EnCase Seminar #1
(Hardware and File System)

FORENSIC-PROOF.COM

PRONEER

Welcome to EnCase Seminar!!

Security is a people problem….

Introduction

Goals

- ✓ 디지털포렌식에 대한 이해 (하드웨어, 파일시스템, 사고 대응)
- ✓ 능동적인 디지털포렌식 증거 획득 기술 습득
- ✓ 디지털포렌식 증거 분석 기술 습득
- ✓ EnCase를 통한 전문적인 분석 기술 습득

Reference

- ✓ **EnCE – The Official EnCase Certified Examiner STUDY GUIDE (second edition)**

Introduction

Outline

- ✓ **Week 1 : Hardware and File system Analysis (Chapter 1, 2)**
- ✓ Week 2 : Acquiring Digital Evidence (Chapter 4)
- ✓ Week 3 : EnCase Concepts and Environment (Chapter 5, 6)
- ✓ Week 4 : Actual Test
- ✓ Week 5 : Actual Test
- ✓ Week 6 : Actual Test
- ✓ Week 7 : Actual Test
- ✓ ..
- ✓ PS : EnScripting

Introduction

- Guidance Software (http://guidancesoftware.com)

Introduction

- Certification Programs

Certification Programs

Guidance Software offers certification programs for private and public sector professionals in the use of EnCase® digital investigation software and their proficiency in industry best practices.

Since 2001, Guidance Software has certified over 2,100 computer forensic investigative professionals with the industry standard EnCase® Certified Examiner (EnCE)® designation.

The EnCase® Certified eDiscovery Practitioner (EnCEP™) program likewise enables eDiscovery practitioners to demonstrate their skills, training and experience in the proper handling of electronically stored information for legal purposes.

Certification candidates must meet professional requirements and pass a rigorous testing program to earn an EnCase certification. The certifications are valid for three years, and require continuing education for renewal.

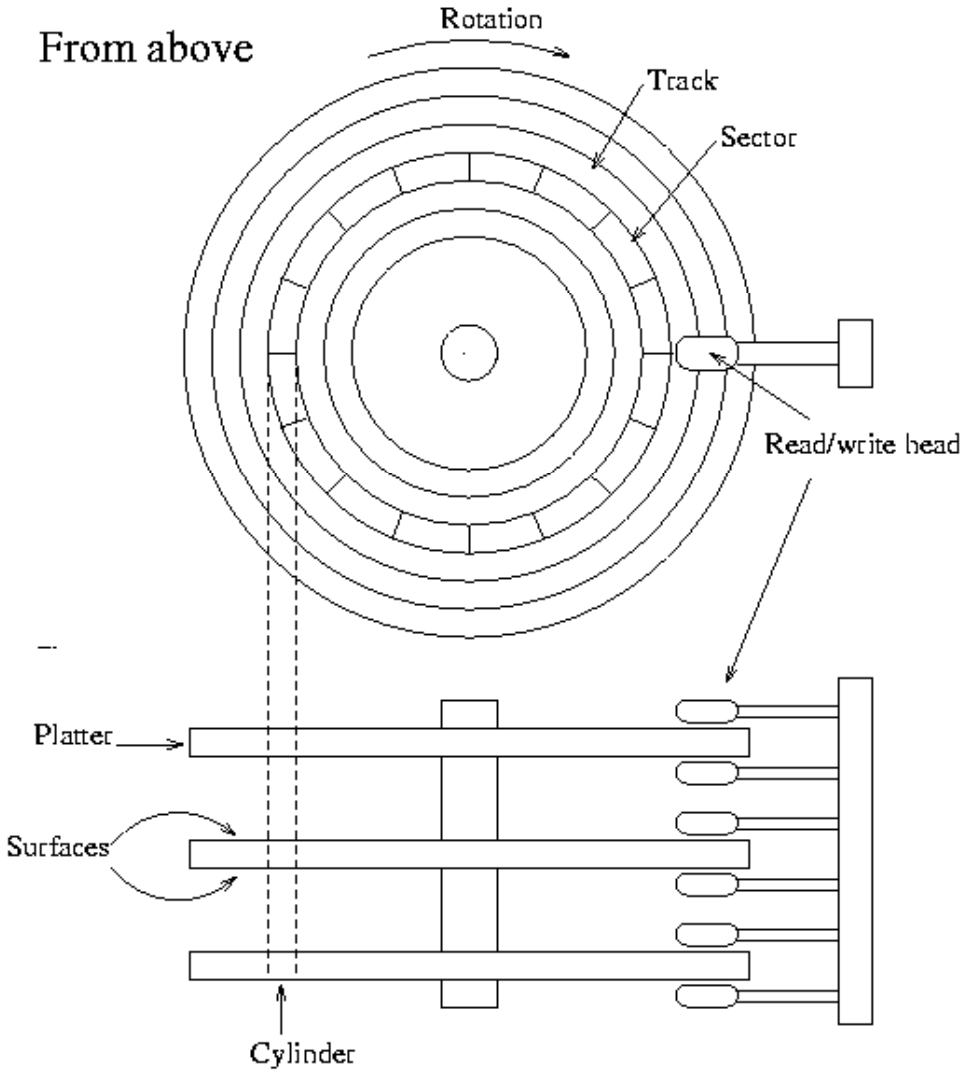
Please select the desired certification for specific program information, frequently asked questions, professional referrals, and forms for certification application or renewal.

File System

- ✓ *MBR*
- ✓ *FAT*
- ✓ *NTFS*

Important factor

- **Addressing**



Important factor

Addressing

✓ CHS (Cylinder, Head, Sector) 방식

- 디스크의 물리적인 구조에 기반한 방식
- 초기 ATA 표준과 BIOS의 지원 비트의 차이로 인해 최대 504MB까지만 지정 가능
- 이후 BIOS 비트 확장으로 8.1GB 까지 지원
- ATA-6부터 표준에서 제외, LBA 방식이 새롭게 대두

Important factor

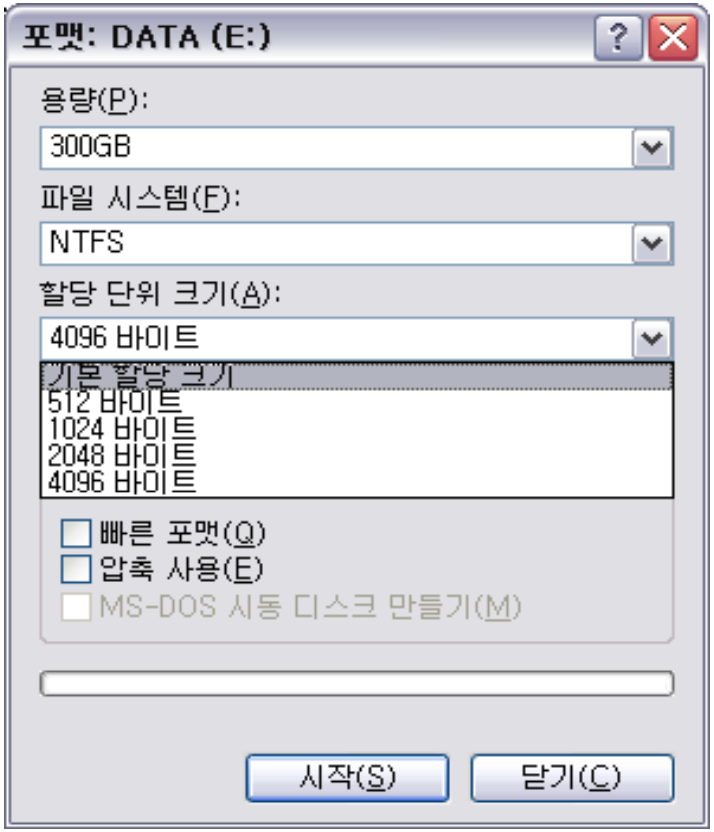
Addressing

✓ LBA (Logical Block Addressing) 방식

- 디스크의 0번 실린더, 0번 헤드, 1번 섹터를 첫 번째 블록으로 지정
- 디스크의 마지막 섹터까지 순차적으로 주소를 지정
- 물리적인 구조를 정보 불필요
- ROM BIOS에 의해 자동적으로 변환

Important factor

- **Cluster**



✓ 디스크 I/O 작업을 줄이기 위한 목적

✓ 4MB 데이터를 저장하는데 4K(1,024번), 512Byte(8,192번)

Important factor

Cluster

FAT32

Volume Size	Cluster Size
32MB – 8GB	4KB
8GB – 16GB	8KB
16GB – 32GB	16KB
32GB -	32KB

NTFS

Volume Size	Cluster Size
- 512MB	512 Byte
512MB – 1GB	1KB
1GB – 2GB	2KB
2GB -	4KB

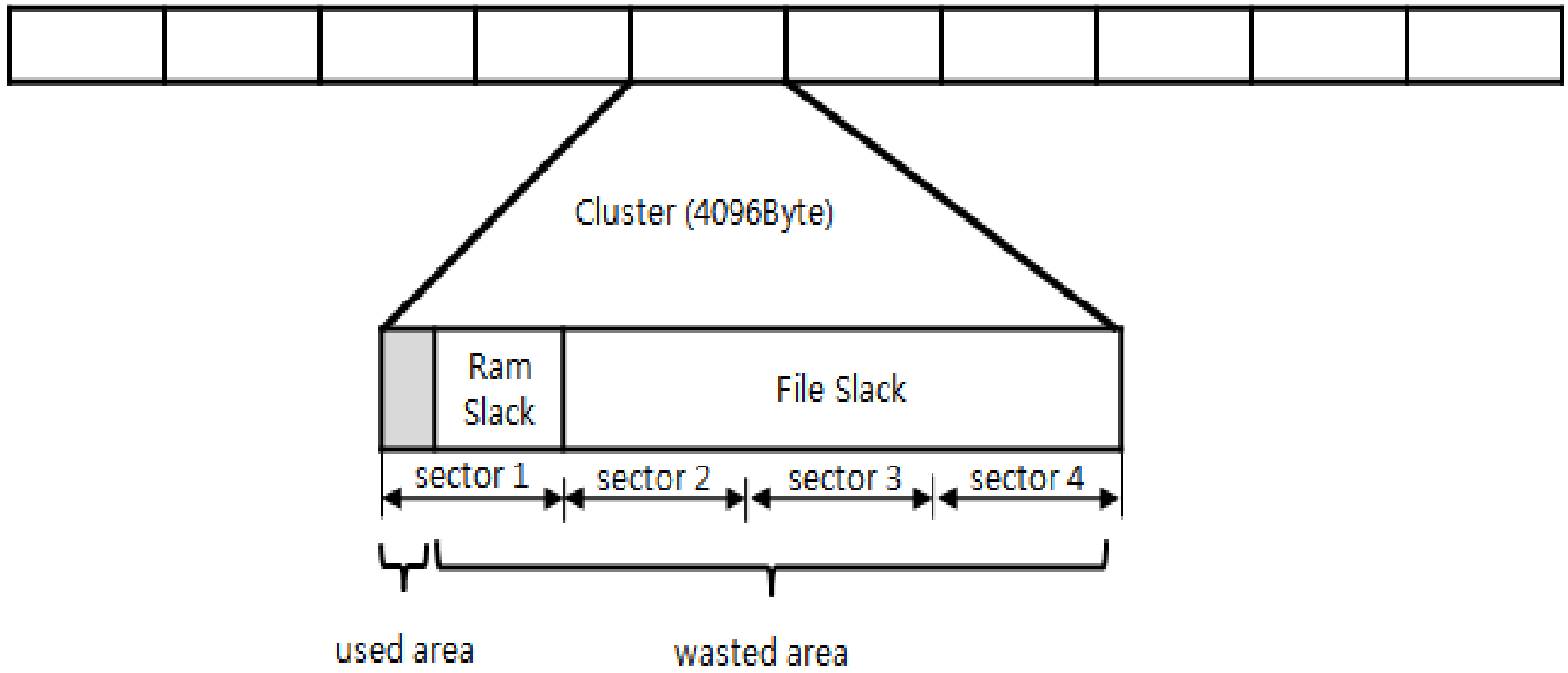
Important factor

Slack Space

- ✓ 물리적인 구조와 논리적인 구조의 차이로 발생하는 낭비 공간
- ✓ 파일에 할당된 공간이지만 사용되지 않는 낭비 공간
  - RAM Slack (Sector Slack)
  - File Slack (Drive Slack)
  - File System Slack
  - Volume Slack

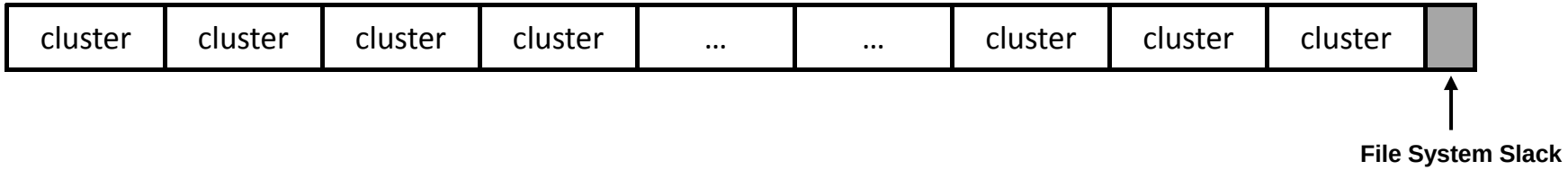
Important factor

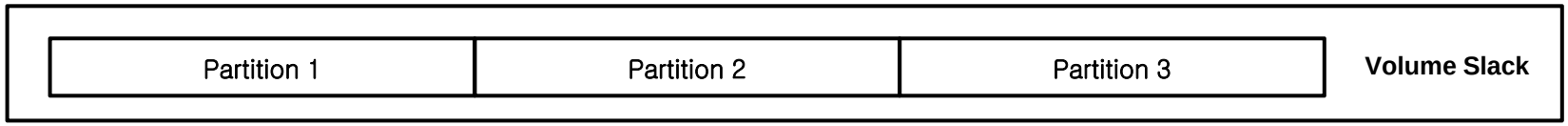
- **Slack Space (RAM Slack & File Slack)**



