File System Forensics
FAT and NTFS

FAT File Systems

File Allocation Table (FAT) File Systems

- Simple and common
- Primary file system for DOS and Windows 9x
- Can be used with Windows NT, 2000, and XP
  - New Technologies File System (NTFS) is default for NT, 2000, and XP
- Supported by all Windows and UNIX varieties
- Used in flash cards and USB thumb drives

The FAT Family

- FAT12, FAT16, FAT32
  - The number refers to the quantity of bits used in the FAT to refer to clusters

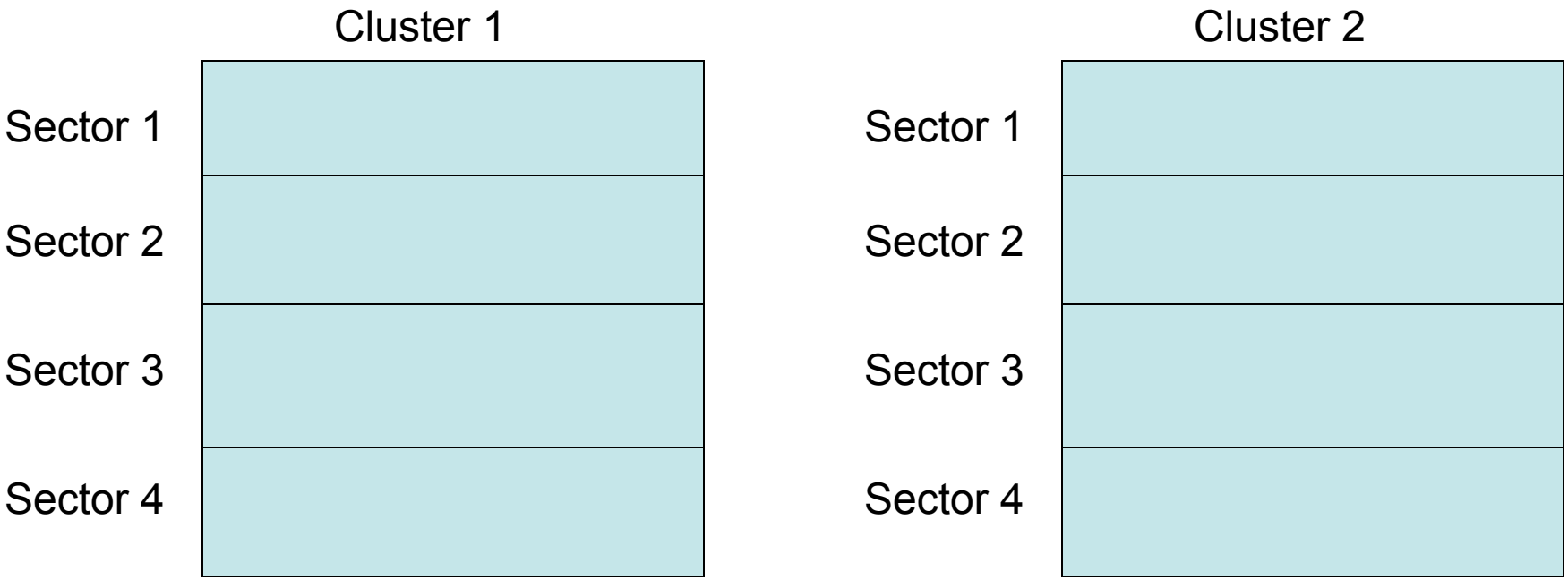
Disk Storage Review

- Data is stored on disks one entire sector at a time
  - A sector is usually 512 bytes
  - If you use only one byte, the system still provides the other 511 bytes for you
  - A sector is the minimum size read from, or written to, a disk
  - A sector is the minimum I/O unit

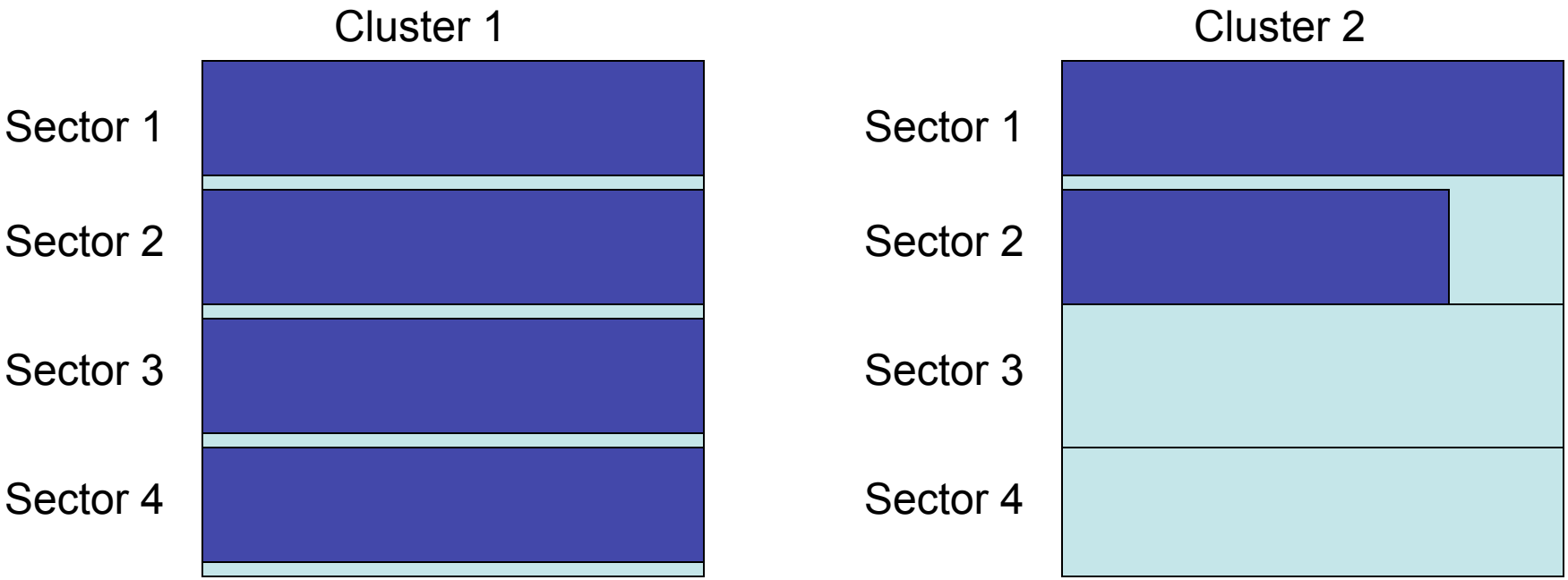
Disk Storage Review (cont.)

- Space is allocated to a file one cluster at a time
  - A cluster is a fixed number of sectors
    - Must be a power of 2 (1,2,…64)
  - Unused sectors retain the data that was on them prior to allocation
  - A cluster is the minimum file allocation unit

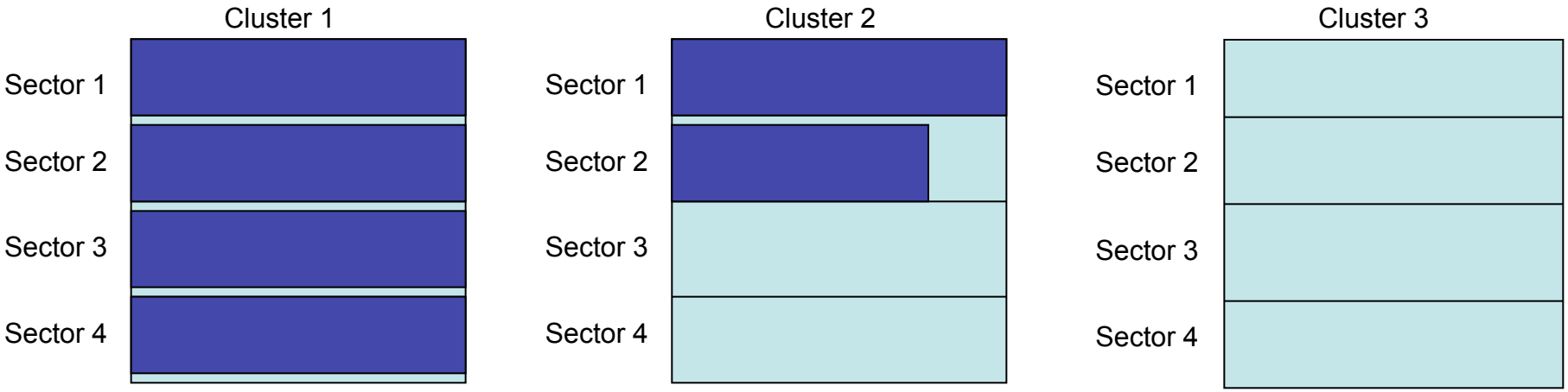
Clusters



File Data (Example 1)



File Data (Example 2)



Slack

- Slack is the space allocated to a file, but unused
  - Space at the end of a sector that remains unused by the file
  - Sectors allocated to the file that the file hasn't yet used
- Slack space often contains useful evidence
  - Unused bytes in an allocated sector are less useful
  - Unused sectors in an allocated cluster retain their original contents and are very useful

Unallocated Clusters

- Many clusters on a modern hard drive are unallocated
- Unallocated clusters may have been allocated earlier though
  - These clusters retain their data until they are reallocated to a new file
  - Deleted files are still recoverable!

Cluster Allocation Algorithms

- First available
  - Always start at the beginning of the file system
  - Fragmented files common
  - Recovery of deleted content better at end of file system

Cluster Allocation Algorithms

- Best fit
  - Search for consecutive clusters that fit the size of file
  - Only works for files that don't grow
- Next available
  - Start search with the cluster that was most recently allocated
  - More balanced for data recovery
  - Used by Windows 98 and XP

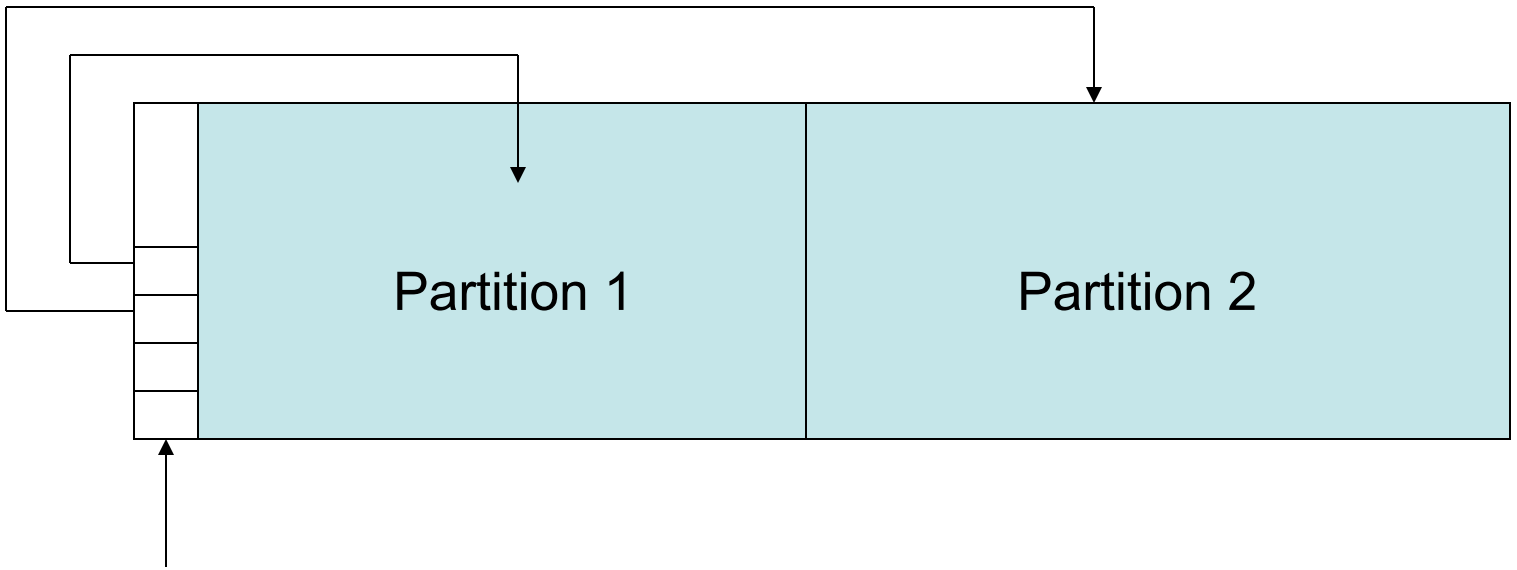
Partitions Review

- The user creates partitions (logical drives or volumes)
  - Creates Master Boot Record with partition table
  - Each partition uses a file system
    - FAT12, FAT16, FAT32, NTFS on Windows systems
    - EXT2, EXT3, UFS1, UFS2 on Linux and UNIX systems
- Recovery tools can often find data even if the disk was repartioned
  - Look for tell-tale symptoms of a file system
  - FAT file systems have 0x55AA in bytes 510 and 511 of the partition, for example

Partitions Review

- MBR in first 512-byte sector on disk
  - Boot code (Bytes 0-445)
  - Partition table (bytes 446-509)
  - Signature (bytes 510-511, value = 0x55AA)
- Partition table has four entries
  - Disk has four primary partitions
  - A primary partition may hold extended partitions

