

8장 파일시스템과 파일 복구 (파일시스템 개론)

박 종 혁 교수

UCS Lab

Tel: 02-970-6702

Email: jhpark1@seoultech.ac.kr

- 학습 목표

- 디지털 포렌식에서 기본 단위는 파일의 추출과 분석을 통해 이루어 지므로, 파일의 저장 및 관리를 책임지는 파일시스템의 이해는 필수적이다.
- 먼저 파일시스템의 이해와 구조를 파악하고, 윈도우 시스템에서 가장 널리 쓰이는 FAT, NTFS 파일시스템에 대해 학습한다.

- 학습 내용

- 파일시스템의 이해
- 파일시스템 분석
- FAT 파일 시스템
- NTFS 파일 시스템

목 차

1. 파일 시스템의 이해

1. 파일 시스템의 종류
2. 파일 시스템의 구조
3. 주소 지정방식
4. 클러스터
5. 슬랙 공간

2. 파티션과 MBR

1. 파티션
2. MBR
3. 확장 파티션

3. FAT 파일 시스템

1. FAT 파일 시스템 소개
2. 예약 영역
3. FAT 영역
4. 데이터 영역
5. 파일의 할당, 삭제로 인한 변화 요소
6. The function of FAT

4. NTFS

1. NTFS 소개
2. NTFS 구조
3. VBR(Volume Boot Record)
4. MFT(Master File Table)
5. 데이터 영역
6. 파일의 할당, 삭제로 인한 변화 요소

5. 디지털 포렌식 관점에서의 파일 시스템 분석

6. 파일 복구

1. 파일 시스템상의 파일 복구
2. 파일 카빙

파일 시스템의 이해

- 파일시스템이란?

- 디지털 데이터를 효과적으로 관리하기 위해 파일을 체계적으로 기록하는 방식
- 사용자에게 파일과 디렉터리를 계층 구조로 데이터를 저장하도록 하는 메커니즘
- 파일이 어디에 저장되어 있는지 조직화하고, 사용자의 데이터를 구조적으로 정의

- 파일 시스템이 필요한 이유?

- 저장 매체의 용량이 증가함에 따라 저장되는 파일의 수도 급격히 증가
- 원하는 파일을 읽고 쓰는 기본적인 기능부터 데이터를 검색, 저장, 관리하기 위한 규약이 필요

파일 시스템의 이해 - 파일 시스템의 종류

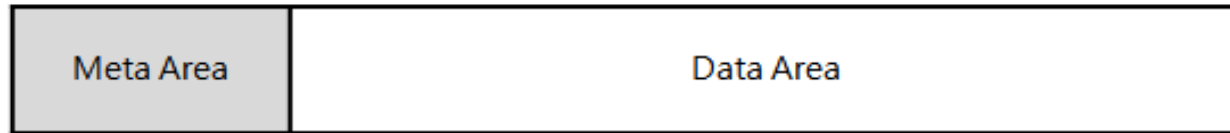
저장매체	운영체제	파일 시스템
디스크장치	Windows	FAT(FAT12,FAT16,FAT32,exFAT), NTFS
	Linux	Ext2, Ext3, Ext4
	Unix-like	UFS
	OS2	HPFS
	Mac OS	HFS, HFS+
	Solaris	ZFS
	HP-UX	ODS-5, VxFS
광학장치		ISO 9660, UDF

파일 시스템의 이해 - 파일 시스템의 구조

• 파일시스템의 구조

- 파일시스템의 기본적인 동작은 운영체제가 각 파일을 사용하기 위해 저장되어 있는 위치로 접근하여 해당 데이터를 읽도록 함
- 또한 데이터의 위치를 파악하기 위하여 사용자가 저장된 파일의 목록을 확인할 수 있도록 지원 함

• 파일시스템의 추상화 구조

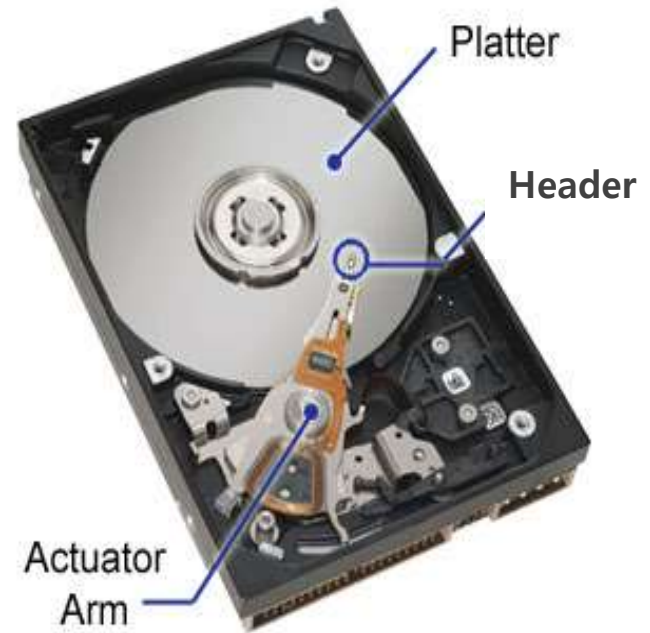


- 사용자가 생성한 파일의 내용은 데이터 영역에 기록
- 메타 영역에는 파일 관리를 위한 파일의 이름, 위치, 크기, 시간 정보 등이 기록
- 파일 시스템은 이러한 메타 정보를 유지 관리함으로써 파일을 효과적으로 관리

파일 시스템의 이해 – 하드 디스크

• Hard disk

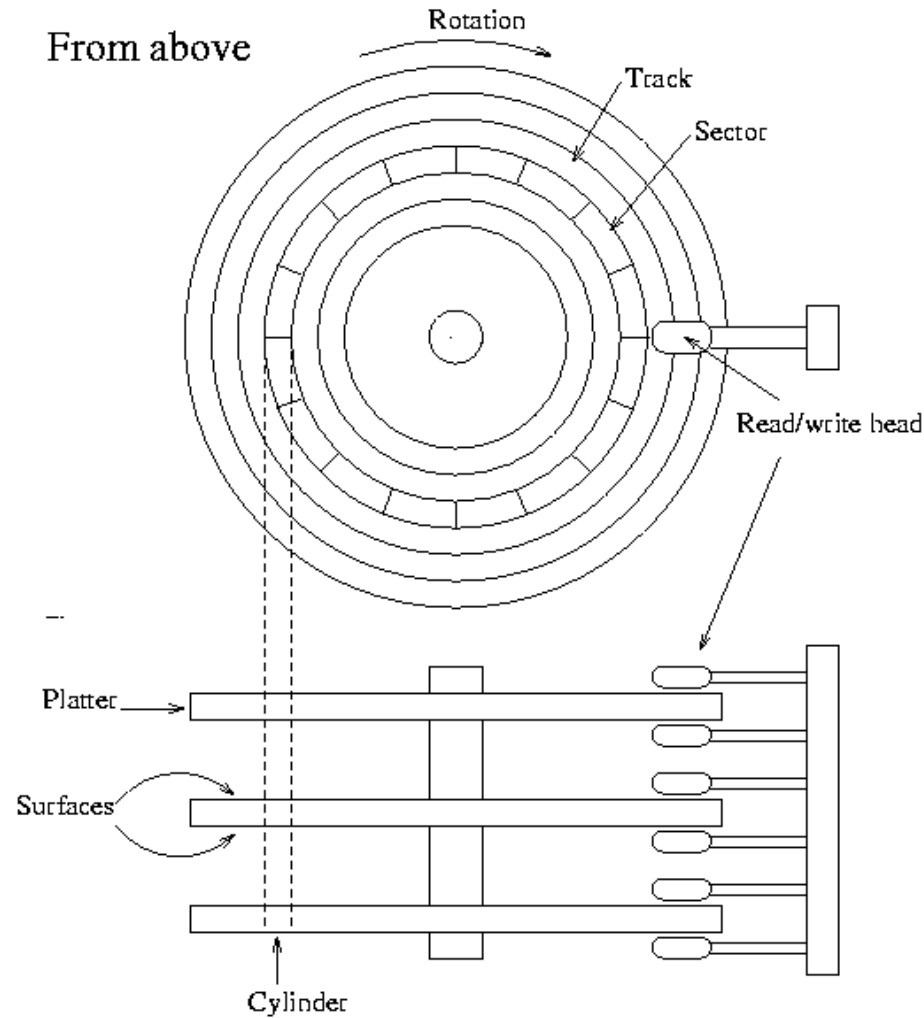
- ✓ 가장 널리 사용되는 비-휘발성 저장 장치
- ✓ Interface or Controller
 - IDE (Integrated Drive Electronics) Controller
 - SCSI (Small Computer System Interface)
 - SATA (Serial Advanced Technology Attachment) Controller
 - RAID (Redundant Array of Inexpensive Disks)



파일 시스템의 이해 - 주소 지정 방식

- Hard disk의 구성

- ✓ Addressing



파일 시스템의 이해 - 주소 지정 방식

- Hard disk

- ✓ Sector

- 데이터 기록의 가장 기본 단위
- 총 571 bytes에서 섹터의 위치를 구분하기 위한 고유 번호 저장 (59 bytes)
- 실제 데이터 저장으로 사용되는 영역은 512 bytes (Sector size)

파일 시스템의 이해 - 주소 지정 방식

• Addressing (주소 지정 방식)

✓ 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 지원이 가능하게 되었지만 대용량 디스크는 지원하지 못함
- ATA-6부터 표준에서 제외, **LBA 방식**이 새롭게 대두

파일 시스템의 이해 - 주소 지정 방식

• Addressing (주소 지정 방식)

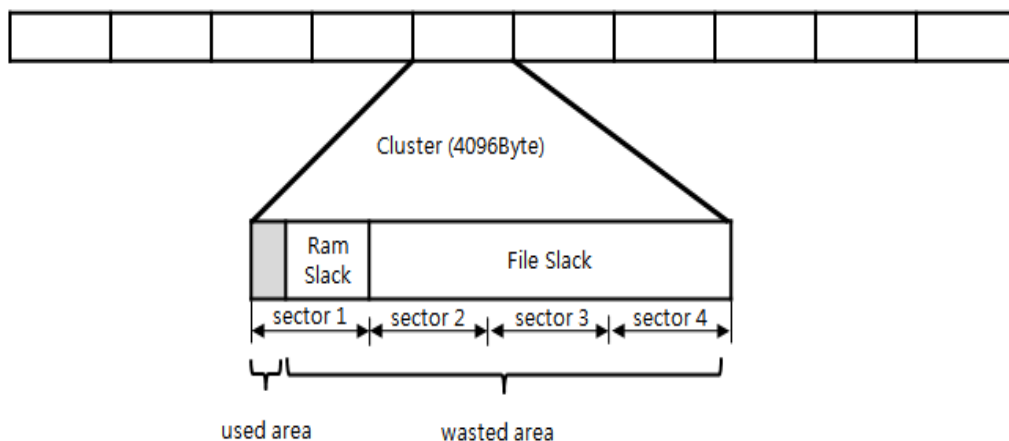
✓ LBA (Logical Block Addressing) 방식

- 디스크의 0번 실린더, 0번 헤드, 1번 섹터를 첫 번째(0번) 블록으로 지정
- 디스크의 마지막 섹터까지 순차적으로 주소를 지정
- 물리적인 구조에 대한 정보 불필요(섹터의 번호만으로 접근 가능)
- 선형적인 섹터 번호가 실제 디스크의 물리적 구조로 변환 되어야 하지만 ROM BIOS에 의해 자동적으로 수행됨
- 초기에는 28bit로 처리하여 약 127GB가 최대 용량이었음
- 현재는 48bit 어드레스 방식을 사용하고 있음

파일 시스템의 이해 - 클러스터

Cluster (클러스터)

- ✓ 클러스터 = 여러 개의 섹터(하드디스크의 물리적 최소 단위)를 묶은 단위
- ✓ 섹터단위로 입출력 처리하면 시간이 오래 걸리므로 여러 개의 섹터를 묶어 한번에 처리

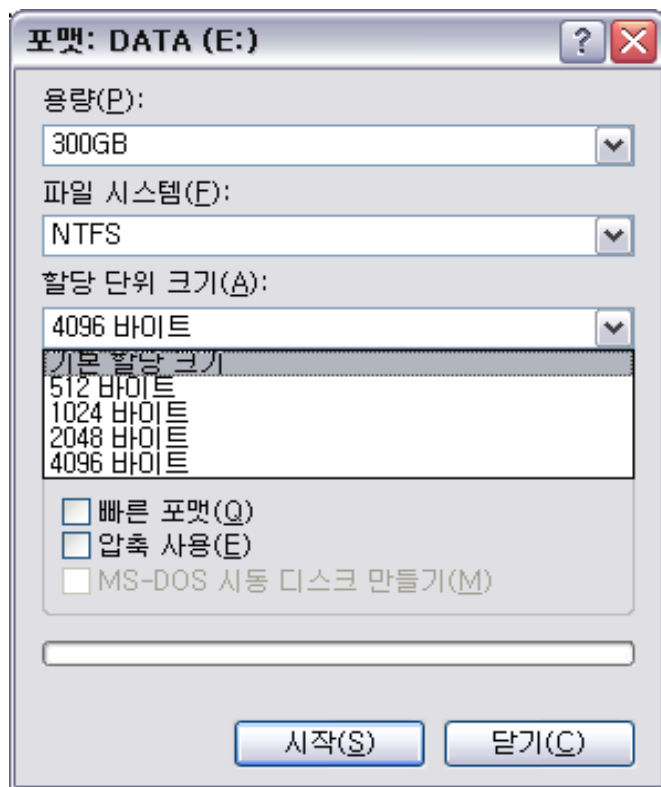


- ✓ 클러스터 크기를 4,096바이트(4KB)로 지정 했을 때, 100바이트의 데이터를 저장하는 경우로 클러스터의 크기만큼 할당됨
 - 3996바이트가 낭비되지만 그럼에도 불구하고 디스크 입출력 횟수를 줄이기 위해 클러스터 단위를 사용
 - 4MB(4,096KB)파일 저장할 때 4KB크기의 클러스터 사용 = 1,024번 입출력 수행
 - 4MB(4,096KB = 4,194,304B)파일 저장할 때 512B크기의 클러스터 사용 = 8,192번 입출력 수행

파일 시스템의 이해 - 클러스터

• Cluster (클러스터)

- ✓ 윈도우 시스템에서 디스크 포맷할 때 클러스터 크기 지정



FAT32에서의 클러스터 크기

볼륨 크기	클러스터 크기
32MB - 8GB	4KB
8GB - 16GB	8KB
16GB - 32GB	16KB
32GB	32KB

NTFS에서의 클러스터 크기

볼륨 크기	클러스터 크기
512MB 이하	512Byte
513MB - 1GB	1KB
1GB - 2GB	2KB
2GB 이상	4KB

파일 시스템의 이해 - 슬랙 공간

• Slack Space

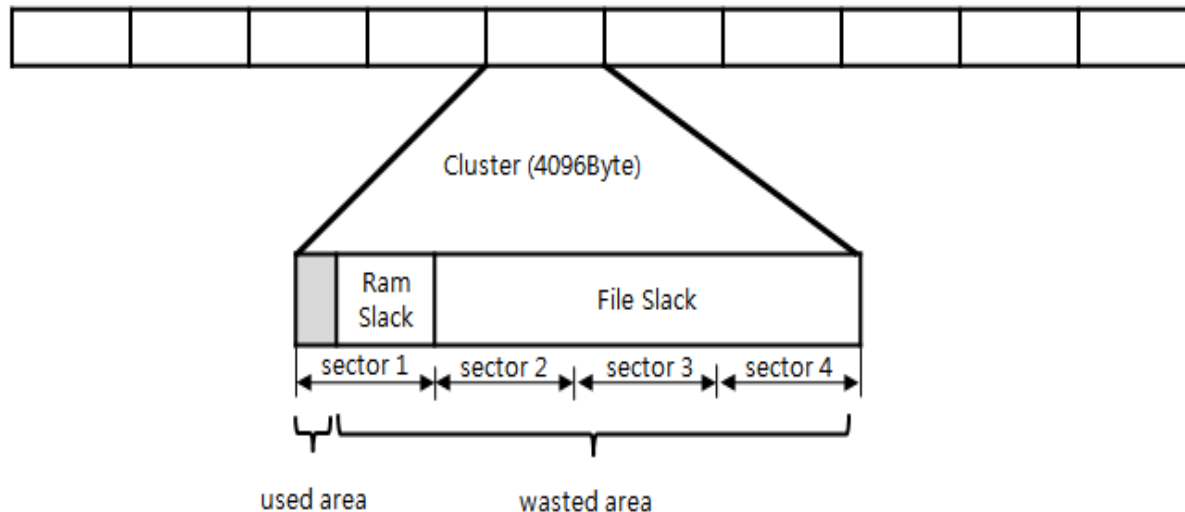
- ✓ 물리적인 구조와 논리적인 구조의 차이로 발생하는 낭비 공간
- ✓ 물리적으로 파일에 할당된 공간이지만 논리적으로 사용 할 수 없는 낭비 공간
- ✓ 디지털 포렌식 관점에서 - 정보 은닉 가능성, 파일 복구와 삭제된 파일의 파편 조사
 - RAM Slack (Sector Slack)
 - File Slack (Drive Slack)
 - 이전에 사용한 데이터가 존재, 흔적 조사에 활용
 - File System Slack
 - Volume Slack

파일 시스템의 이해 - 슬랙 공간

Slack Space (RAM Slack & File Slack)

✓ RAM Slack (Sector Slack)

- 램에 저장 되어있는 데이터가 디스크에 저장될 때 512 바이트씩 기록되는 특성 때문에 발생하는 공간으로 섹터 슬랙(Sector Slack)이라고도 함
- 지정되는 파일 크기가 512 바이트의 배수가 아닐 경우 발생
- 여분 바이트 0x00 값으로 기록
- 램 슬랙을 이용하면 파일의 끝을 알 수 있기 때문에 삭제된 파일 복구 시 유용하게 사용

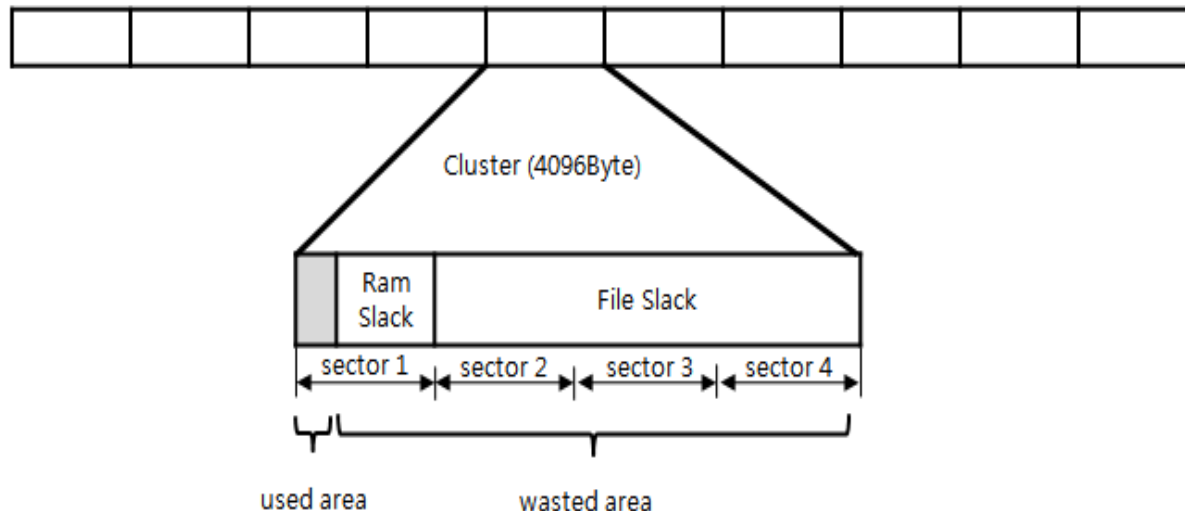


파일 시스템의 이해 - 슬랙 공간

Slack Space (RAM Slack & File Slack)