Important factor

- **Slack Space (File System Slack & Volume Slack)**



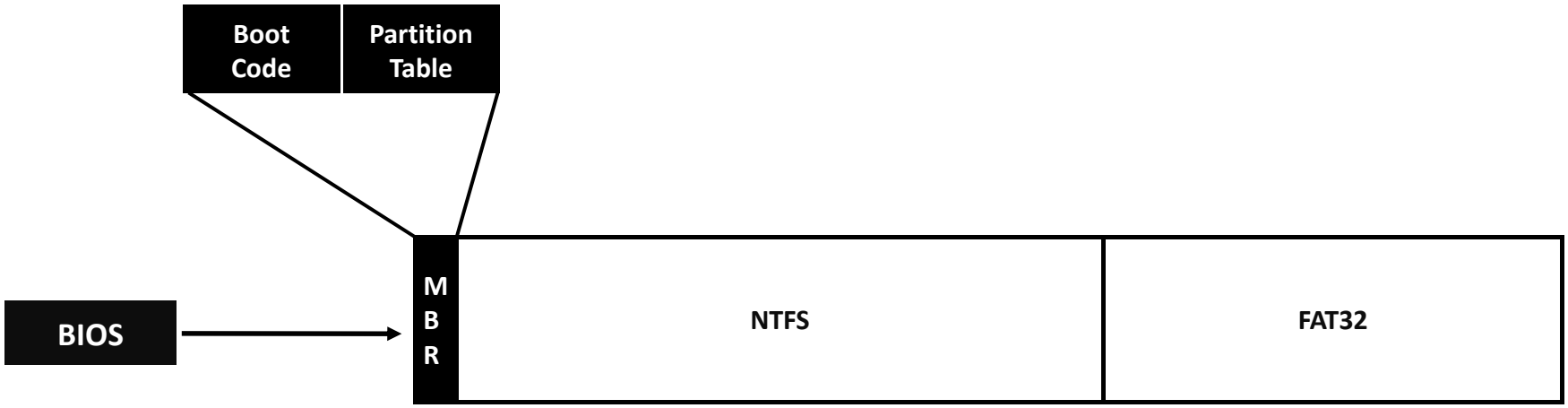


File System

- ✓ ***MBR***
- ✓ *FAT*
- ✓ *NTFS*

Master Boot Record

- **MBR called by BIOS**

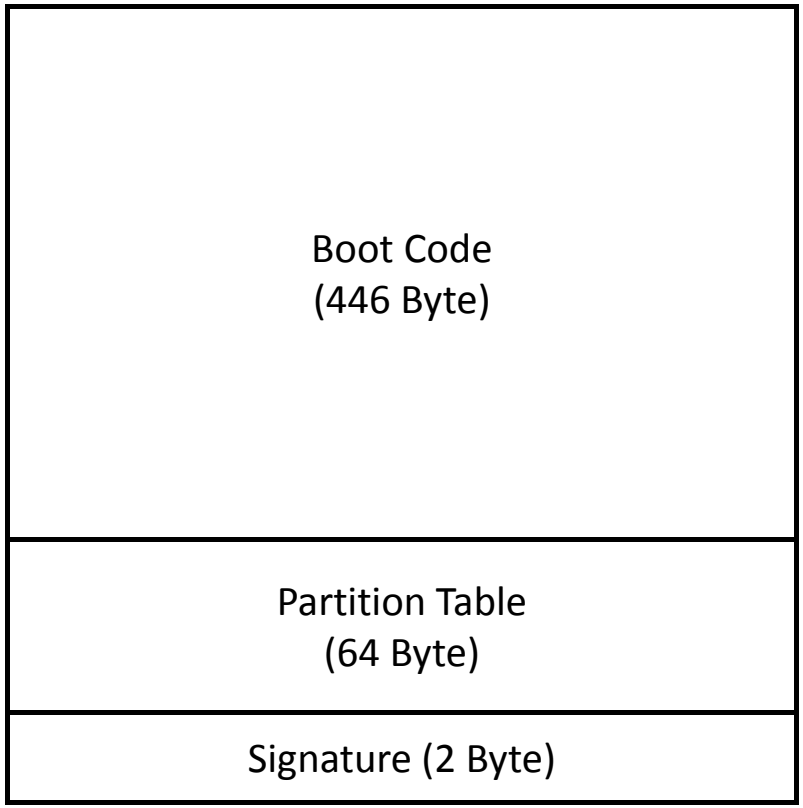


✓ 디스크의 첫 번째 섹터(LBA 0)에 위치하는 512Byte 크기의 영역

✓ Boot Code, Partition Table, Signature

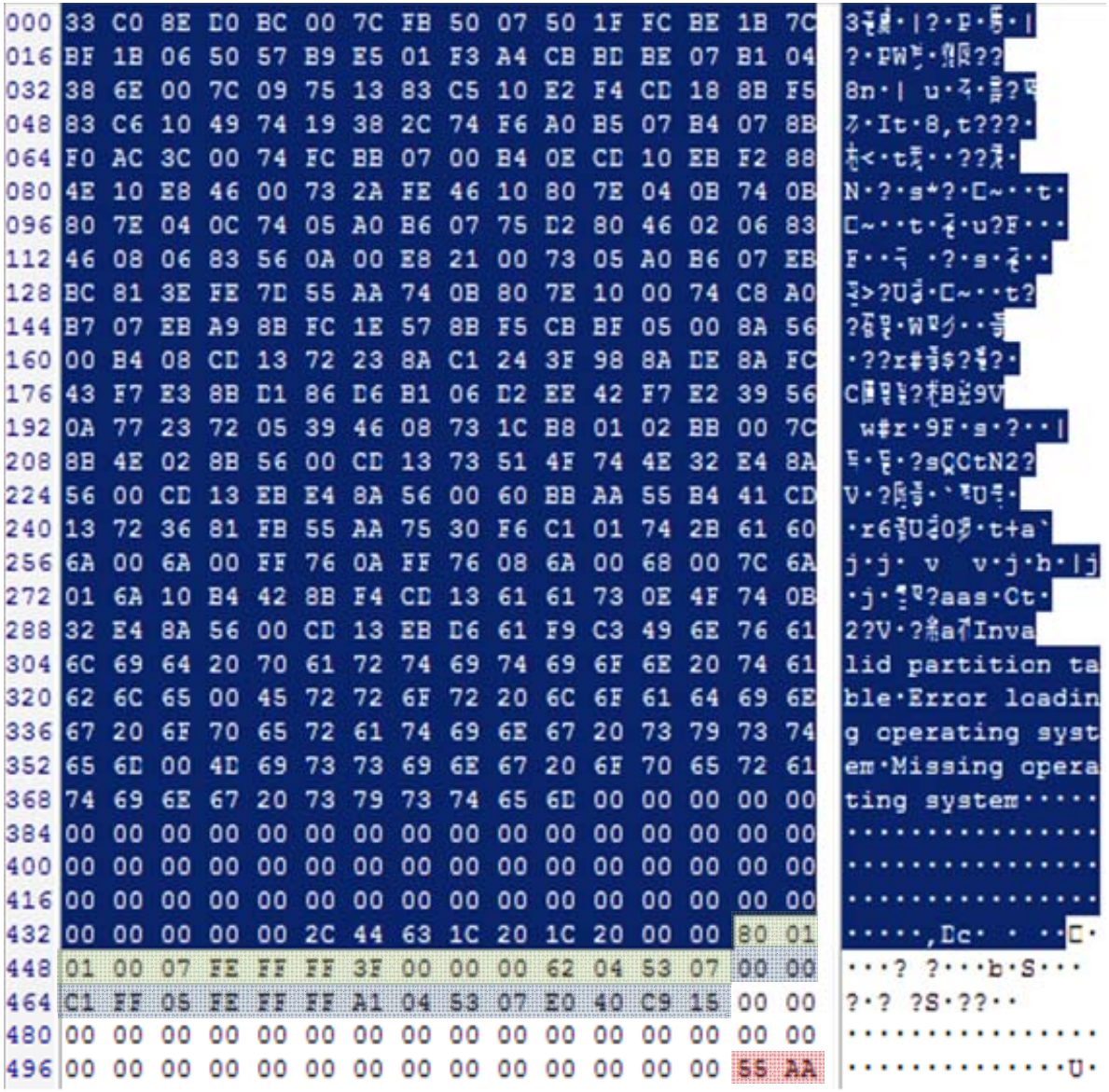
Master Boot Record

- **MBR Structure**



Master Boot Record

- MBR Dump



```
000 33 C0 8E D0 BC 00 7C FB 50 07 50 1F FC BE 1B 7C  3?????|?·P·P·??|
016 BF 1B 06 50 57 B9 E5 01 F3 A4 CB BD BE 07 B1 04  ?·PW?·??ó??????
032 38 6E 00 7C 09 75 13 83 C5 10 E2 F4 CD 18 8B F5  8n·|·u·?·??ôí??
048 83 C6 10 49 74 19 38 2C 74 F6 A0 B5 07 B4 07 8B  ?·It·8,t???·???
064 F0 AC 3C 00 74 FC BB 07 00 B4 0E CD 10 EB F2 88  ?¬<·t??··??í?·?
080 4E 10 E8 46 00 73 2A FE 46 10 80 7E 04 0B 74 0B  N·?·s*?·?~··t·
096 80 7E 04 0C 74 05 A0 B6 07 75 D2 80 46 02 06 83  ?~··t·?·u??F···
112 46 08 06 83 56 0A 00 E8 21 00 73 05 A0 B6 07 EB  F···V·· ·?·s·?··
128 BC 81 3E FE 7D 55 AA 74 0B 80 7E 10 00 74 C8 A0  ?>?U?·?~··t??
144 B7 07 EB A9 8B FC 1E 57 8B F5 CB BF 05 00 8A 56  ?????·W??·?·?
160 00 B4 08 CD 13 72 23 8A C1 24 3F 98 8A DE 8A FC  ·??r#??$?????·
176 43 F7 E3 8B D1 86 D6 B1 06 D2 EE 42 F7 E2 39 56  C?????????B??9V
192 0A 77 23 72 05 39 46 08 73 1C B8 01 02 BB 00 7C  ·w#r·9F·s·?··|
208 8B 4E 02 8B 56 00 CD 13 73 51 4F 74 4E 32 E4 8A  ?·?·V·??sQOtN2??
224 56 00 CD 13 EB E4 8A 56 00 60 BB AA 55 B4 41 CD  V·??????·`?U??·
240 13 72 36 81 FB 55 AA 75 30 F6 C1 01 74 2B 61 60  ·r6??U?u0???·t+a`
256 6A 00 6A 00 FF 76 0A FF 76 08 6A 00 68 00 7C 6A  j·j· v  v·j·h·|j
272 01 6A 10 B4 42 8B F4 CD 13 61 61 73 0E 4F 74 0B  ·j·??B??aas·Ot·
288 32 E4 8A 56 00 CD 13 EB D6 61 F9 C3 49 6E 76 61  2??V·??a??Inva
304 6C 69 64 20 70 61 72 74 69 74 69 6F 6E 20 74 61  lid partition ta
320 62 6C 65 00 45 72 72 6F 72 20 6C 6F 61 64 69 6E  ble·Error loadin
336 67 20 6F 70 65 72 61 74 69 6E 67 20 73 79 73 74  g operating syst
352 65 6D 00 4D 69 73 73 69 6E 67 20 6F 70 65 72 61  em·Missing opera
368 74 69 6E 67 20 73 79 73 74 65 6D 00 00 00 00 00  ting system·····
384 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ················
400 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ················
416 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ················
432 00 00 00 00 00 2C 44 63 1C 20 1C 20 00 00 80 01  ·····,Dc· · ··??·
448 01 00 07 FE FF FF 3F 00 00 00 62 04 53 07 00 00  ···? ?···b·S···
464 C1 FF 05 FE FF FF A1 04 53 07 E0 40 C9 15 00 00  ?·? ?S·??··
480 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ················
496 00 00 00 00 00 00 00 00 00 00 00 00 00 00 55 AA  ··············U·
```

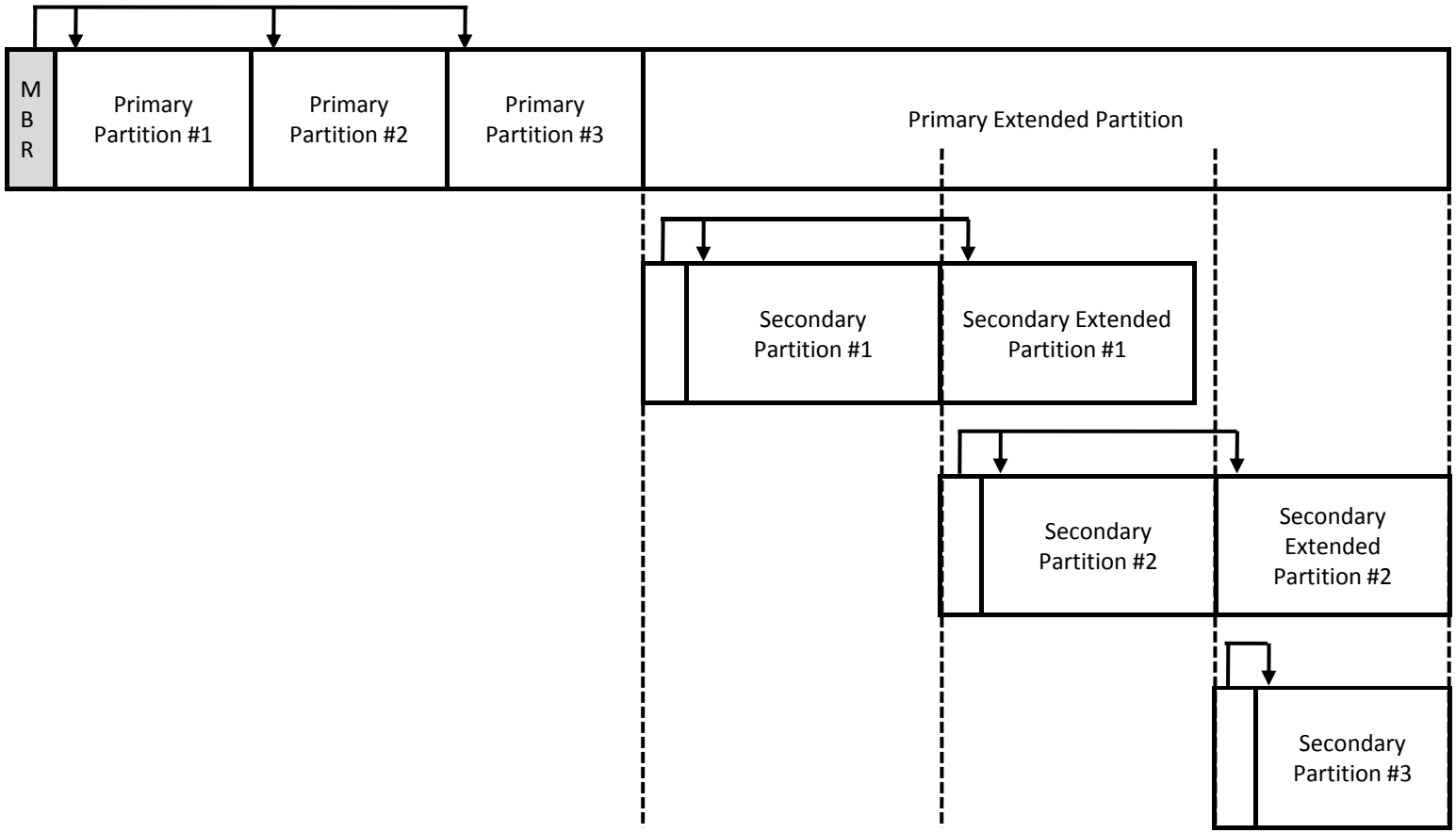
Master Boot Record

Partitions

- ✓ MBR 구조에 따르면 4개의 파티션만 생성 가능
- ✓ 멀티 부팅을 위해서라면 4개 모두 주 파티션으로 설정
- ✓ 3개의 주 파티션과 하나의 확장 파티션 생성
- ✓ 확장 파티션 내부의 논리 파티션 생성
- ✓ MS-DOS 환경에서는 기본적으로 30개의 논리 파티션 지원