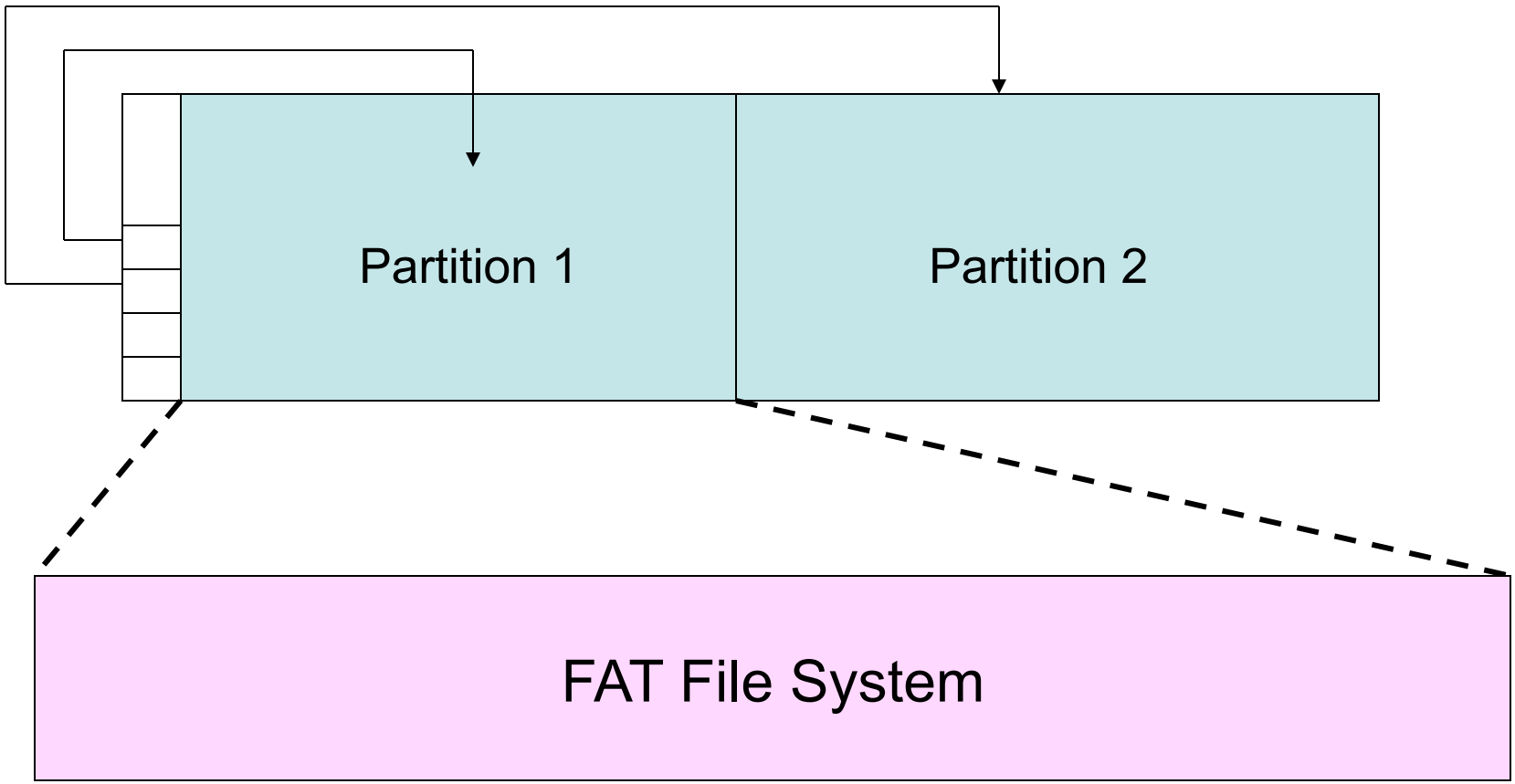
Disk



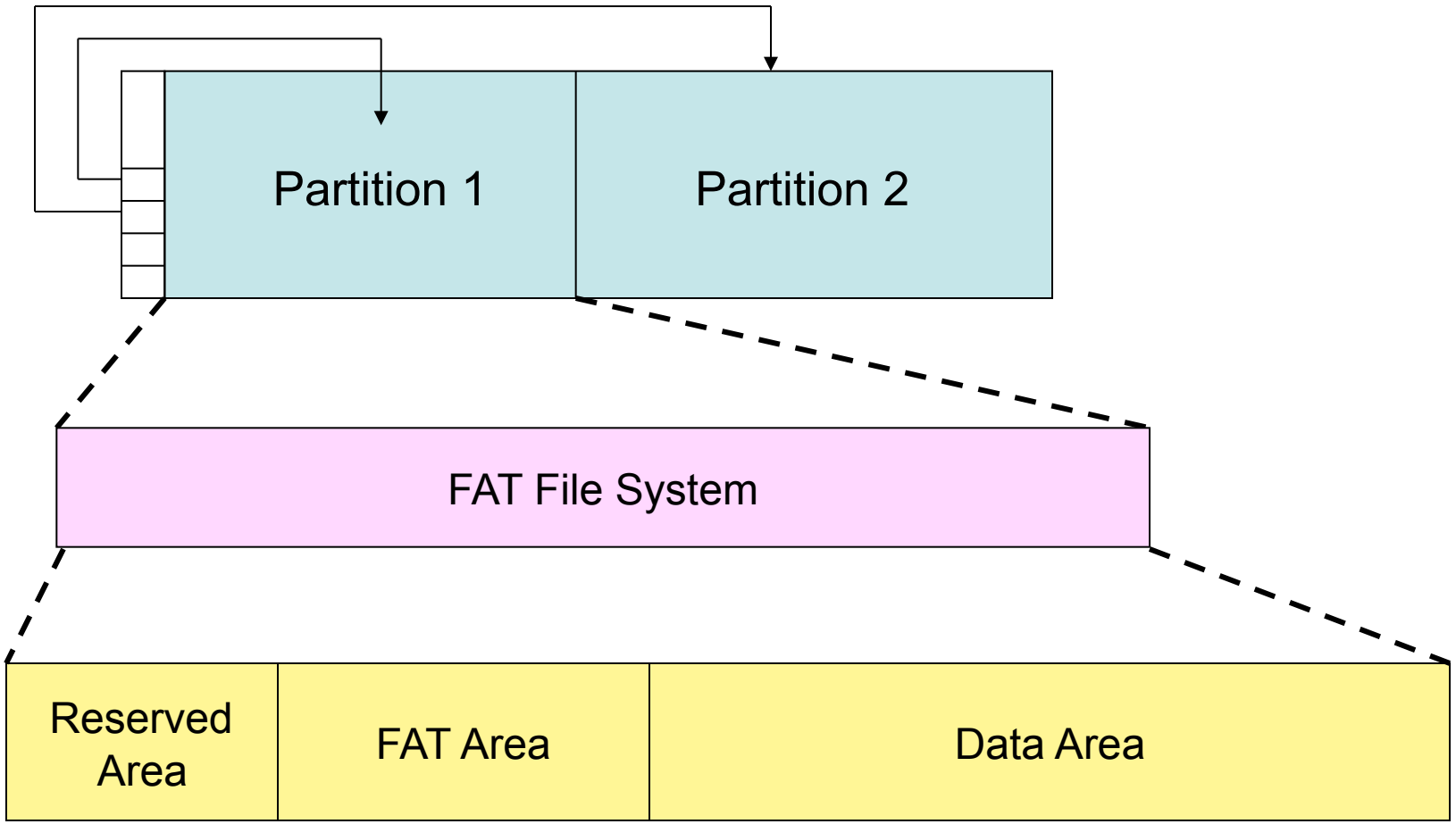
File Systems

- 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 in the Master File Table (MFT) on newer Windows
  - Exact details depend on operating system

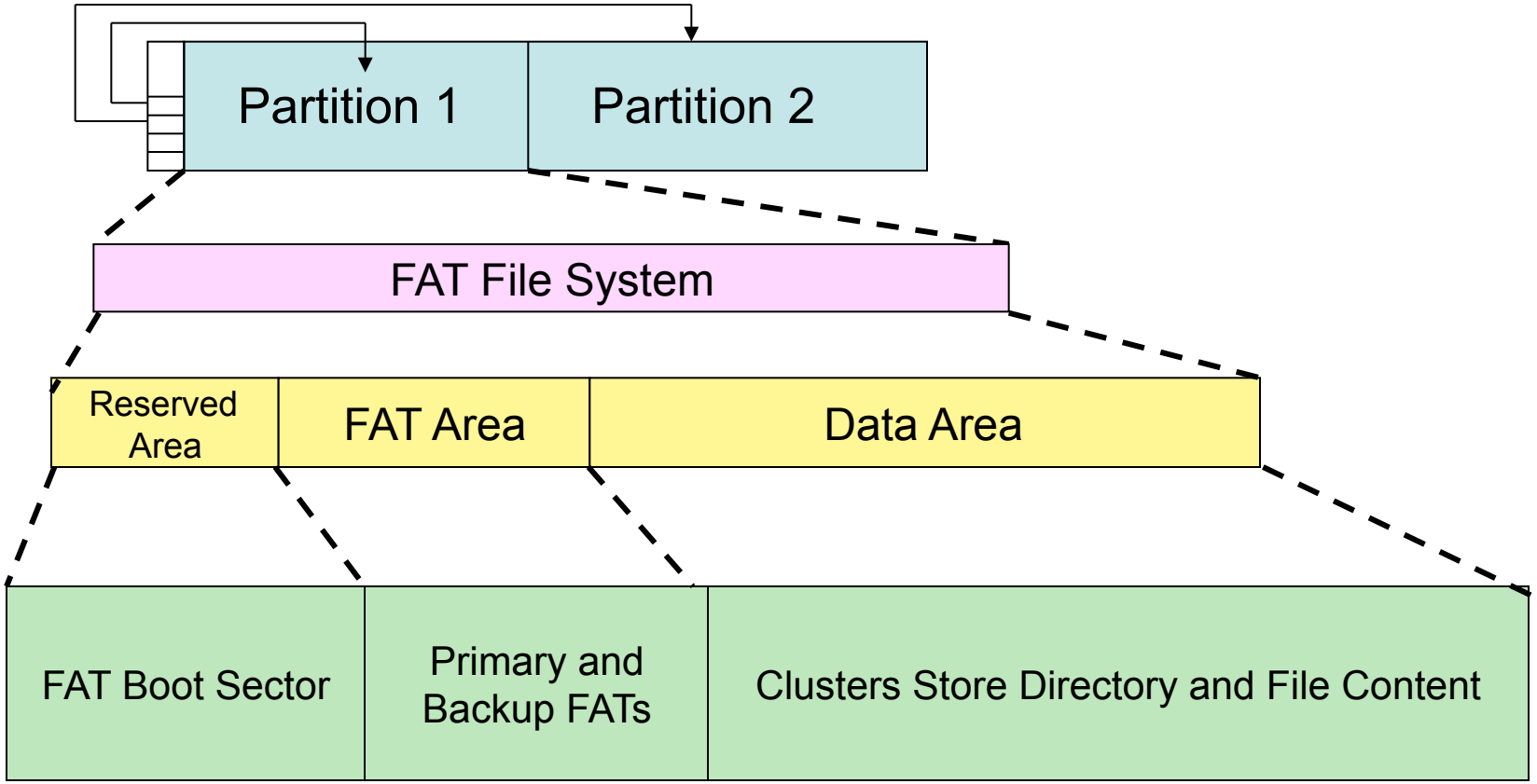
Partition Holds a File System from the FAT Family