✓ File Slack (Drive Slack)

- 클러스터의 사용으로 인해 낭비되는 공간 중 램 슬랙을 제외한 부분으로 드라이브 슬랙(Drive Slack)이라고도 함
- 파일 슬랙을 이용하면 특정 파일이 해당 저장 매체에 존재하였는지 규명 가능
 - 존재 여부를 알아야 할 파일을 클러스터 단위로 나눈 후, 각 클러스터의 마지막 부분과 파일 슬랙 중 일치하는 부분이 있는지 확인
- 최하단의 디스크 입출력은 섹터 단위로 진행되므로 0x00으로 기록되는 램 슬랙과 다르게 이전의 데이터가 그대로 남아있음(I/O는 섹터단위로 진행)



파일 시스템의 이해 - 슬랙 공간

⋮

Sector 1(512 byte)

Hex dump of Sector 1 (512 bytes) showing data and code. A green box highlights the first 480 bytes, and a red box highlights the last 32 bytes, labeled "RAM slack".

```
0375 74 49 74 65 6D 43 6F 75 6E 74 20 47 65 74 49 74 65 6D 43 6F 75 6E 74 20 28
0400 29 20 69 66 6D 5F 76 69 65 77 54 61 62 20 28 20 28 20 28 29 20 20 0D 0A 09
0425 20 31 20 3E 20 20 20 0D 0A 09 20 21 20 20 20 3D 21 3D 20 3D 3D 21 3D 20 30
0450 20 30 20 20 20 20 0D 0A 09 20 29 20 20 20 0D 0A 09 20 7D 20 7B 7D 20 7B 7D 20
0475 0A 0D 0A 09 20 0A 09 0D 0A 09 20 29 3B 20 29 3B 20 0A 20 2F 23 20 2F 2F 23
0500 20 0A 0D 0A 09 20 2F 20 2F 2F 35 00 09 20 2F 20 2F 2F 20 00 FF FF FF FF 82
0525 79 47 11 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0550 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0575 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0600 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

```
tItemCount GetItemCount (
) ifm_viewTab ( ( ()
1 >      !   != == != 0
0      )      } {} {}
      ); );  /# // #
      / // s / // -YYYYY,
yG.....
.....
.....
```

RAM slack

⋮

Sector 2(512 byte)

Hex dump of Sector 8 (512 bytes) showing data and code. A green box highlights the first 480 bytes, and a light green box highlights the last 32 bytes, labeled "File slack".

```
3850 70 EF 68 C6 CA 01 28 66 83 F4 68 C6 CA 01 AF 4C 4C 9C 3F D4 CA 01 C1 AC B8
3875 EF 68 C6 CA 01 00 E0 D8 00 00 00 00 00 03 D6 D8 00 00 00 00 20 00 00 00
3900 00 00 00 00 08 03 E4 C2 89 D5 0C D3 7C C7 2E 00 7A 00 69 00 70 00 00 00
3925 00 00 00 00 00 00 00 00 00 00 10 00 00 00 02 00 00 00 FF FF FF FF 82 79
3950 47 11 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
3975 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
4000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
4025 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
4050 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
4075 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 32 00
```