Master Boot Record

Partitions

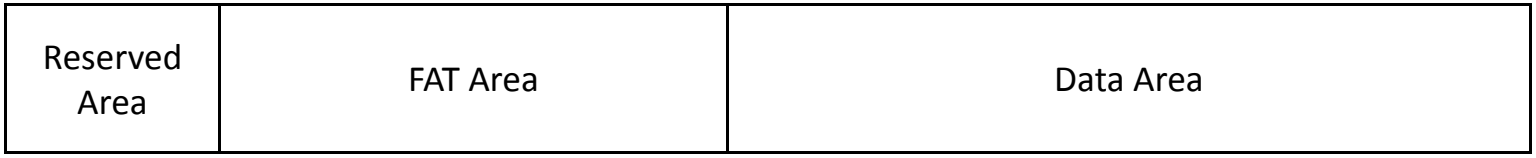


File System

- ✓ *MBR*
- ✓ ***FAT***
- ✓ *NTFS*

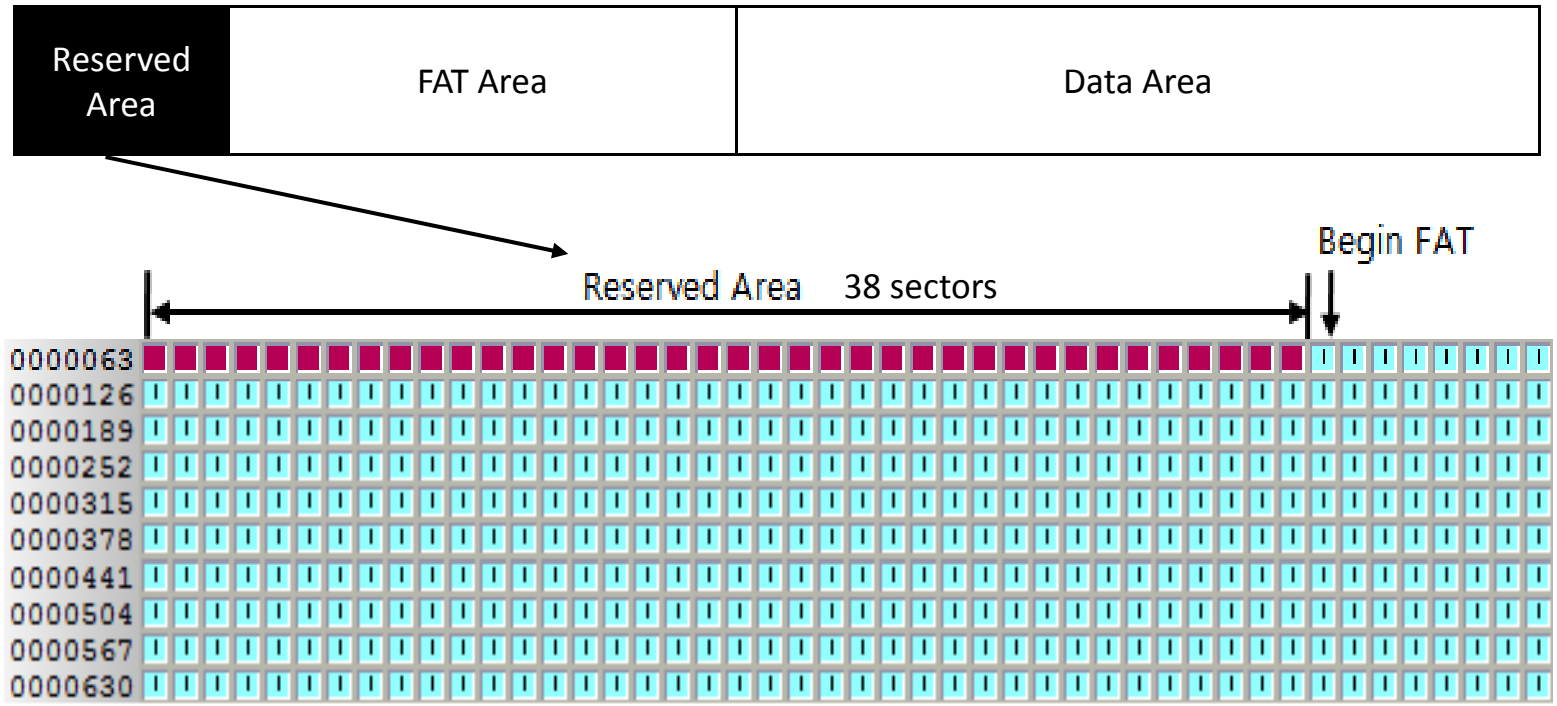
File Allocation Table

- **Structure**



File Allocation Table

- **Structure – Reserved Area**

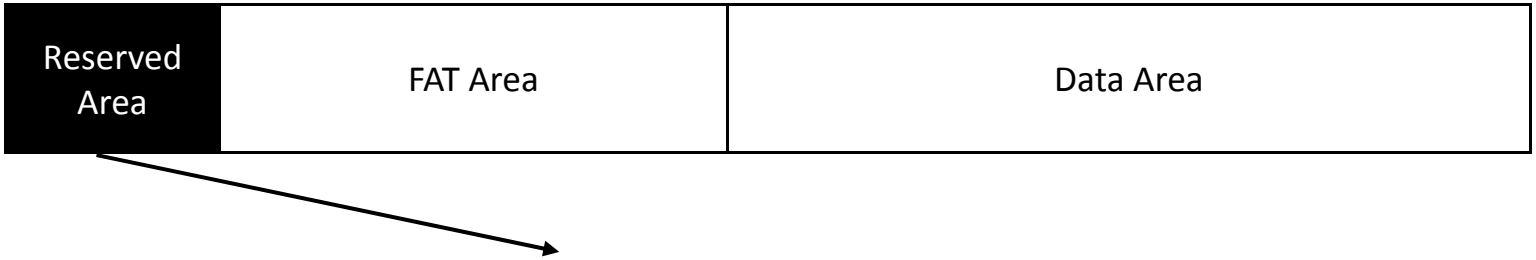


✓ FAT12/16 : 1 Sector(default)

✓ FAT32 : 32 Sectors(default)

File Allocation Table

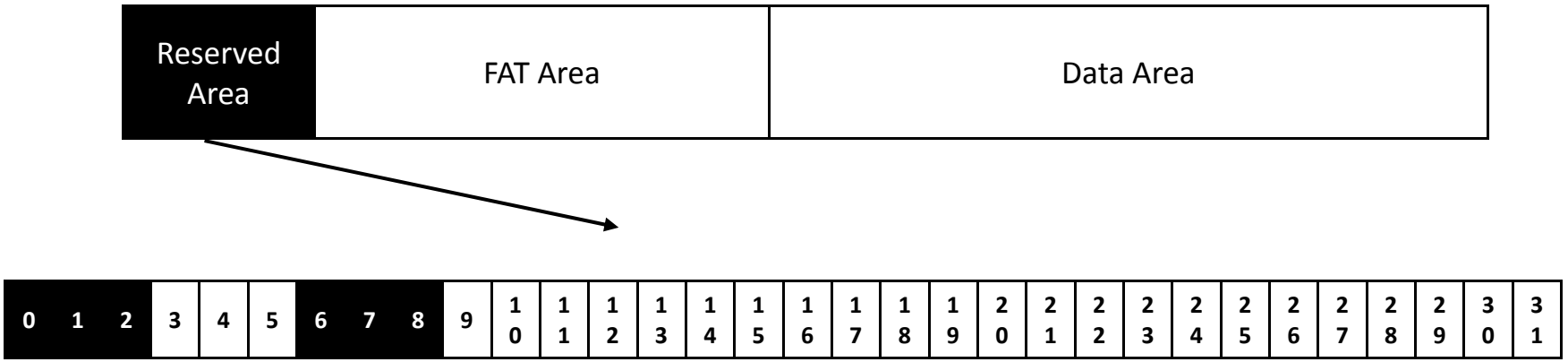
- **Structure – Reserved Area**



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

File Allocation Table

Structure – Reserved Area

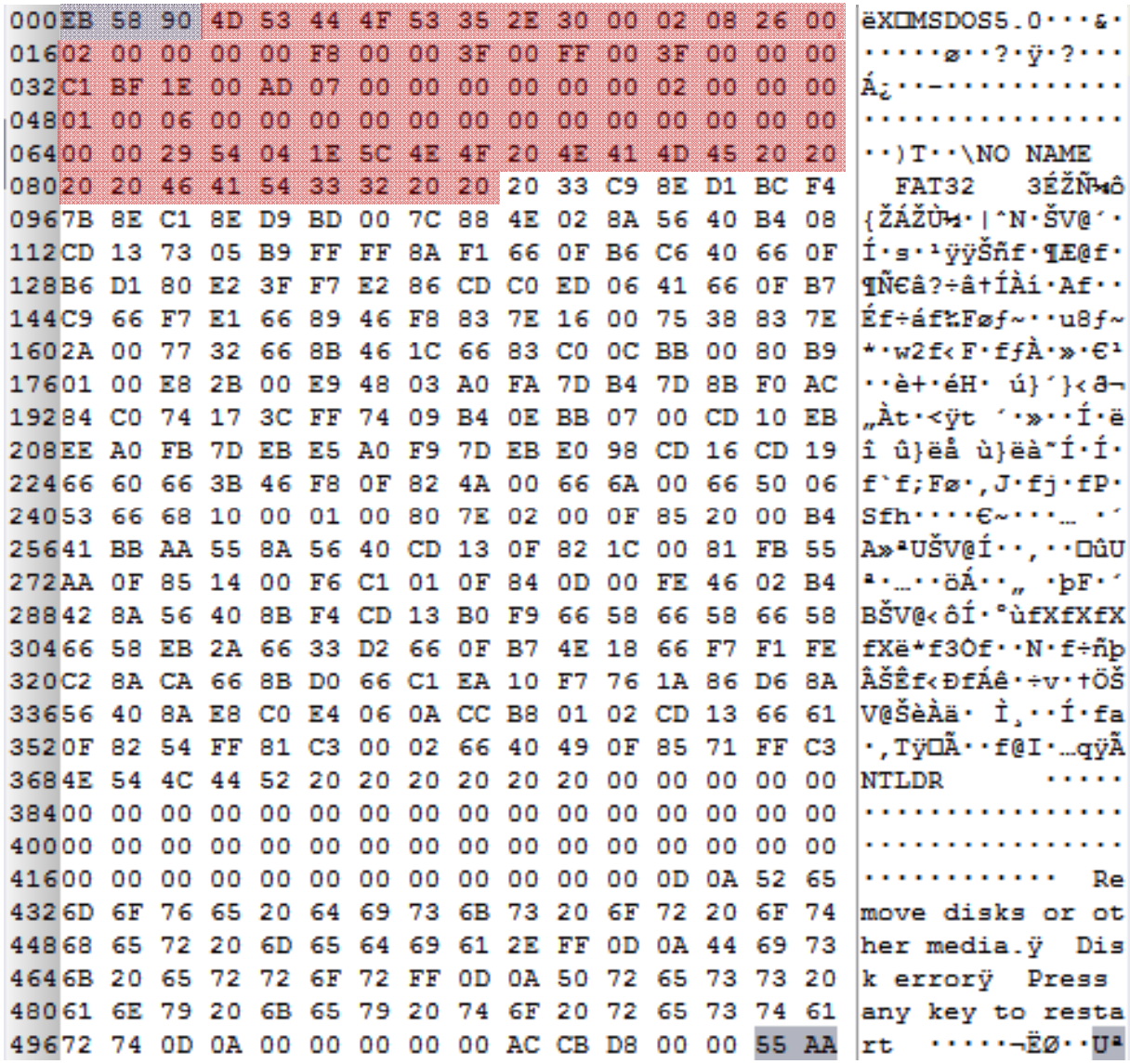


- ✓ 0, 6 : Volume Boot Sector
- ✓ 1, 7 : File System Information(FSINFO) Structure
- ✓ 2, 8 : Bootstrap code

File Allocation Table

Volume Boot Sector

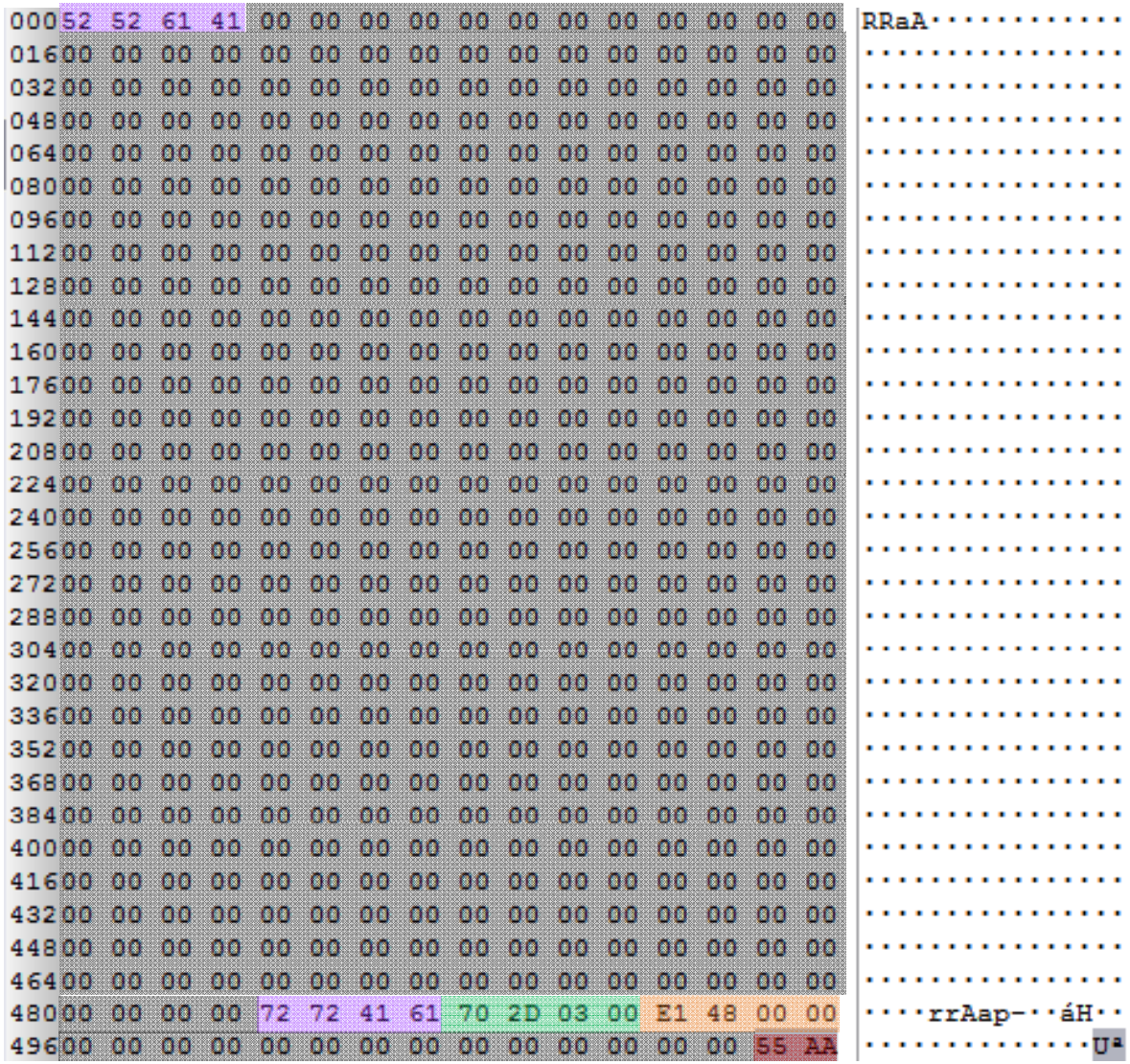
FAT Format	Byte Range	Description
FAT12/16	0 – 2	Jump command to boot code
FAT32		
FAT12/16	3 – 61	BIOS Parameter Block(BPB)
FAT32	3 – 89	
FAT12/16	62 – 509	Boot code Error message
FAT32	90 – 509	
FAT12/16	510 - 511	Signature (0x55AA)
FAT32		



File Allocation Table

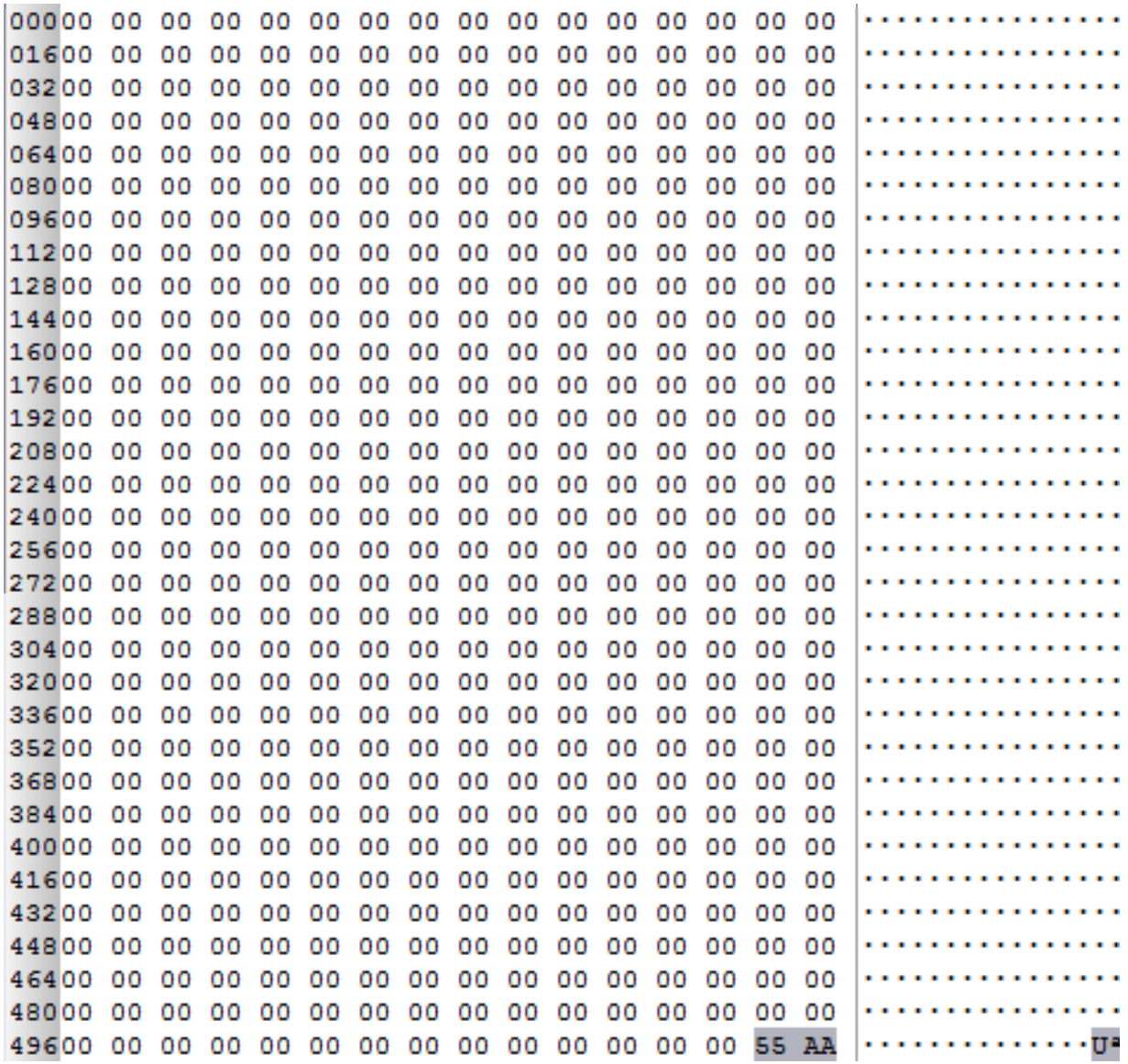
File System Information (FAT32)