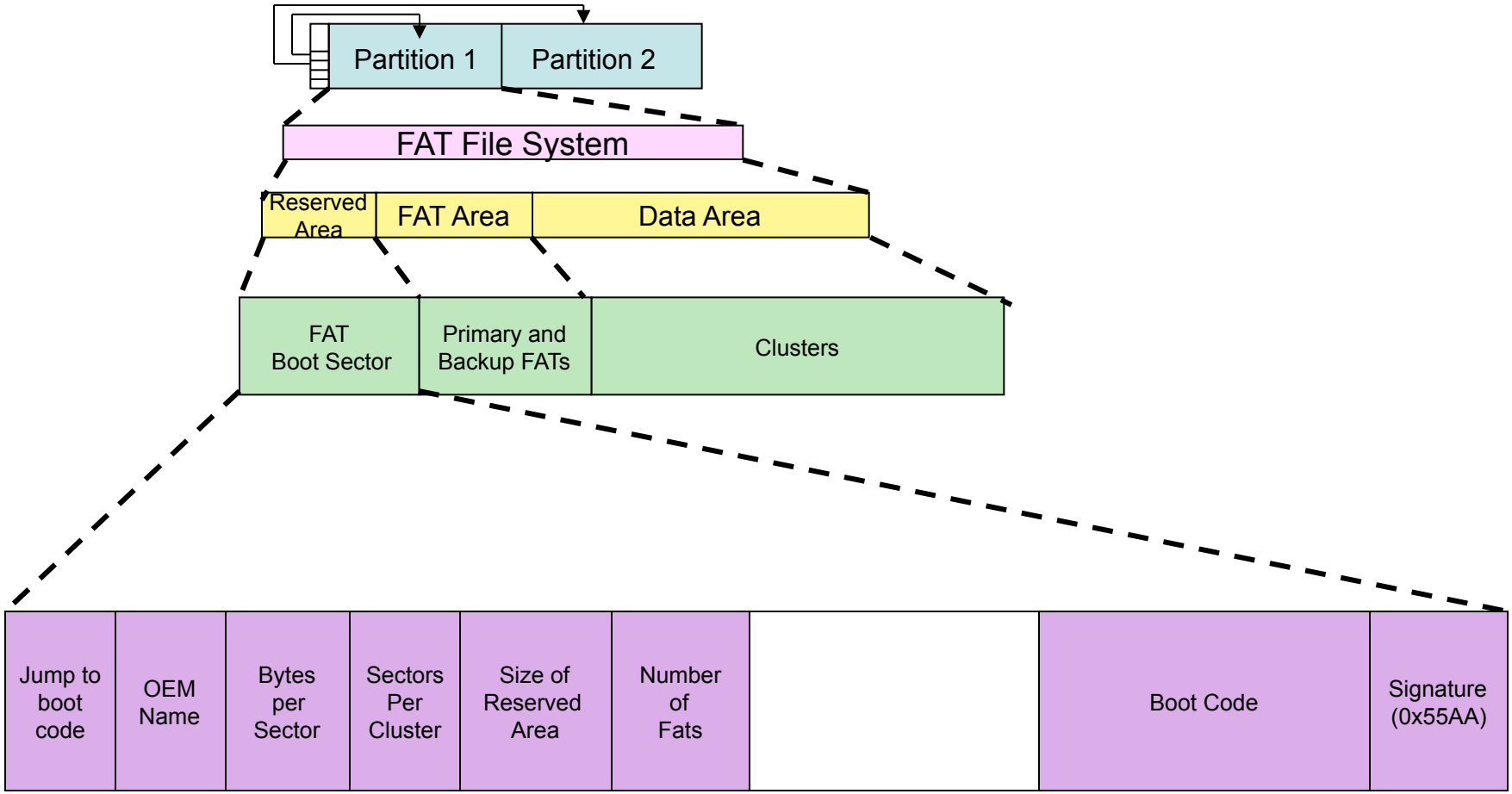
FAT Family File System



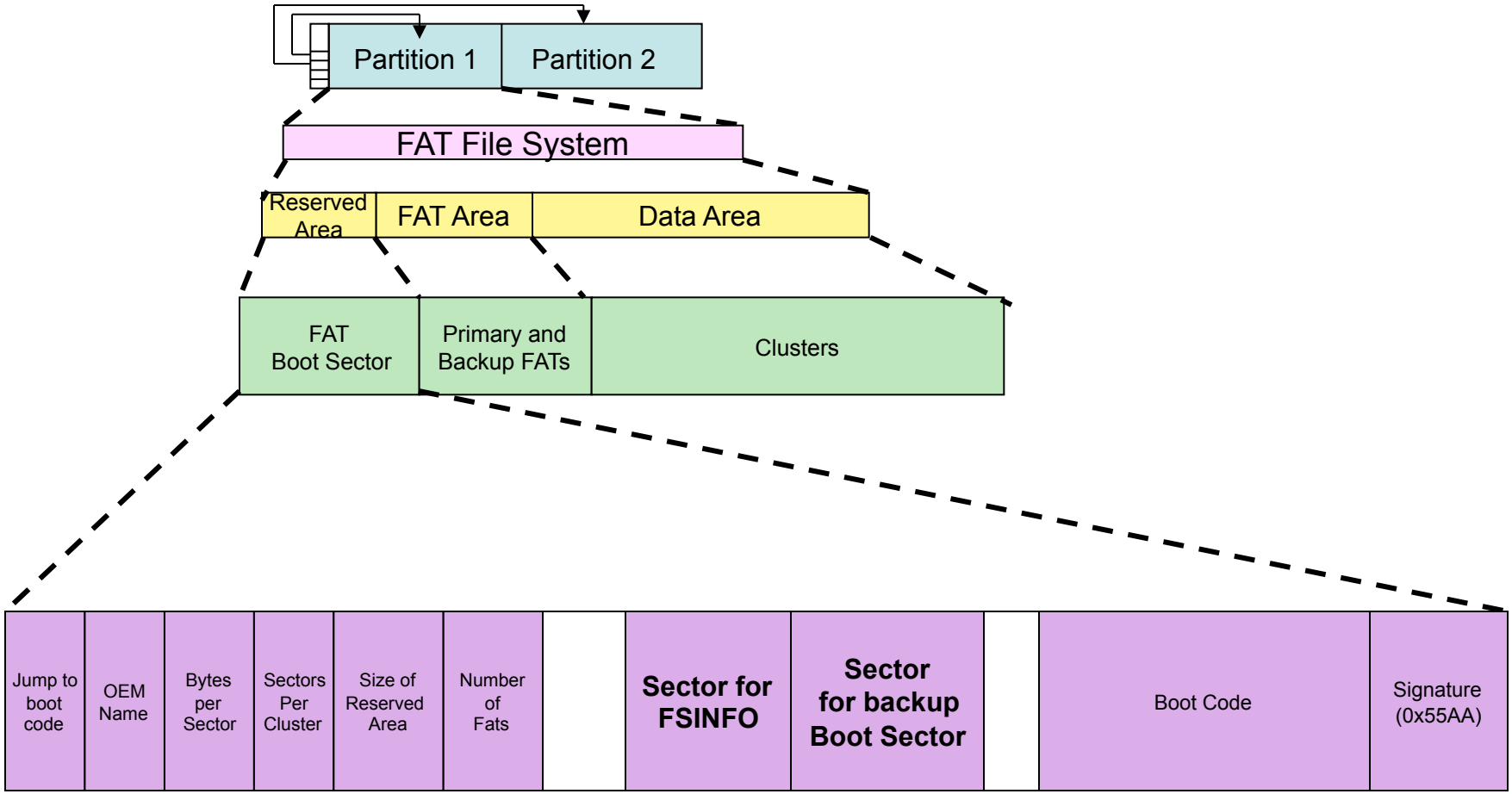
FAT File System Layout



FAT File System Boot Sector



FAT32 Boot Sector



FAT32 FSINFO

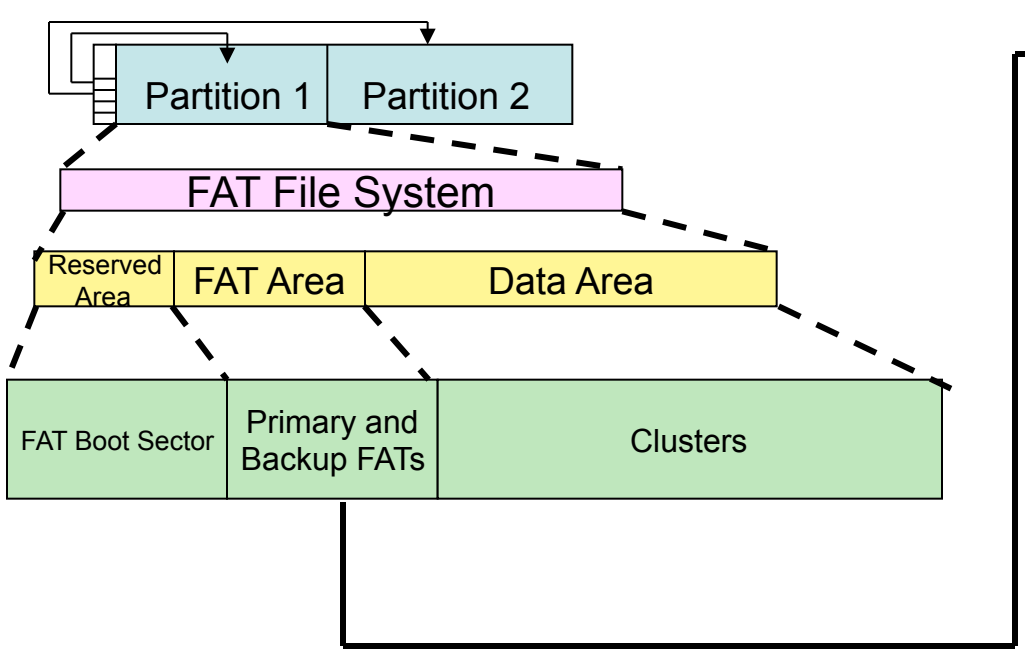
Hints about where the OS can find free clusters

Byte Range	Description
0-3	Signature (0x41615252)
4-483	Not Used
484-487	Signature (0x61417272)
488-491	Number of free clusters
492-495	Next free cluster
496-507	Not Used
508-511	Signature (0x55AA0000)

FAT Entries

- 12, 16, or 32 bits
- First addressable cluster is cluster 2
- In FAT16, non-addressable cluster 0 stores the media type
  - 0xF0 means removable
  - 0xF8 means non-removable
  - Duplicates byte 21 of volume boot record
- In FAT16, non-addressable cluster 1 stores the dirty status of the file system

File Allocation Table Concepts

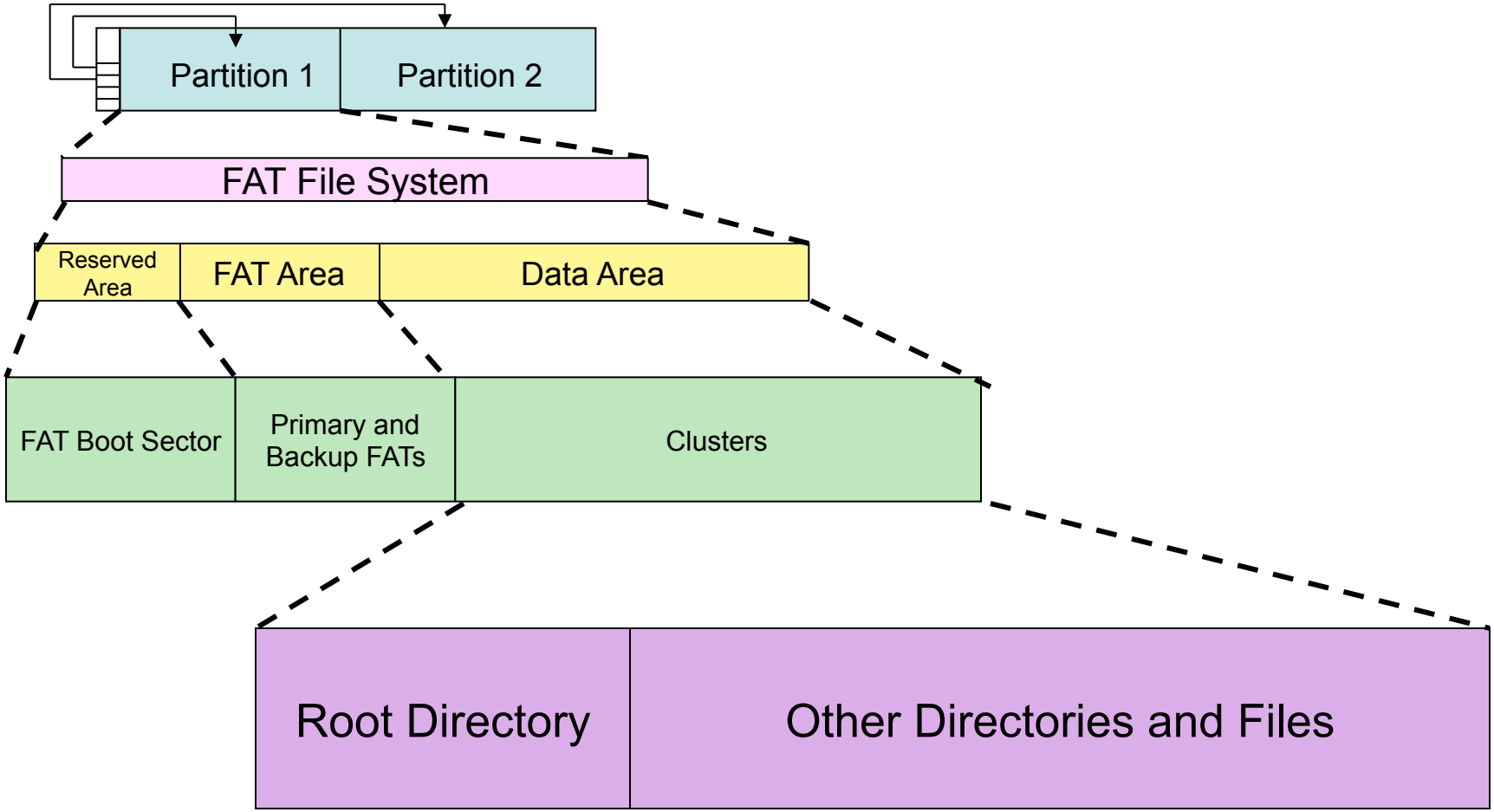


FAT		
Cluster	Entry	
000	…	
001	…	
002	000	not allocated
003	004	next cluster
004	072	next cluster
…	…	
072	FFF	end of file
…	…	

End of File and Bad Cluster

- End-of-file marker
  - Greater than 0xFF8 for FAT12
  - Greater than 0xFFF8 for FAT16
  - Greater than 0xFFFF FFF8 for FAT32
- Bad cluster
  - 0xFF7 for FAT12
  - 0xFFF7 for FAT16
  - 0x FFFF FFF7 for FAT32