```
pïhEÊ·(ffôhEÊ·~LLœ?ÔÊ·Á-,
ihEÊ--àø-----Öø-----
-----äÅ%Ö·Ó|Ç·z·i·p····
-----·YYYYY,y
B-----
-----
-----
```

File slack

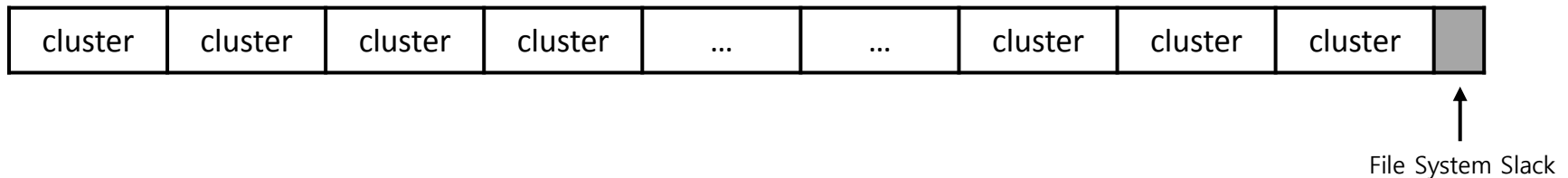
Sector 8(512 byte)

파일 시스템의 이해 - 슬랙 공간

• Slack Space (File System Slack & Volume Slack)

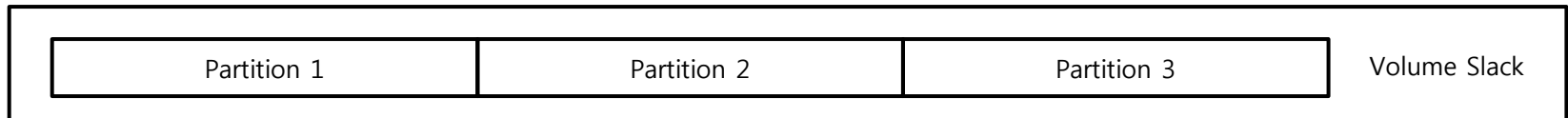
✓ File System Slack

- 파일시스템 할당 크기와 볼륨 크기간의 차이로 인해 발생하는 공간
 - 1,026KB 볼륨에 4KB 클러스터 사용하는 파일 시스템 구성하면 마지막 2KB이 파일 시스템 슬랙이 됨



✓ Volume Slack

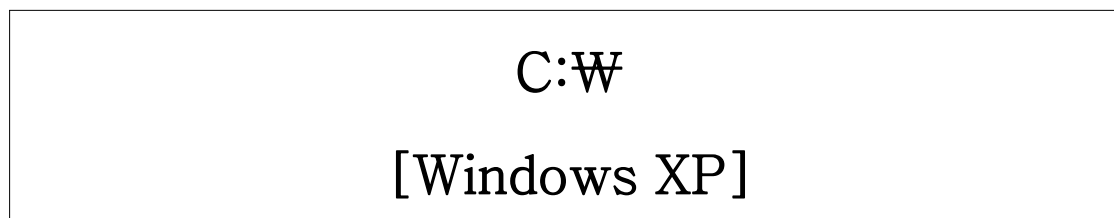
- 전체 볼륨 크기와 할당된 파티션 크기의 차이로 인해 발생하는 공간



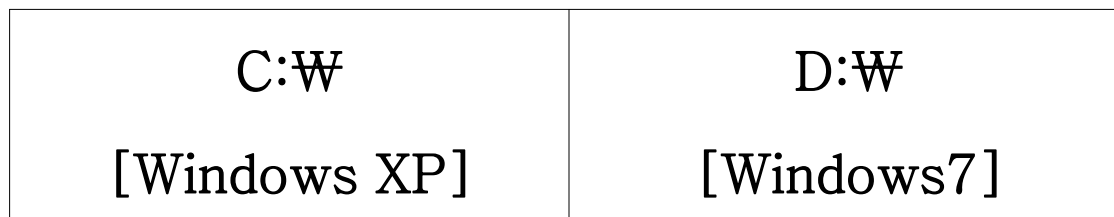
파티션과 MBR – 파티션

- 파티션(Partition)

- 저장매체의 저장 공간을 논리적으로 분할한 것
- 시스템은 부팅 과정에서 파티션의 크기, 위치, 설치된 운영체제 등을 파악하여 그에 맞게 구동해야 함
- 그러한 정보 담고 있는 부분이 Boot Record(BR) 영역, windows는 파티션의 첫 번째 섹터에 위치



단일 파티션



다중 파티션

파티션과 MBR – MBR(Master Boot Record)

- **MBR (Master Boot Record)**

- 분할된 파티션에서 각 파티션의 BR영역을 관리하는 영역
- MBR은 저장매체의 첫 번째 섹터(LBA 0)에 위치하는 512 바이트 크기의 영역
- 446 바이트의 부트 코드(Boot Code) 영역, 64 바이트의 파티션 테이블 (Partition Table) 영역, 2 바이트의 시그니처(Signature) 영역

BR	C:W [Windows XP]
----	---------------------

단일 파티션에서의 Boot Record

MBR	BR	C:W [Windows XP]	BR	D:W [Windows7]
-----	----	---------------------	----	-------------------

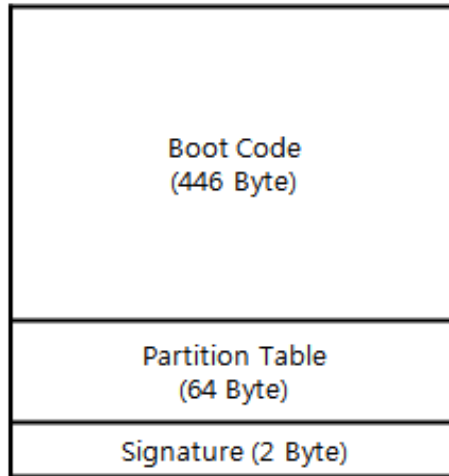
단wnd 파티션에서의 Boot Record

파티션과 MBR – MBR(Master Boot Record)

- 부트 코드 영역

- 컴퓨터가 부팅될 때 수행되는 코드로, 파티션 테이블에서 부팅 가능한 파티션을 찾아 해당 파티션의 부트 섹터(Boot Sector)를 호출하는 역할을 수행

MBR 데이터 구조



범위(Byte Range)		설명
10진수	16진수	
0 - 445	0x0000 - 0x01BD	Boot code
446 - 461	0x01BE - 0x01CD	Partition table entry #1
462 - 477	0x01CE - 0x01DD	Partition table entry #2
478 - 493	0x01DE - 0x01ED	Partition table entry #3
494 - 509	0x01EE - 0x01FD	Partition table entry #4
510 - 511	0x01FE - 0x01FF	Signature (0x55AA)

파티션과 MBR – MBR(Master Boot Record)

• 파티션 테이블 영역

- 16 바이트씩 총 4개의 파티션 정보가 저장
- 첫 번째 값인 부트 플래그(Bootable Flag)는 해당 파티션이 부팅 가능한 파티션인지를 나타내며, 부팅 가능한 파티션일 경우 해당 부트 플래그의 값이 0x80
- MBR의 부트 코드는 파티션 테이블을 검색하여 부트 플래그 값이 0x80 값을 갖는 파티션의 부트 섹터 위치로 점프하는 역할을 수행
- 포렌식 조사 시, 보이는 파티션 영역 및 디스크 전 영역을 조사, 필요에 따라 MBR영역 직접 해석
 - MBR영역과 실제 파일 시스템이 시작하는 영역 사이에 악의적 코드 삽입하여 운영 체제 시작 전에 동작하도록 구성된 악성 코드 존재

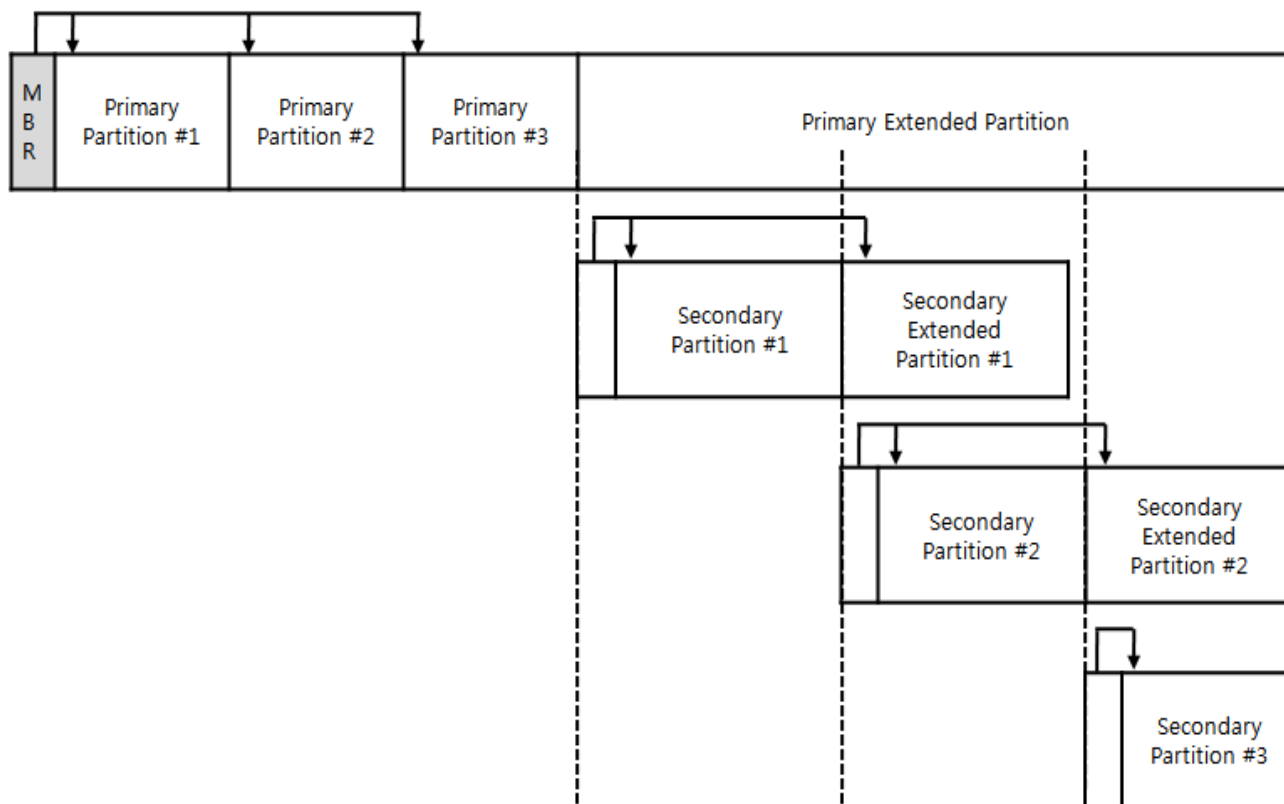
파티션 테이블 데이터 구조

범위(Byte Range)		설명
10진수	16진수	
0 - 0	0x0000 - 0x0000	Bootable flag
1 - 3	0x0001 - 0x0003	CHS 주소 방식의 시작 위치
4 - 4	0x0004 - 0x0004	Partition 유형
5 - 7	0x0005 - 0x0007	CHS 주소 방식의 끝 위치
8 - 11	0x0008 - 0x000B	LBA 주소 방식의 시작 위치
12 - 15	0x000C - 0x000F	총 섹터 개수

파티션과 MBR - 확장 파티션

• 확장 파티션(Extended Partition)

- MBR영역에서 파티션 정보를 표현하는 공간은 64바이트로 총 4개까지만 파티션을 분할 할 수 밖에 없는 한계를 극복하기 위해 나온 개념
- 마지막 4번째 파티션 테이블이 가리키는 위치가 또 다른 MBR 영역을 가리켜 추가로 4개의 파티션을 담을 수 있도록 하는 구조



FAT 파일 시스템 - FAT 파일 시스템 소개

• FAT (File Allocation Table) History

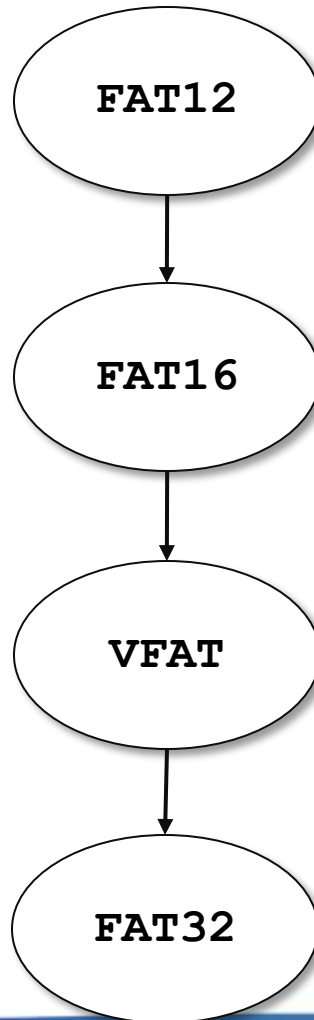
MS-DOS 파일 시스템에서 파일의 위치정보를 기록한 테이블을 지칭

FAT 뒤의 숫자는 파일시스템에서 관리하는 클러스터의 개수를 의미

FAT 형식	최대 표현 가능한 클러스터
FAT12	4,084
FAT16	65,524
FAT32	67,092,481

FAT12의 경우 12비트를 사용하여 클러스터 위치를 표현

: $2^{12} = 4,096$ 개의 클러스터를 표현, 하지만 미리 예약된 12개의 클러스터가 존재하기 때문에 최대 4,084개의 클러스터를 표현



1980년대 초 (QDOS)
플로피디스크용으로 처음 개발

1980년대 말
하드디스크를 지원하기 위해 개발

1995년
FAT의 성능 향상, 긴 파일 이름이 가능해짐

1996년
VFAT을 확장, 고용량 하드디스크 지원

FAT 파일 시스템 – FAT 파일 시스템 소개

• FAT12

- ✓ 5.25인치 플로피디스크에 파일을 저장하기 위해 개발
- ✓ 1983년 MS-DOS 2.0이 출시될 때 FAT12에 계층형 디렉터리가 지원됨

• FAT16

- ✓ HDD 기술의 발달로 개인용 컴퓨터에서 사용할 파일시스템의 필요성 증가
- ✓ 1988년 MS-DOS 4.0에서 FAT16 발표

FAT 파일 시스템 – FAT 파일 시스템 소개

• VFAT or Fast FAT

- ✓ 1995년 MS는 Windows 95에서 FAT 파일시스템의 성능과 기능을 향상시킴
- ✓ 32 bits 보호 모드(Protected Mode) – 성능 향상
- ✓ 독점 모드(Exclusive Mode) – 동시에 하나의 파일에 접근할 경우 처리
- ✓ LFNs (Long File Names) 지원
 - 이전의 FAT 파일시스템은 8.3 File naming, 대문자만 가능
 - 최대 255 문자까지 파일명으로 적용 가능
 - 이전 버전의 DOS와 호환성 유지
- ✓ 볼륨의 최대 용량은 2GB로 파일 시스템의 용량 표현 문제가 여전히 존재함

FAT 파일 시스템 - FAT 파일 시스템 소개

• FAT32

- ✓ 1996년 MS는 Windows 95 OSR2에서 FAT32 사용
- ✓ 클러스터 표현 비트 수 28 bits (4 bits 예약 영역)
- ✓ 볼륨의 최대 용량은 32GB로 제한

• FAT 파일시스템의 비교

구분	FAT12	FAT16	FAT32
사용 용도	FDD	Small HDD	Large HDD
클러스터 표현 비트 수	12 bits	16 bits	32 bits (28 bits만 사용)
최대 클러스터 개수	4,084 개	65,524 개	67,092,281 개
최대 볼륨 크기	16 MB	2 GB	2 TB (32GB로 제한됨)
파일의 최대 크기	볼륨 크기만큼	볼륨 크기만큼	4 GB
디렉토리당 최대 파일 개수	X	65,535 개	65,535 개
루트 디렉토리의 파일 개수 제한	있음	있음	없음

FAT 파일 시스템 - FAT 파일 시스템 소개

• FAT 파일 시스템의 호환성

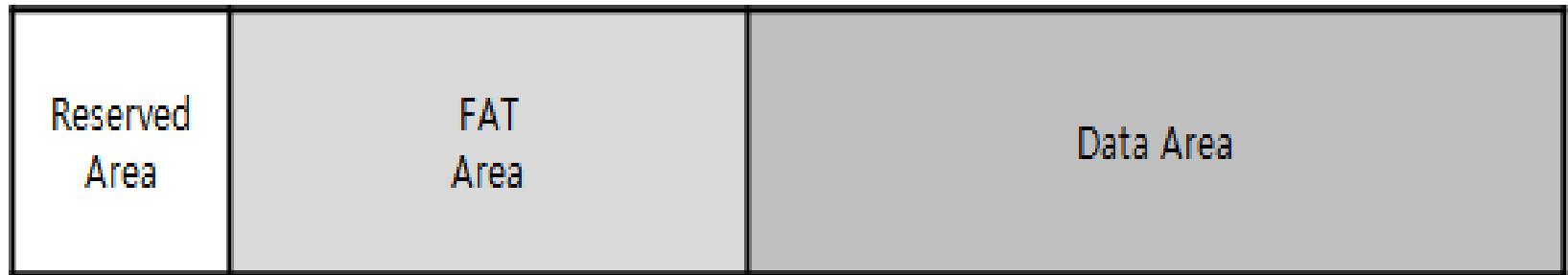
- ✓ 1996년 MS는 Windows 95 OSR2에서 FAT32 사용
- ✓ 클러스터 표현 비트 수 28 bits (4 bits 예약 영역)
- ✓ 볼륨의 최대 용량은 32GB로 제한

Operating System	FAT12	FAT16	FAT32	NTFS
MS-DOS	●	●		
Windows 95	●	●		
Windows 95 OSR2	●	●	●	
Windows 98	●	●	●	
Windows ME	●	●	●	
Windows NT 4.0	●	●		●
Windows 2000	●	●	●	●
Windows XP	●	●	●	●

FAT 파일 시스템 - FAT 파일 시스템 소개

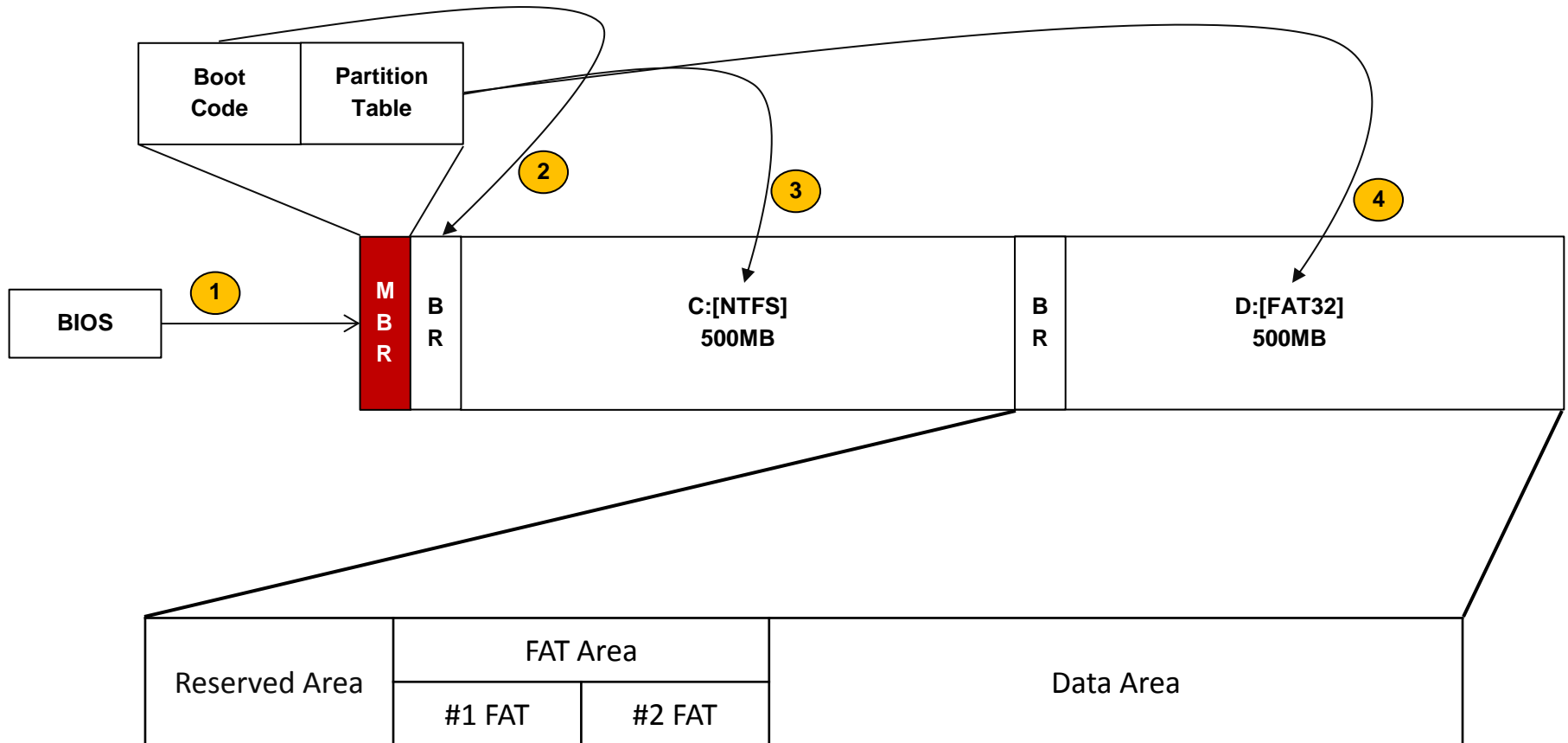
- FAT 파일 시스템의 구조

- ✓ 예약 영역 / FAT 영역 / 데이터 영역



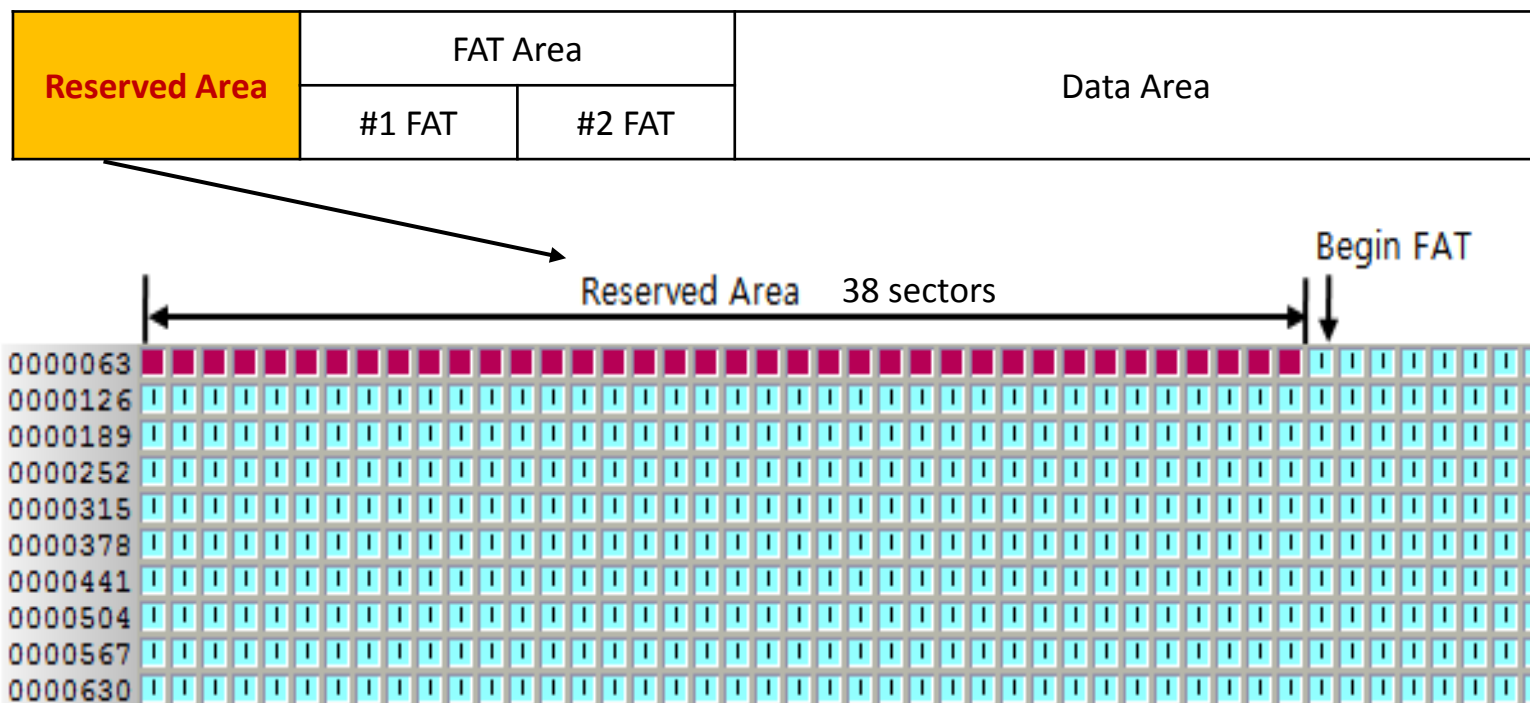
FAT 파일 시스템 - FAT 파일 시스템 소개

• FAT (File Allocation Table) Layout



FAT 파일 시스템 - 예약 영역

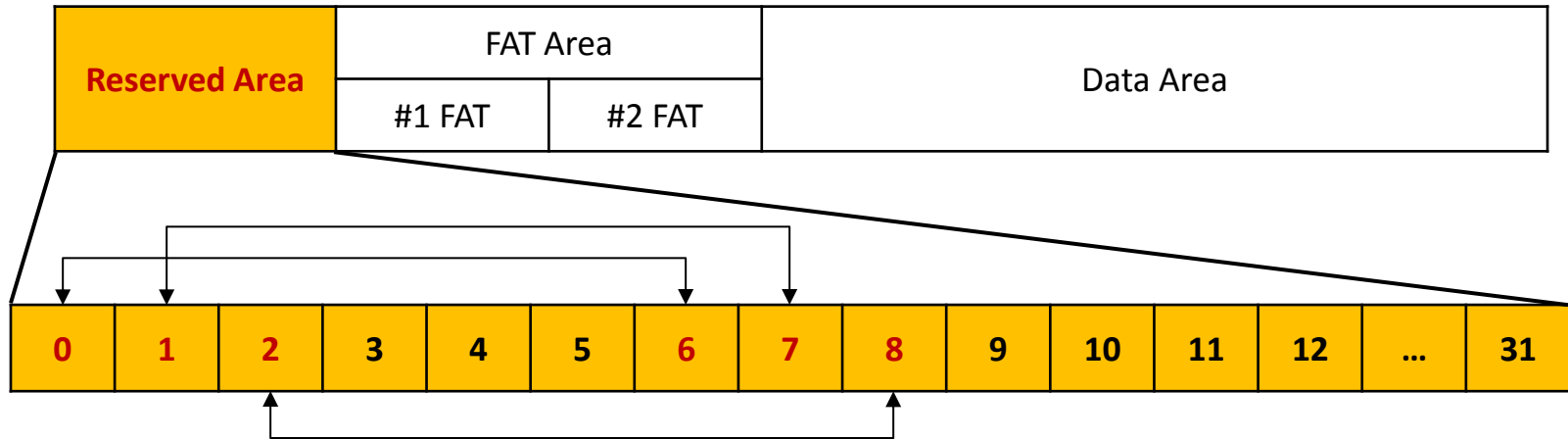
• Structure – Reserved Area



- ✓ FAT파일 시스템에서 가장 앞에 위치하는 구조로 여러 개의 섹터를 포함
- ✓ FAT12/16 : 1 Sector(default) 사용
- ✓ FAT32 : 32 Sectors(default) 사용

FAT 파일 시스템 - 예약 영역

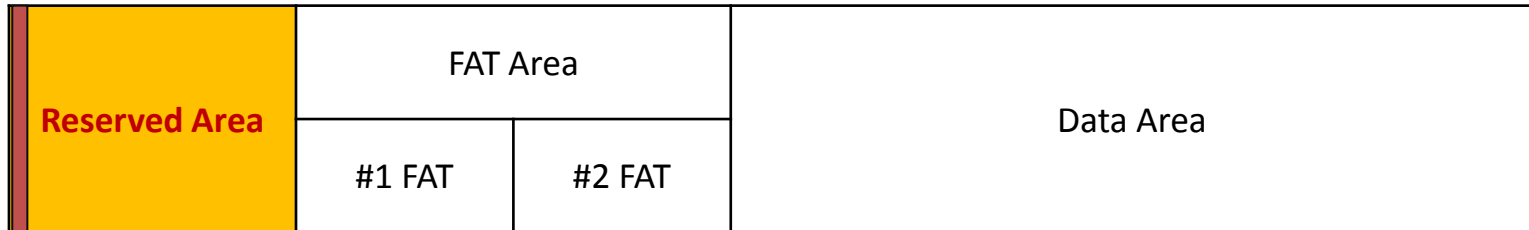
Structure - Reserved Area



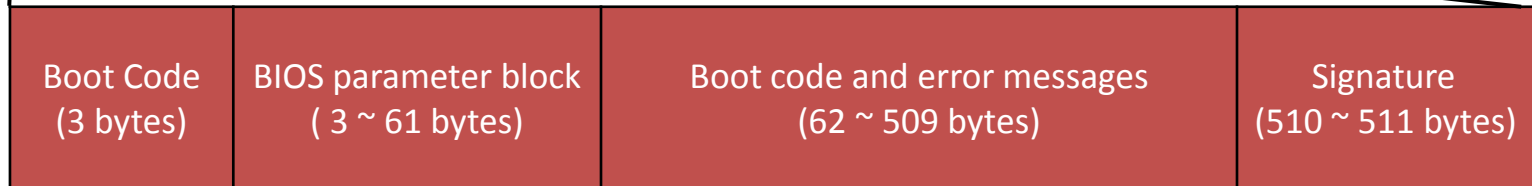
- ✓ FAT32 - 예약영역 중 6개의 섹터만 사용, 나머지는 만약을 대비해 예약해 둔 것
- ✓ 예약 영역의 크기는 가변적 이지만 0, 1, 2, 6, 7, 8번 섹터는 미리 정해져 있음
 - 0, 6 : Volume Boot Sector (0번은 부트섹터로 사용, 만약을 대비해 6번에 백업)
 - 1, 7 : File System Information(FSINFO) Structure
(7번에 백업, FSINFO구조체는 운영체제에게 비 할당 클러스터의 첫 위치와 전체 비 할당 클러스터의 수를 알려줌으로써 저장할 데이터를 빠르게 할당할 수 있도록 도와줌)
 - 2, 8 : Additional bootstrap code (부트 섹터의 부트 코드 영역이 부족할 경우 추가적으로 사용할 수 있는 영역, 일반적으로 비어 있음)

FAT 파일 시스템 - 예약 영역

- Structure – First Sector(Volume Boot Sector) of Reserved Area

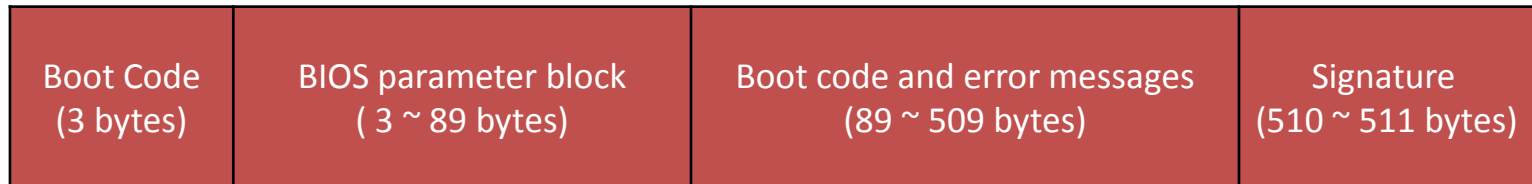


FAT 12/16



첫 번째 섹터는 부트 코드를 포함하고 있는 부트 섹터로 사용, FAT12/14은 예약 영역 = 부트 섹터

FAT 32



FAT 파일 시스템 - 예약 영역

- ✓ 첫 번째 섹터는 부트 코드를 포함하고 있는 부트 섹터(Boot Sector)로 사용
- ✓ FAT12/16은 예약 영역이 곧 부트 섹터
- ✓ 부트 섹터에는 제일 먼저 BPB(BIOS Parameter Block)을 지나 부트 코드로 점프하기 위한 명령어가 위치
- ✓ 부트 코드는 파일 시스템의 여러 설정 정보를 나타내는 BPB를 참조하여 시스템 부팅
- ✓ 부팅과정이 실패하면 미리 설정된 오류 메시지를 출력

FAT파일 시스템 부트 섹터 구조

FAT 형식	범위(Byte Range)		설명
	십진수	십육진수	
FAT12/16	0 - 2	0x0000 - 0x0002	Jump command to boot code
FAT32			
FAT12/16	3 - 61	0x0003 - 0x003D	BIOS parameter block(BPB)
FAT32	3 - 89	0x0003 - 0x0059	
FAT12/16	62 - 509	0x003E - 0x01FD	Boot code and error message
FAT32	90 - 509	0x005A - 0x01FD	
FAT12/16	510 - 511	0x01FE - 0x01FF	Signature (0x55AA)
FAT32			

FAT 파일 시스템 - 예약 영역

- Structure – FAT 12/16/32 부트 섹터의 공통된 데이터 구조

범위		설명
10 진수	16 진수	
0 ~ 2	0x00 ~ 0x02	Assembly jump instruction to bootstrap code (0xEB3C90)
3 ~ 10	0x03 ~ 0x0A	OEM ID (Win95 = MSWIN4.0, Win98 = MSWIN4.1 Win2K/XP/VISTA = MSDOS5.0, Linux = mkdosfs)
11 ~ 12	0x0B ~ 0x0C	Bytes per Sector
13	0x0D	Sectors per Cluster
14 ~ 15	0x0E ~ 0x0F	Reserved Sector count (FAT 12/16 = 1)
16	0x10	Number of FAT tables
17 ~ 18	0x11 ~ 0x12	Root director entry count (FAT12/16 = 512, FAT32 = 0)
19 ~ 20	0x13 ~ 0x14	Total Sector - 16 bits (FAT12/16 = variable, FAT32 = 0)
21	0x15	Media Type
22 ~ 23	0x16 ~ 0x17	FAT size - 16 bits (FAT12/16 = variable, FAT32 = 0)
24 ~ 25	0x18 ~ 0x19	Sector per Track (typically 32 for hard disk)
26 ~ 27	0x1A ~ 0x1B	Number of Heads (typically 255 for hard disk)
28 ~ 31	0x1C ~ 0x1D	Hidden Sectors
32 ~ 35	0x20 ~ 0x23	Total Sector - 32 bits

FAT 파일 시스템 - 예약 영역

- Structure – FAT 12/16 부트 섹터의 추가적인 데이터 구조

범위		설명
10 진수	16 진수	
36	0x24	INT 0x13 drive number (Floppy = 0x00, Hard disk = 0x80)
37	0x25	Not used
38	0x26	Boot signature
39 ~ 42	0x27 ~ 0x2A	Volume serial number
43 ~ 53	0x2B ~ 0x35	Volume label (ASCII)