Byte Range	Description
0 – 3	Signature (0x41615252)
4 – 483	Not Used
484-487	Signature (0x61417272)
488-491	Number of free cluster
492-495	Next free cluster
496-507	Not used
508-511	Signature (0xAA550000)



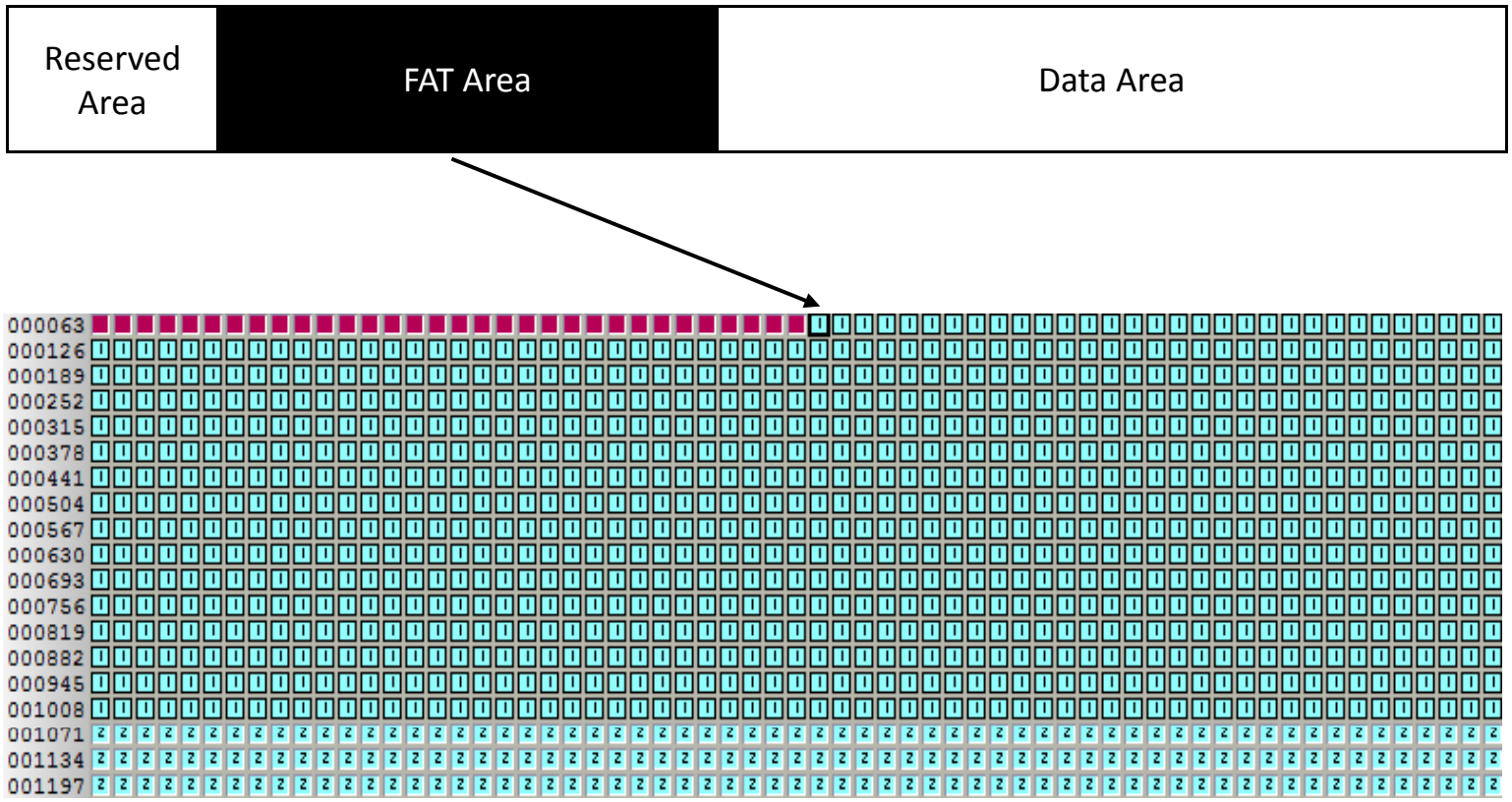
File Allocation Table

- Bootstrap code



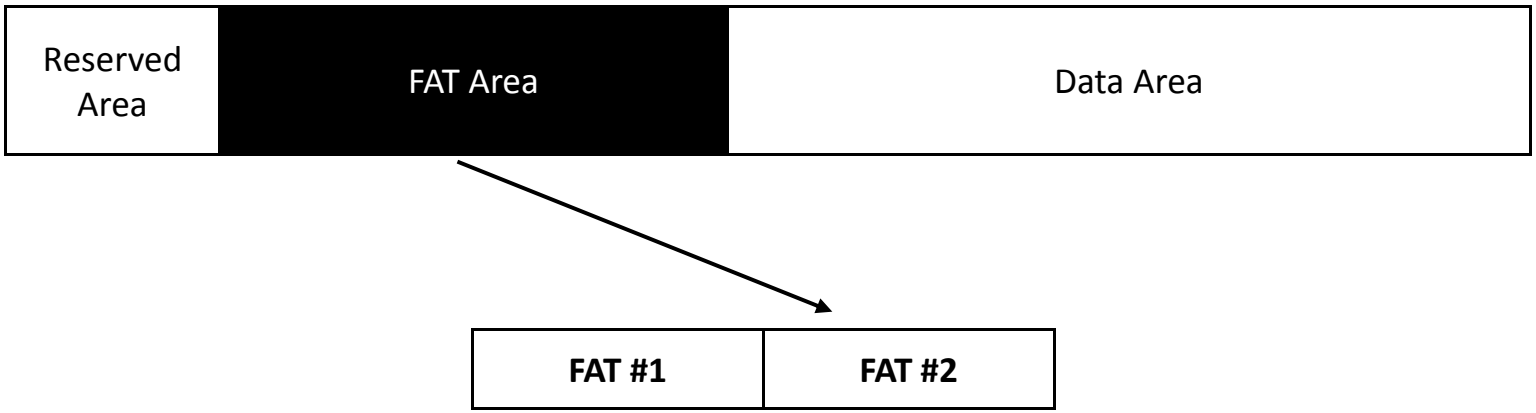
File Allocation Table

- **Structure – FAT Area**



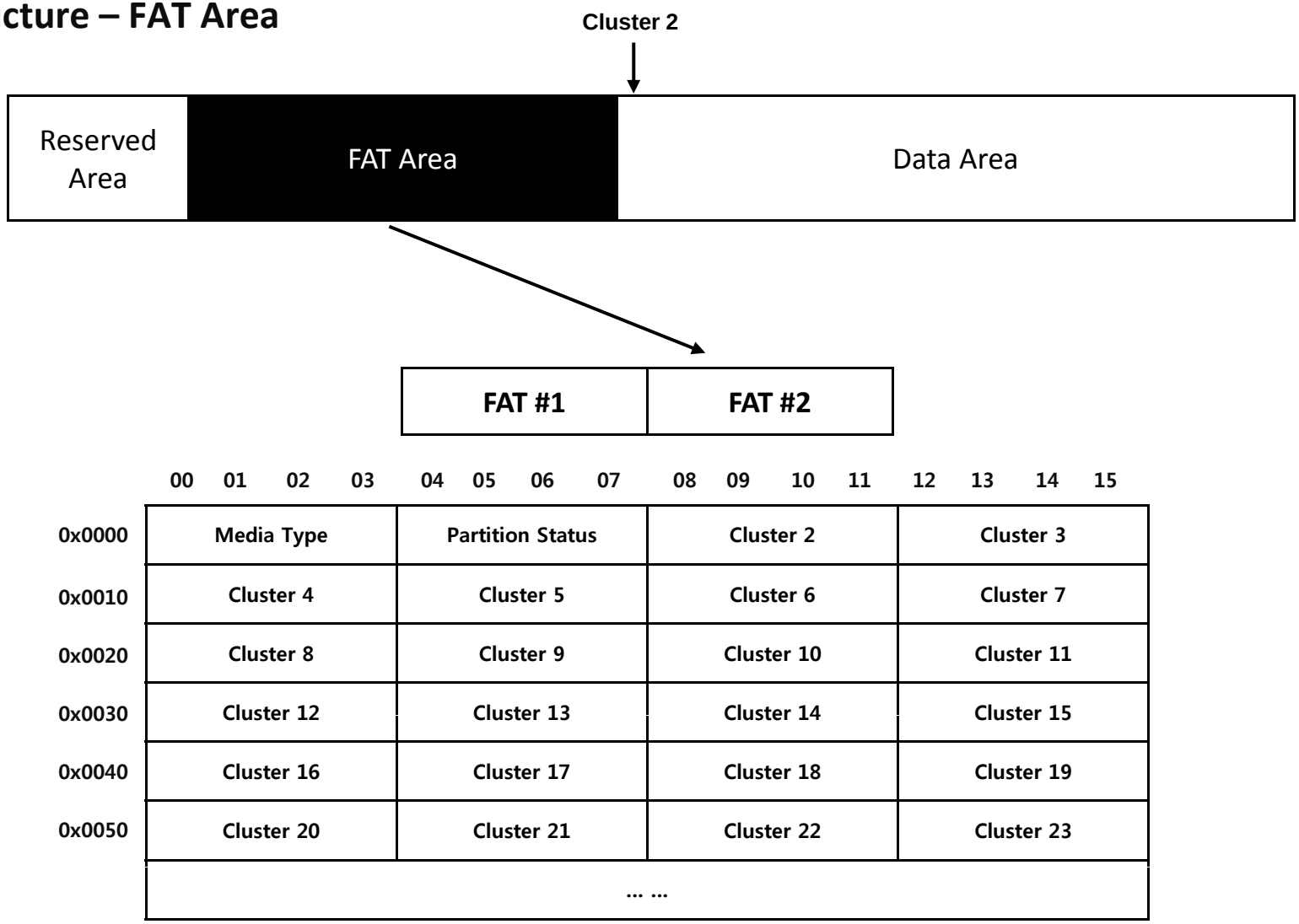
File Allocation Table

- **Structure – FAT Area**



File Allocation Table

Structure – FAT Area



	00 01 02 03	04 05 06 07	08 09 10 11	12 13 14 15
0x0000	Media Type	Partition Status	Cluster 2	Cluster 3
0x0010	Cluster 4	Cluster 5	Cluster 6	Cluster 7
0x0020	Cluster 8	Cluster 9	Cluster 10	Cluster 11
0x0030	Cluster 12	Cluster 13	Cluster 14	Cluster 15
0x0040	Cluster 16	Cluster 17	Cluster 18	Cluster 19
0x0050	Cluster 20	Cluster 21	Cluster 22	Cluster 23
	... ...			

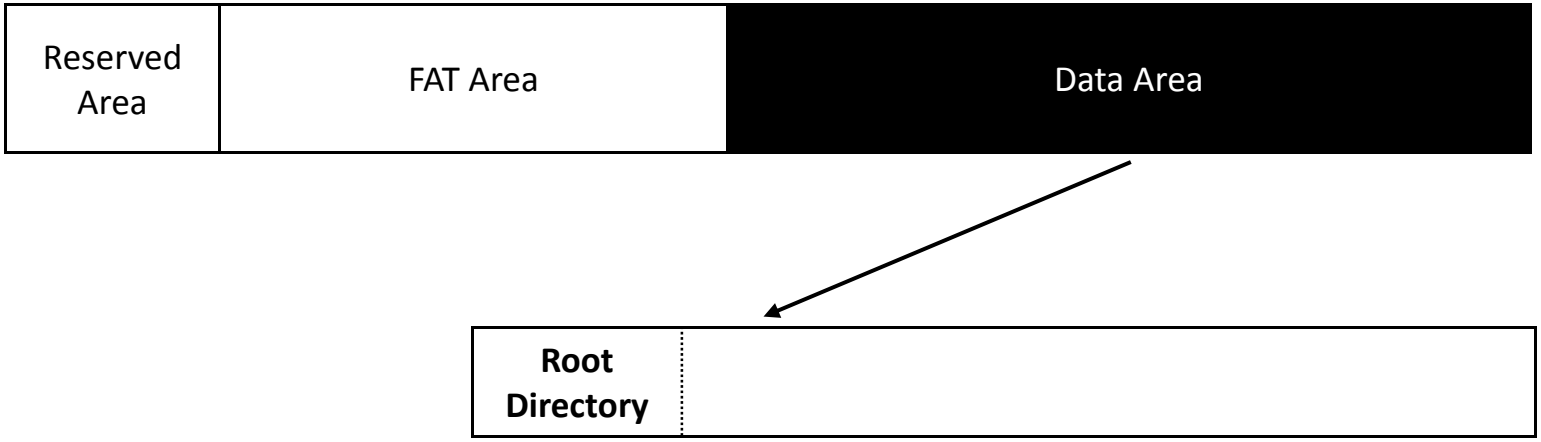
File Allocation Table

- FAT Dump

```
000 F8 FF FF 0F FF FF FF FF FF FF FF 0F FF FF FF 0F  øÿÿ·ÿÿÿÿÿÿÿ·ÿÿÿ·
016 05 00 00 00 06 00 00 00 07 00 00 00 08 00 00 00  ················
032 FF FF FF 0F 0A 00 00 00 FF FF FF 0F 0C 00 00 00  ÿÿÿ· ···ÿÿÿ·····
048 0D 00 00 00 0E 00 00 00 0F 00 00 00 10 00 00 00   ···············
064 FF FF FF 0F 12 00 00 00 13 00 00 00 14 00 00 00  ÿÿÿ·············
080 15 00 00 00 16 00 00 00 17 00 00 00 18 00 00 00  ················
096 19 00 00 00 1A 00 00 00 1B 00 00 00 1C 00 00 00  ················
112 1D 00 00 00 1E 00 00 00 1F 00 00 00 FF FF FF 0F  ············ÿÿÿ·
128 21 00 00 00 22 00 00 00 23 00 00 00 24 00 00 00  !···"···#···$···
144 25 00 00 00 26 00 00 00 27 00 00 00 28 00 00 00  %···&···'···(···
160 29 00 00 00 2A 00 00 00 2B 00 00 00 2C 00 00 00  )···*···+···,···
176 2D 00 00 00 2E 00 00 00 2F 00 00 00 30 00 00 00  -···.···/···0···
192 31 00 00 00 32 00 00 00 33 00 00 00 34 00 00 00  1···2···3···4···
208 35 00 00 00 36 00 00 00 37 00 00 00 38 00 00 00  5···6···7···8···
224 39 00 00 00 3A 00 00 00 3B 00 00 00 3C 00 00 00  9···:···;···<···
240 3D 00 00 00 3E 00 00 00 3F 00 00 00 40 00 00 00  =···>···?···@···
256 41 00 00 00 42 00 00 00 43 00 00 00 44 00 00 00  A···B···C···D···
272 45 00 00 00 46 00 00 00 47 00 00 00 48 00 00 00  E···F···G···H···
288 49 00 00 00 4A 00 00 00 4B 00 00 00 4C 00 00 00  I···J···K···L···
304 4D 00 00 00 4E 00 00 00 4F 00 00 00 50 00 00 00  M···N···O···P···
320 51 00 00 00 52 00 00 00 53 00 00 00 54 00 00 00  Q···R···S···T···
336 55 00 00 00 56 00 00 00 57 00 00 00 58 00 00 00  U···V···W···X···
352 59 00 00 00 5A 00 00 00 5B 00 00 00 5C 00 00 00  Y···Z···[···\···
368 5D 00 00 00 5E 00 00 00 5F 00 00 00 60 00 00 00  ]···^···_···`···
384 61 00 00 00 62 00 00 00 63 00 00 00 64 00 00 00  a···b···c···d···
400 65 00 00 00 66 00 00 00 67 00 00 00 68 00 00 00  e···f···g···h···
416 69 00 00 00 6A 00 00 00 6B 00 00 00 6C 00 00 00  i···j···k···l···
432 6D 00 00 00 6E 00 00 00 6F 00 00 00 70 00 00 00  m···n···o···p···
448 71 00 00 00 72 00 00 00 73 00 00 00 74 00 00 00  q···r···s···t···
464 75 00 00 00 76 00 00 00 77 00 00 00 78 00 00 00  u···v···w···x···
480 79 00 00 00 7A 00 00 00 7B 00 00 00 7C 00 00 00  y···z···{···|···
496 7D 00 00 00 7E 00 00 00 7F 00 00 00 80 00 00 00  }···~···□···€···
```