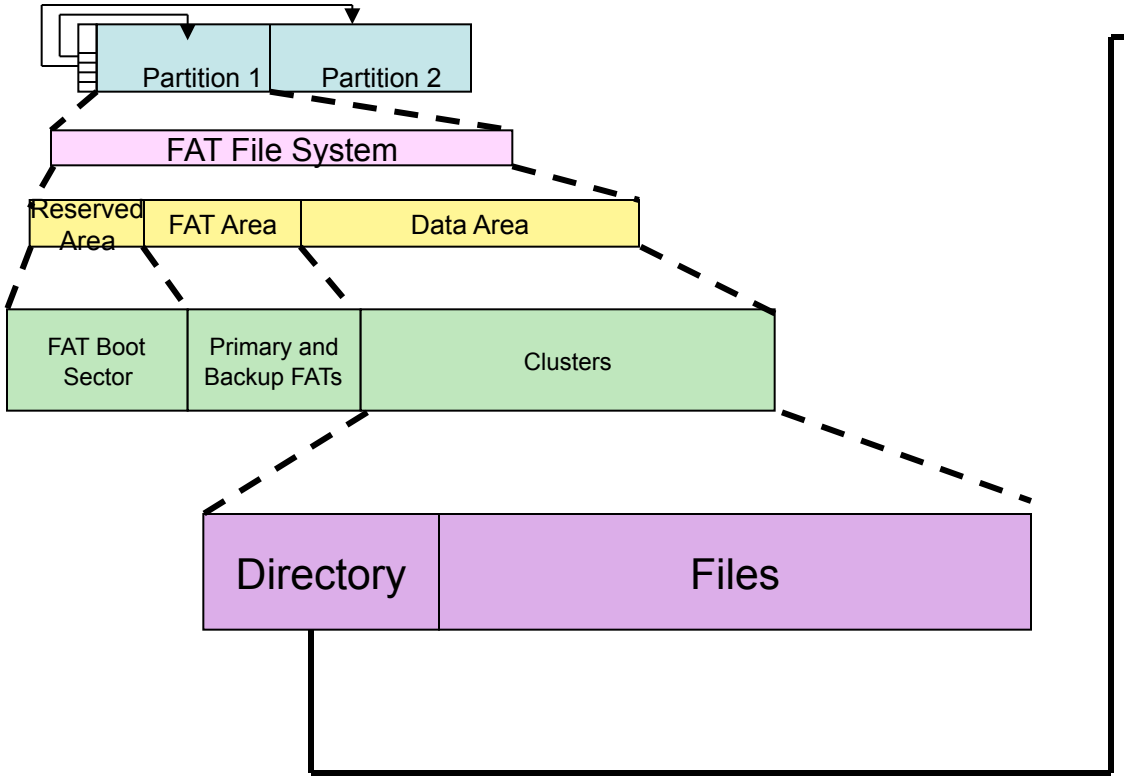
Data Area Concepts



Root Directory

- Fixed length in FAT12/16
  - 32 sectors
  - Each entry is 32 bytes
  - 512 entries total
  - Starts before cluster 2
- Not fixed length in FAT32
  - Starts at cluster 2
  - Each entry is still 32 bytes

FAT Directories



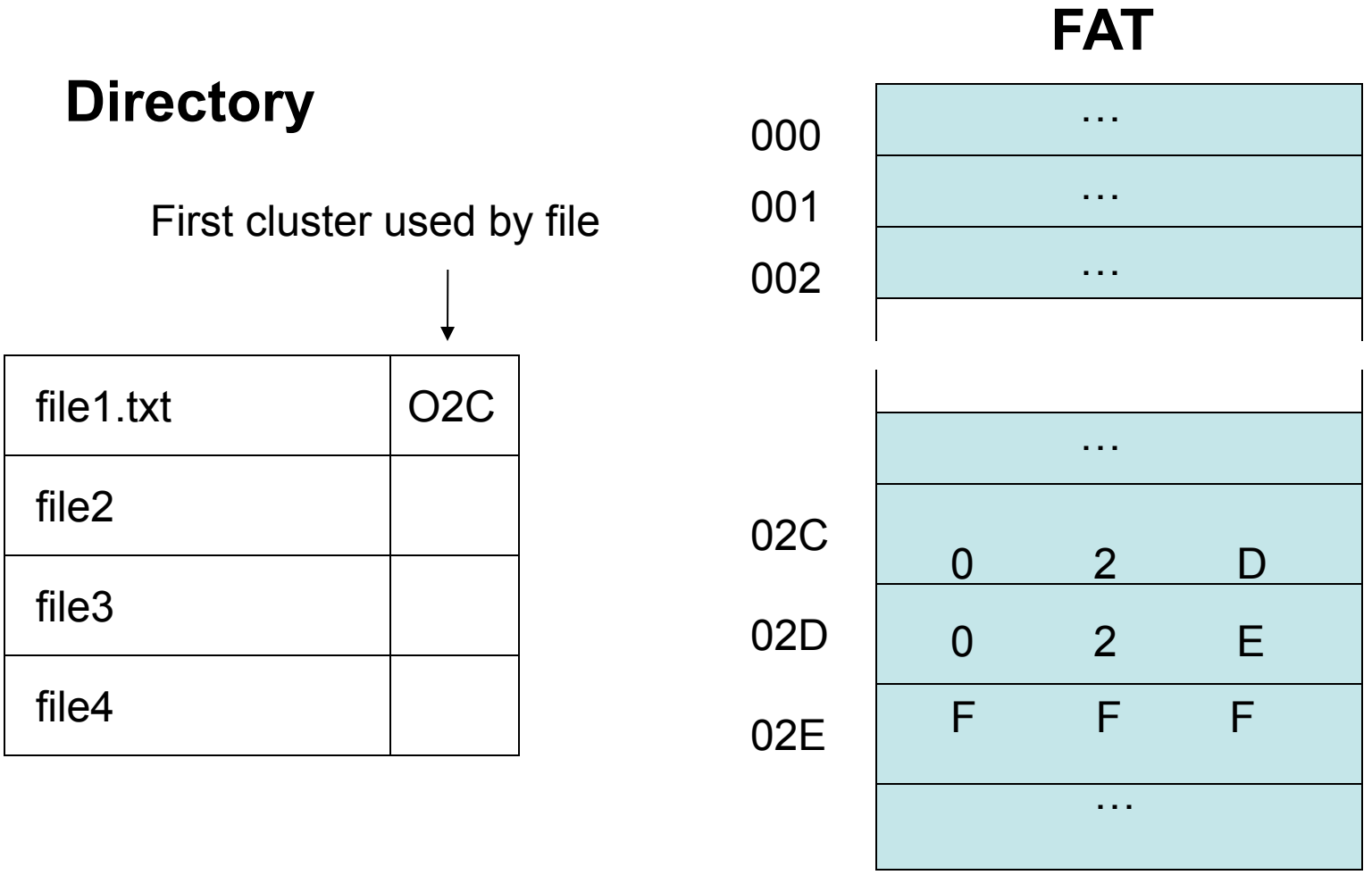
Directory Entry
Long File Name
8.3 Filename
File attributes (read only, hidden, system, long file name, directory, archive, etc.)
Created time/day
Accessed day
Modified time/day
First cluster address
Size of file (0 for directory)

Deleting a FAT File

Deleting dir1\file1.txt

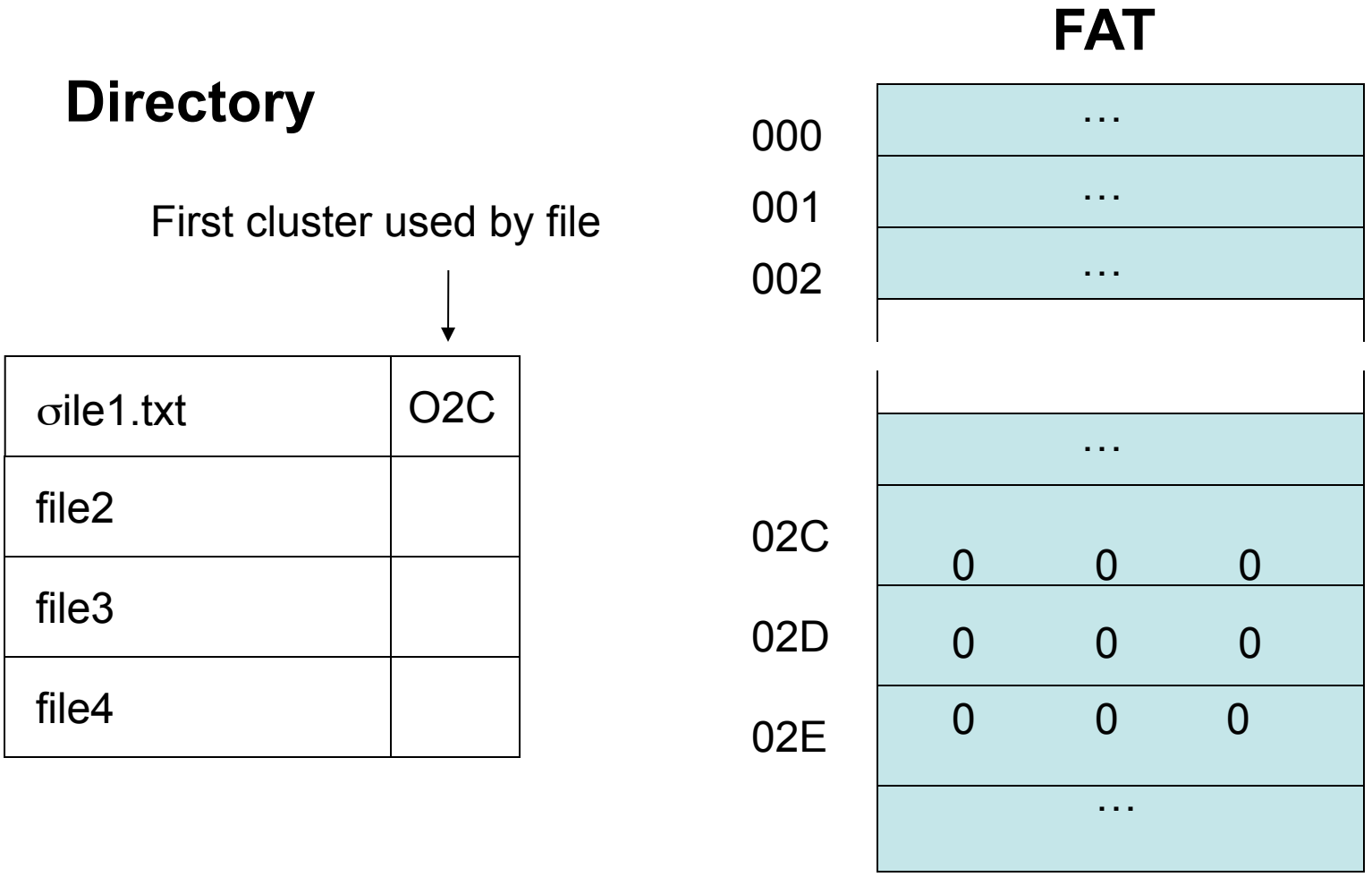
1. Read Fat Boot Sector (sector 0 of the volume) to understand structure and location of Reserved, FAT, and Data areas
2. Locate dir1 in Root Directory; determine its starting cluster
3. Go to dir1 cluster; determine starting cluster for file1.txt
4. Set FAT entries for file1.txt to 0
5. Change filename to σile1.txt in dir1 directory
   – First character becomes 0xE5

Directory and FAT



Directory and FAT

Deleted file



Recovering Files

- Easy if file isn't fragmented and clusters haven't been reallocated!
  - Go to directory entry
  - Change the first character of the file name from 0xE5 to original (or guess if original can't be derived)
  - Go to FAT for first cluster
  - Get that cluster and the next consecutive clusters (depending on size of file)

It's Not Perfect

- Potential problems
  - Fragmented files
  - Clusters that have been overridden
  - Missing directories or directory entries
    - Although the dot and dot dot entries may help
- Best bet will be when fragmentation is minimal and the deletion was recent
  - Usually errors in recovery are obvious
  - Partial recovery is better than nothing!