54 ~ 61	0x36 ~ 0x3D	File system type
62 ~ 509	0x3E ~ 0x01FD	Boot code and error message
510 ~ 511	0x01FE ~ 0x01FF	Signature (0x55AA)

FAT 파일 시스템 - 예약 영역

- Structure – FAT 32 부트 섹터의 추가적인 데이터 구조

범위		설명
10 진수	16 진수	
36 ~ 39	0x24 ~ 0x27	FAT size 32
40 ~ 41	0x28 ~ 29	Ext flags
42 ~ 43	0x2A ~ 2B	FAT32 volume version
44 ~ 47	0x2C ~ 0x2F	Root directory cluster offset
48 ~ 49	0x30 ~ 0x31	FSINFO (File System Information) offset
50 ~ 51	0x32 ~ 0x33	Backup boot sector offset
52 ~ 63	0x34 ~ 0x3F	Reserved
64	0x40	INT 0x13 drive number (Floppy = 0x00, Hard Disk = 0x80)
66	0x41	Not used (typically 0)
66	0x42	Boot signature
67 ~ 70	0x43 ~ 0x46	Volume serial number
71 ~ 81	0x47 ~ 0x51	Volume label (ASCII)
82 ~ 89	0x52 ~ 0x59	File system type
90 ~ 509	0x60 ~ 0x01FD	Boot code and error message
510 ~ 511	0x01FE ~ 0x01FF	Signature (0x55AA)

FAT 파일 시스템 - 예약 영역

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		

FAT 파일 시스템 부트 섹터 구조

Jump command

BIOS Parameter Block

000	EB 58 90	4D 53 44 4F 53 35 2E 30 00 02 08 26 00	äXIMSDOS5.0···&·
016	02 00 00	00 00 F8 00 00 3F 00 FF 00 3F 00 00 00	·····ø··?·ÿ?·
032	C1 BF 1E 00	AD 07 00 00 00 00 00 00 00 02 00 00 00	Á¿··-········
048	01 00 06 00	00 00 00 00 00 00 00 00 00 00 00 00 00	··········
064	00 00 29 54	04 1E 5C 4E 4F 20 4E 41 4D 45 20 20	··)T··\NO NAME
080	20 20 46 41	54 33 32 20 20 20 33 C9 8E D1 BC F4	FAT32 3ÉŽŃ·ó
096	7B 8E C1 8E	D9 BD 00 7C 88 4E 02 8A 56 40 B4 08	{ŽÁŽŮ· ·^N·ŠV@·`·
112	CD 13 73 05	B9 FF FF 8A F1 66 0F B6 C6 40 66 0F	Í·s··ÿÿŠñf·¶E@f·
128	B6 D1 80 E2	3F F7 E2 86 CD C0 ED 06 41 66 0F B7	¶ŃÈá?·â·+ÍÁí·Af·
144	C9 66 F7 E1	66 89 46 F8 83 7E 16 00 75 38 83 7E	Éf·á·f·F·øf·~···u8f~
160	2A 00 77 32	66 8B 46 1C 66 83 C0 0C BB 00 80 B9	·+·w2f·<F·ff·À·»·€¹
176	01 00 E8 2B	00 E9 48 03 A0 FA 7D B4 7D 8B F0 AC	··è·+·é·H· ú·}·'·<·ø·
192	84 C0 74 17	3C FF 74 09 B4 0E BB 07 00 CD 10 EB	„Át·<·ÿt·`·»··Í·ä
208	EE A0 FB 7D	EB E5 A0 F9 7D EB E0 98 CD 16 CD 19	í·ù·}·è·à·ù·}·è·à·í·í·
224	66 60 66 3B	46 F8 0F 82 4A 00 66 6A 00 66 50 06	f·`·f·;·F·ø·,·J·f·j·f·P·
240	53 66 68 10	00 01 00 80 7E 02 00 0F 85 20 00 B4	Sfh······€~·····`·`·
256	41 BB AA 55	8A 56 40 CD 13 0F 82 1C 00 81 FB 55	A»·°UŠV@Í·,·,·°ÓU
272	AA 0F 85 14	00 F6 C1 01 0F 84 0D 00 FE 46 02 B4	·°·····ø·Á·,·,·°pF·`·
288	42 8A 56 40	8B F4 CD 13 B0 F9 66 58 66 58 66 58	BŠV@<·óÍ·°·ù·f·X·f·X·f·X·
304	66 58 EB 2A	66 33 D2 66 0F B7 4E 18 66 F7 F1 FE	f·X·è·+·f·3·0·f···N·f·+·ñ·p·
320	C2 8A CA 66	8B D0 66 C1 EA 10 F7 76 1A 86 D6 8A	ÅŠÉf·<·ø·f·Á·è·+·v·+·t·ÖŠ
336	56 40 8A E8	C0 E4 06 0A CC B8 01 02 CD 13 66 61	V@Šè·À·à· Ì·,·,·Í·f·a·
352	0F 82 54 FF	81 C3 00 02 66 40 49 0F 85 71 FF C3	·,·Tÿ·°Á·°·f·@I·...qÿ·Á
368	4E 54 4C 44	52 20 20 20 20 20 00 00 00 00 00 00	NTLDR ·····
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 0D 0A 52 65	·········· Re
432	6D 6F 76 65	20 64 69 73 6B 73 20 6F 72 20 6F 74	move disks or ot
448	68 65 72 20	6D 65 64 69 61 2E FF 0D 0A 44 69 73	her media.ÿ Dis
464	6B 20 65 72	72 6F 72 FF 0D 0A 50 72 65 73 73 20	k errorÿ Press
480	61 6E 79 20	6B 65 79 20 74 6F 20 72 65 73 74 61	any key to resta
496	72 74 0D 0A	00 00 00 00 00 00 00 AC CB D8 00 00 55 AA	rt ·····-Èø·U²

FAT 파일 시스템 - 예약 영역

- 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)

FAT32의 FSINFO 구조체 영역 데이터 구조

000	52	52	61	41	00	00	00	00	00	00	00	00	00	00	00	00	00	RRaA.....
016	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
032	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
048	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
064	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
096	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
112	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
128	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
144	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
160	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
176	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
192	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
208	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
224	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
240	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
256	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
272	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
288	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
304	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
320	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
336	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
352	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
368	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
384	00	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	00
416	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
432	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
448	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
464	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
480	00	00	00	00	72	72	41	61	70	2D	03	00	E1	48	00	00	00rrAap-...áH..
496	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	55	AAU²

FAT32의 FSINFO 구조체 섹터 덤프

FAT 파일 시스템 - FAT 영역

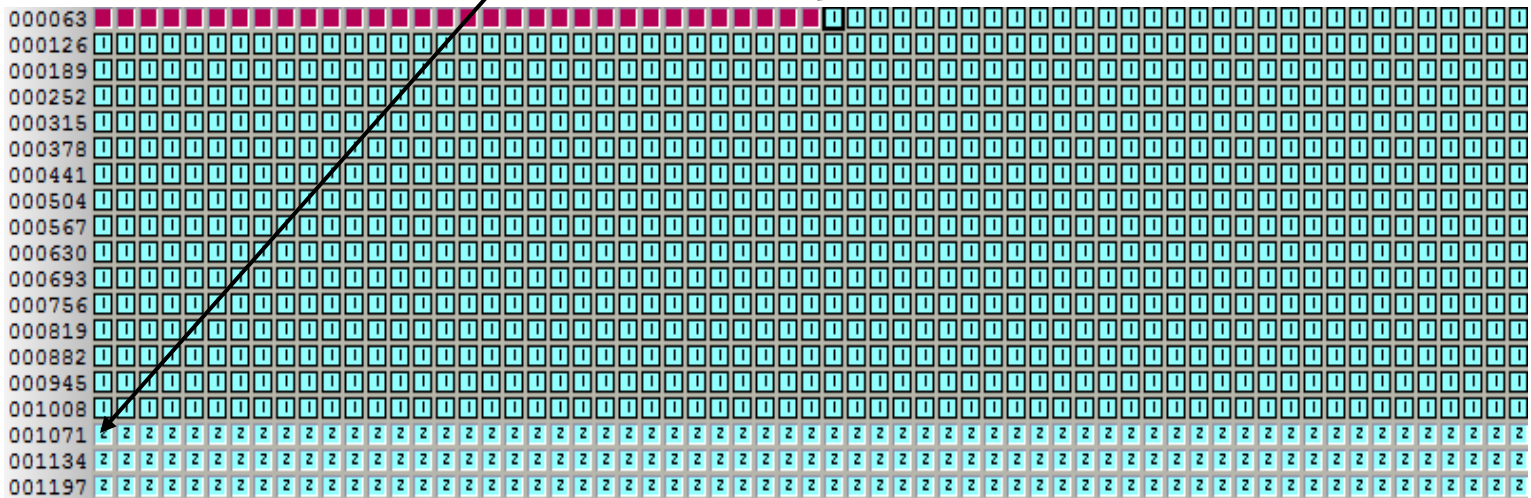
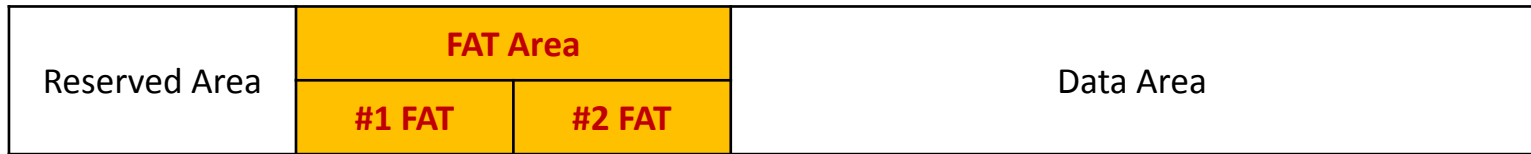
• Structure – FAT Area



- ✓ 저장된 파일의 클러스터 할당 관계를 표현 - FAT 12/16, FAT 32
- ✓ FAT (File Allocation Table) Area - #1 FAT, #2 FAT(Backup)
- ✓ FAT16은 16비트, FAT32는 32비트를 사용해 데이터 영역의 시작 클러스터부터 마지막 클러스터까지 할당 상태를 표시

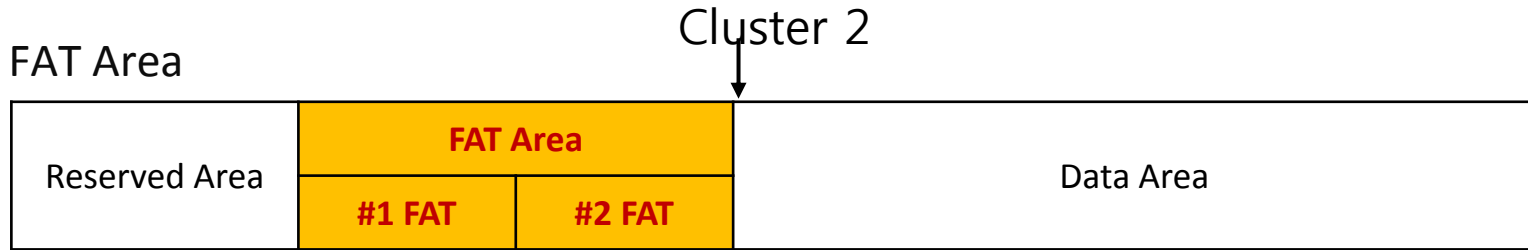
FAT 파일 시스템 - FAT 영역

- Structure – FAT Area



FAT 파일 시스템 - FAT 영역

Structure - FAT Area



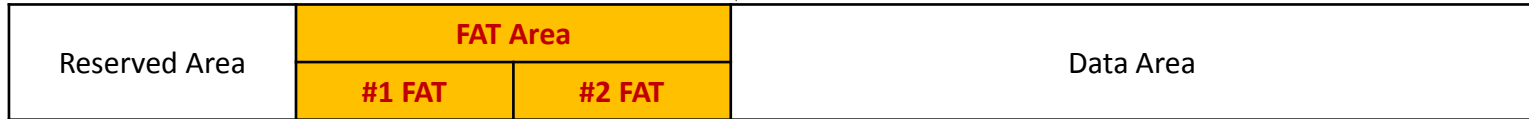
- FAT32의 FAT영역의 첫 번째 섹터 내용
- 각 4바이트는 "FAT Entry"라고 불림
- FAT Entry 0,1번은 저장 매체 종류와 파티션 상태를 표현하기 위해 예약됨
- FAT Entry 2번부터 데이터 영역의 클러스터와 대응
- 데이터영역의 시작 클러스터 번호는 2번
이므로 해당 클러스터의 상태가 4바이트로 표현

	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			
															

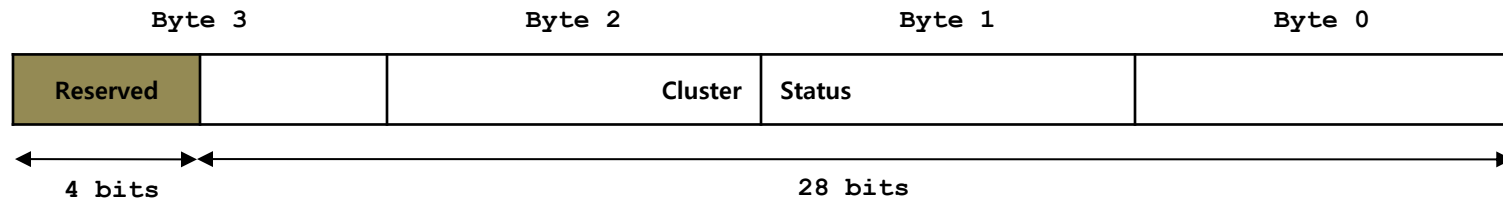
FAT 파일 시스템 - FAT 영역

- Structure – FAT Area (Entry Type)

Cluster 2



FAT 12	FAT 16	FAT 32	설명
0x000	0x0000	0x?0000000	Unallocated (Free) cluster
0x001	0x0001	0x?0000001	Reserved cluster
0x002	0x0002 ~ 0xEFFF	0x?0000002 ~ 0x?FFFFFFEF	Allocated cluster
0xFF0 ~ 0xFF6	0xFFFF0 ~ 0xFFFF6	0x?FFFFFF0 ~ 0x?FFFFFF6	Reserved cluster
0xFF7	0xFFFF7	0x?FFFFFF7	Bad cluster
0xFF8	0xFFFF8 ~ 0xFFFF	0x?FFFFFF8 ~ 0x?FFFFFFF	End-of-file marker



FAT 파일 시스템 – FAT 영역

• Structure – FAT Area (Entry Type)

- 비할당 상태일 경우, 0x00의 값을 가진다. 따라서 운영체제는 새로운 파일 및 디렉터리를 저장하고자 할 경우 FAT 영역에서 FAT Entry 값이 0x00인 클러스터를 찾아 할당한다.
- 할당 상태일 경우, FAT Entry의 값은 그 클러스터를 점유하고 있는 파일의 다음 데이터가 있는 클러스터를 가리킨다. 파일의 마지막 데이터가 있는 클러스터이면 마지막을 나타내는 특정 값을 사용한다. FAT12는 0xFF8보다 큰 값을 사용하고 FAT16은 0xFFF8보다 큰 값을 사용한다. 그리고 FAT32는 0x0FFF FFF8보다 큰 값을 사용한다. 만약 파일이 하나의 클러스터만 사용한다면 이 값을 사용하게 된다.
- 만약 배드 섹터가 포함된 클러스터가 발견될 경우 FAT12에서는 0xFF7, FAT16은 0xFFF7, FAT32는 0x0FFF FFF7 값을 사용해 표시하게 된다. 표시된 클러스터는 이후에 사용되지 않는다.

FAT 파일 시스템 – FAT 영역

• Structure – FAT Area (Entry 0 and 1)

✓ FAT Entry 0 – Media Type

Byte 3	Byte 2	Byte 1	Byte 0
0xFF	0xFF	0xFF	Media Type

FAT 16	FAT 32	설명
0xFF??	0x?FFFFFF??	FAT Entry 0 (Media Type)

✓ FAT Entry 1 – Volume Status

FAT 16	FAT 32	설명
0x8000	0x80000000	Clean Shutdown Bit Mask (상위 1 번째 비트)
0x4000	0x40000001	Hard Error Bit Mask (상위 2 번째 비트)

FAT 파일 시스템 - FAT 영역

• FAT Dump

FAT32인 경우 4 바이트를 FAT Entry로 사용하기 때문에 한 섹터 당 128개의 FAT Entry를 표현할 수 있다.

FAT Entry 0, 1번은 앞서 살펴본 바와 같이 특별한 의미로 사용된다.

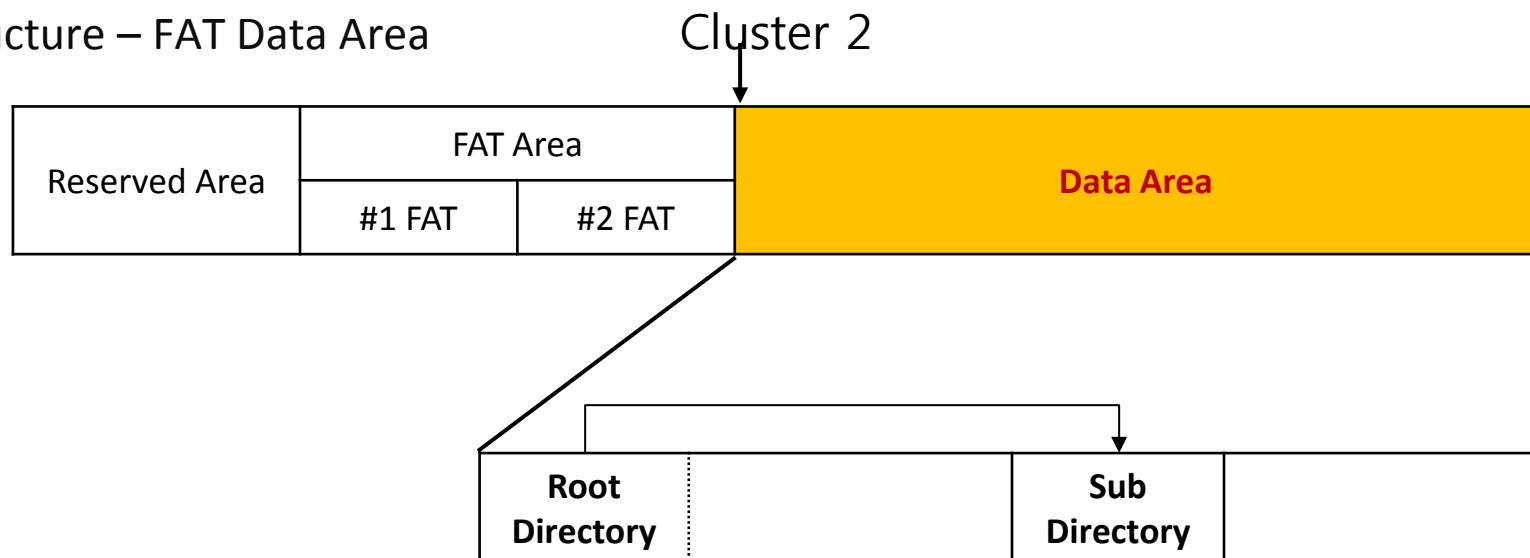
실제 데이터 영역의 시작은 FAT Entry 2번부터 시작한다. FAT Entry 2번의 값이 0x0FFF FFFF 값을 가지므로 해당 클러스터를 사용하는 파일은 하나의 클러스터만 사용하는 것을 알 수 있다.