File Allocation Table

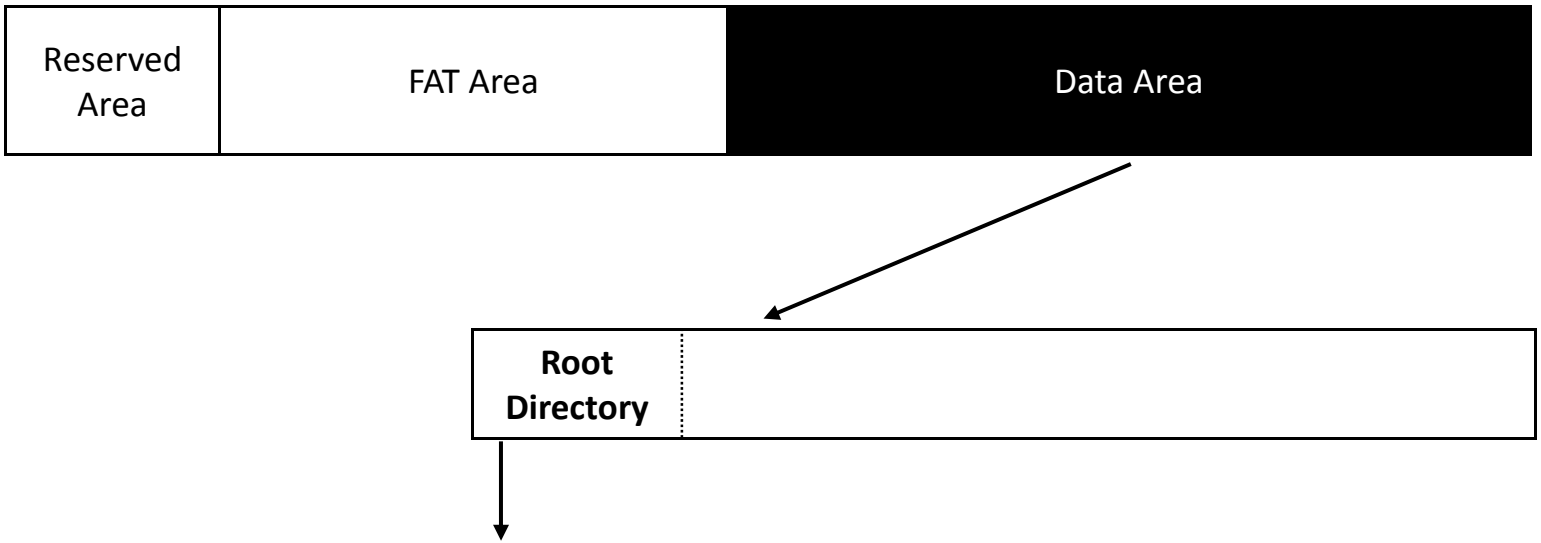
Structure – Data Area



✓ 모든 파일과 디렉터리는 Directory Entry로 표현됨

File Allocation Table

- **Structure – Data Area**



	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0x00	Name								Extension			Attr	Reserved		Create Time	
0x10	Created Date				Last Accessed Date				Starting Cluster Hi			Last Written Time			Last Written Date	
	Starting Cluster Low				File Size											

File Allocation Table

LFN – Long File Name

File Allocation Table

- **Root Directory**


bin


out-CIST-PRONEER-20090904172828


PRESENT


runner.exe

```
000 43 4F 46 45 45 20 20 20 20 20 20 08 00 00 00 00  COFEE      ·····
016 00 00 00 00 00 00 0E 71 24 3B 00 00 00 00 00 00  ·······q$;······
032 E5 44 45 4C 54 45 4D 50 20 20 20 22 00 80 17 5A  åDELTEMP   "·€·Z
048 24 3B 24 3B 00 00 05 71 24 3B 03 00 00 70 60 3D  $;$;···q$;···p`=
064 E5 53 00 44 00 45 00 4C 00 54 00 0F 00 81 45 00  åS·D·E·L·T···□E·
080 4D 00 50 00 31 00 00 00 FF FF 00 00 FF FF FF FF  M·P·1···ÿÿ··ÿÿÿÿ
096 E5 44 45 4C 54 45 7E 31 20 20 20 22 00 17 04 71  åDELTE~1   "···q
112 24 3B 24 3B 00 00 05 71 24 3B 00 00 00 00 00 00  $;$;···q$;······
128 E5 53 00 44 00 45 00 4C 00 4D 00 0F 00 5D 46 00  åS·D·E·L·M···]F·
144 54 00 30 00 30 00 30 00 30 00 00 00 30 00 30 00  T·0·0·0·0···0·0·
160 E5 44 45 4C 4D 46 7E 31 20 20 20 22 00 1A 04 71  åDELMF~1   "···q
176 24 3B 24 3B 00 00 05 71 24 3B 00 00 00 00 00 00  $;$;···q$;······
192 42 49 4E 20 20 20 20 20 20 20 20 10 08 A0 0D 71  BIN        ··  q
208 24 3B 24 3B 03 00 0E 71 24 3B 02 00 00 00 00 00  $;$;···q$;······
224 52 55 4E 4E 45 52 20 20 45 58 45 20 18 B2 15 71  RUNNER  EXE ·²·q
240 24 3B 2A 3B 03 00 AC 90 55 38 FE 00 69 60 0A 00  $;*;··¬□U8þ·i` ·
256 43 37 00 32 00 38 00 32 00 38 00 0F 00 CA 00 00  C7·2·8·2·8···Ê··
272 FF FF FF FF FF FF FF FF FF FF 00 00 FF FF FF FF  ÿÿÿÿÿÿÿÿÿÿ··ÿÿÿÿ
288 02 45 00 45 00 52 00 2D 00 32 00 0F 00 CA 30 00  ·E·E·R·-·2···Ê0·
304 30 00 39 00 30 00 39 00 30 00 00 00 34 00 31 00  0·9·0·9·0···4·1·
320 01 6F 00 75 00 74 00 2D 00 43 00 0F 00 CA 49 00  ·o·u·t·-·C···ÊI·
336 53 00 54 00 2D 00 50 00 52 00 00 00 4F 00 4E 00  S·T·-·P·R···O·N·
352 4F 55 54 2D 43 49 7E 31 20 20 20 10 00 8B 8E 8B  OUT-CI~1   ··‹Ž‹
368 24 3B 24 3B 03 00 8F 8B 24 3B D8 0D 00 00 00 00  $;$;··□‹$;Ø ····
384 52 45 43 59 43 4C 45 52 20 20 20 17 00 BB 82 7E  RECYCLER   ··»‚~
400 28 3B 28 3B 00 00 83 7E 28 3B 7F 48 00 00 00 00  (;(;··ƒ~(;□H····
416 41 55 54 4F 52 55 4E 20 49 4E 46 27 18 A8 83 7E  AUTORUN INF'·¨ƒ~
432 28 3B 2B 3B 00 00 8B 7E 28 3B 85 48 1A 01 00 00  (;+;··‹~(;…H····
448 E5 49 00 6D 00 6A 00 69 00 6E 00 0F 00 8C 2E 00  åI·m·j·i·n···Œ.·
464 62 00 6D 00 70 00 00 00 FF FF 00 00 FF FF FF FF  b·m·p···ÿÿ··ÿÿÿÿ
480 E5 4D 4A 49 4E 20 20 20 42 4D 50 20 00 4E 86 7E  åMJIN   BMP ·N†~
496 28 3B 28 3B 00 00 6F 7E 28 3B 86 48 6F FA 0F 00  (;(;··o~(;†Hoú··
```

File System

- ✓ *MBR*
- ✓ *FAT*
- ✓ ***NTFS***

New Technology File System

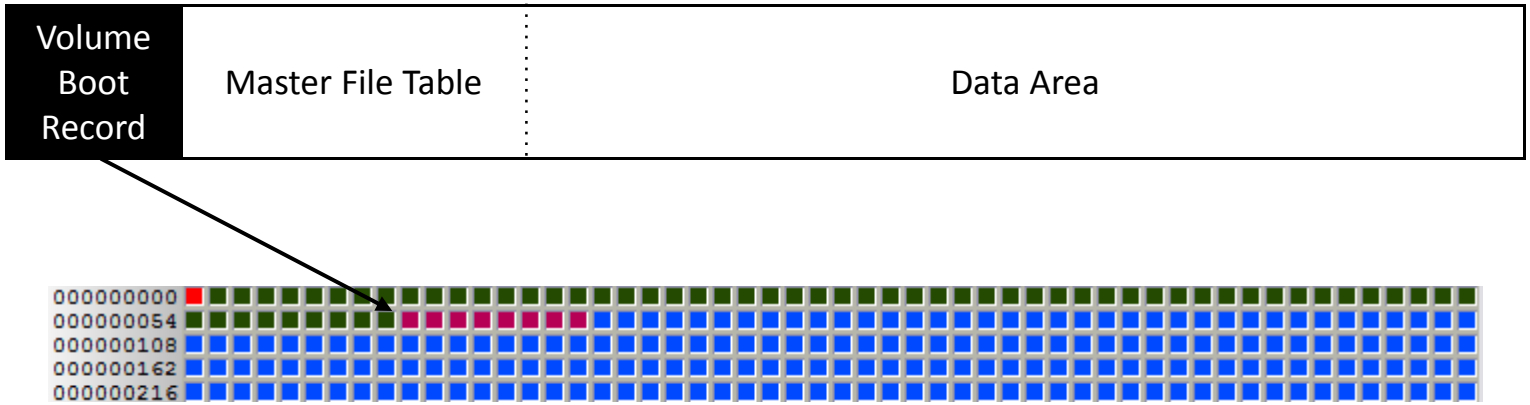
Characteristics

- ✓ Update Sequence Number (USN) Journal
- ✓ Alternate Data Stream (ADS)
- ✓ Sparse File
- ✓ File Compression
- ✓ Volume Shadow Copy (VSS)
- ✓ Encrypting File System (EFS)
- ✓ Quotas
- ✓ Supporting Unicode
- ✓ Dynamic Bad Cluster Allocation

New Technology File System

- **Structure**

Structure – Volume Boot Record (VBR)



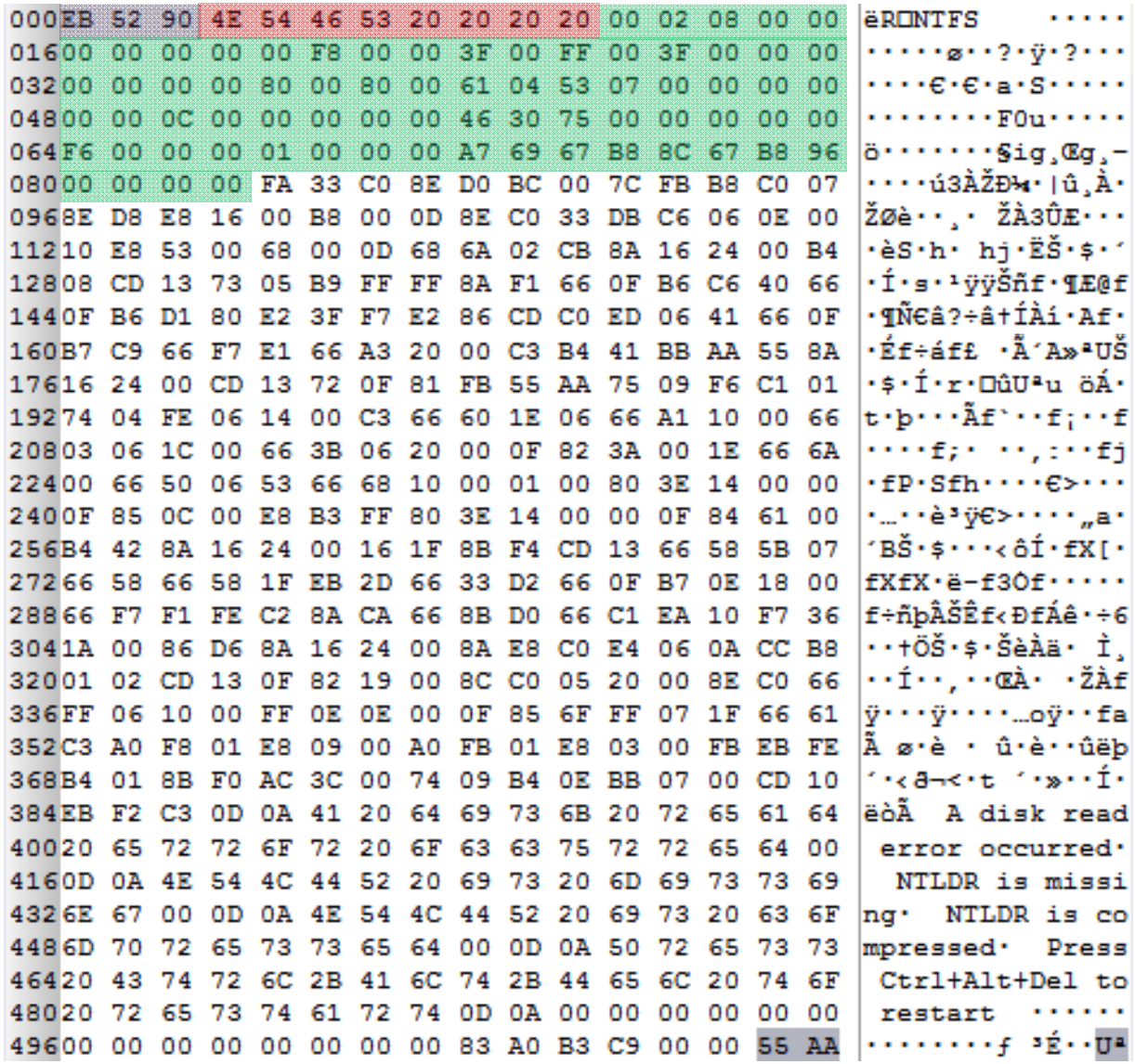
Cluster Size	VBR Size
512Byte	1
1KB	2
2KB	4
4KB	8