New Technologies File System (NTFS)

NTFS

- Default file system for Windows NT, 2000, XP, and Windows Server 2003
- No published spec from Microsoft that describes the on-disk layout
- Windows 2000 and XP use a newer form of NTFS called NTFS5.
- 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

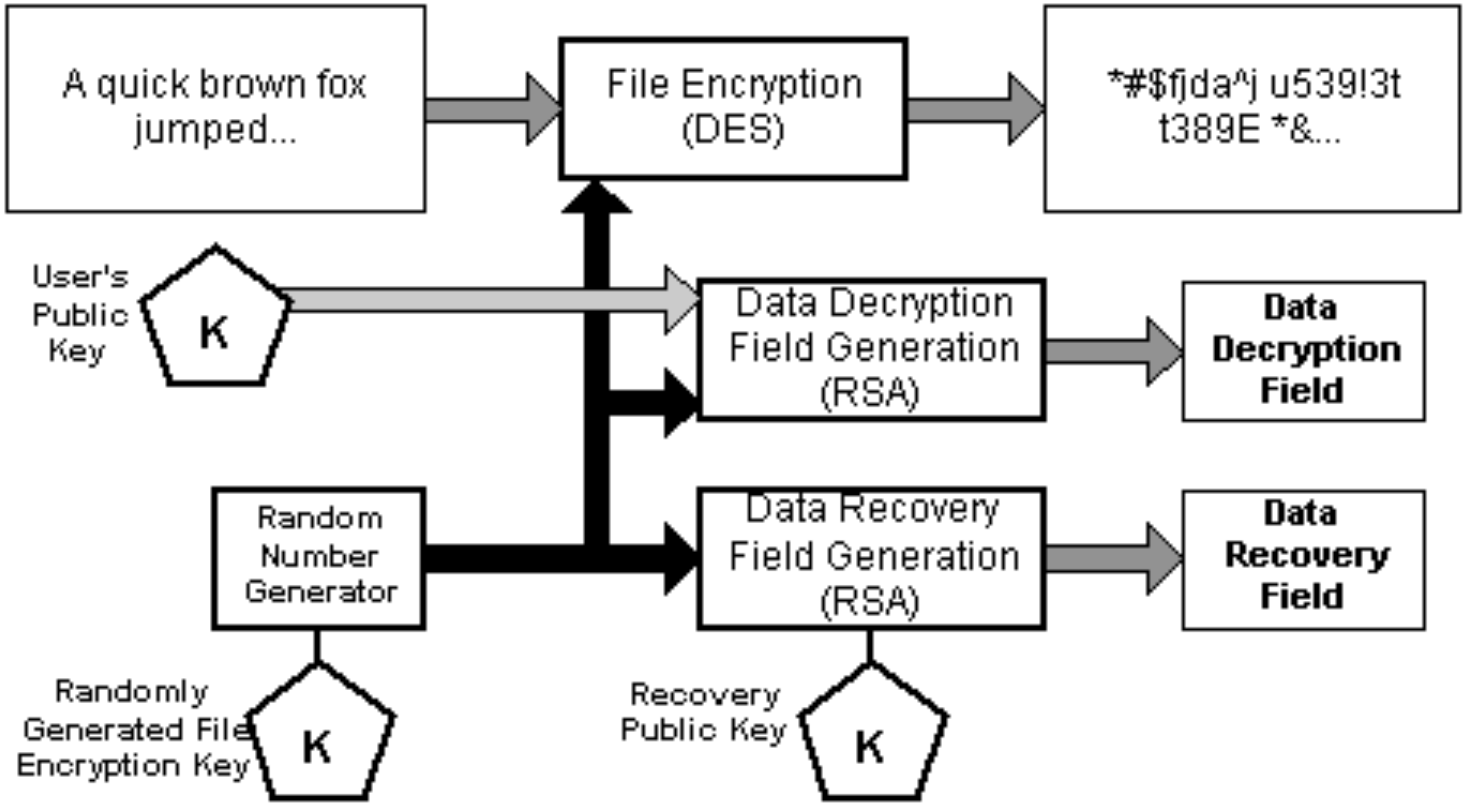
Encrypting File System (EFS)

- 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
- Encrypts FEK with RSA
- 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

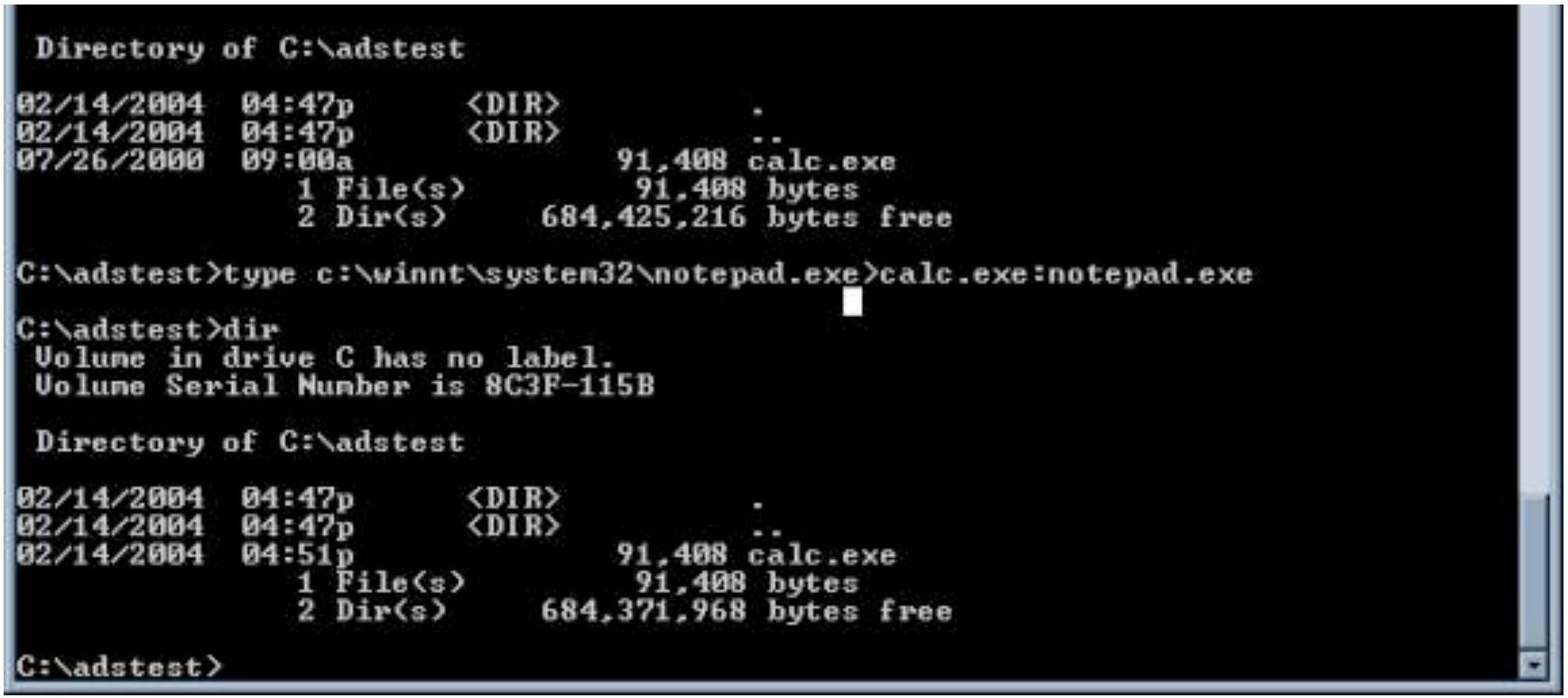
Alternate Data Streams

- 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

Creating an ADS

- To create an ADS named foo to go with the file.txt file, use the following DOS command
  - echo "Hello There" > file.txt:foo

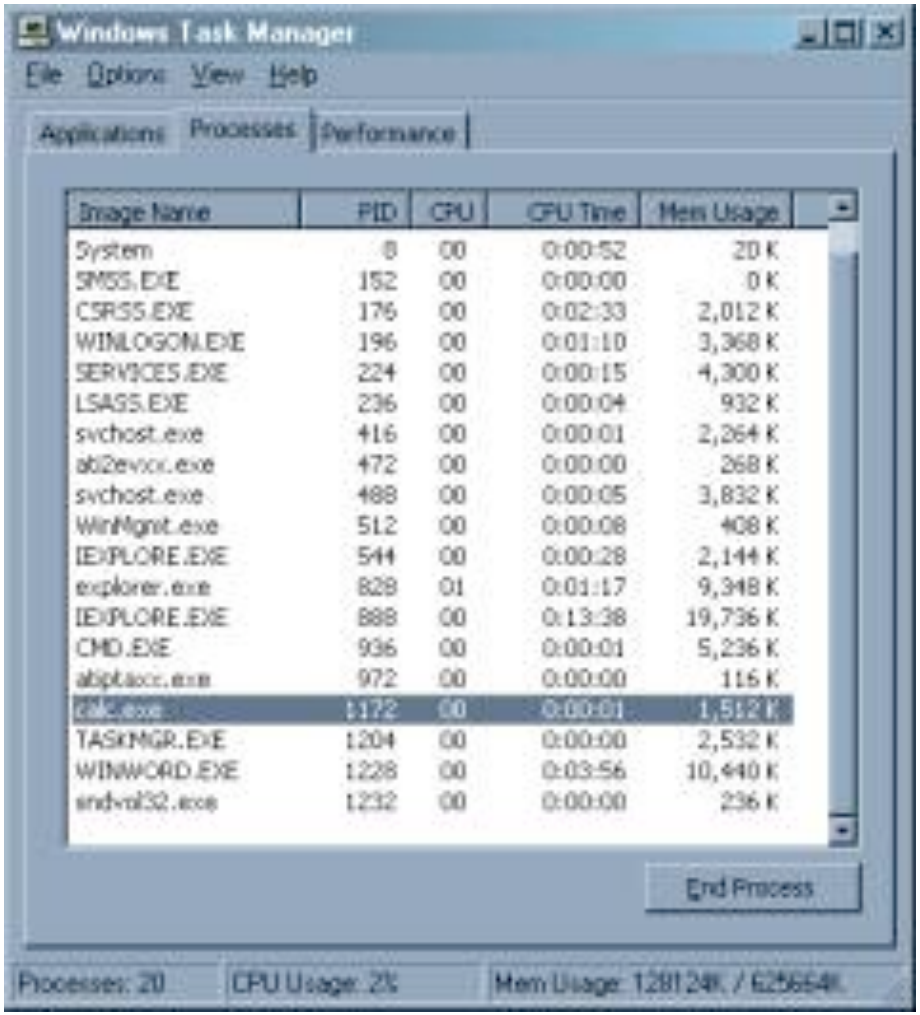
Another ADS Example



Source: [WindowSecurity.com]

Start the Program

What Program Is Running?



NTFS Basic Concepts

- Everything is a file
- Files have attributes
  - $SOME_UPPER_CASE_THING
    - $FILE_NAME
    - $STANDARD_INFORMATION
      - Creation, altered, accessed times; flags (read only, hidden, system, archive, etc.)
    - $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

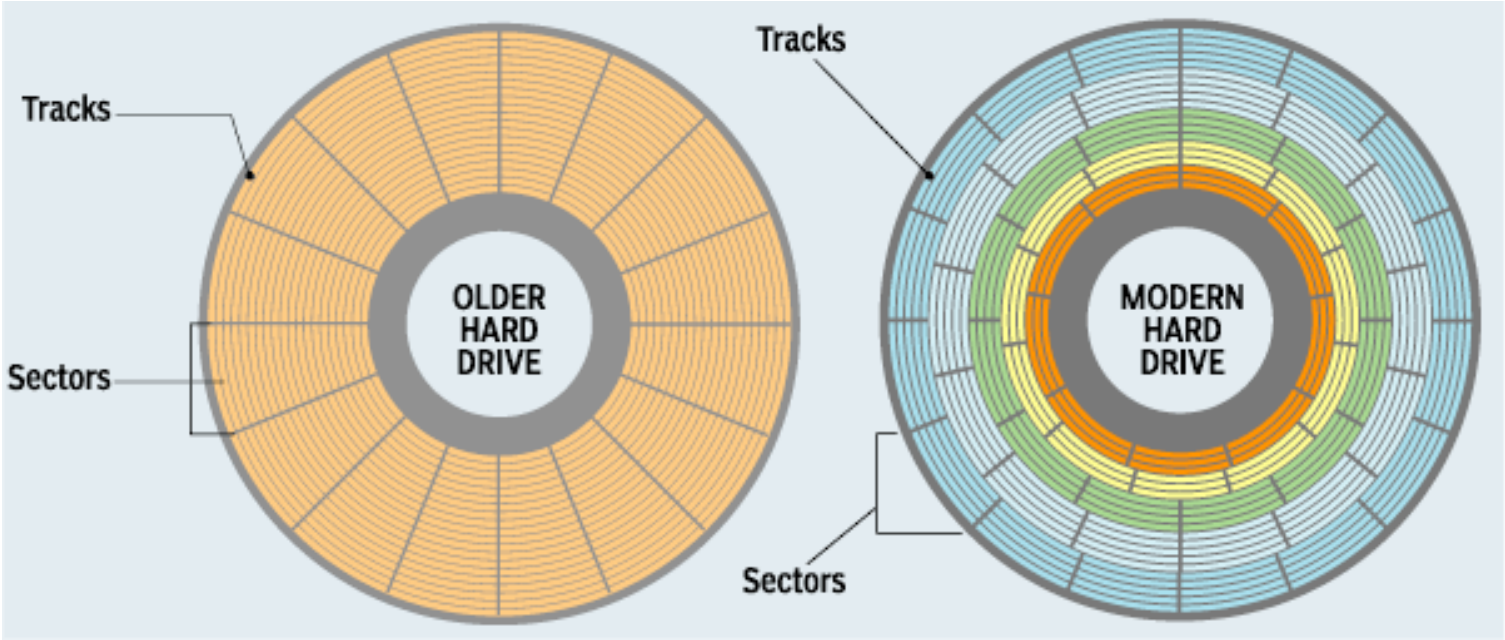
- A resident attribute stores its content in the MFT entry
- A non-resident attribute stores its content in external clusters
- Non resident attributes are stored in cluster runs
- The attribute header gives the starting cluster address and its run length

Non-Resident Attributes

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

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 $2^1=2$, $2^2=4$, $2^3=8$, $2^4=16$