FAT Entry 4번부터 할당된 파일은 모두 5개의 클러스터를 사용하는 파일임을 알 수 있다. 이처럼 FAT 영역의 정보로 특정 파일이 점유하고 있는 클러스터의 위치를 알 수 있다. 하지만 해당 파일의 이름, 확장자, 시간 정보, 정확한 크기 등을 알 수가 없다. 이러한 정보는 다음에서 살펴볼 데이터 영역의 디렉터리 엔트리를 통해 확인할 수 있다.

000F8	FF	FF	0F	FF	FF	FF	FF	FF	FF	FF	0F	FF	FF	FF	0F	00000000
01605	00	00	00	06	00	00	00	07	00	00	00	08	00	00	00	00000000
032FF	FF	FF	0F	0A	00	00	00	FF	FF	FF	0F	0C	00	00	00	00000000
0480D	00	00	00	0E	00	00	00	0F	00	00	00	10	00	00	00	00000000
064FF	FF	FF	0F	12	00	00	00	13	00	00	00	14	00	00	00	00000000
08015	00	00	00	16	00	00	00	17	00	00	00	18	00	00	00	00000000
09619	00	00	00	1A	00	00	00	1B	00	00	00	1C	00	00	00	00000000
1121D	00	00	00	1E	00	00	00	1F	00	00	00	FF	FF	FF	0F	00000000
12821	00	00	00	22	00	00	00	23	00	00	00	24	00	00	00	00000000
14425	00	00	00	26	00	00	00	27	00	00	00	28	00	00	00	00000000
16029	00	00	00	2A	00	00	00	2B	00	00	00	2C	00	00	00	00000000
1762D	00	00	00	2E	00	00	00	2F	00	00	00	30	00	00	00	00000000
19231	00	00	00	32	00	00	00	33	00	00	00	34	00	00	00	00000000
20835	00	00	00	36	00	00	00	37	00	00	00	38	00	00	00	00000000
22439	00	00	00	3A	00	00	00	3B	00	00	00	3C	00	00	00	00000000
2403D	00	00	00	3E	00	00	00	3F	00	00	00	40	00	00	00	00000000
25641	00	00	00	42	00	00	00	43	00	00	00	44	00	00	00	00000000
27245	00	00	00	46	00	00	00	47	00	00	00	48	00	00	00	00000000
28849	00	00	00	4A	00	00	00	4B	00	00	00	4C	00	00	00	00000000
3044D	00	00	00	4E	00	00	00	4F	00	00	00	50	00	00	00	00000000
32051	00	00	00	52	00	00	00	53	00	00	00	54	00	00	00	00000000
33655	00	00	00	56	00	00	00	57	00	00	00	58	00	00	00	00000000
35259	00	00	00	5A	00	00	00	5B	00	00	00	5C	00	00	00	00000000
3685D	00	00	00	5E	00	00	00	5F	00	00	00	60	00	00	00	00000000
38461	00	00	00	62	00	00	00	63	00	00	00	64	00	00	00	00000000
40065	00	00	00	66	00	00	00	67	00	00	00	68	00	00	00	00000000
41669	00	00	00	6A	00	00	00	6B	00	00	00	6C	00	00	00	00000000
4326D	00	00	00	6E	00	00	00	6F	00	00	00	70	00	00	00	00000000
44871	00	00	00	72	00	00	00	73	00	00	00	74	00	00	00	00000000
46475	00	00	00	76	00	00	00	77	00	00	00	78	00	00	00	00000000
48079	00	00	00	7A	00	00	00	7B	00	00	00	7C	00	00	00	00000000
4967D	00	00	00	7E	00	00	00	7F	00	00	00	80	00	00	00	00000000

FAT 파일 시스템 - 데이터 영역

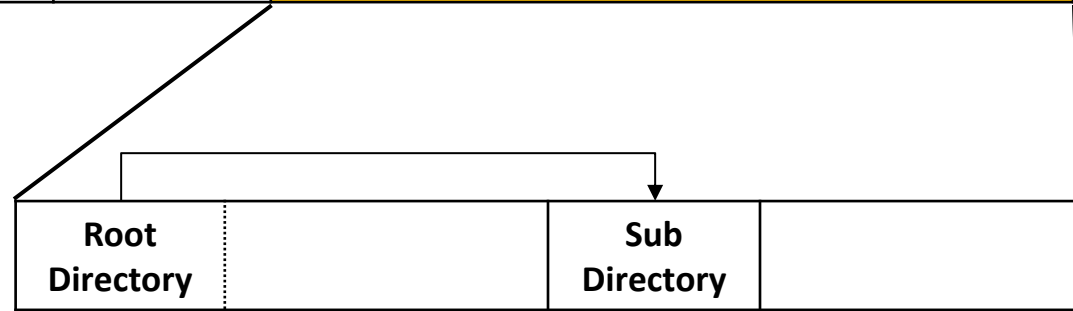
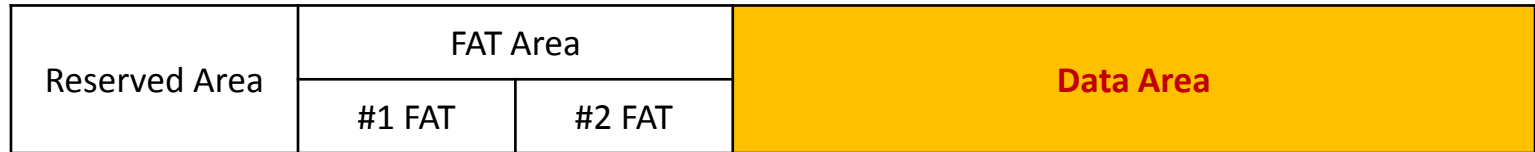
Structure - FAT Data Area



- ✓ 최상위 루트 디렉터리가 가장 중요
 - FAT12/16 : FAT Entry 2번, FAT32 : 어느 곳이나 (기본적으로는 FAT Entry 2번)
- ✓ 데이터 = 디렉터리 + 파일
- ✓ 모든 파일과 디렉터리는 하위디렉터리 및 파일의 이름, 확장자, 시간 정보, 크기 등을 표현하기 위해 Directory Entry로 표현됨

FAT 파일 시스템 - 데이터 영역

• Structure – FAT 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									

Directory Entry 구조

FAT 파일 시스템 - 데이터 영역

Structure – FAT Data Area (Directory Entry – Name)

	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			

- ✓ 첫 번째 바이트가 0xE5 값을 가진다면 해당 엔트리는 삭제되었음을 의미
 - 첫 번째 바이트 값만 변경되고 나머지 값은 초기화하지 않기 때문에 파일의 정보 및 내용을 복구할 수 있음
- ✓ 첫 번째 바이트가 0x00의 값을 가진다면 해당 엔트리는 사용되지 않는 엔트리
 - 이후에 엔트리가 존재하지 않는다는 것을 의미하므로 더 이상 검색할 필요가 없음
- ✓ 파일 이름 또는 디렉토리 이름의 문자 제한
 - 영어 대문자: A ~ Z (소문자는 대문자로 변환)
 - 숫자: 0 ~ 9
 - 특수 문자: \$ % ` - _ @ ~ ! () { } ^ # &

FAT 파일 시스템 - 데이터 영역

Structure – FAT Data Area (Directory Entry – Name)

	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 Name	Name + Extender (11 bytes)											
FOO.BAR	F	O	O							B	A	R
FILEDATA.DOC	F	I	L	E	D	A	T	A		D	O	C
foo	F	O	O									
foo.bar	F	O	O							B	A	R
Pickle.A	P	I	C	K	L	E				A		
.BIG	잘못된 표현. Name[0]에 0x20이 올 수 없음											
HELLO!.JPG	잘못된 표현. 특수("!") 문자 허용 안됨											

FAT 파일 시스템 - 데이터 영역

• Structure – FAT Data Area (Directory Entry – Attribute)

	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			

Attribute	설 명
0000 0001	Read only
0000 0010	Hidden file
0000 0100	System file
0000 1000	Volume label
0000 1111	Long file name (LFN)
0001 0000	Directory
0010 0000	Archive

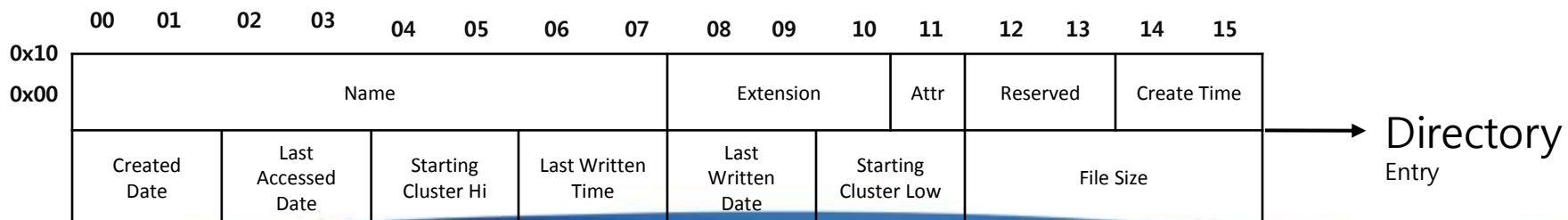
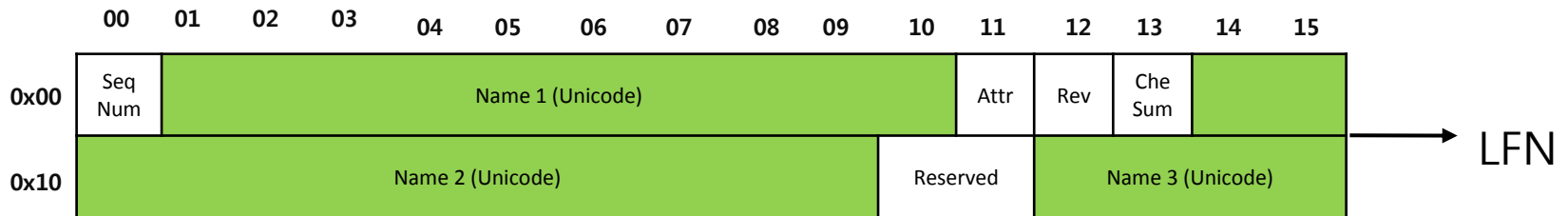
Directory Entry의 속성

FAT 파일 시스템 - 데이터 영역

• LFN – Long File Name

- 8바이트보다 긴 파일 이름 표현 위해 사용(유니코드로 표현)
- 여러 개의 LFN Entry 사용할 경우 순서번호는 이러한 여러 개의 LFN Entry의 관계를 나타냄

만약 마지막 LFN 이면 LAST_LONG_ENTRY (0x40) | N의 값을 저장



FAT 파일 시스템 - 데이터 영역

LFN Entry 데이터 구조

범위(Byte Range)		설 명
십진수	십육진수	
0 - 0	0x0000 - 0x0000	Sequence number or status byte
1 - 10	0x0001 - 0x000A	LFN character 1-5 (Unicode)
11 - 11	0x000B - 0x000B	Attributes
12 - 12	0x000C - 0x000C	Reserved
13 - 13	0x000D - 0x000D	Checksum
14 - 25	0x000E - 0x0019	LFN character 6-11 (Unicode)
26 - 27	0x001A - 0x001B	Reserved
28 - 31	0x001C - 0x001F	LFN character 12-13 (Unicode)

FAT 파일 시스템 - 데이터 영역

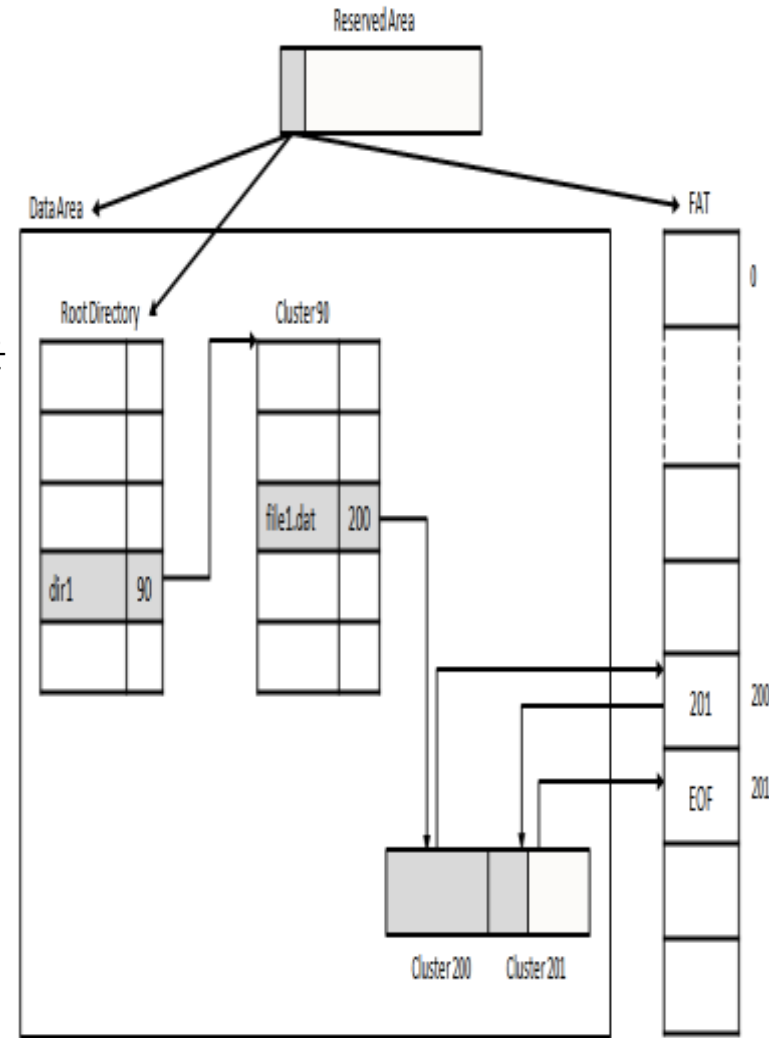
000	02 20 00 46 00 69 00 6C 00 65 00	0F 00 36 20 00	· · F i l e · · · · 6 ·
016	53 00 79 00 73 00 74 00 65 00 00 00	6D 00 2E 00	S y s t e m · · ·
032	01 55 00 6E 00 64 00 65 00 72 00	0F 00 36 73 00	· U n d e r · · · · 6 s ·
048	74 00 61 00 6E 00 64 00 69 00 00 00	6E 00 67 00	t a n d i · · · · n g ·
064	55 4E 44 45 52 53 7E 31 48 57 50	20 00 32 2A 03	U N D E R S ~ 1 H W P · 2 * ·
080	27 3B 28 3B 00 00 35 02 27 3B 92 00	00 00 80 0D 00	' ; (; · · 5 ' ; ' · · E ·
096	55 54 46 38 20 20 20 20 43 20 20	20 18 73 2A 03	U T F 8 C ' s * ·
112	27 3B 27 3B 00 00 42 88 24 3B 42 02	9C 2F 00 00	' ; ' ; · · B * \$; B * æ / · ·
128	43 70 00 70 00 74 00 78 00 00 00 0F	00 B9 FF FF	C p p t x · · · · · y y
144	FF FF FF FF FF FF FF FF FF 00 00	FF FF FF FF	y y y y y y y y y y · · y y y y
160	02 20 00 46 00 69 00 6C 00 65 00	0F 00 B9 20 00	· · F i l e · · · · 1 ·
176	53 00 79 00 73 00 74 00 65 00 00 00	6D 00 2E 00	S y s t e m · · ·
192	01 55 00 6E 00 64 00 65 00 72 00	0F 00 B9 73 00	· U n d e r · · · · s ·
208	74 00 61 00 6E 00 64 00 69 00 00 00	6E 00 67 00	t a n d i · · · · n g ·
224	55 4E 44 45 52 53 7E 31 50 54 20 00	8B 2A 03	U N D E R S ~ 1 P P T < * ·
240	27 3B 27 3B 00 00 0D B1 26 3B 48 02	B2 B6 00 00	' ; ' ; · · ± & ; H * ¶ · ·
256	E5 50 00 4E 00 47 00 00 00 FF FF	0F 00 09 FF FF	ã p n g · · · · y y · · y y
272	FF FF FF FF FF FF FF FF FF 00 00	FF FF FF FF	y y y y y y y y y y · · y y y y
288	E5 76 00 6F 00 6C 00 75 00 6D 00	0F 00 09 65 00	ã v o l u m e · · · e ·
304	5F 00 73 00 6C 00 61 00 63 00 00 00	6B 00 2E 00	· · s l a c k · · · · k · ·
320	E5 4F 4C 55 4D 45 7E 31 50 4E 47	20 00 A7 2A 03	ã O L U M E ~ 1 P N G ' S * ·
336	27 3B 28 3B 00 00 B0 8A 26 3B 5F 02	97 09 00 00	' ; (; · · ° Š & ; _ - · ·
352	00 00 00 00 00 00 00 00 00 00 00	00 00 00 00	· · · · · · · · · · · · · · · ·
368	00 00 00 00 00 00 00 00 00 00 00	00 00 00 00	· · · · · · · · · · · · · · · ·
384	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	· · · · · · · · · · · · · · · ·
416	00 00 00 00 00 00 00 00 00 00 00	00 00 00 00	· · · · · · · · · · · · · · · ·
432	00 00 00 00 00 00 00 00 00 00 00	00 00 00 00	· · · · · · · · · · · · · · · ·
448	00 00 00 00 00 00 00 00 00 00 00	00 00 00 00	· · · · · · · · · · · · · · · ·
464	00 00 00 00 00 00 00 00 00 00 00	00 00 00 00	· · · · · · · · · · · · · · · ·
480	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 00	· · · · · · · · · · · · · · · ·

Directory Entry 덤프

- Understanding File System.HWP
이름의 길이가 8 바이트를 초과하기 때문에 추가적으로 2개의 LFN 엔트리를 사용해 표현
- utf8.c
이름의 길이가 8 바이트를 초과하지 않아 하나의 디렉터리 엔트리로 표현
- volume_slack.png
2개의 LFN 엔트리를 사용해 표현되었다. 하지만 디렉터리 엔트리의 첫 바이트가 0xE5 값을 가지는 것으로 보아 현재 삭제되었음

FAT 파일 시스템 - 파일의 할당으로 인한 변화 요소

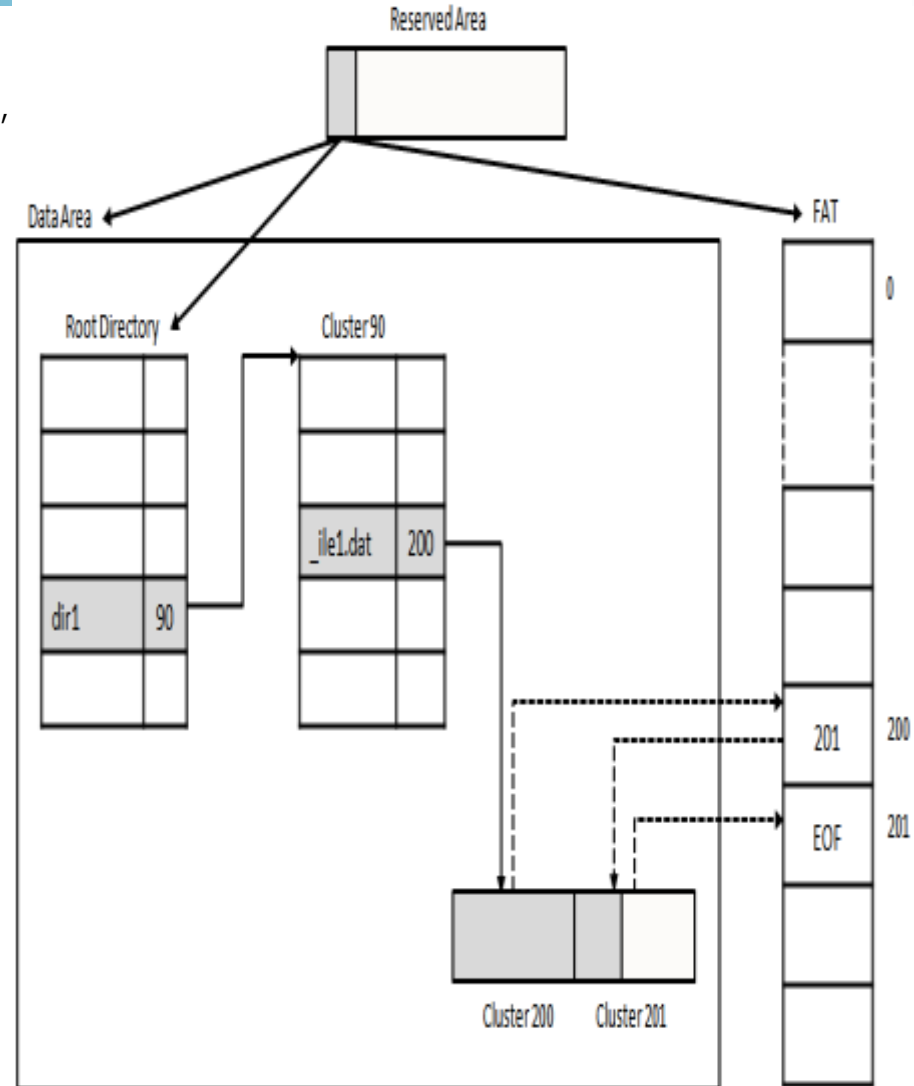
1. 예약된 영역의 부트 섹터에 있는 BPB에서 데이터 영역, 루트 디렉터리, FAT 영역의 위치를 얻어온다.
2. Wdir1 디렉터리의 정보를 얻기 위해 루트 디렉터리에서 Wdir1에 해당하는 디렉터리 엔트리를 검색한다. 그 결과 Wdir1 디렉터리 정보를 가지는 시작 클러스터가 90번임을 확인했다.
