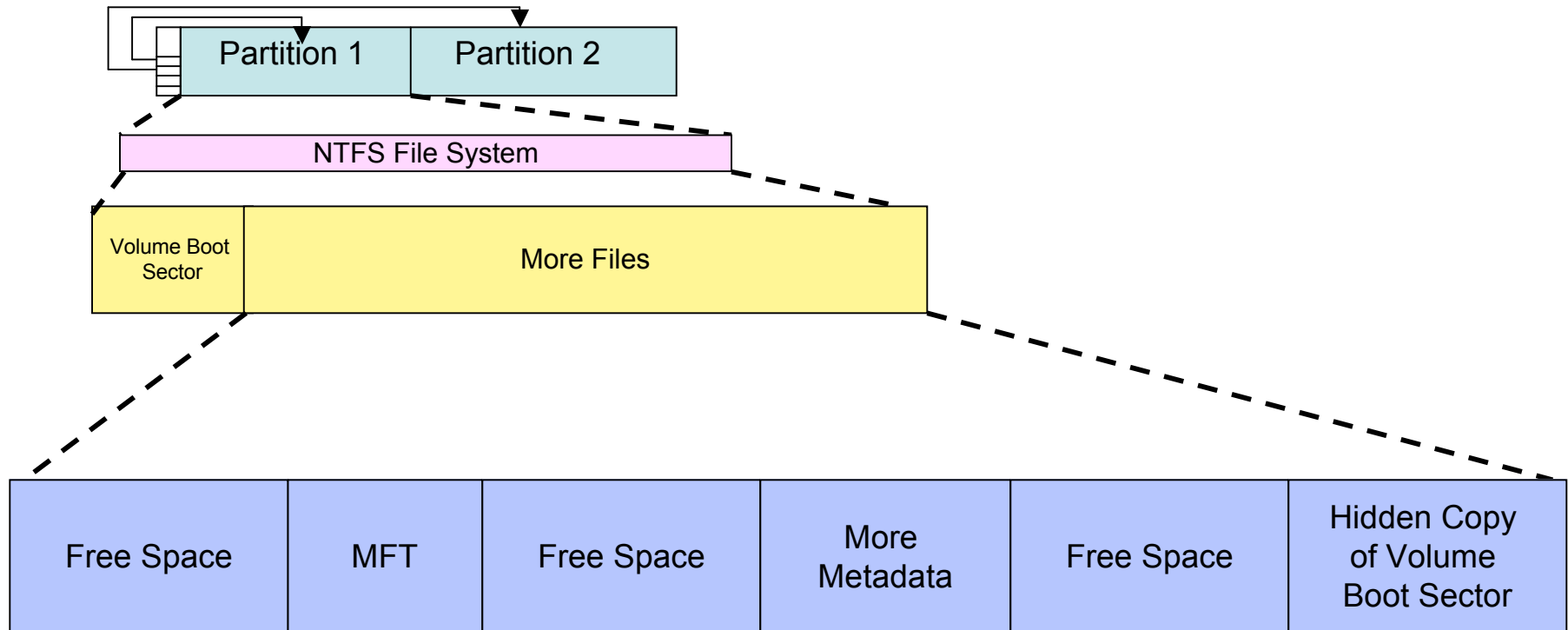
NTFS: Everything Is a File



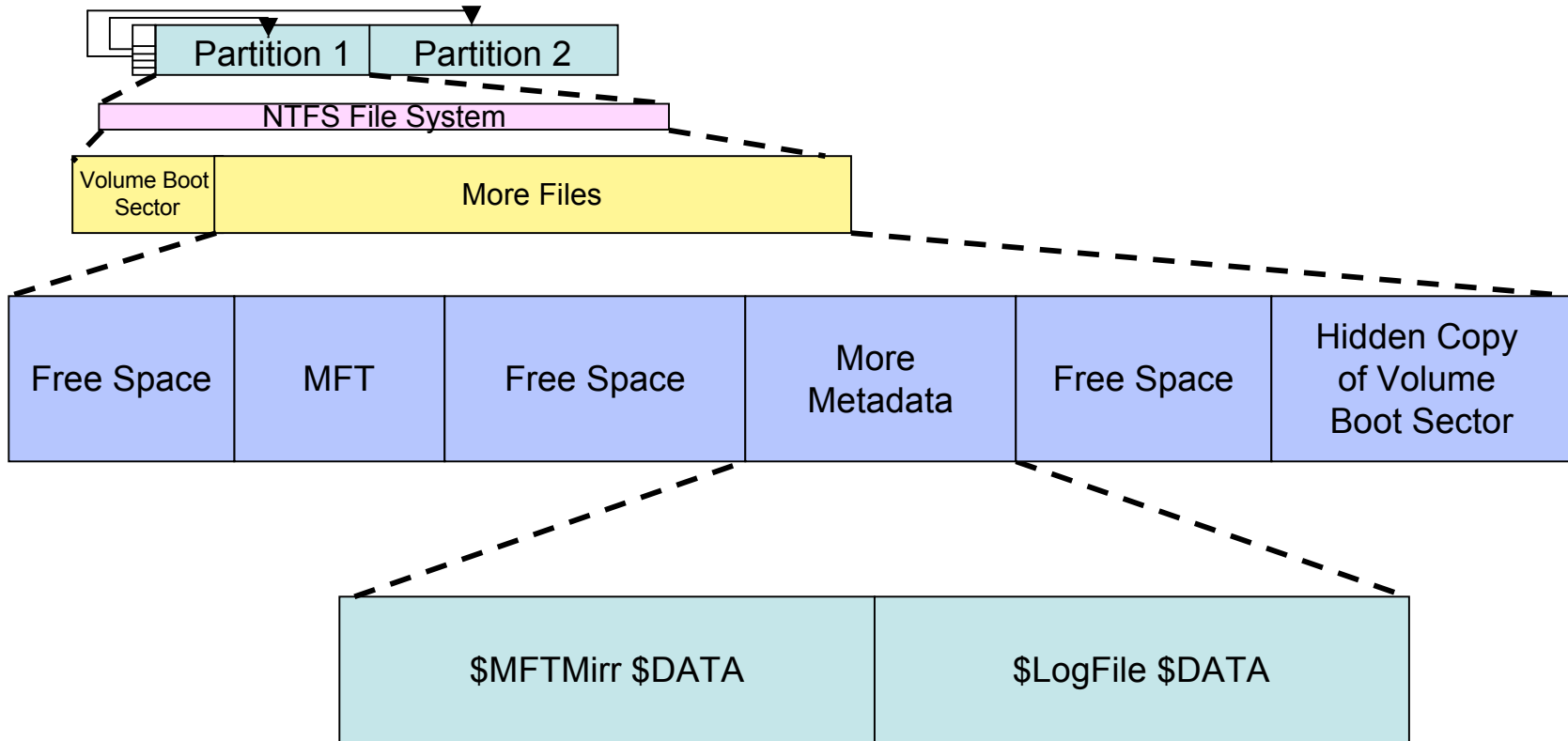
NTFS Volume Boot Sector



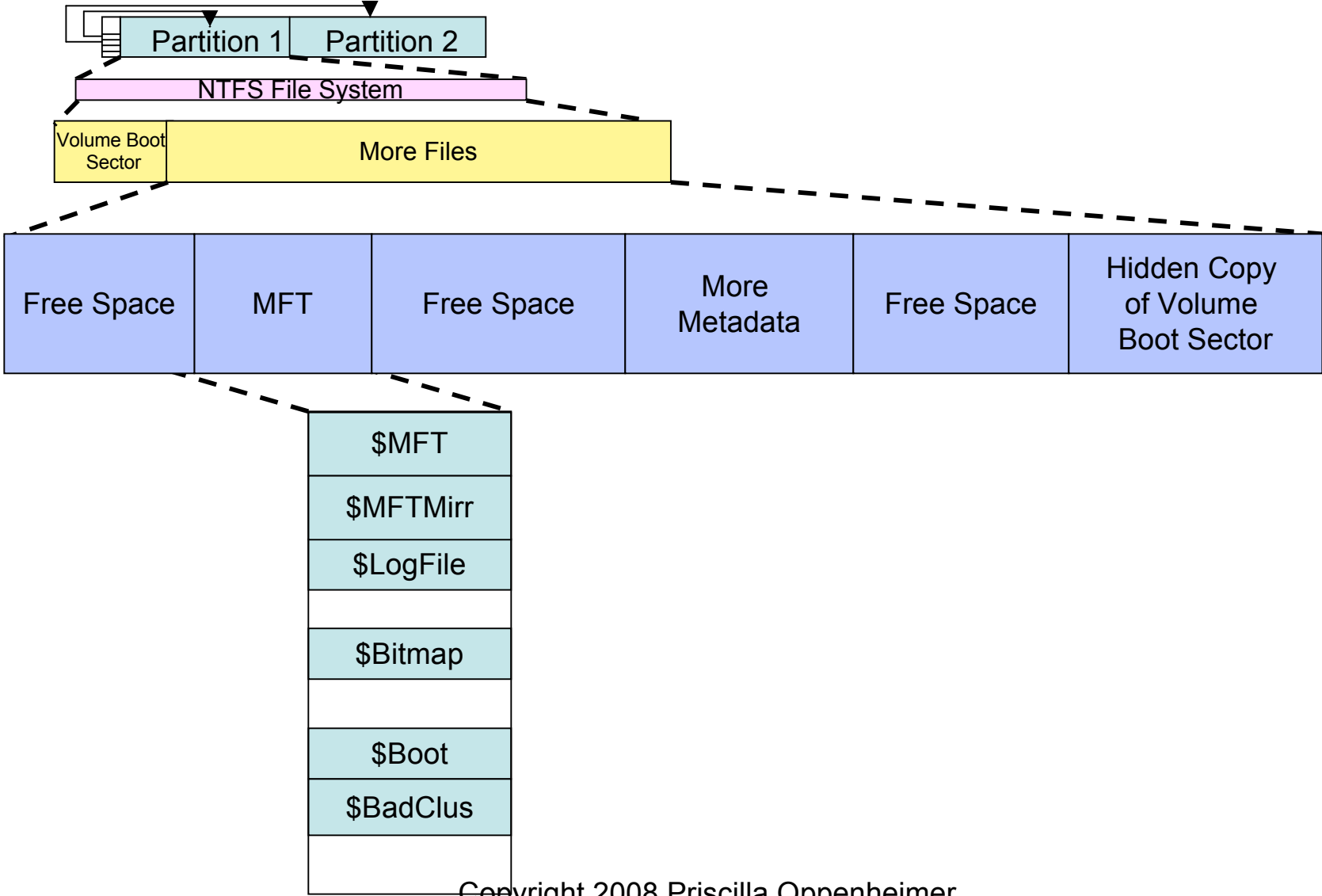
A Freshly Formatted NTFS Volume



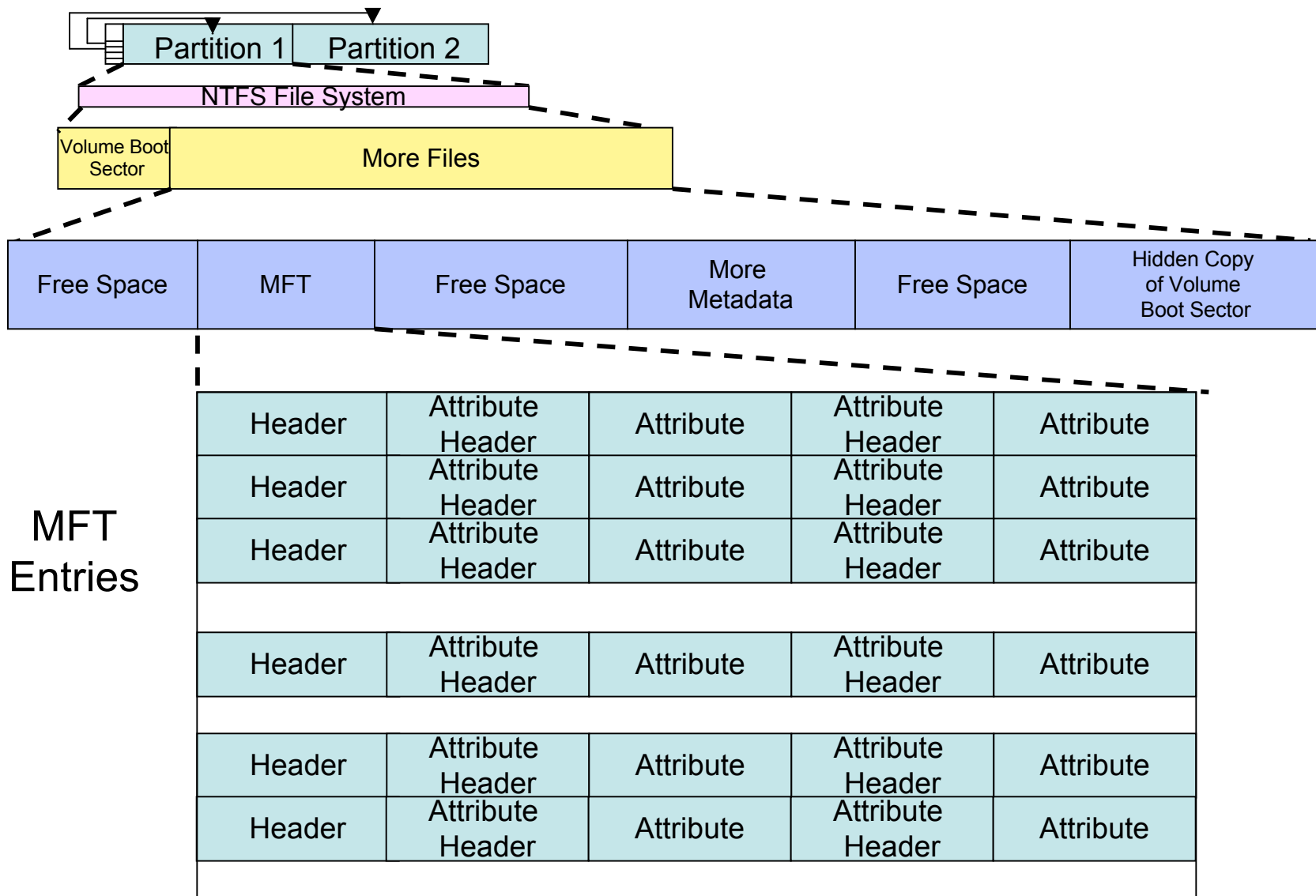