✓ VBR : Volume Boot Sector + Bootstrap code

New Technology File System

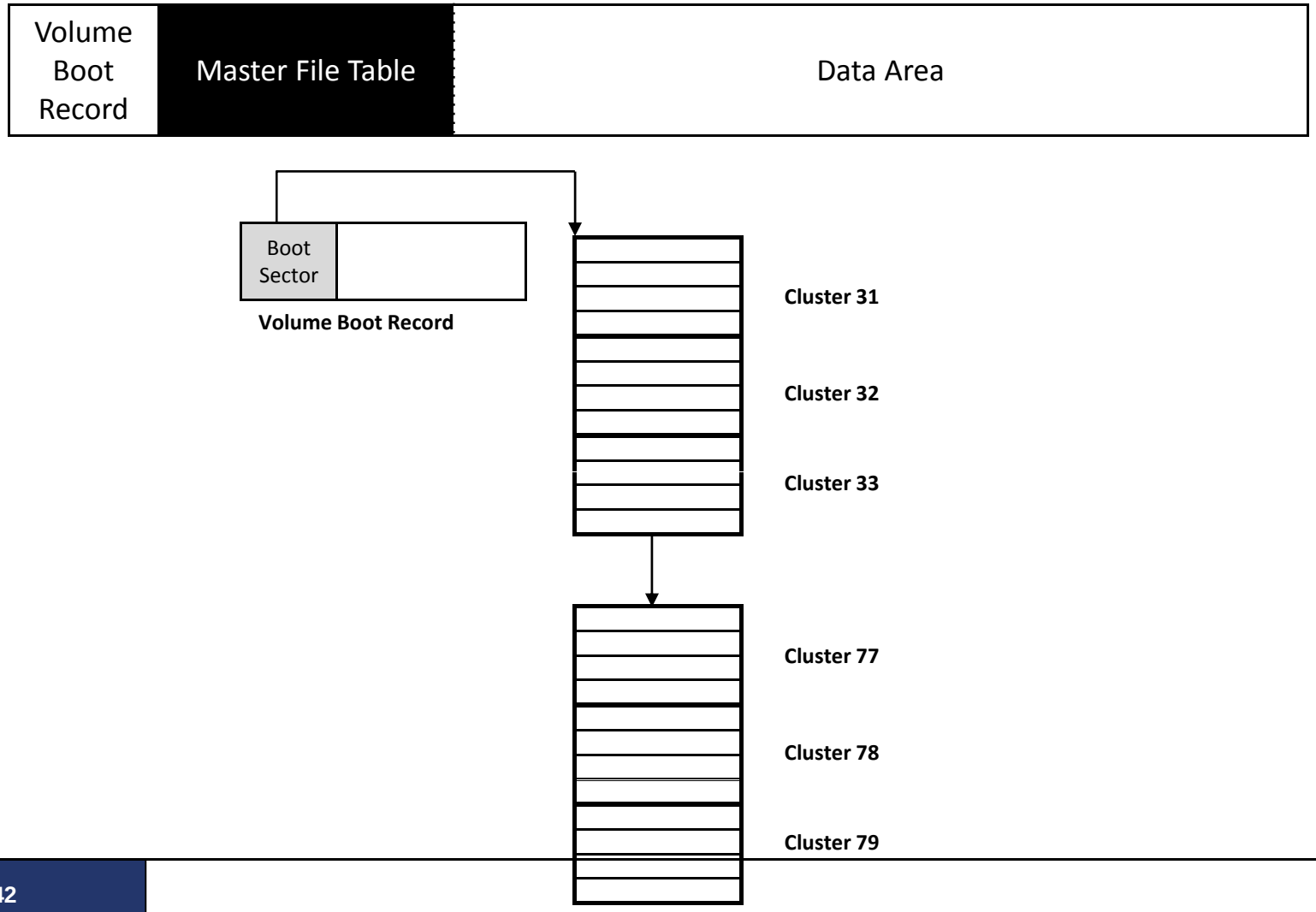
Volume Boot Sector

Byte Range	Description
0 – 2	Jump Instruction
3 – 10	OEM ID
11 – 83	BIOS Parameter Block
84 – 509	Boot Code
510 - 511	Signature



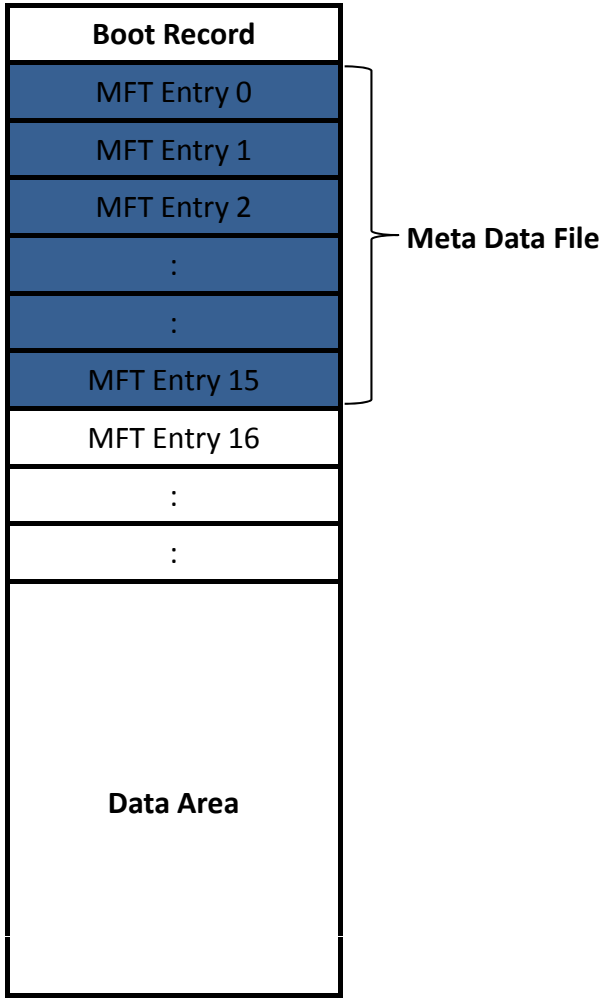
New Technology File System

- **Structure – Master File Table (MFT)**



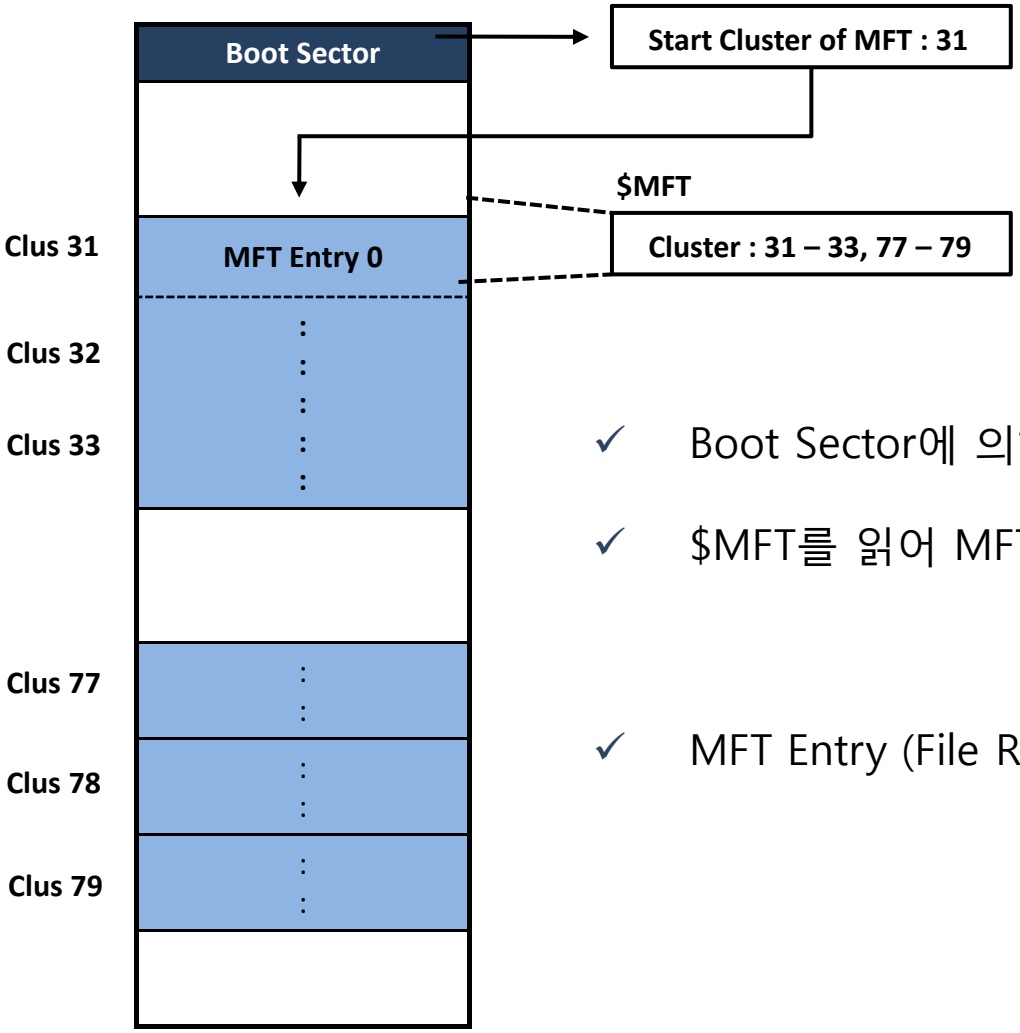
New Technology File System

MFT Layout



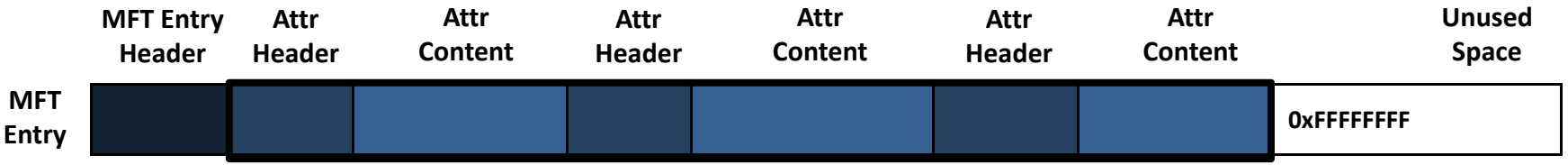
Entry Number	Entry File Name	Description
0	$MFT	NTFS 상의 모든 파일들의 MFT 레코드 정보
1	$MFTMirr	MFT 파일의 일부 백업본
2	$LogFile	메타데이터의 트랜잭션 저널 정보
3	$Volume	볼륨의 레이블, 식별자, 버전 등의 정보
4	$AttrDef	속성의 식별자, 이름, 크기 등의 정보
5	.	볼륨의 루트 디렉터리
6	$Bitmap	볼륨의 클러스터 할당 정보
7	$Boot	부팅 가능할 경우 부트섹터 정보
8	$BadClus	배드섹터를 가지는 클러스터 정보
9	$Secure	파일의 보안, 접근제어와 관련된 정보
10	$Upcase	모든 유니코드 문자의 대문자
11	$Extend	$ObjID, $Quota, $Reparse points, $UsnJrnl의 추가적인 레코드를 위해 사용
12 – 15		미래를 위해 예약
-	$ObjId	파일 고유의 ID 정보 (Windows 2000 -)
-	$Quota	사용자별 할당량 정보 (Windows 2000 -)
-	$Reparse	Reparse Point 에 대한 정보 (Windows 2000 -)
-	$UsnJrnl	파일, 디렉터리의 변경 정보 (Windows 2000 -)

MFT Layout



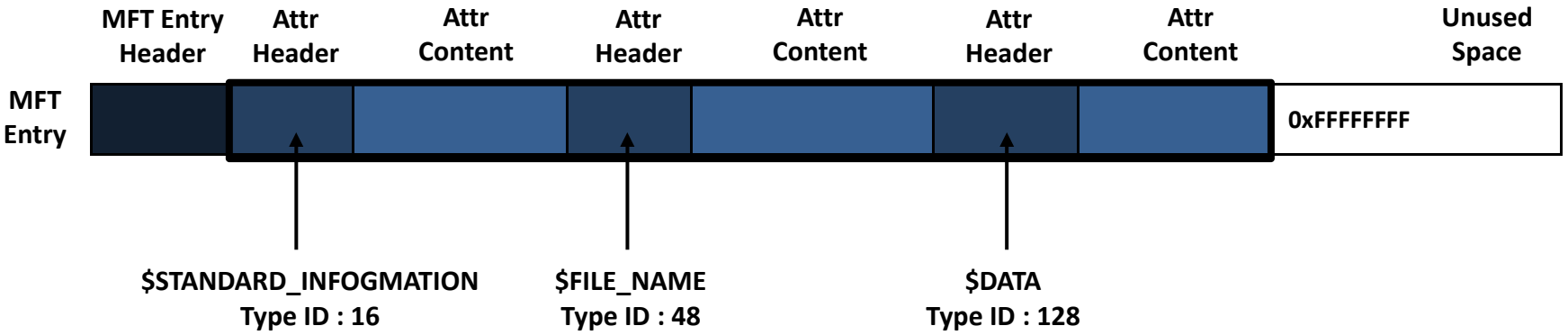
- ✓ Boot Sector에 의해 MFT 시작 위치로 이동
- ✓ $MFT를 읽어 MFT 전체의 레이아웃 판단
- ✓ MFT Entry (File Record) : 1,024Byte(1K) 고정

MFT Entry (File Record)



- ✓ **MFT Entry** : 파일, 디렉터리에 대한 속성 정보
- ✓ **Attribute** : 해당 속성을 표현하는 구조 (파일이름, 시간정보, 크기, 소유자, 데이터 등)
- ✓ 각 데이터 특성에 따라 MFT Entry는 다양한 속성으로 구성

MFT Entry (File Record)



- ✓ **${uppercase}** : 속성을 의미 (단, $MFT 제외)
- ✓ 기본적인 파일은 3가지 속성을 가짐
  - $STANDARD_INFORMATION
  - $FILE_NAME
  - $DATA

New Technology File System

Attributes

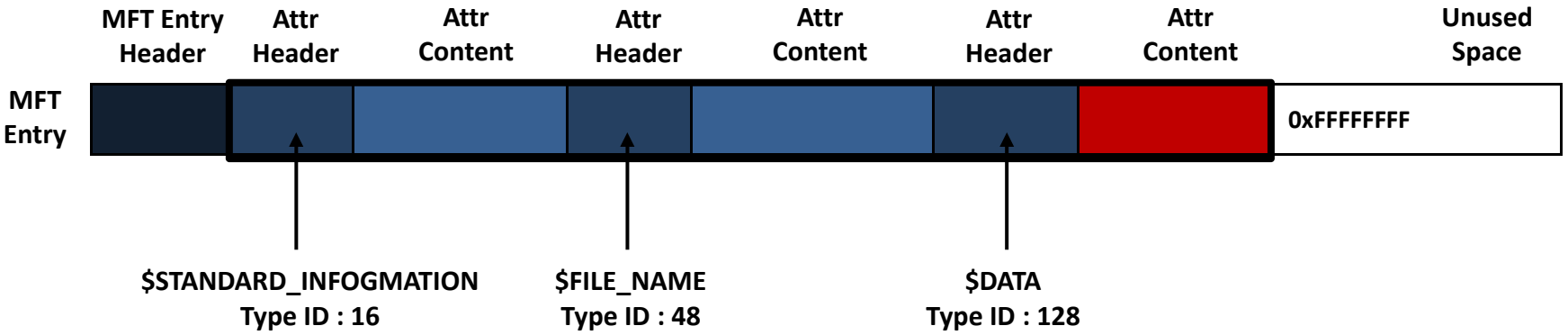
Attr Type Num	Attr Name	Description
16	**$STANDARD_INFORMATION**	최근 접근 시간, 생성 시간, 소유자, 보안 아이디
32	**$ATTRIBUTE_LIST**	속성을 찾을 수 있는 속성 리스트
48	**$FILE_NAME**	파일 이름(유니코드)
64	**$VOLUME_VERSION**	볼륨 정보, 오직 1.2 버전에만 존재 (Windows NT)
64	**$OBJECT_ID**	16바이트로 이루어진 파일, 디렉터리의 고유 값, 3.0 이상에서만 존재
80	**$SECURITY_DESCRIPTOR**	파일의 접근 제어와 보안 속성
96	**$VOLUME_NAME**	볼륨 이름과 관련 정보
112	**$VOLUME_INFORMATION**	파일 시스템의 버전과 여러 FLAG
128	**$DATA**	파일의 내용
144	**$INDEX_ROOT**	인덱스 트리의 루트 노드
160	**$INDEX_ALLOCATION**	인덱스 트리와 연결된 노드
176	**$BITMAP**	할당 정보를 관리하는 속성
192	**$SYMBOLIC_LINK**	심볼릭 링크 정보, 오직 1.2 버전에만 존재 (Windows NT)
192	**$REPARSE_POINT**	심볼릭 링크에서 사용하는 reparse의 위치 정보, 3.0 이상에서만 존재
208	**$EA_INFORMATION**	OS/2 응용 프로그램과 호환성을 위해 사용
224	**$EA**	OS/2 응용 프로그램과 호환성을 위해 사용
256	**$LOGGED_UTILITY_STREAM**	암호화된 속성의 정보와 Key 값 (Windows 2000 이상)