3. 클러스터 90번에서 file1.dat 파일의 메타 정보를 저장할 디렉터리 엔트리를 찾는다. 첫 번째 바이트가 0xE5 값을 갖는 삭제된 디렉터리 엔트리가 없다면 마지막의 0x00 값을 갖는 새로운 디렉터리 엔트리를 할당한다.
4. file1.dat 파일의 내용을 저장할 클러스터를 찾기 위해 FAT 영역을 검색하여 0x00 값을 갖는 FAT Entry를 찾은 결과 클러스터 200번에 해당하는 FAT Entry 200번이 선택되었다.
5. FAT Entry 200번에 "0x0FFF FFFF" 값을 기록하고, 디렉터리 엔트리의 시작 클러스터 번호를 200번으로 기록한다.
6. file1.dat 파일의 처음 4,096 바이트의 내용을 클러스터 200번에 기록한다. 그 결과 1,904 바이트의 내용이 남았다.
7. 남은 1,904 바이트의 내용을 다시 기록하기 위해 FAT 영역에서 또 다른 빈 FAT Entry를 찾는다. 그 결과 FAT Entry 201번이 선택되었다.
8. FAT Entry 200번의 값을 FAT Entry 201을 가리키도록 "201"로 수정한 후 FAT Entry 201번의 값은 클러스터 체인의 마지막을 나타내는 "0x0FFF FFFF" 값으로 기록한다.
9. 남은 1,904 바이트의 내용을 클러스터 201번에 기록한다. 기록 결과 144 바이트의 램 슬랙과 2,048 바이트의 파일 슬랙이 생성되었다.



Wdir1\file1.dat 파일 할당 시 FAT 파일 시스템 변화

FAT 파일 시스템 - 파일의 삭제로 인한 변화 요소

1. 예약된 영역의 부트 섹터에 있는 BPB에서 데이터 영역, 루트 디렉터리, FAT 영역의 위치를 얻어온다.
2. `\dir1` 디렉터리의 정보를 얻기 위해 루트 디렉터리에서 `\dir1`에 해당하는 디렉터리 엔트리를 검색한다. 그 결과 `\dir1` 디렉터리 정보를 가지는 시작 클러스터가 90번임을 확인했다.
3. 클러스터 90번에서 디렉터리 엔트리들을 검색하여 `file1.dat` 파일 이름을 가지는 디렉터리 엔트리를 찾는다. 그 결과 해당 파일의 시작 클러스터가 200번임이 확인되었다.
4. FAT Entry의 클러스터 체인을 확인 결과 해당 파일이 클러스터 200, 201번을 사용한다는 것이 확인되었다.
5. FAT Entry 200, 201번의 값을 `0x00`으로 초기화한다.
6. `file1.dat` 파일의 디렉터리 엔트리의 오프셋 `0x00` 위치의 값을 `0xE5`로 변경한다.



`\dir1\file1.dat` 파일 삭제 시 FAT 파일 시스템 변화

FAT 파일 시스템 – The Function of FAT

- How a File is stored (1/4)

The screenshot displays the EnCase Forensic Training interface. The top menu bar includes File, Edit, View, Tools, and Help. Below the menu is a toolbar with icons for New, Open, Save, Print, Add Device, Search, Refresh, and Find. The main window is divided into several panes:

- Left Pane:** Shows a tree view of the file system. The 'Home' folder is selected, and its contents are listed: 'Entries', 'File Extents', and 'Permissions'. The 'Entries' folder is expanded, showing a sub-folder '4'.
- Table View:** A table with columns: Name, Filter, In Report, File Ext, File Type, and File Category. The table contains three entries:

	Name	Filter	In Report	File Ext	File Type	File Category
<input type="checkbox"/>	1	1ST				
<input type="checkbox"/>	2	2ND				
<input type="checkbox"/>	3	LongDirectory				
- Bottom Left Pane:** A hex dump view showing memory addresses and their corresponding hexadecimal and ASCII values. The address 0248 is highlighted, showing the ASCII string 'LONGFI~1TXT'. The address 0310 is also highlighted, showing the ASCII string 'LONGFI~1TXT'. The address 0496 is also highlighted, showing the ASCII string 'LONGFI~1TXT'.
- Bottom Right Pane:** A tree view of the EnScript project, showing folders for Examples, Forensic, Include, Main, and Source Processor.

The status bar at the bottom of the window displays the path: Case 1\W4\WC\LongDirectory (PS 5360 LS 5328 CL 5 SO 288 FO 288 LE 32).

FAT 파일 시스템 – The Function of FAT

- How a File is stored (2/4)

The screenshot displays the EnCase Forensic Training interface. On the left, a file explorer shows the path 'Case 1\W4\WC\WLongDirectory'. The main window shows a 'Bookmark Data' dialog box with a 'Data Type' list containing 'Partition Entry', 'DOS Directory Entry', 'Win95 Info File Record', 'Win2000 Info File Record', 'GUID', 'UUID', 'SID', and 'Styles'. The 'Destination Folder' is set to 'Bookmarks'. Below the dialog, a 'File Record' table is shown for 'Case 1\W4\WC\WLongDirectory'. The table has columns for Name, Created, Written, Accessed, Size, and Cluster. The 'Cluster' column for the file 'LONGFI~1.TXT' is highlighted with a red box and contains the value '7'.

Name	Created	Written	Accessed	Size	Cluster
LONGFI~1.TXT	04/23/10 05:27:39오전	04/23/10 05:28:10오전	04/23/10	24	7

FAT 파일 시스템 – The Function of FAT

- How a File is stored (4/4)

The screenshot displays the EnCase Forensic Training software interface. The main window shows a FAT file system view with a grid of file entries. The grid has columns for address, permissions, and data. The data column shows a sequence of '2' characters, indicating a file with a specific content. The bottom status bar shows the current file path and details.

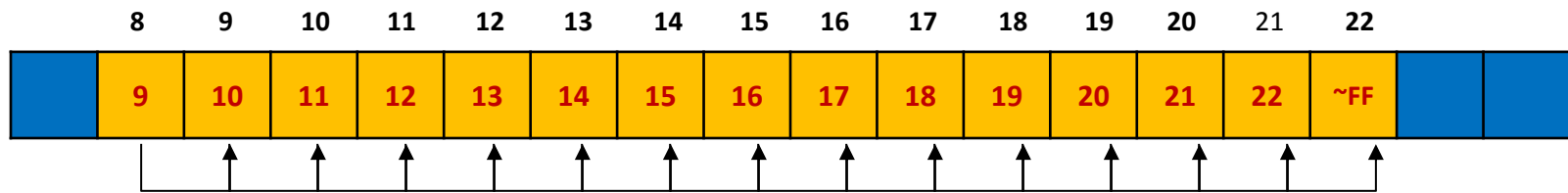
Address	Permissions	Data
0005160	2	2
0005203	2	2
0005246	2	2
0005289	2	2
0005332	2	2
0005375	2	2
0005418	2	2
0005461	2	2
0005504	2	2

Case 1\W4\WC\LongDirectory\LongFileNmae.txt (PS 5376 LS 5344 CL 7 SO 000 FO 0 LE 1)

FAT 파일 시스템 – The Function of FAT

• The Effects of Deleting and Undeleting Files

✓ C:\1ST\file1234.txt (58,242 bytes) 파일의 FAT Entry 구성



Root Directory

Name	Starting Cluster
1ST	3

Cluster 3

File1234.txt	8

FAT 파일 시스템 – The Function of FAT

- The Effects of Deleting and Undeleting Files

- ✓ C:\1ST\file1234.txt (58,242 bytes) 파일의 위치 찾기

The screenshot displays the EnCase Forensic Training interface. The top menu includes File, Edit, View, Tools, and Help. Below the menu is a toolbar with icons for New, Open, Save, Print, Add Device, Search, Refresh, and Find. The main window is divided into several panes:

- Left Pane:** Shows a tree view of the file system. The 'Home' folder is expanded, showing 'File Extents' and 'Permissions'. The 'File Extents' pane shows a list of file entries, with '1ST' selected.
- Table Pane:** A grid view showing file entries with columns for file number, size, and status. The entries are: 0005117, 0005160, 0005203, 0005246, 0005289, 0005332, 0005375, 0005418, and 0005461. The entry 0005332 is highlighted in blue.
- Bottom Pane:** A hex dump view showing the raw data of the selected file. The hex values are: 048 FF FF FF FF FF FF FF FF FF 00 00 FF FF FF FF, 064 B5 DF 20 20 20 20 20 20 20 20 10 00 39 56 2B, 080 97 3C 97 3C 00 00 57 2B 97 3C 03 00 00 00 00, 096 31 53 54 20 20 20 20 20 20 20 20 10 00 39 56 2B, 112 97 3C 97 3C 00 00 57 2B 97 3C 03 00 00 00 00, 128 B5 00 AC C8 B9 B0 C6 C0 C9 00 00 0F 00 81 FF FF, 144 FF FF FF FF FF FF FF FF FF 00 00 FF FF FF FF, 160 B5 A1 B8 B6 BF EC C1 F6 20 20 20 10 00 49 5D 2B, 176 97 3C 97 3C 00 00 60 2B 97 3C 04 00 00 00 00. The corresponding ASCII values are: yyyyyyyyyy...yyyy, ã¸ ..9V+, □<□<...W+□<....., 1ST ..9V+, □<□<...W+□<....., ã·-È'·ÈÈÈ...□yy, ã;·Œ;·iÁö ..I]+, □<□<...+□<.....

The bottom status bar shows the case information: Case 1 W4WC (PS 5336 LS 5304 CL 2 SO 096 FO 96 LE 32).

FAT 파일 시스템 – The Function of FAT

• The Effects of Deleting and Undeleting Files

- ✓ C:\1ST\file1234.txt (58,242 bytes) 파일의 위치 찾기

The screenshot displays the EnCase Forensic Training interface. On the left, a file list shows '1ST' selected. The main window shows a 'Bookmark Data' dialog box with 'DOS Directory Entry' selected in the 'Data Type' list. Below the dialog, a 'File Record' table is visible.

Name	Created	Written	Accessed	Size	Cluster
1ST	04/23/10 05:26:44오전	04/23/10 05:26:46오전	04/23/10	0	3

FAT 파일 시스템 – The Function of FAT

• The Effects of Deleting and Undeleting Files

- ✓ C:\1ST\file1234.txt (58,242 bytes) 파일의 위치 찾기

The screenshot displays the EnCase Forensic Training interface. The top menu bar includes File, Edit, View, Tools, and Help. Below the menu is a toolbar with icons for New, Open, Save, Print, Add Device, Search, Refresh, and Find. The main window is divided into several panes:

- Left Pane:** Shows a tree view of the file system. The root is 'C', with subfolders '1ST' and '2ND' visible.
- Table Pane:** A grid showing file entries. The first column lists file IDs (e.g., 0005117, 0005160, 0005203, 0005246, 0005289, 0005332, 0005375, 0005418, 0005461). The second column shows a grid of colored cells (green and blue) representing file attributes.