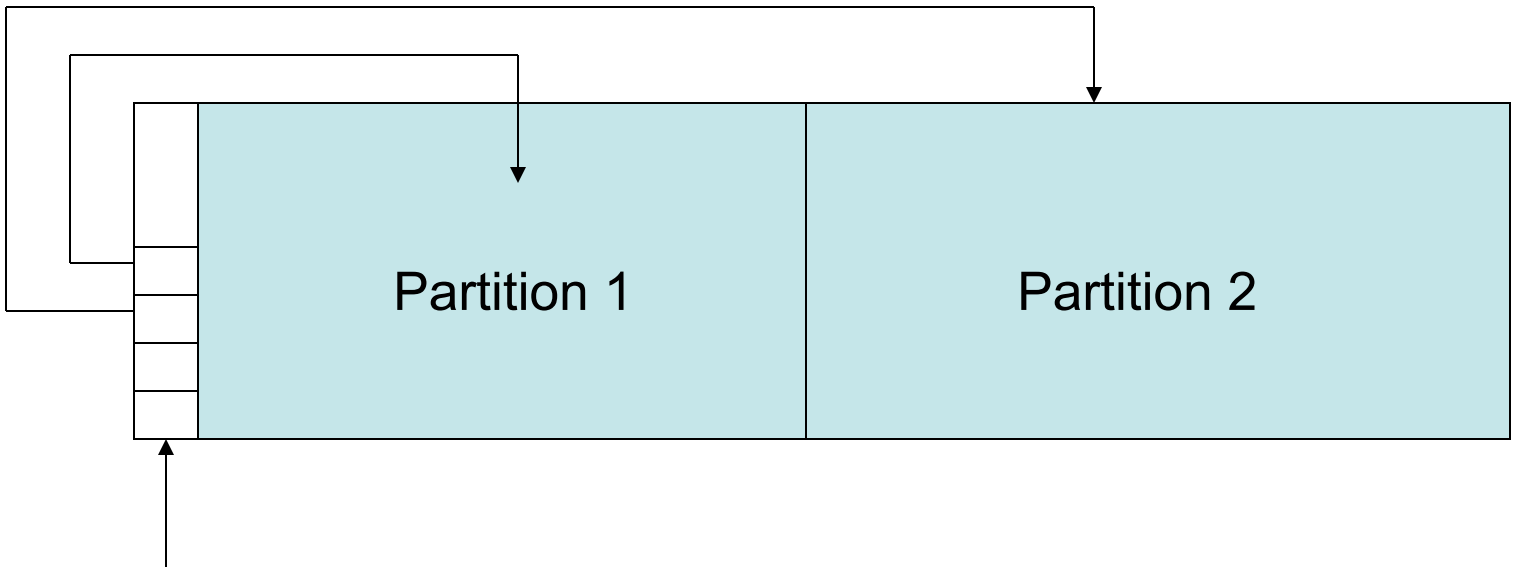
Partitions

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

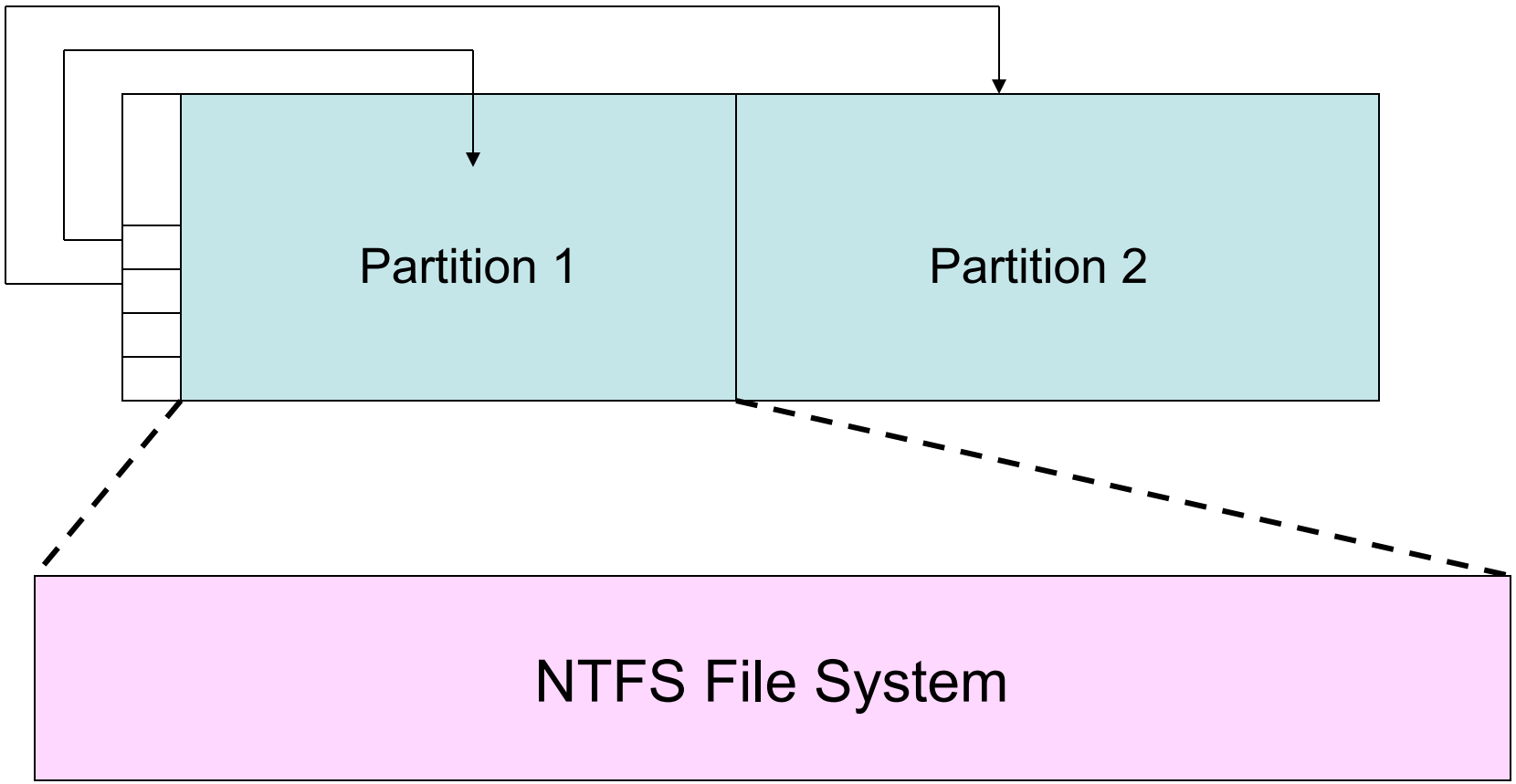
File Systems

- 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

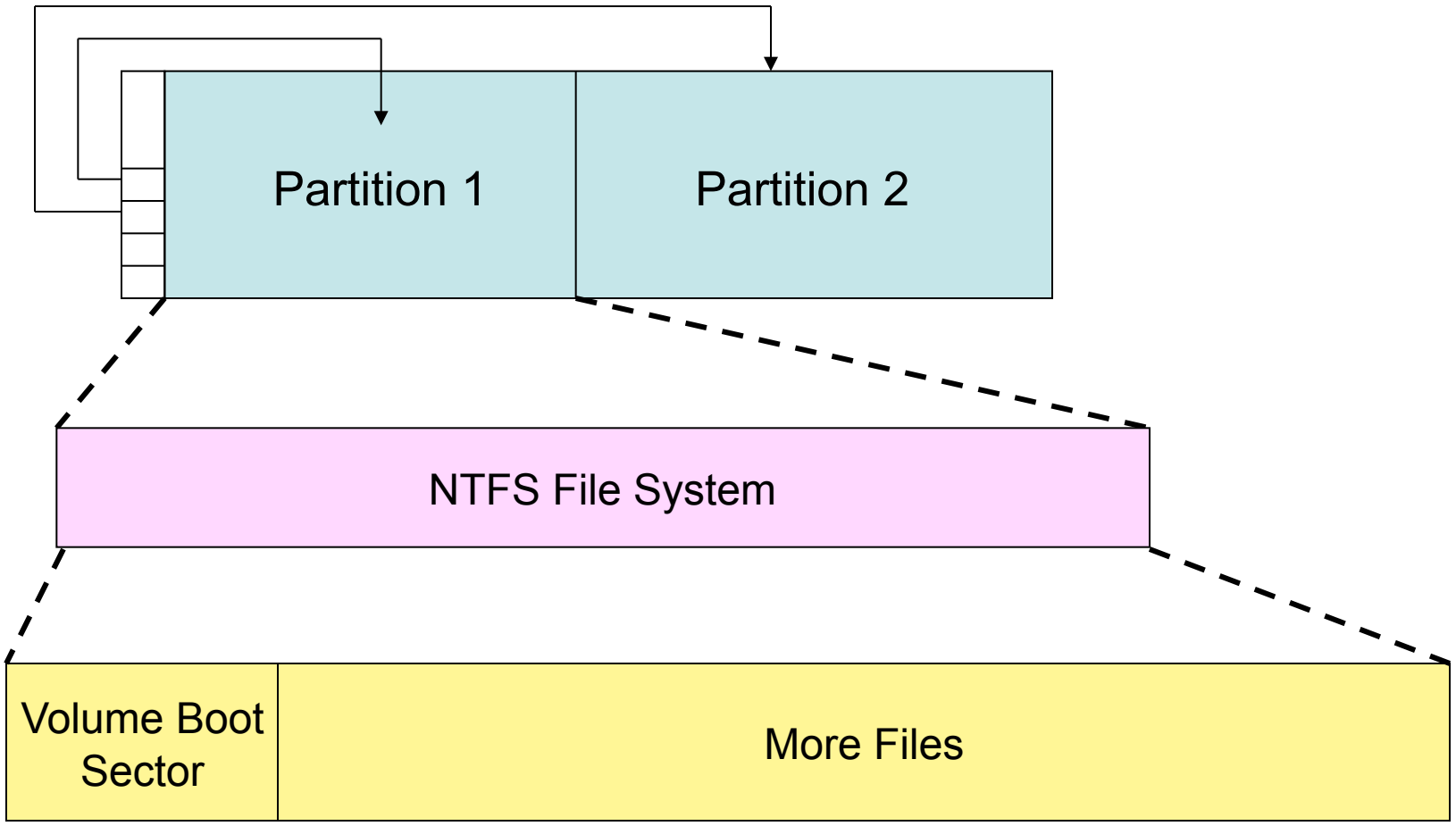
DOS Disk Review



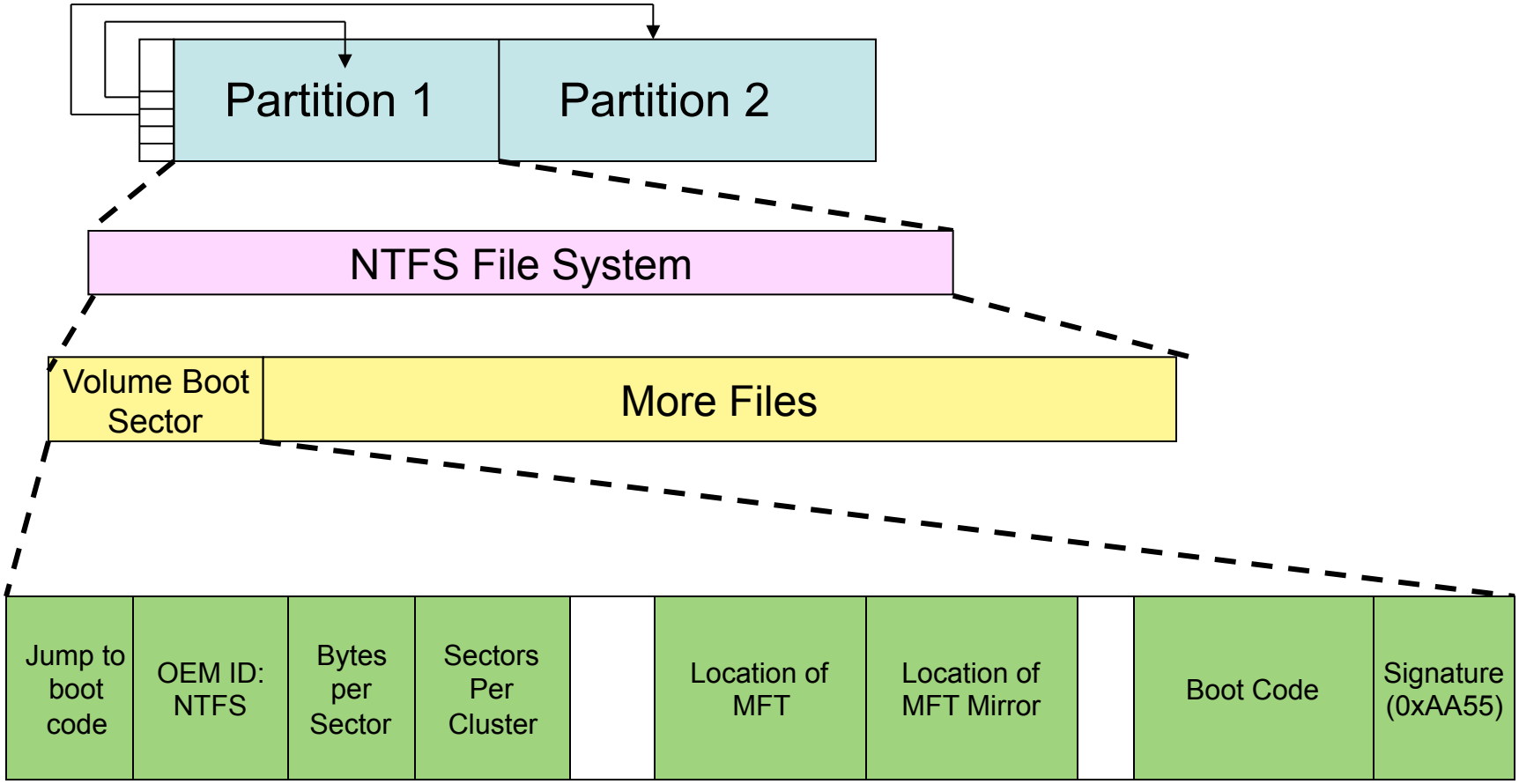
Partition Holds an NTFS File System



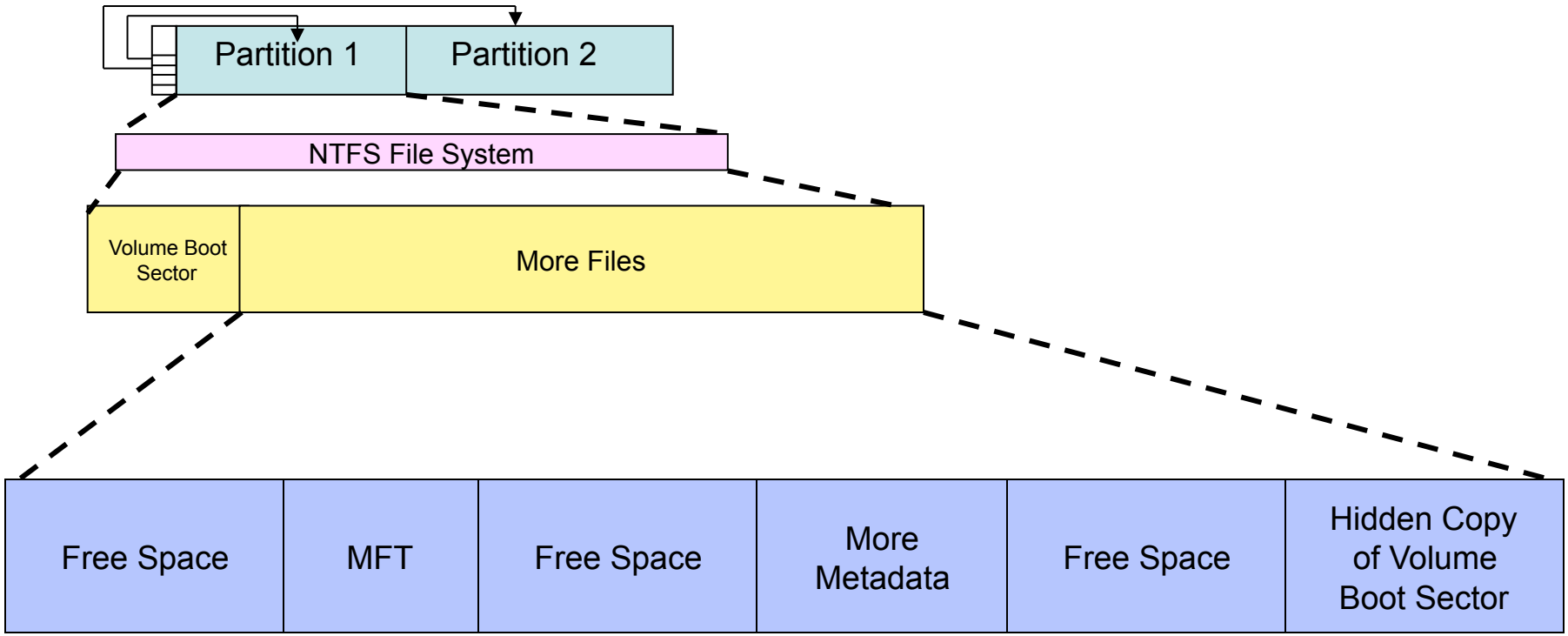
NTFS: Everything Is a File



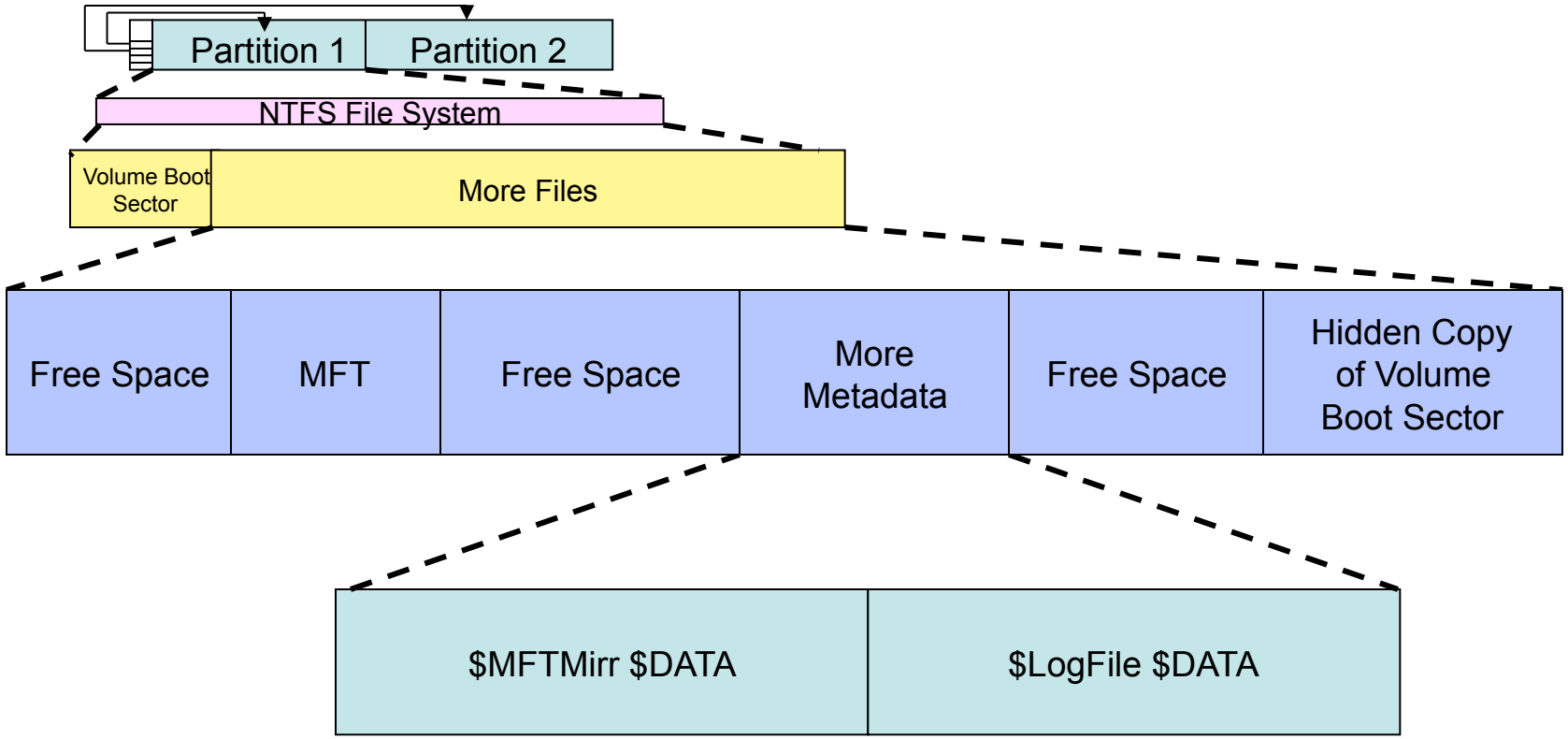
NTFS Volume Boot Sector



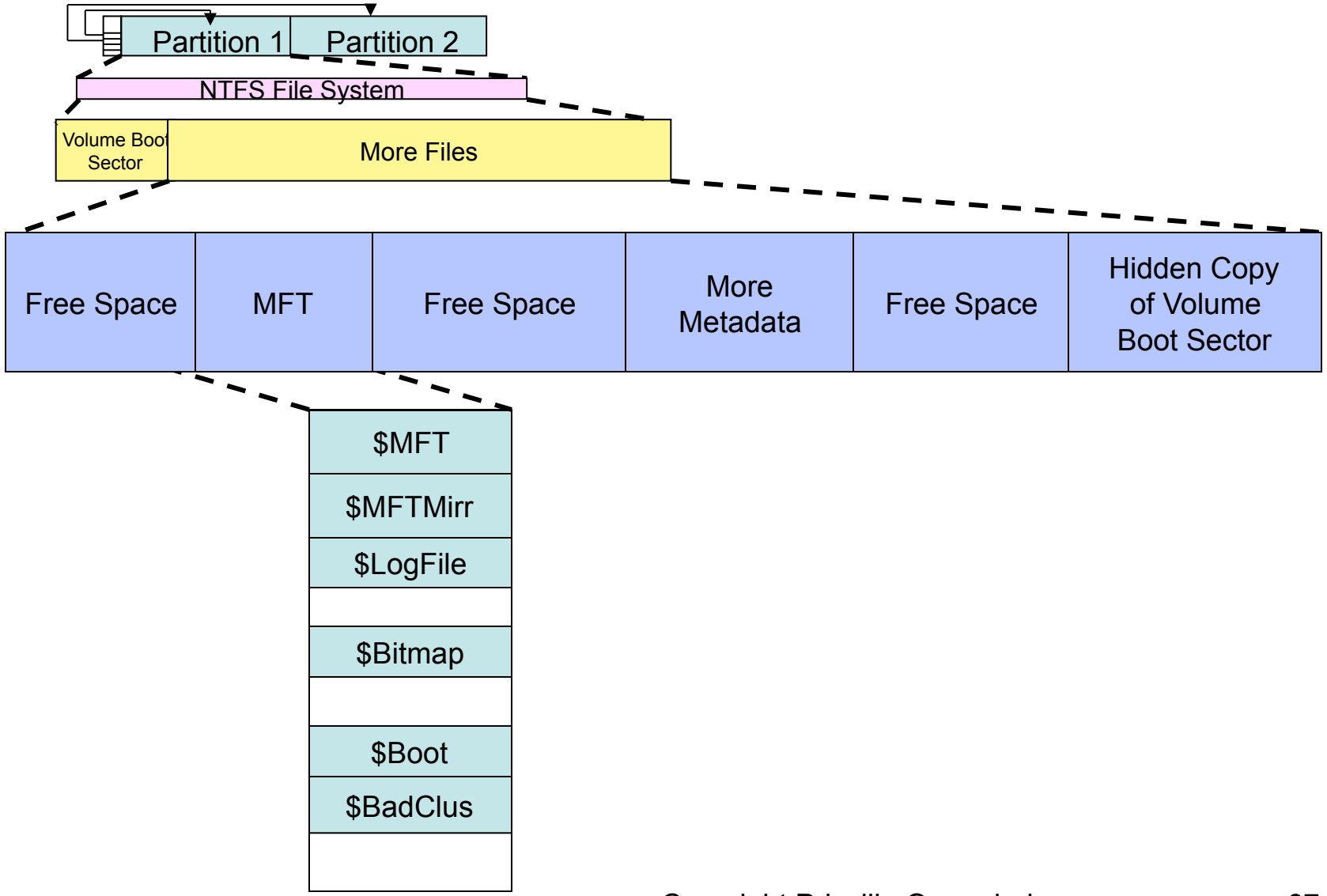
A Freshly Formatted NTFS Volume



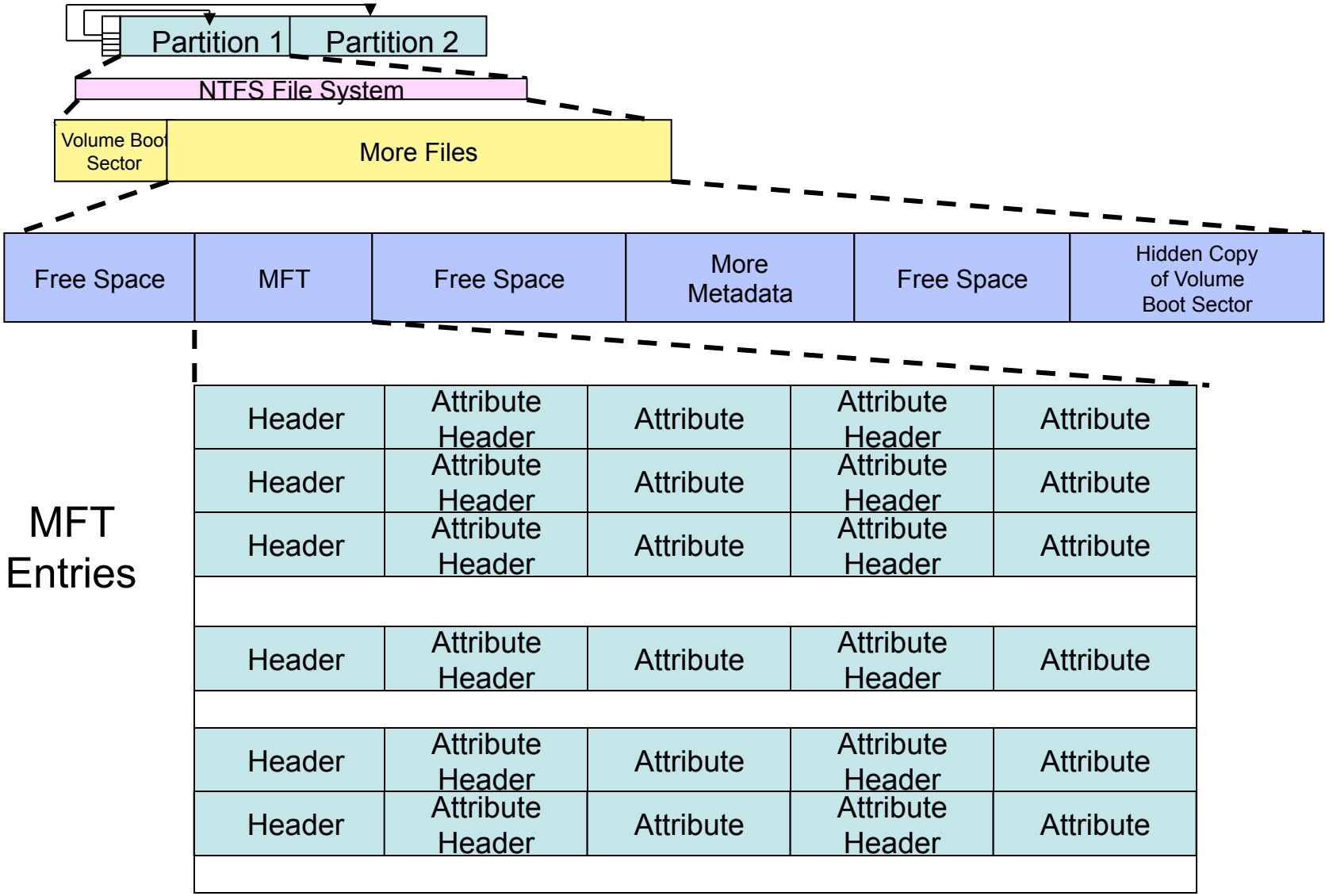