Metadata in Center of Volume



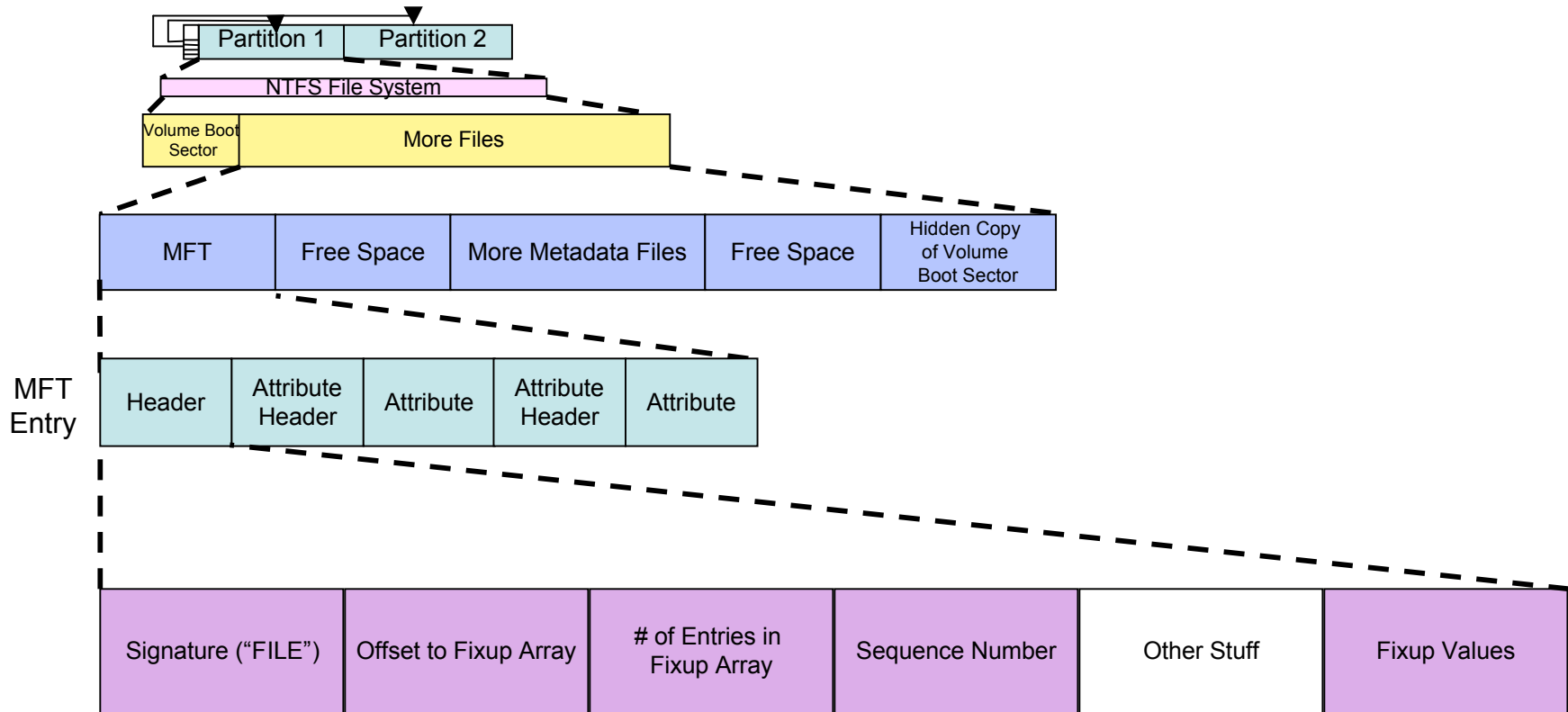
MFT



MFT Attributes



MFT Entry Header

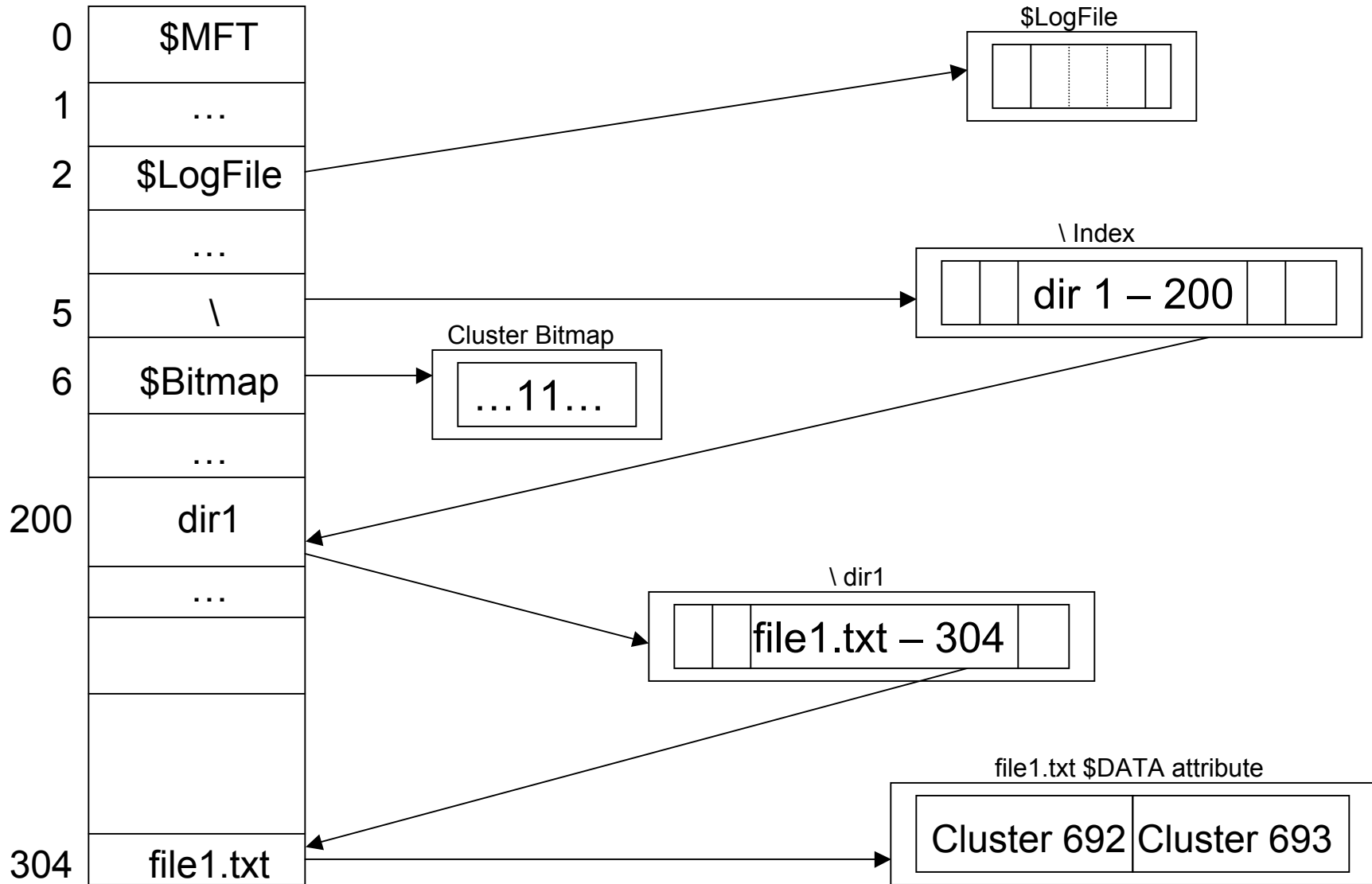


Creating an NTFS File

Creating dir1\file1.txt