- Bottom Pane:** A hex dump view showing memory addresses and their corresponding hexadecimal values. The address 06446 is highlighted, and the corresponding hex value is 46 49 4C 45 31 32 33 34 54 58 54 20 18 49 AC 32. The text 'FILE1234TXT · I-2' is visible in the hex dump.
- Right Pane:** A tree view of the 'EnScript' directory, showing subfolders like 'Examples', 'Forensic', 'Include', 'Main', and 'Source Processor'.

The status bar at the bottom shows the current case information: Case 1 W4WCW1ST (PS 5344 L5 5312 CL 3 SO 064 FO 64 LE 32).

FAT 파일 시스템 – The Function of FAT

• The Effects of Deleting and Undeleting Files

- ✓ C:\1ST\file1234.txt (58,242 bytes) 파일의 위치 찾기

The screenshot displays the EnCase Forensic Training interface. On the left, a file explorer shows the directory structure: C:\1ST. The main window shows a 'Bookmark Data' dialog box with a 'Data Type' list where 'DOS Directory Entry' is selected. Below the dialog, a 'File Record' table is visible, showing the details for 'FILE1234.TXT'.

Name	Created	Written	Accessed	Size	Cluster
FILE1234.TXT	04/23/10 06:21:24오전	04/23/10 06:20:20오전	04/23/10	58242	8

FAT 파일 시스템 – The Function of FAT

• The Effects of Deleting and Undeleting Files

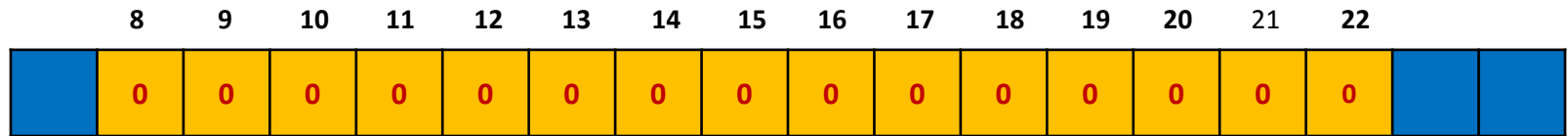
- ✓ C:\1ST\file1234.txt (58,242 bytes) 파일의 위치 찾기

The screenshot displays the EnCase Forensic Training application. The main window shows a file system analysis of a FAT volume. The left pane shows a tree view of the file system with folders '1ST' and '2ND'. The central pane shows a grid of data points, likely representing file entries or clusters. The bottom pane shows a hex and ASCII view of the data, with the ASCII view displaying a series of dots and some characters. The status bar at the bottom indicates 'Case 1 W4WC WPrimary FAT (PS 68 LS 36 SO 032 FO 32 LE 60)'.

FAT 파일 시스템 – The Function of FAT

• The Effects of Deleting and Undeleting Files

- ✓ C:\1ST\file1234.txt (58,242 bytes) 파일의 삭제



Root Directory

Name	Starting Cluster
1ST	3

Cluster 3

0xE5 ile1234.txt	8

- **Introduction**

- ✓ New Technology File System
- ✓ FAT의 한계점을 개선한 파일시스템
- ✓ Windows NT 이후부터 사용

버 전	운 영 체 제
1.0	Windows NT 3.1
1.1	Windows NT 3.5
1.2	Windows NT 3.51
3.0	Windows 2000
3.1 – 5.1	Windows XP
5.2	Windows 2003
6.0	Windows Vista, 2008, 7

• Features

특징	설명
USN 저널 (Update Sequence Number Journal)	파일의 모든 변경 내용을 기록하는 로그 시스템 오류 발생으로 재부팅 될 경우 잘못된 처리 작업을 롤백(Rollback)
ADS (Alternate Data Stream)	파일당 하나 이상의 데이터 스트림을 저장할 수 있도록 지원 파일 이름, 소유자, 시간 정보 등을 스트림 통해 표현, 데이터도 하나의 데이터 스트림으로 표현 추가된 ADS는 정보 은닉 용도로 사용될 수 있음
Sparse 파일	파일 데이터가 대부분 0일 경우 실제 데이터는 기록하지 않고 정보만 기록
파일 압축	LZ77의 변형된 알고리즘을 사용하여 파일 데이터 압축
EFS (Encrypting File System)	파일을 암호화 하는 기능으로 빠른 암,복호화를 위해 FEK(File Encryption Key)를 통한 대칭키 방식의 암호화 수행
VSS (Volume Shadow Copy Service)	윈도우2003부터 지원, 새롭게 덮여 쓰인 파일, 디렉터리에 대해 백업본을 유지하여 USN저널과 함께 좀 더 안전한 복구를 도움
Quotas	사용자 별 디스크 사용량 제한
유니코드 지원	다국어 지원 (파일, 디렉터리, 볼륨 이름 모두 유니코드로 저장)
대용량 지원	이론상 Exa Byte(2^{64}), 실제로는 약 16 TB (2^{44})
동적 배드 클러스터 재할당	배드섹터 발생 클러스터의 데이터를 자동으로 새로운 클러스터로 복사, 배드섹터 발생 클러스터는 플래그 통해 더 이상 사용되지 않도록 함

- NTFS 구조



- ✓ NTFS는 파일, 디렉터리 및 메타 정보까지 파일 형태로 관리
- ✓ VBR영역 : 부트 섹터, 추가적인 부트 코드가 저장되는 부분
- ✓ MFT영역 : 파일과 디렉토리를 관리하기 위한 MFT Entry의 집합체
 - 각 파일은 위치, 시간 정보, 크기, 파일 이름 등을 MFT Entry라는 특별한 구조로 저장
 - 크기가 가변적, 해당 MFT가 모두 사용되면 동적으로 클러스터를 추가로 할당해 MFT영역의 크기를 증가 시키므로 파일 시스템의 여러 부분에 조각나 분포될 수 있음
 - MFT(Master File Table)은 NTFS 상의 모든 MFT Entry들의 배열
 - MFT Entry 0 ~ 15번은 파일 시스템 생성시 함께 생성되어 특별한 용도로 사용
- ✓ DATA영역 : 파일의 실제 내용이 저장되는 공간, 내용만 저장됨

NTFS – VBR(Volume Boot Record)

- **VBR (Volume Boot Record)**

- ✓ NTFS로 포맷된 드라이브의 가장 앞부분에 위치, 부트 섹터와 추가적인 부트 코드 저장

클러스터 크기에 따른 VBR크기

클러스터 크기(Byte)	VBR 크기(Sector)
512	1
1K	2
2K	4
4K	8

- ✓ VBR의 첫 섹터에는 부트 코드를 포함한 부트 섹터가 위치
- ✓ 부트 섹터는 FAT의 부트 섹터와 유사한 구조로 BPB를 포함

NTFS 부트 섹터 구조

범위(Byte Range)		설 명
십진수	십육진수	
0 - 2	0x0000 - 0x0002	Jump command to boot code
3 - 10	0x0003 - 0x000A	OEM ID
11 - 83	0x000B - 0x0053	BIOS Parameter Block
84 - 509	0x0054 - 0x01FD	Boot code and error message
510 - 511	0x01FE - 0x01FF	Signature

NTFS – VBR(Volume Boot Record)

• VBR

- ✓ NTFS의 부트 섹터가 FAT 부트 섹터와 크게 다른 점은 BPB영역

NTFS 부트 섹터 데이터 구조

범위(Byte Range)		설 명
십진수	십육진수	
0 - 2	0x0000 - 0x0002	Jump command to boot code (usually 0xEB5290)
3 - 10	0x0003 - 0x000A	OEM ID (typically "NTFS ")
11 - 12	0x000B - 0x000C	Bytes per sector
13 - 13	0x000D - 0x000D	Sectors per cluster
14 - 15	0x000E - 0x000F	Reserved sectors
16 - 18	0x0010 - 0x0012	Always 0
19 - 20	0x0013 - 0x0014	Unused
21 - 21	0x0015 - 0x0015	Media descriptor
22 - 23	0x0016 - 0x0017	Always 0
24 - 25	0x0018 - 0x0019	Sectors per track
26 - 27	0x001A - 0x001B	Number of heads
28 - 31	0x001C - 0x001F	Hidden sectors
32 - 35	0x0020 - 0x0023	Unused
36 - 39	0x0024 - 0x0027	Unused
40 - 47	0x0028 - 0x002F	Total sectors
48 - 55	0x0030 - 0x0037	Logical cluster Number for the file \$MFT
56 - 63	0x0038 - 0x003F	Logical cluster Number for the file \$MFTMirr
64 - 67	0x0040 - 0x0043	Clusters per file record segment
68 - 71	0x0044 - 0x0047	Clusters per index block
72 - 79	0x0048 - 0x004F	Volume serial number
80 - 83	0x0050 - 0x0053	Checksum
84 - 509	0x0054 - 0x01FD	Boot code and error message
510 - 511	0x01FE - 0x01FF	Signature ("0x55AA")

NTFS 부트 섹터 덤프

Jump command	OEM ID	BIOS Parameter Block	
000 EB 52 90	4E 54 46 53	20 20 20 20	00 02 08 00 00
016 00 00 00	00 00 F8 00	00 00 3F 00	FF 00 3F 00 00 00
032 00 00 00	00 80 00 80	00 5F A2 98	25 00 00 00 00 00
048 00 00 00	00 00 00 00	00 91 22 01	00 00 00 00 00 00
064 F6 00 00	00 01 00 00	00 00 8E D5	02 10 08 03 10 F0
080 00 00 00	00 FA 33 C0	8E D0 BC 00	7C FB B8 C0 07
096 8E D8 E8	16 00 B8 00	0D 8E C0 33	DB C6 06 0E 00
112 10 E8 53	00 68 00 0D	68 6A 02 CB	8A 16 24 00 B4
128 08 CD 13	73 05 B9 FF	FF 8A F1 66	0F B6 C6 40 66
144 0F B6 D1	80 E2 3F F7	E2 86 CD C0	ED 06 41 66 0F
160 B7 C9 66	F7 E1 66 A3	20 00 C3 B4	41 BB AA 55 8A
176 16 24 00	CD 13 72 0F	81 FB 55 AA	75 09 F6 C1 01
192 74 04 FE	06 14 00 C3	66 60 1E 06	66 A1 10 00 66
208 03 06 1C	00 66 3B 06	20 00 0F 82	3A 00 1E 66 6A
224 00 66 50	06 53 66 68	10 00 01 00	80 3E 14 00 00
240 0F 85 0C	00 E8 B3 FF	80 3E 14 00	00 0F 84 61 00
256 B4 42 8A	16 24 00 16	1F 8B F4 CD	13 66 58 5B 07
272 66 58 66	58 1F EB 2D	66 33 D2 66	0F B7 0E 18 00
288 66 F7 F1	FE C2 8A CA	66 8B D0 66	C1 EA 10 F7 36
304 1A 00 86	D6 8A 16 24	00 8A E8 C0	E4 06 0A CC B8
320 01 02 CD	13 0F 82 19	00 8C C0 05	20 00 8E C0 66
336 FF 06 10	00 FF 0E 0E	00 0F 85 6F	FF 07 1F 66 61
352 C3 A0 F8	01 E8 09 00	A0 FB 01 E8	03 00 FB EB FE
368 B4 01 8B	F0 AC 3C 00	74 09 B4 0E	BB 07 00 CD 10
384 EB F2 C3	0D 0A 41 20	64 69 73 6B	20 72 65 61 64
400 20 65 72	72 6F 72 20	6F 63 63 75	72 72 65 64 00
416 0D 0A 4E	54 4C 44 52	20 69 73 20	6D 69 73 73 69
432 6E 67 00	0D 0A 4E 54	4C 44 52 20	69 73 20 63 6F
448 6D 70 72	65 73 73 65	64 00 0D 0A	50 72 65 73 73
464 20 43 74	72 6C 2B 41	6C 74 2B 44	65 6C 20 74 6F
480 20 72 65	73 74 61 72	74 0D 0A 00	00 00 00 00 00
496 00 00 00	00 00 00 00	00 83 A0 B3	C9 00 00 55 AA

NTFS – MFT(Master File Table)

- MFT Entry

NTFS 구조



MFT Entry 구조

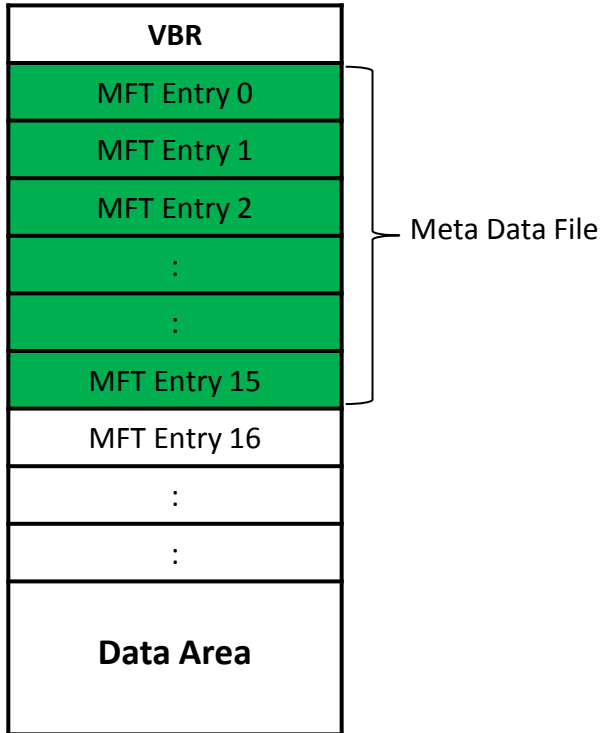


- ✓ **MFT Entry** : 파일의 메타 정보(위치, 시간 정보, 크기, 파일 이름 등)들을 저장 (NTFS는 모두 파일 형태로 관리)
- ✓ **Attributes** : 각 메타 정보 표현 (시간, 이름, 내용 등)
- ✓ 메타 정보의 크기에 따라 각 파일은 하나 이상의 MFT Entry를 가짐
- ✓ 0 ~ 15번은 파일 시스템을 생성할 때 함께 생성 되어 NTFS의 다양한 특성을 지원
- ✓ 사용자에게 의해 파일이 생성될 때마다 새로운 MFT Entry가 할당되어 해당 파일의 정보를 유지 및 관리

NTFS – MFT(Master File Table)

예약된 MFT Entry

• Master File Table

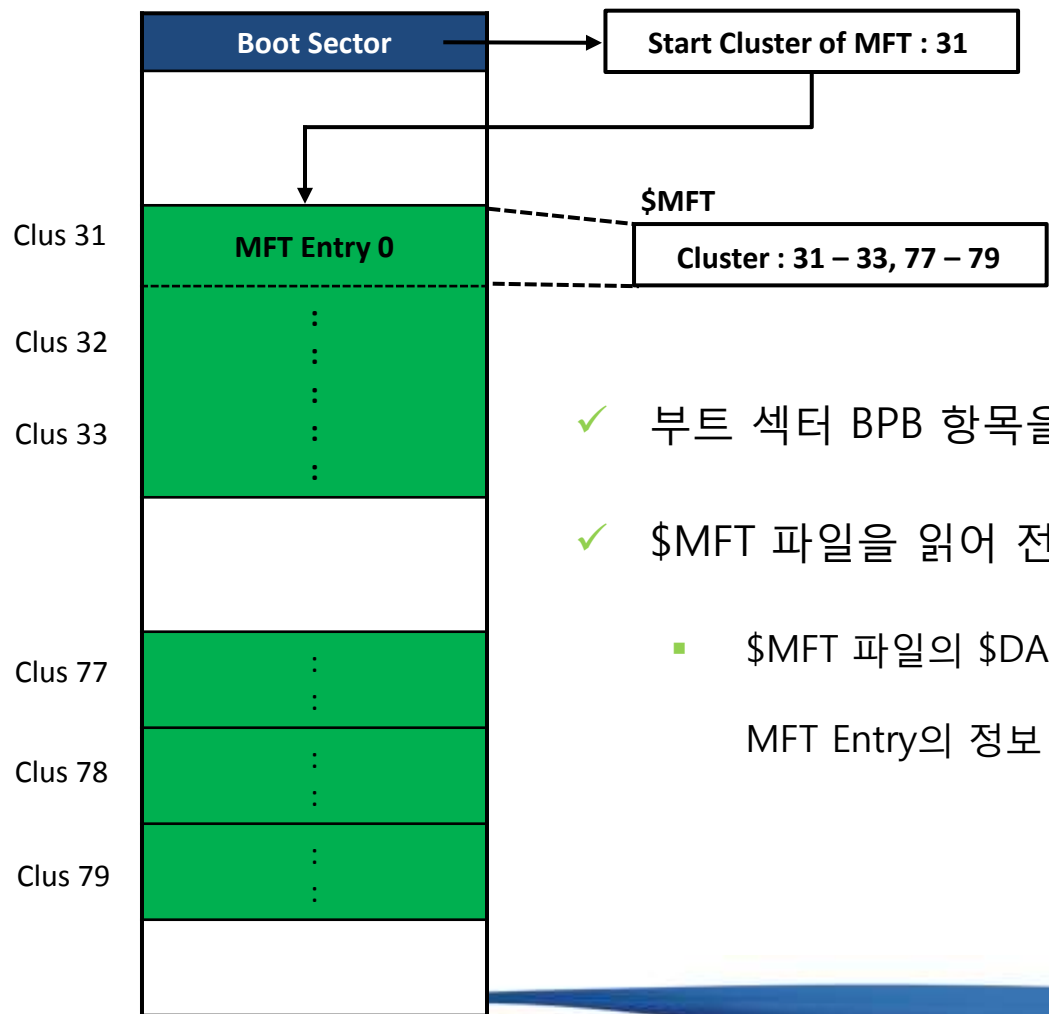


Entry 번호	Entry 이름	설명