New Technology File System

Attributes

Attr Type Num	Attr Name	Description
16	**$STANDARD_INFORMATION**	최근 접근 시간, 생성 시간, 소유자, 보안 아이디
32	**$ATTRIBUTE_LIST**	속성을 찾을 수 있는 속성 리스트
48	**$FILE_NAME**	파일 이름(유니코드)
64	**$VOLUME_VERSION**	볼륨 정보, 오직 1.2 버전에만 존재 (Windows NT)
64	**$OBJECT_ID**	16바이트로 이루어진 파일, 디렉터리의 고유 값, 3.0 이상에서만 존재
80	**$SECURITY_DESCRIPTOR**	파일의 접근 제어와 보안 속성
96	**$VOLUME_NAME**	볼륨 이름과 관련 정보
112	**$VOLUME_INFORMATION**	파일 시스템의 버전과 여러 FLAG
128	**$DATA**	파일의 내용
144	**$INDEX_ROOT**	인덱스 트리의 루트 노드
160	**$INDEX_ALLOCATION**	인덱스 트리와 연결된 노드
176	**$BITMAP**	할당 정보를 관리하는 속성
192	**$SYMBOLIC_LINK**	심볼릭 링크 정보, 오직 1.2 버전에만 존재 (Windows NT)
192	**$REPARSE_POINT**	심볼릭 링크에서 사용하는 reparse의 위치 정보, 3.0 이상에서만 존재
208	**$EA_INFORMATION**	OS/2 응용 프로그램과 호환성을 위해 사용
224	**$EA**	OS/2 응용 프로그램과 호환성을 위해 사용
256	**$LOGGED_UTILITY_STREAM**	암호화된 속성의 정보와 Key 값 (Windows 2000 이상)

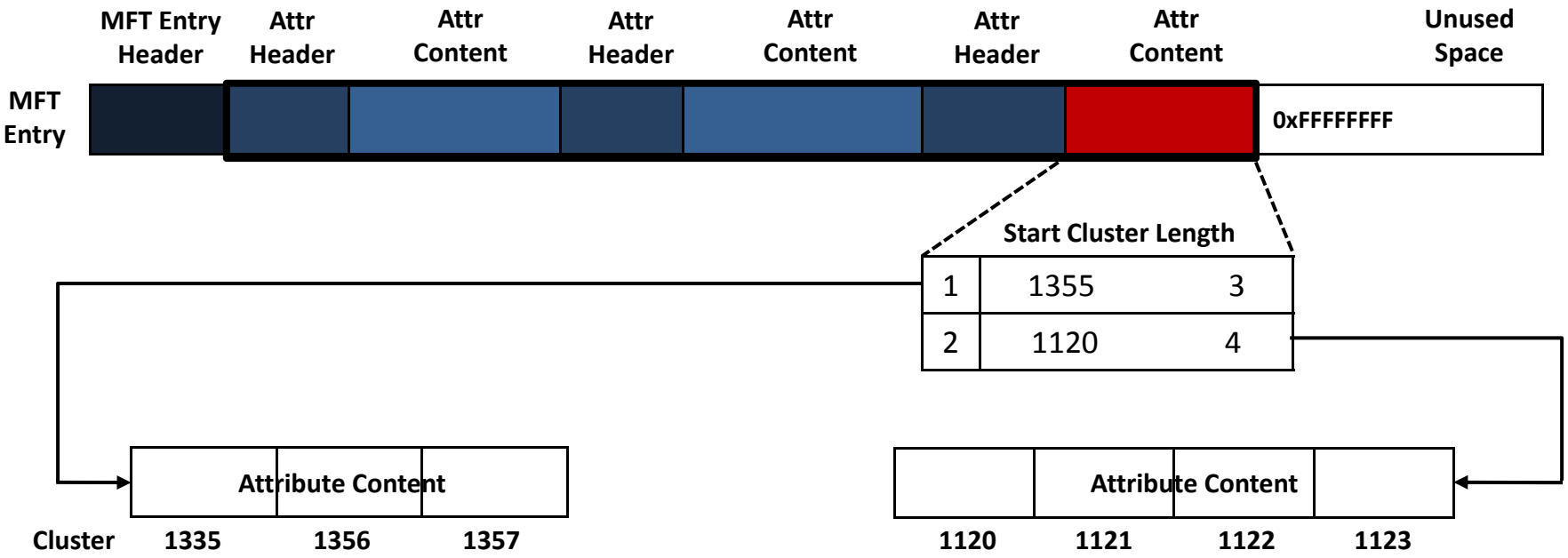
Attributes – Resident & Non-resident



- ✓ **Resident** : $DATA 속성 내용에 데이터가 기록 (700Byte 이하)
- ✓ **Non-resident** : 데이터 700Byte 이상일 경우 별도의 클러스터를 할당하여 저장

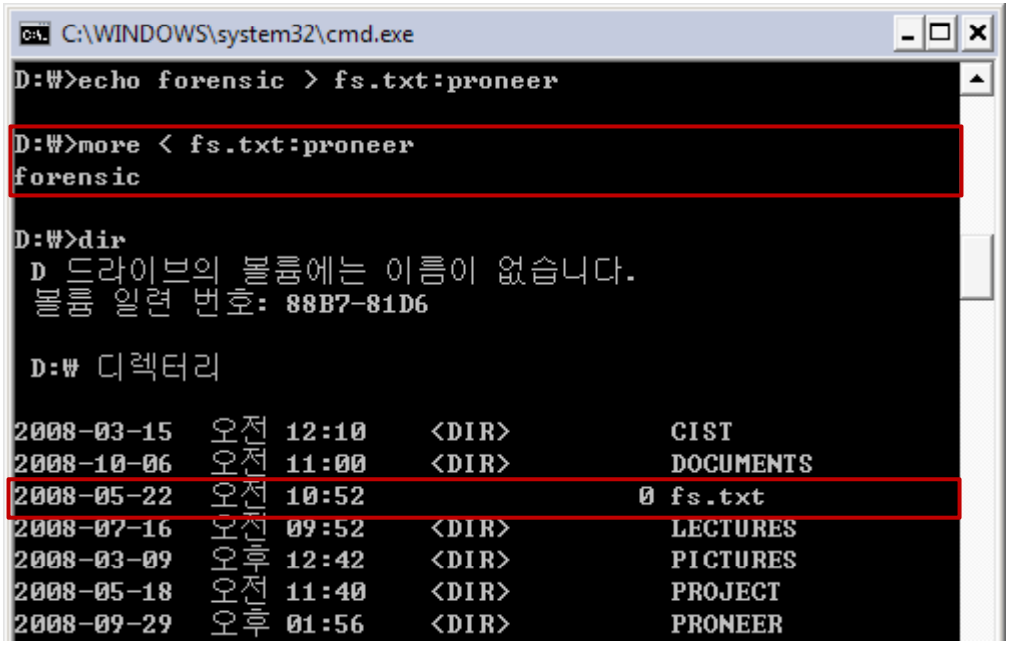
New Technology File System

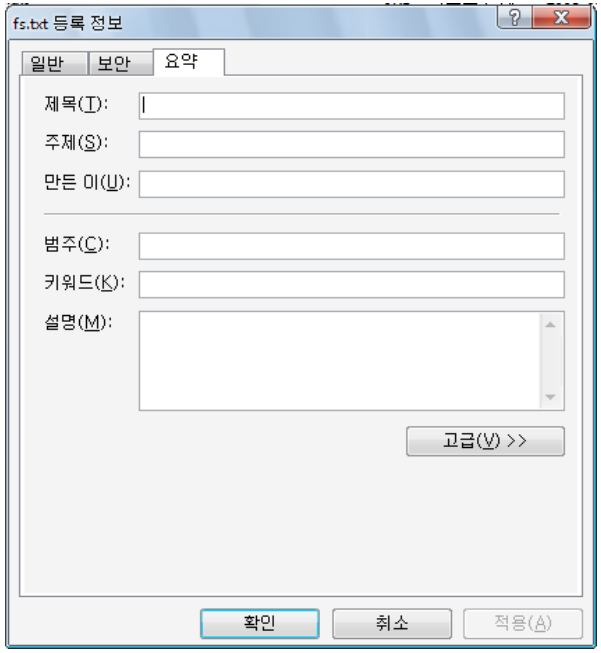
- Attributes – Cluster Runs



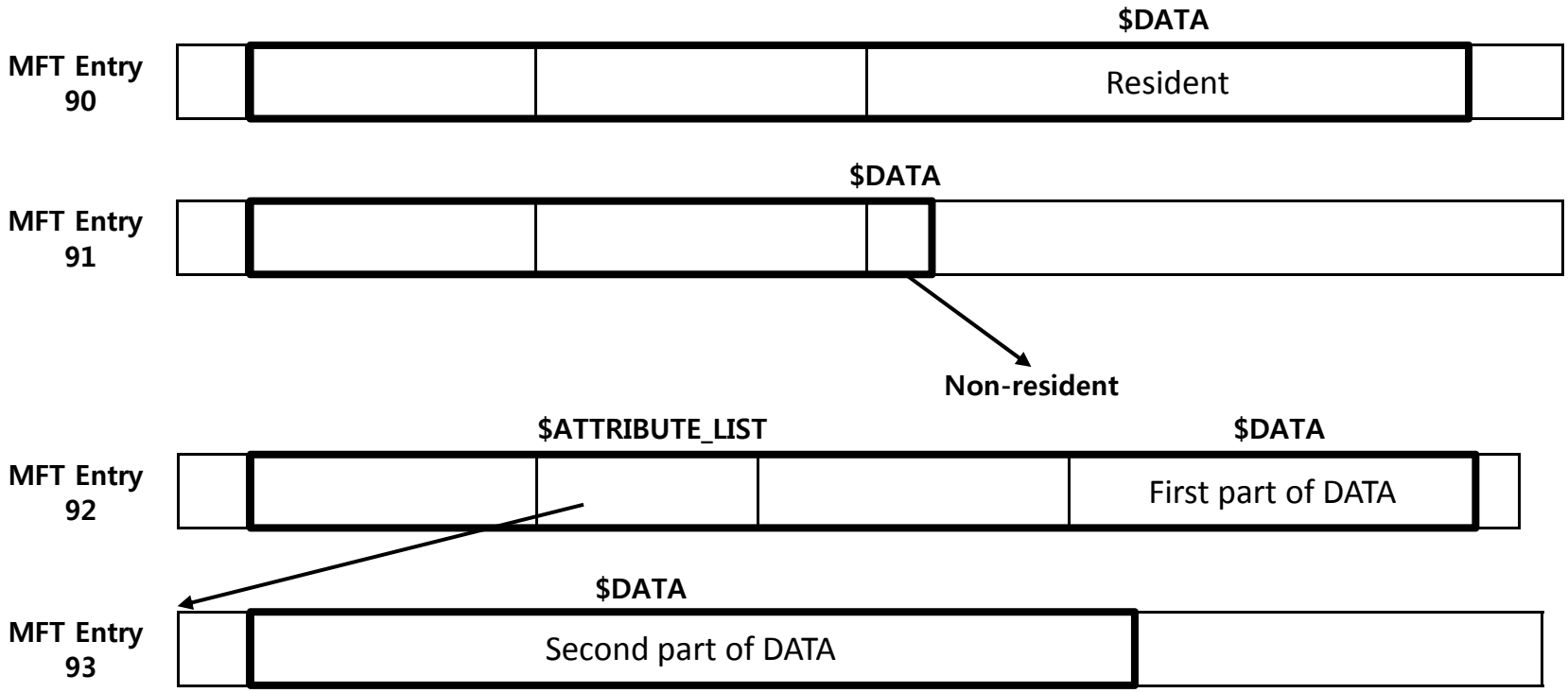
Alternate Data Stream

- ✓ 파일, 디렉터리에 포함되는 추가적인 데이터 속성
- ✓ ADS 속성은 반드시 속성 이름이 있어야 함 (기본적인 $DATA는 없어도 됨)
- ✓ ADS 속성은 파일 크기에 포함되지 않음 (첫 번째 $DATA 속성만 포함)





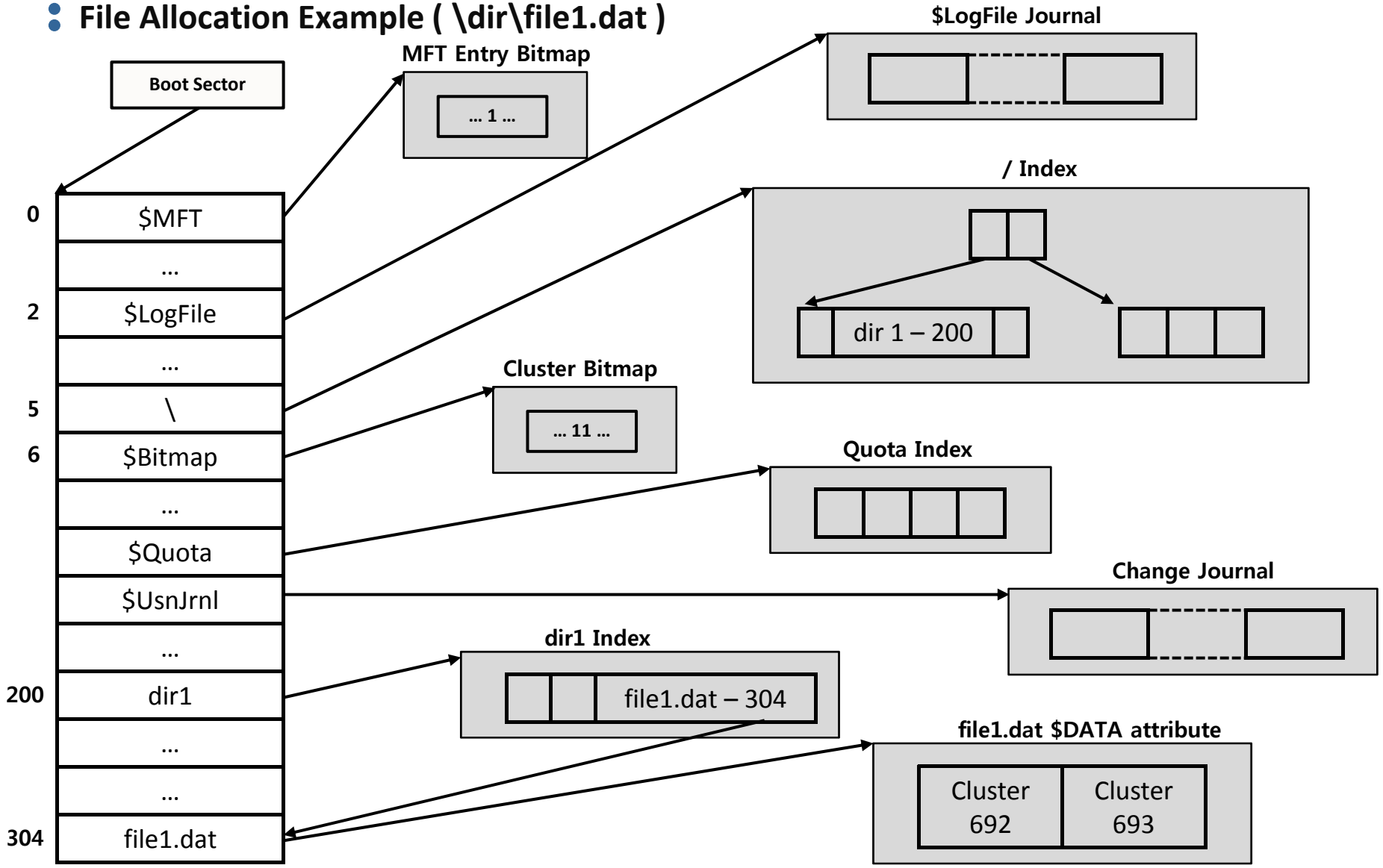
Analysis Consideration



- ✓ Resident MFT Entry가 재할당 될 경우 이전 데이터는 복구 어려움
- ✓ Non-resident MFT Entry가 재할당 될 경우 Application-level에서 파일 복구 가능