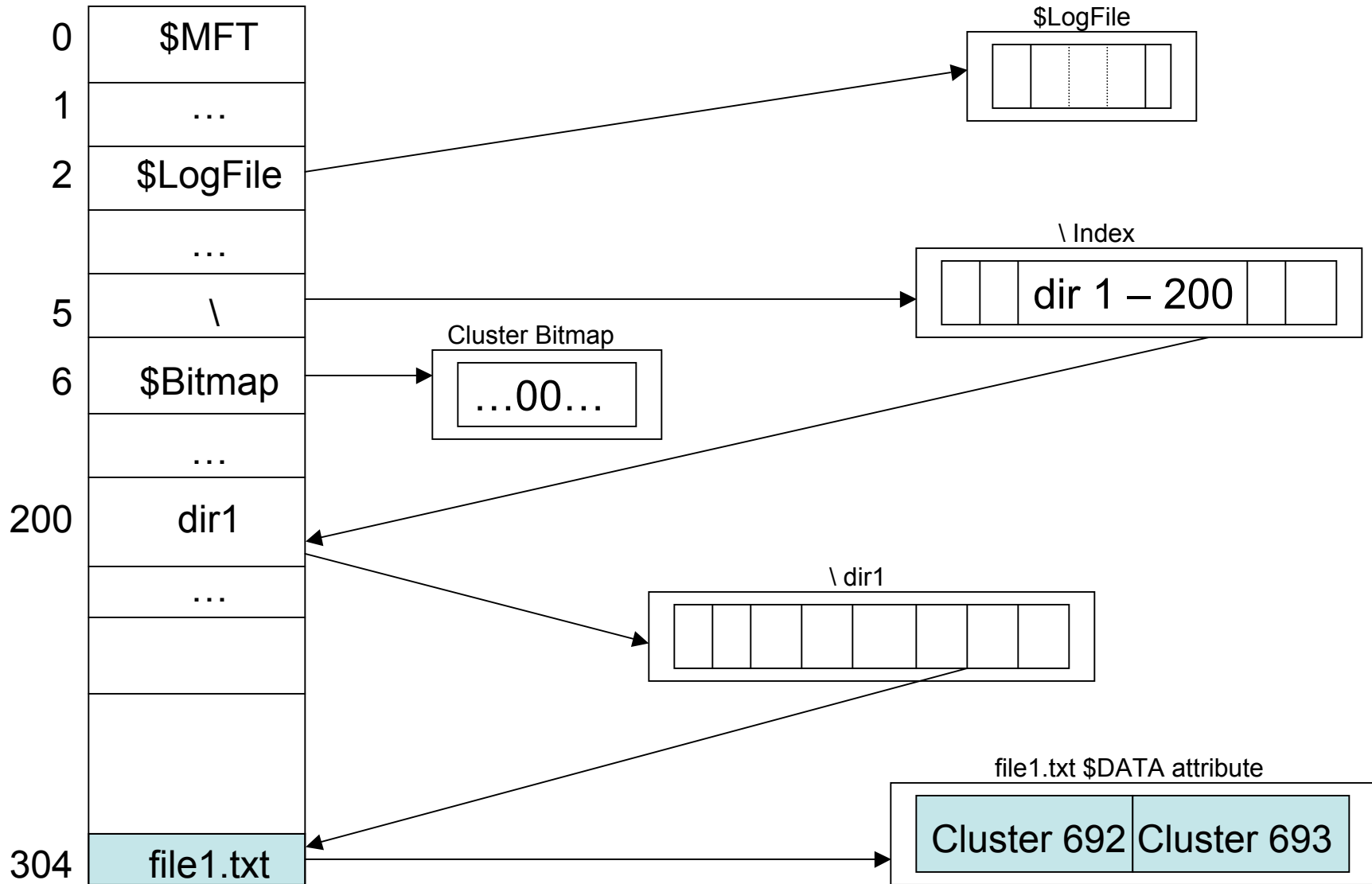
1. Read volume boot sector to locate MFT.
2. Read first entry in MFT to determine layout of MFT.
3. Allocate an MFT entry for the new file.
4. Initialize MFT entry with \$STANDARD_INFORMATION, etc.
5. Check MFT \$Bitmap to find free clusters, using best-fit algorithm.
6. Set corresponding \$Bitmap bits to 1.
7. Write file content to clusters and update \$DATA attribute with starting address of cluster run and run length.
8. Read root directory (MFT entry 5), traverse index, and find dir1.
9. Read \$INDEX_ROOT attribute for dir1 and determine where file1.txt should go.
10. Create new index entry; resort index tree.
11. Enter steps in \$LogFile (as each step is taken).

An NTFS File



NTFS File Deleted

Data in blue boxes is unallocated



Deleting an NTFS File

Deleting dir1\file1.txt

1. Read volume boot sector to locate MFT.
2. Read first entry in MFT to determine layout of MFT.
3. Read root directory (MFT entry 5), traverse index, and find dir1.
4. Read \$INDEX_ROOT for dir1 entry and find file1.txt entry.
5. Remove filename entry from index; move other entries over.
6. Set MFT \$Bitmap entries to 0.
7. Enter steps in \$LogFile (as each step is taken).