Metadata in Center of Volume



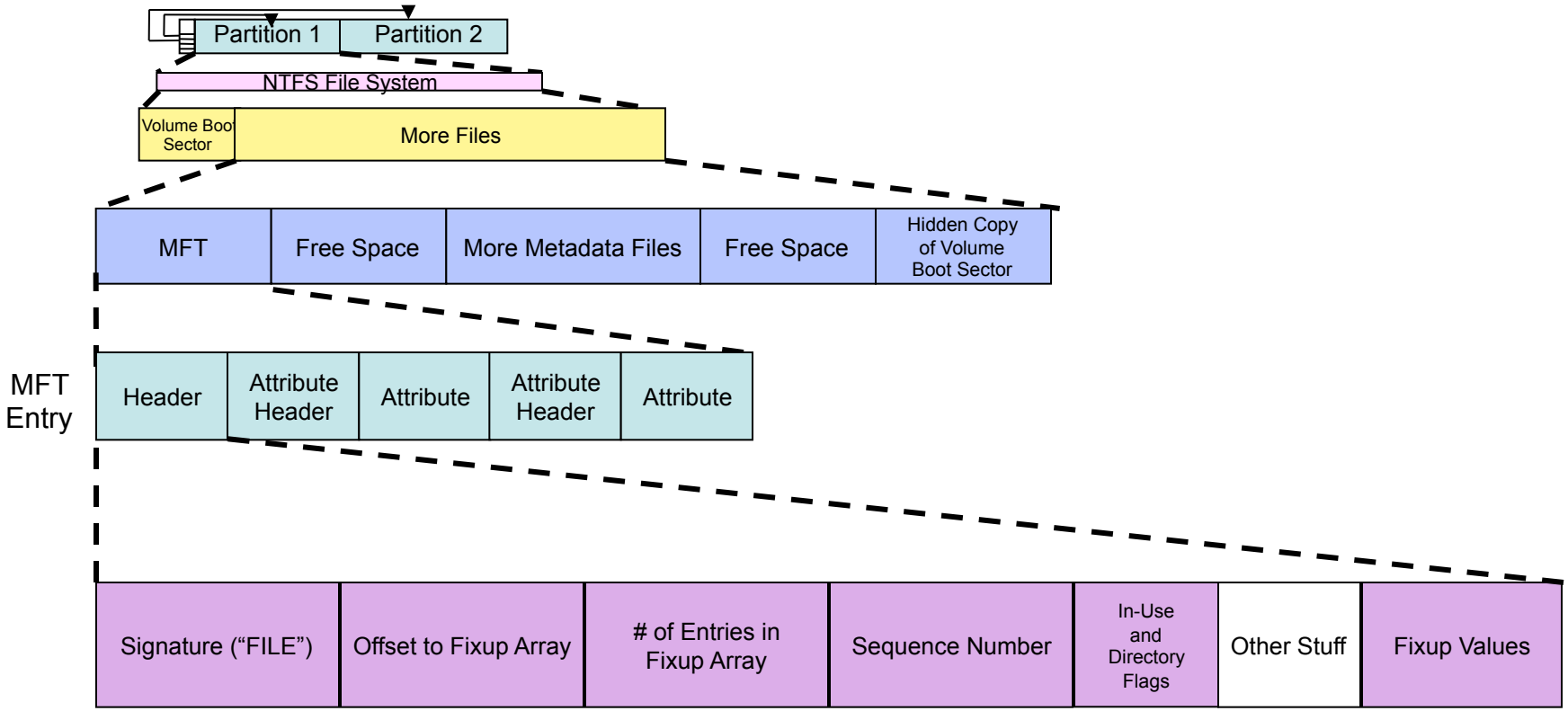
MFT



MFT Attributes



MFT Entry Header

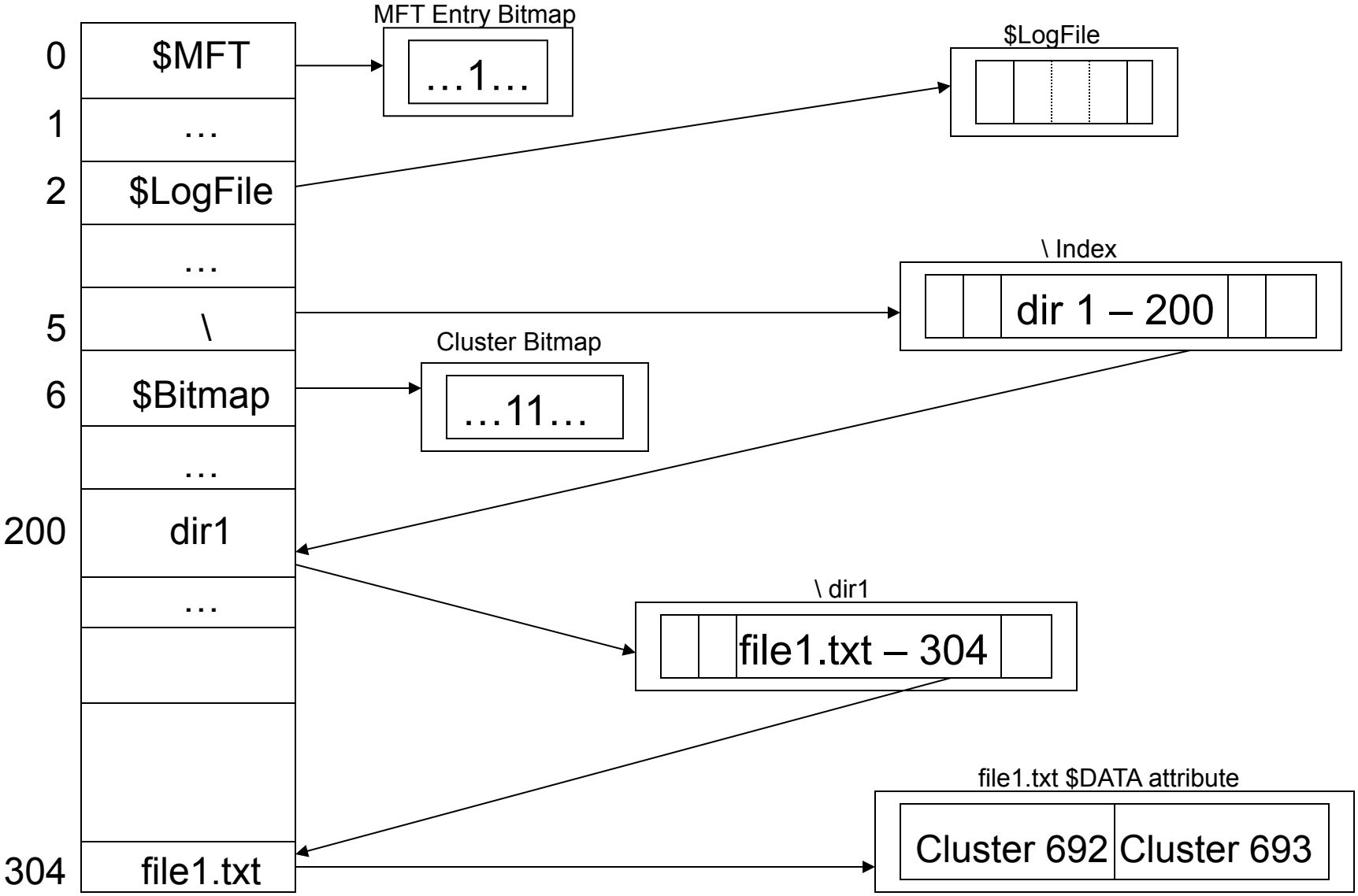


Creating an NTFS File

Creating dir1\file1.txt

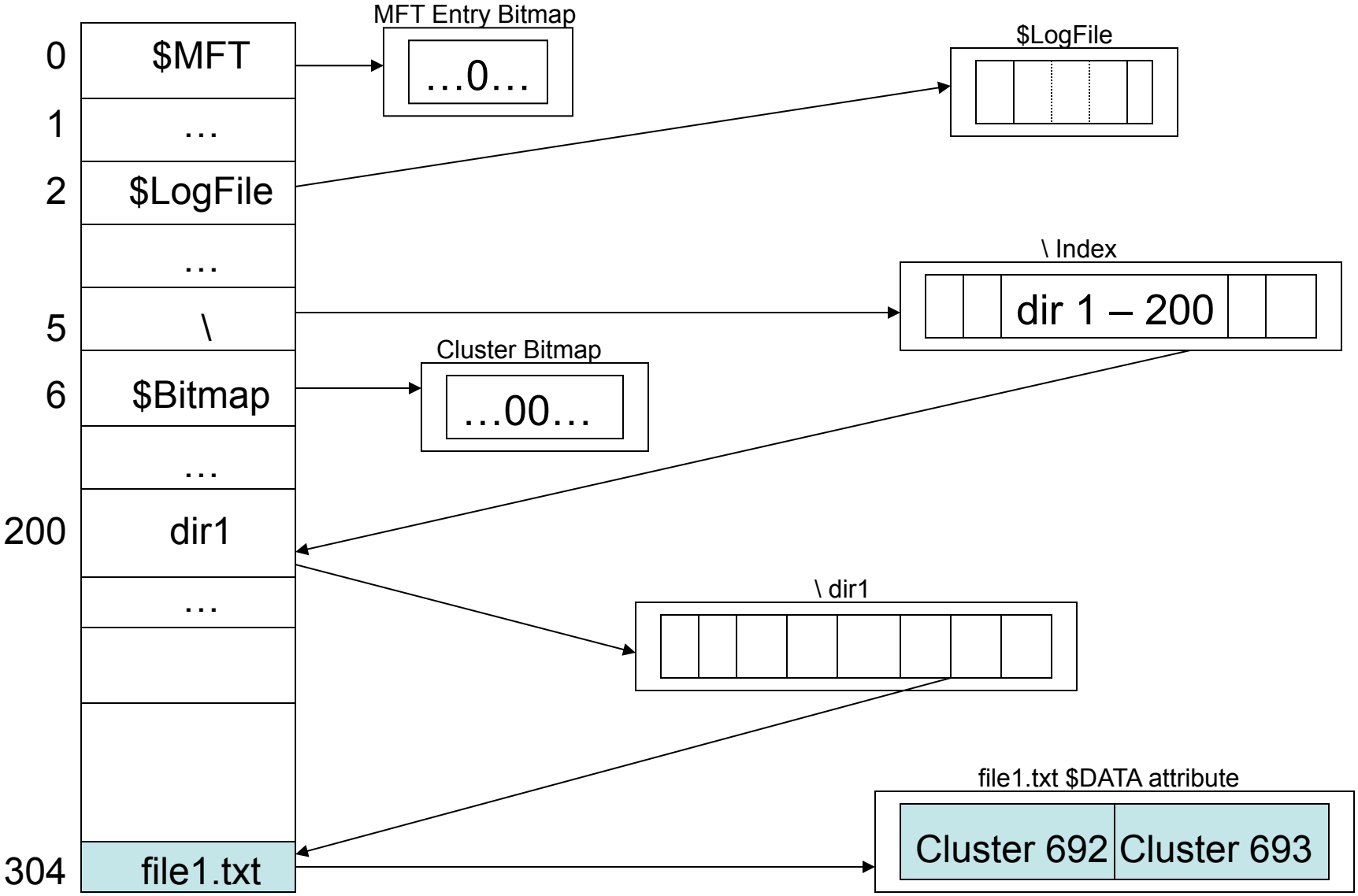
1. Read volume boot sector to locate MFT.
2. Read first entry in MFT to determine layout of MFT.
3. Allocate an MFT entry for the new file.
4. Initialize MFT entry with $STANDARD_INFORMATION, In-Use Flag, etc.
5. Check MFT $Bitmap to find free clusters, using best-fit algorithm.
6. Set corresponding $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


MFT Entry Bitmap
$LogFile
0
$MFT
…1…
1
…
2
$LogFile
…
\ Index
5
\
dir 1 – 200
Cluster Bitmap
6
$Bitmap
…11…
…
200
dir1
…
\ dir1
file1.txt – 304
file1.txt $DATA attribute
304
file1.txt
Cluster 692
Cluster 693

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. Unallocate MFT entry and clean In-Use Flag.
7. Set MFT $Bitmap entries to 0.
8. Enter steps in $LogFile (as each step is taken).

Summary

- NTFS is more complicated than FAT but also has more scalability, reliability, and security features
- Forensics analysis and recovery of files is possible especially if $MFT or $MFTMirr are in good shape
- Recovery challenges include compression and encryption