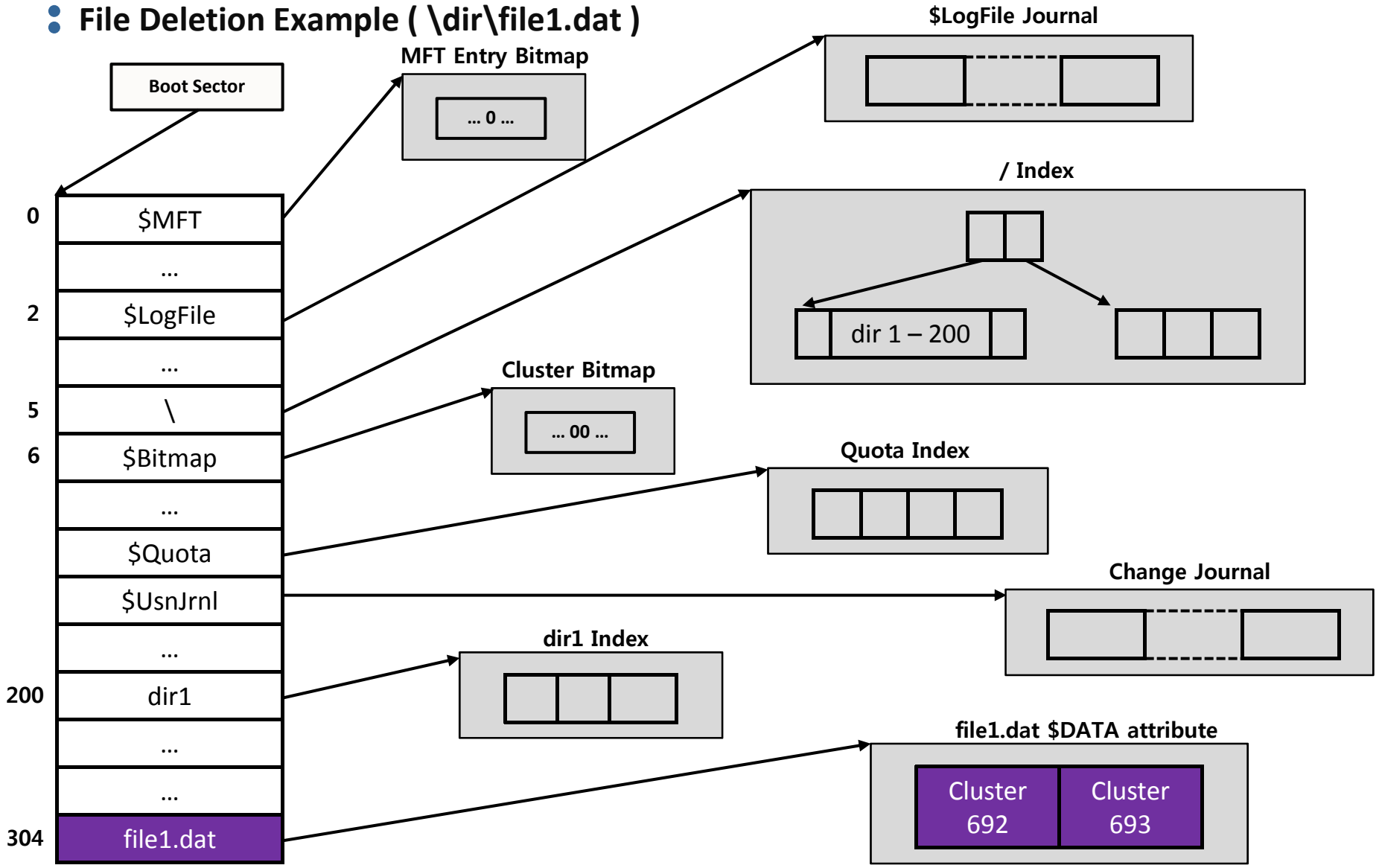
New Technology File System

- File Allocation Example (\dir\file1.dat)



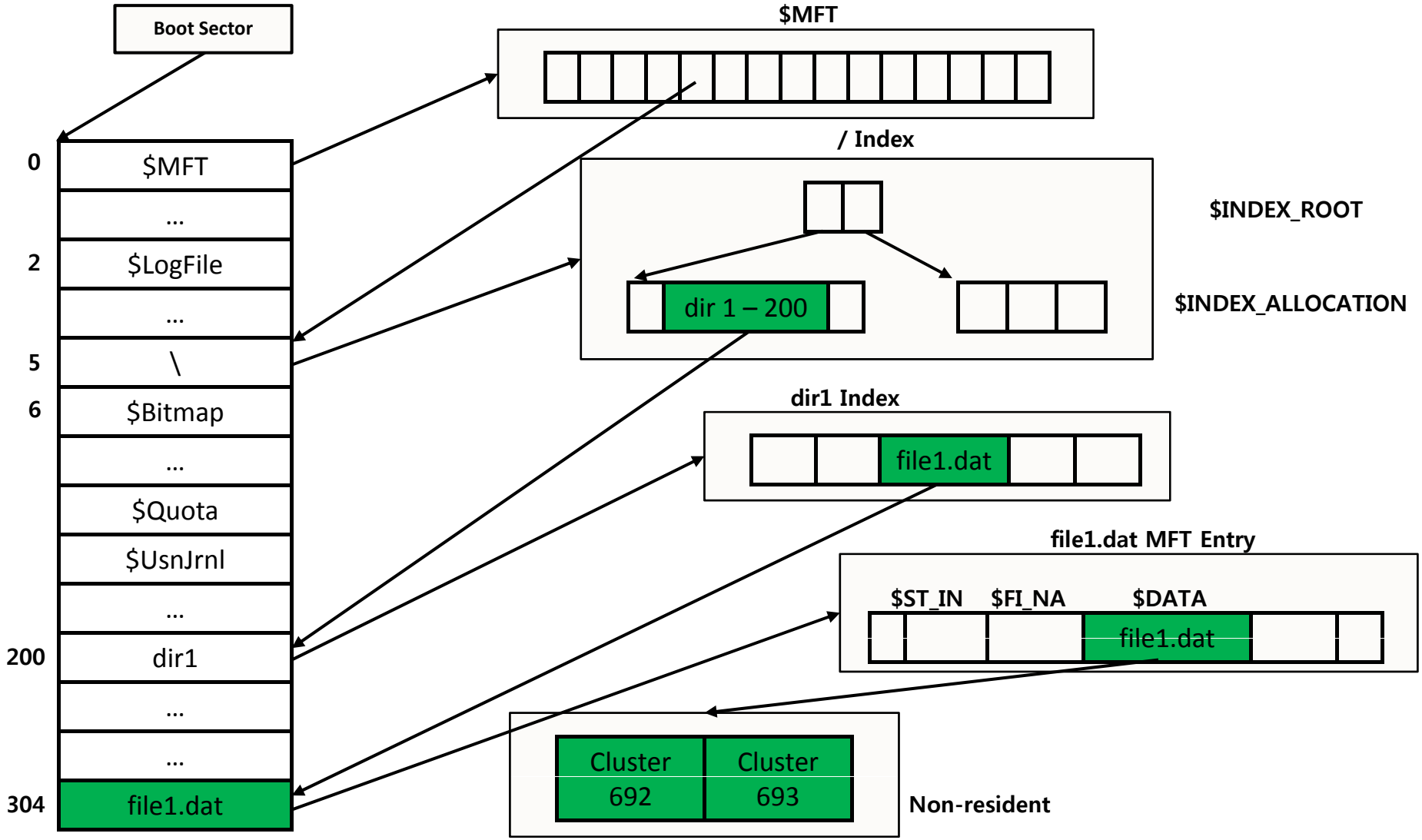
New Technology File System

File Deletion Example (\dir\file1.dat)



New Technology File System

- File Search Example (\dir\file1.dat)



Toward forensically sound file system analysis

- ✓ *MBR*
- ✓ *FAT*
- ✓ *NTFS*

Toward forensically sound file system analysis

Unallocated Clusters Analysis

- ✓ 대용량의 하드디스크 사용으로 많은 양의 공간이 비할당 영역일 가능성
- ✓ 비할당된 클러스터는 이전 데이터가 남아 있을 가능성
  - 포맷하기 이전의 데이터
  - 포맷한 후 할당되었다가 삭제된 데이터
- ✓ 비할당 클러스터 판별
  - FAT : FAT 영역에서 0x00 값을 갖는 클러스터
  - NTFS : 클러스터 비트맵에서 0x00 값을 갖는 클러스터

Toward forensically sound file system analysis

Deleted Files Analysis

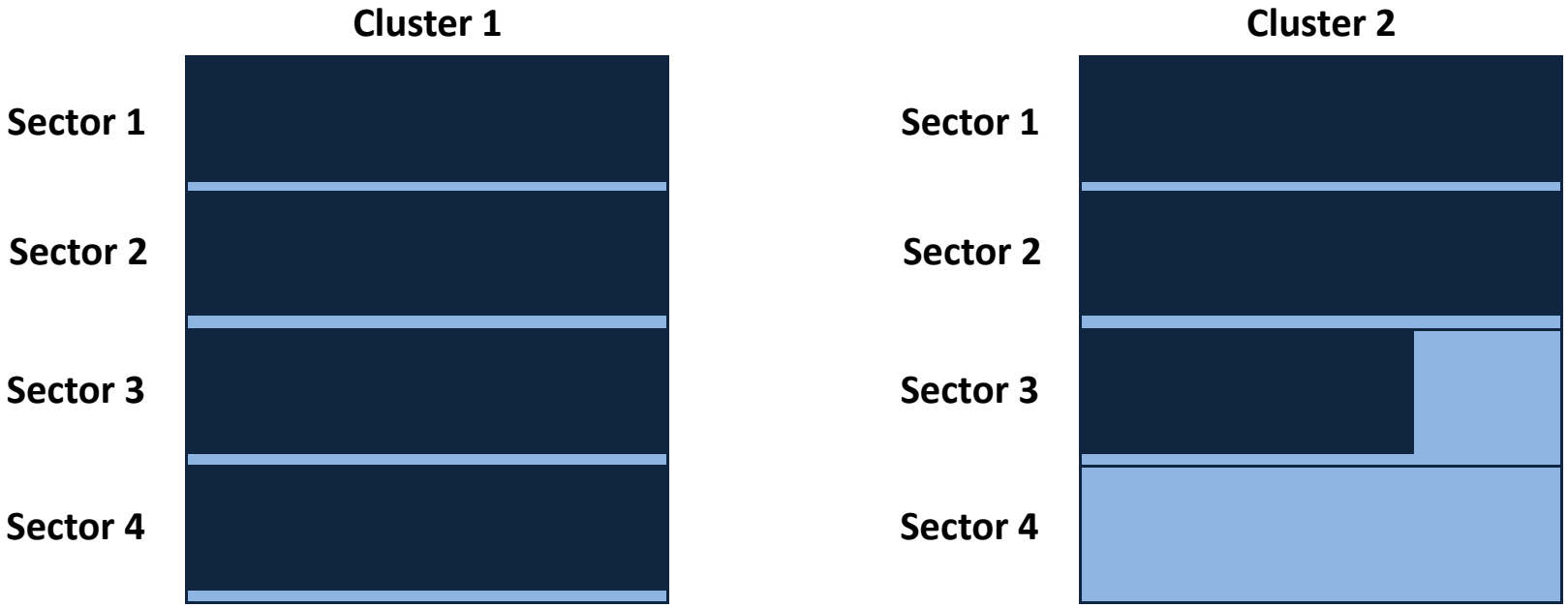
- 삭제한 파일은 다른 파일보다 우선 분석

✓ 삭제된 파일 판별 판별

- FAT : Root Directory부터 삭제된 Directory Entry 검색(value : 0xE5, offset : 0x00)
- NTFS : $MFT 내의 MFT Entry bitmap에서 0x00 값을 가지는 MFT Entry 조사

Toward forensically sound file system analysis

- **Slack Space Analysis**



- ✓ **File Slack** : 이전에 할당되었던 데이터가 남아 있을 가능성
- ✓ **File System Slack, Volume Slack :** 마찬가지
- ✓ 의도적으로 데이터를 은닉할 가능성

Toward forensically sound file system analysis

Timestamp Analysis

- ✓ 사건이 발생한 시점을 중심으로 데이터 분석
- ✓ 시간의 흐름 파악이 중요
- ✓ 시간의 역전 및 의도적인 조작이 발생했는지 파악

- ✓ 시간 정보 위치
  - ✓ FAT : 해당 파일, 디렉터리의 Directory Entry
    (create time, last written time, created date,
    last accessed date, last written date)
  - ✓ NTFS : 해당 파일의 속성
    ($STANDARD_INFORMATION, $FILE_NAME)

Toward forensically sound file system analysis

Signature Analysis

- ✓ 파일 시그니처와 확장자가 일치하는지 검사
- ✓ 확장자 변경을 통해 의도적으로 파일을 은폐할 가능성
- ✓ 확장자 위치
  - ✓ FAT : 해당 파일의 Directory Entry
  - ✓ NTFS : 해당 파일의 $FILE_NAME 속성

Toward forensically sound file system analysis

- ✓ ***MBR***
- ✓ *FAT*
- ✓ *NTFS*

Toward forensically sound file system analysis

Boot Code Analysis

- ✓ 사용자에 의해 의도적인 boot code 수정
- ✓ 실제 부트 코드를 특정 위치에 백업
- ✓ 의도한 작업 실행 후 실제 부트 코드가 실행되도록 수정

- ✓ 방지 대책
  - ✓ boot code 분석을 통해 오프셋 0x00의 값이 0x80을 갖는 파티션의 시작 위치로 점프하는지 판별

Toward forensically sound file system analysis

Partition Table Analysis

- ✓ 각 파티션의 크기의 합과 전체 디스크 볼륨의 차이를 계산
- ✓ Volume Slack이 존재할 가능성
- ✓ Volume Slack이 존재할 경우 해당 영역을 대상으로 실행 코드나 실행파일 검색

Toward forensically sound file system analysis

- ✓ *MBR*
- ✓ ***FAT***
- ✓ *NTFS*

Toward forensically sound file system analysis

Wasted Area Analysis

- ✓ MBR과 Reserved Area 간의 낭비되는 공간 조사
- ✓ Reserved Area 내의 낭비되는 섹터 (0,1,2,6,7,8 섹터 제외) 조사
- ✓ Reserved Area 내의 bootstrap code 영역 조사 (2, 8 섹터)
- ✓ FSINFO의 Not Used 영역 조사

Toward forensically sound file system analysis

- ✓ *MBR*
- ✓ *FAT*
- ✓ ***NTFS***

Toward forensically sound file system analysis

Wasted Area Analysis

- ✓ MBR과 Volume Boot Record 간의 낭비되는 공간 조사
- ✓ Volume Boot Record에서 boot sector를 제외한 bootstrap code 영역 조사
- ✓ MFT Entry 12-15번 영역 조사

Toward forensically sound file system analysis

$BadClus Entry Analysis

- ✓ $BadClus Entry(8)는 배드 섹터가 포함된 클러스터를 관리
- ✓ 정상적인 클러스터를 $BadClus에 등록 후 의도한 데이터 저장
- ✓ $BadClus에 등록된 클러스터는 할당된 클러스터로 표시

Toward forensically sound file system analysis

$Boot Entry Analysis

- ✓ $Boot는 보통 boot sector의 내용을 저장
- ✓ $Boot의 데이터크기는 무제한
- ✓ $Boot의 $DATA 의 크기를 늘려 데이터 숨김

- ✓ 탐지 방안
  - boot sector의 내용과 $Boot의 $DATA 내용을 비교

Toward forensically sound file system analysis

- **$LogFile Entry Analysis**

  - ✓ $LogFile은 메타데이터의 트랜잭션 정보의 로그 저장
  - ✓ $LogFile을 통해 최근 사용자가 수행한 작업을 재 구성

Toward forensically sound file system analysis

Encrypting File System(EFS) Analysis

- ✓ 암호화된 파일은 다른 파일보다 우선 분석
- ✓ 사용자의 private key는 login password에 의해 암호화되어 registry 저장,
- ✓ login password (brute force)
- ✓ 암호화 작업 동안 임시 plaintext 파일 생성(EFS0.TMP) 후 삭제 ➔ 복구 가능
- ✓ 암호화 설정 확인
  - 해당 파일의 $STANDARD_INFORMATION 속성

Toward forensically sound file system analysis

Alternate Data Stream Analysis

- ✓ ADS(Alternate Data Stream) 파일은 Explorer를 통해 확인 불가
- ✓ 의도적으로 숨기기 위해 활용할 가능성

- ✓ ADS 파일 조사 방안
  - 전체 MFT Entry를 대상으로 $DATA 속성이 두 개 이상인 파일을 검색

Conclusion

- **This week**
  - ~~✓ Week 1 : Hardware and File system Analysis (Chapter 1, 2)~~
  - ✓ Week 2 : Acquiring Digital Evidence (Chapter 4)
  - ✓ Week 3 : EnCase Concepts and Environment (Chapter 5, 6)
  - ✓ Week 4 : Actual Test
  - ✓ Week 5 : Actual Test
  - ✓ Week 6 : Actual Test
  - ✓ Week 7 : Actual Test
  - ✓ ..
  - ✓ PS : EnScripting

Conclusion

- **Next week**
  - ~~**Week 1 : Hardware and File system Analysis (Chapter 1, 2)**~~
  - **Week 2 : Acquiring Digital Evidence (Chapter 4)**
    - Booting a Computer Using the EnCase Boot Disk
    - Drive-to-Drive DOS Acquisition
    - Network Acquisitions
    - FastBloc Acquisitions
    - FastBloc SE Acquisitions
    - LinEn Acquisitions
    - Enterprise and FIM Acquisitions

Question and Answer