0	\$MFT	NTFS 상의 모든 파일들의 MFT Entry 정보 가짐
1	\$MFTMirr	\$MFT 파일의 일부 백업본
2	\$LogFile	메타데이터의 트랜잭션 저널 정보
3	\$Volume	볼륨의 레이블, 식별자, 버전 등의 정보
4	\$AttrDef	속성의 식별자, 이름, 크기 등의 정보
5	.	볼륨의 루트 디렉터리
6	\$Bitmap	볼륨의 클러스터 할당 정보
7	\$Boot	볼륨이 부팅 가능할 경우 부트 섹터 정보
8	\$BadClus	배드 섹터를 가지는 클러스터 정보
9	\$Secure	파일의 보안, 접근제어와 관련된 정보
10	\$Upcase	모든 유니코드 문자의 대문자
11	\$Extend	\$ObjID, \$Quota, \$Reparse points, \$UsnJrnl 등의 추가적인 파일의 정보를 기록하기 위해 사용
12 – 15		예약 영역
16 -		포맷 후 생성되는 파일의 정보를 위해 사용
-	\$ObjId	파일 고유의 ID 정보 (Windows 2000 -)
-	\$Quota	사용량 정보 (Windows 2000 -)
-	\$Reparse	Reparse Point 에 대한 정보 (Windows 2000 -)
-	\$UsnJrnl	파일, 디렉터리의 변경 정보 (Windows 2000 -)

- ✓ MFT는 파일 시스템의 여러 영역에 조각나서 존재 가능하므로, \$MFT파일이 MFT 전체 정보를 유지 관리 하는 역할 수행
- ✓ 각 파일들 분석 위해 전체 MFT 정보를 획득이 선행되어야 함

NTFS – MFT(Master File Table)

- Master File Table Layout - \$MFT 파일 획득 방법



- ✓ 부트 섹터 BPB 항목을 읽은 후 MFT 시작 위치 파악
- ✓ \$MFT 파일을 읽어 전체적인 MFT의 레이아웃을 판단
 - \$MFT 파일의 \$DATA 속성 정보를 확인하여 조각나 있는 MFT Entry의 정보 확인

NTFS – MFT(Master File Table)



MFT Entry 구조

- ✓ MFT Entry는 1,024 바이트의 고정된 크기
- ✓ 42 Bytes의 고정된 헤더, Fixup배열, 속성이 위치
- ✓ 속성들의 정보가 많아져 하나 이상의 MFT Entry를 사용할 경우 마지막 MFT Entry의 끝에 End Marker가 표시(0xFFFFFFFF)
- ✓ Fixup 배열은 저장하고자 하는 데이터가 하나 이상의 섹터를 사용할 경우 각 섹터의 마지막 2바이트를 따로 저장하여 두는 것
 - MFT Entries
 - INDEX Records
 - RCRD Records
 - RSTR Records

NTFS – MFT(Master File Table)

MFT Entry 데이터 구조

범위(Byte Range)		설 명
십진수	십육진수	
0 - 3	0x0000 - 0x0003	Signature("FILE")
4 - 5	0x0004 - 0x0005	Offset to fixup array
6 - 7	0x0006 - 0x0007	Number of entries in fixup array
8 - 15	0x0008 - 0x000F	\$LogFile Sequence Number (LSN)
16 - 17	0x0010 - 0x0011	Sequence value
18 - 19	0x0012 - 0x0013	Link count
20 - 21	0x0014 - 0x0015	Offset to first attribute
22 - 23	0x0016 - 0x0017	Flags (in-use and directory)
24 - 27	0x0018 - 0x001B	Used size of MFT Entry
28 - 31	0x001C - 0x001F	Allocated size of MFT Entry
32 - 39	0x0020 - 0x0027	File reference to base record
40 - 41	0x0028 - 0x0029	Next attribute ID
42 - 1023	0x002A - 0x03FF	Attributes and fixup values

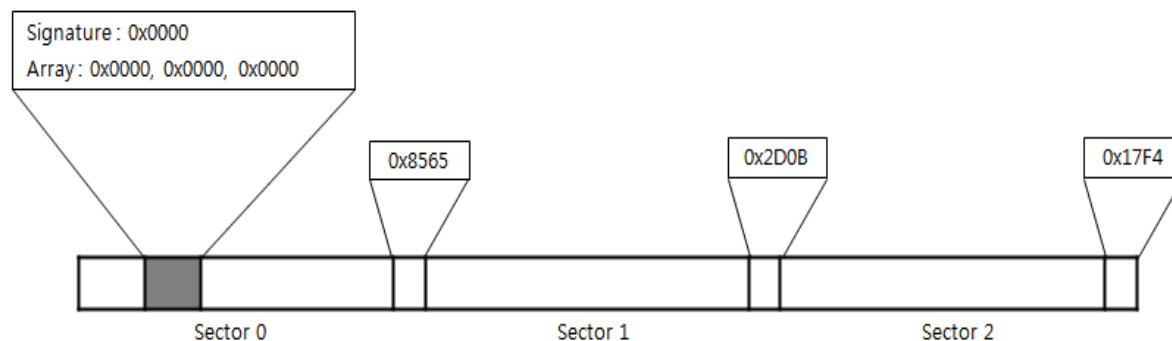
MFT Entry Header

Fixup Array &
Attributes

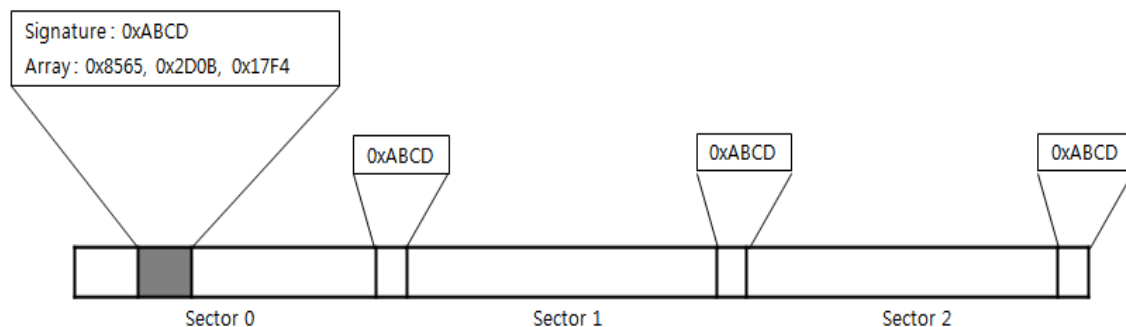
NTFS – MFT(Master File Table)

- **Fixup**

- ✓ 3개의 섹터를 사용하는 데이터므로 각 섹터의 마지막 2바이트를 별도로 저장해 Fixup 배열을 만든 후 마지막 2 바이트 공간에 별도의 시그니처를 저장
 - 해당 데이터가 저장되는 섹터의 이상 유무 점검 위해



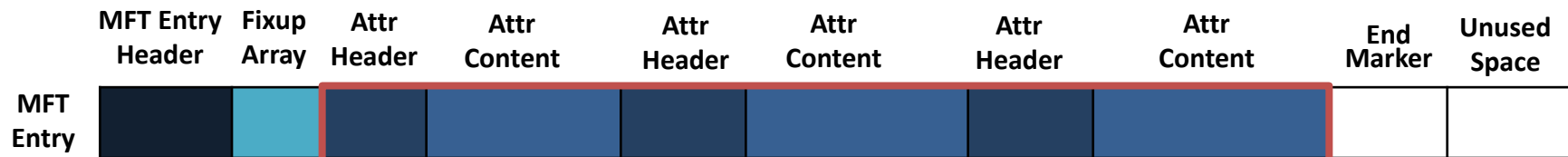
Fixup 적용되기 이전 상태



Fixup 적용된 후의 상태

NTFS – MFT(Master File Table)

- MFT Attributes



- ✓ 각 파일의 메타 정보(시간정보, 이름, 내용 등)는 속성이라는 구조를 통해 관리
- ✓ 각 속성은 속성 헤더와 속성 내용을 가짐
- ✓ 파일의 특성에 따라 다양한 속성을 가짐

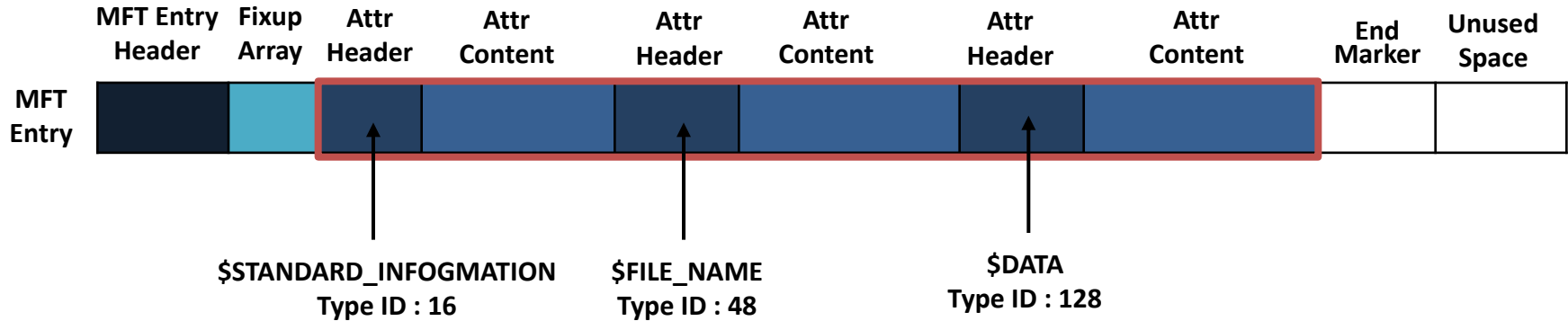
NTFS – MFT(Master File Table)

- MFT 속성 리스트

Attr Type Num	Attr Name	Description
16	\$STANDARD_INFORMATION	파일의 생성.접근.수정 시간, 소유자 등의 일반적인 정보
32	\$ATTRIBUTE_LIST	추가적인 속성들의 리스트
48	\$FILE_NAME	파일 이름(유니코드), 파일의 생성.접근.수정 시간
64	\$VOLUME_VERSION	볼륨 정보 (Windows NT 1.2 버전에만 존재)
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 이름
112	\$VOLUME_INFORMATION	파일 시스템의 버전과 다양한 플래그
128	\$DATA	파일 내용
144	\$INDEX_ROOT	인덱스 트리의 루트 노드
160	\$INDEX_ALLOCATION	인덱스 트리의 루트와 연결된 노드
176	\$BITMAP	\$MFT와 인덱스의 할당 정보 관리
192	\$SYMBOLIC_LINK	심볼릭 링크 정보 (Windows 2000+)
192	\$REPARSE_POINT	심볼릭 링크에서 사용하는 reparse point 정보 (Windows 2000+)
208	\$EA_INFORMATION	OS/2 응용 프로그램과 호환성을 위해 사용 (HPFS)
224	\$EA	OS/2 응용 프로그램과 호환성을 위해 사용 (HPFS)
256	\$LOGGED_UTILITY_STREAM	암호화된 속성의 정보와 키 값 (Windows 2000+)

NTFS – MFT(Master File Table)

• MFT Attributes



- ✓ 기본적인 파일은 위와 같이 3개의 속성을 가짐
 - **\$STANDARD_INFORMATION** : 파일의 생성.접근.수정 시간, 소유자 등의 정보
 - **\$FILE_NAME** : 파일 이름(유니코드), 파일의 생성.접근.수정 시간
 - **\$DATA** : 파일 내용
- ✓ 해당 속성이 Resident, Non-resident 이냐에 따라 속성 헤더 항목의 데이터 구조 다름

NTFS – MFT(Master File Table)

\$STANDARD_INFORMATION 데이터 구조

범위(Byte Range)		설명
십진수	십육진수	
~	~	Attribute Header
0 - 7	0x0000 - 0x0007	Creation time
8 - 15	0x0008 - 0x000F	File altered time
16 - 23	0x0010 - 0x0017	MFT altered time
24 - 31	0x0018 - 0x001F	File accessed time
32 - 35	0x0020 - 0x0023	Flags
36 - 39	0x0024 - 0x0027	Maximum number of versions
40 - 43	0x0028 - 0x002B	Version number
44 - 47	0x002C - 0x002F	Class ID
48 - 51	0x0030 - 0x0033	Owner ID (version 3.0+)
52 - 55	0x0034 - 0x0037	Security ID (version 3.0+)
56 - 63	0x0038 - 0x003F	Quota Charged (version 3.0+)
64 - 71	0x0040 - 0x0047	Update Sequence Number(USN) (version 3.0+)

\$FILE_NAME 데이터 구조

범위(Byte Range)		설명
십진수	십육진수	
~	~	Attribute Header
0 - 7	0x0000 - 0x0007	File reference of parent directory
8 - 15	0x0008 - 0x000F	File creation time
16 - 23	0x0010 - 0x0017	File modification time
24 - 31	0x0018 - 0x001F	MFT modification time
32 - 39	0x0020 - 0x0027	File access time
40 - 47	0x0028 - 0x002F	Allocated size of file
48 - 55	0x0030 - 0x0037	Real size of file
56 - 59	0x0038 - 0x003B	Flags
60 - 63	0x003C - 0x003F	Reparse value
64 - 64	0x0040 - 0x0040	Length of name
65 - 65	0x0041 - 0x0041	Namespace
66 -	0x0042 -	Name

NTFS – MFT(Master File Table)

일반적인 속성 헤더의 데이터 구조

범위(Byte Range)		설명
십진수	십육진수	
0 - 3	0x0000 - 0x0003	Attribute type identifier
4 - 7	0x0004 - 0x0007	Length of attribute
8 - 8	0x0008 - 0x0008	Non-resident flag
9 - 9	0x0009 - 0x0009	Length of name
10 - 11	0x000A - 0x000B	Offset to name
12 - 13	0x000C - 0x000D	Flags
14 - 15	0x000E - 0x000F	Attribute identifier

추가적인 Resident 속성 헤더의 데이터 구조

범위(Byte Range)		설명
십진수	십육진수	
0 - 15	0x0000 - 0x000F	General header
16 - 19	0x0010 - 0x0013	Size of content
20 - 21	0x0014 - 0x0015	Offset to content

추가적인 Non-Resident 속성 헤더의 데이터 구조

범위(Byte Range)		설명
십진수	십육진수	
0 - 15	0x0000 - 0x000F	General header
16 - 23	0x0010 - 0x0017	Starting Virtual Cluster Number(VCN) of the runlist
24 - 31	0x0018 - 0x001F	Ending VCN of the runlist
32 - 33	0x0020 - 0x0021	Offset to the runlist
34 - 35	0x0022 - 0x0023	Compression unit size
36 - 39	0x0024 - 0x0027	Unused
40 - 47	0x0028 - 0x002F	Allocated size of attribute content
48 - 55	0x0030 - 0x0037	Actual size of attribute content
56 - 63	0x0038 - 0x003F	Initialized size of attribute content

NTFS – MFT(Master File Table)

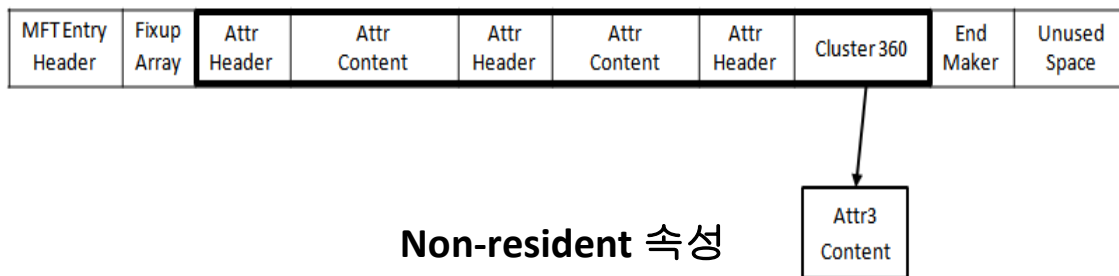
• Resident 속성과 Non-resident 속성

✓ Resident 속성

- MFT Entry 내에 속성 헤더와 속성 내용이 모두 저장되는 경우
- 속성 내용이 많아지면 여러 개의 MFT Entry를 사용
- MFT Entry로 감당 하지 못할 경우 별도의 클러스터에 해당 속성을 저장, 속성 내용에는 저장한 클러스터의 위치 정보만 저장

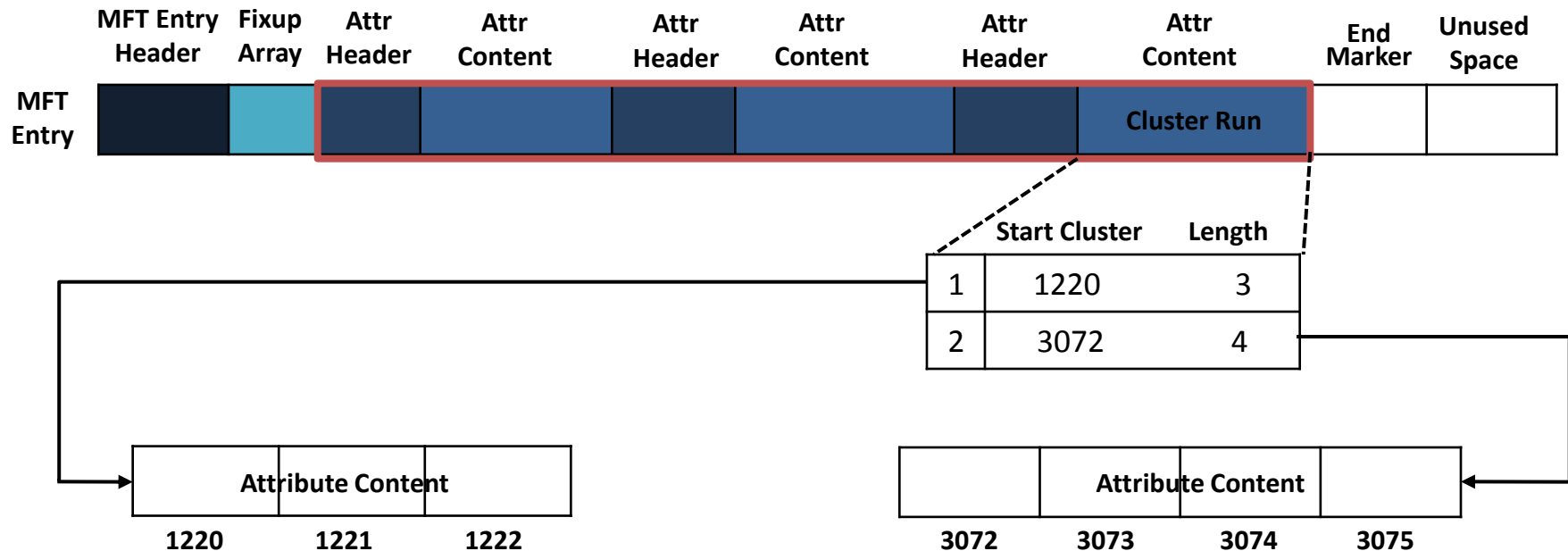
✓ Non-resident 속성

- 형태는 속성 내용이 증가할 수 있는 \$ATTRIBUTE_LIST나 \$DATA속성
 - \$DATA 속성 내용이 700 바이트 이하일 경우 Resident 속성으로 저장, 넘을 경우 Non-resident 속성으로 저장
- 파일이 크기가 700 바이트 보다 작으면 별도의 클러스터 할당 없이 해당 MFT Entry에 함께 저장



NTFS – MFT(Master File Table)

- Cluster Run



- ✓ Non-resident로 저장된 속성의 크기가 매우 클 경우 저장하기 위한 많은 클러스터가 할당
- ✓ Cluster Run을 사용하여 클러스터의 시작 위치와 수를 표현
 - (\$DATA) Cluster Run 형태로 해당 \$DATA 속성 내용을 관리

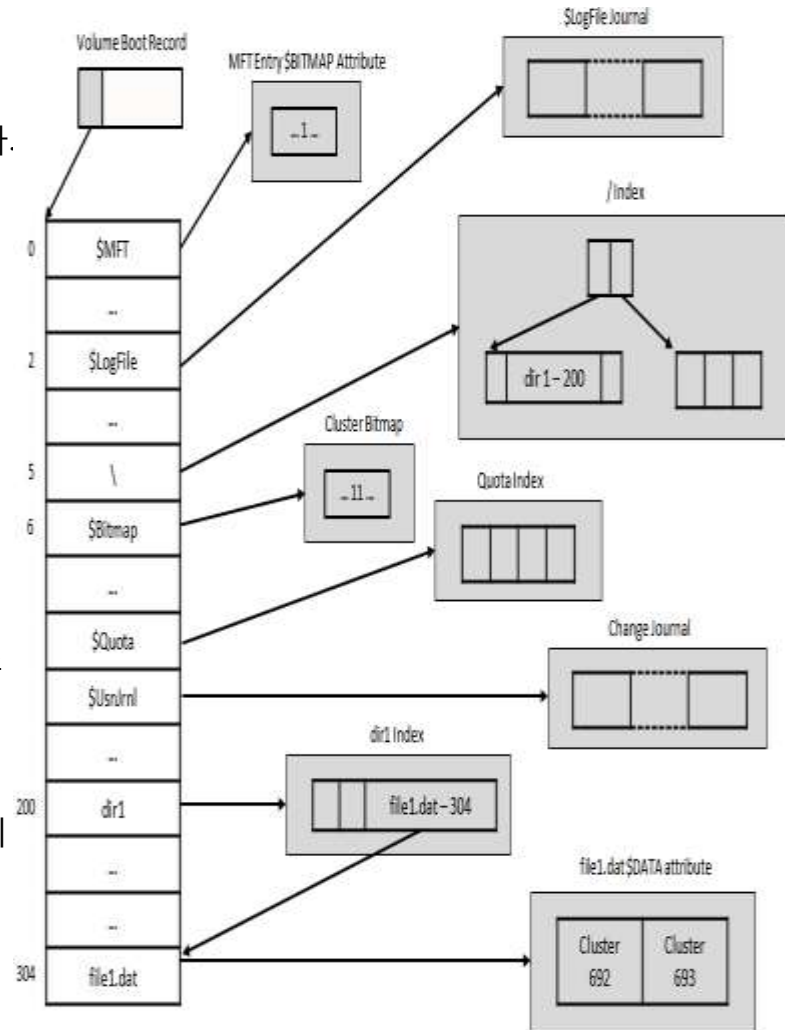
- **Data Area**

- ✓ DATA 영역은 파일의 실제 내용이 저장되는 공간
- ✓ 모든 관련 정보는 MFT Entry의 속성을 통해 관리되므로 특별한 구조 없이 내용만 저장

NTFS – 파일의 할당으로 인한 변화 요소

₩dir1 디렉터리는 이미 생성되어 있고 클러스터 크기는 2,048 바이트, 파일 크기는 4,000 바이트이라 하자.

1. VBR의 BPB 정보에 MFT 시작위치 정보를 얻어 MFT 시작위치로 이동한다.
2. \$MFT(MFT Entry 0) 파일을 읽어 전체 MFT 구조를 파악한다.
3. \$MFT 파일의 \$BITMAP 속성에서 현재 사용되지 않는 MFT Entry를 검색한다. MFT Entry 304번이 미사용 중이므로 Entry를 할당한 후 \$BITMAP 속성에서 해당 Entry 위치의 비트를 1로 세트한다.
4. MFT Entry 304번을 초기화한 후 \$STANDARD_INFORMATION, \$FILE_NAME 속성을 기록하고 MFT Entry의 in-use 플래그를 세트한다.
5. file1.dat는 2개의 클러스터가 필요하므로 \$Bitmap(MFT Entry 6) 파일의 클러스터 할당 정보에서 사용할 클러스터를 검색한다. 할당 알고리즘에 의해 연속된 2개의 클러스터 692, 693번이 선택된다. 그리고 해당 클러스터에 해당하는 비트를 1로 세트한다.
6. 루트 디렉터리(MFT Entry 5)파일에서 ₩dir1의 위치를 검색한다.
7. ₩dir1의 위치인 MFT Entry 200번에서 새로운 파일에 대한 인덱스 엔트리를 생성하면 인덱스가 재배열된다. 이 경우 디렉터리의 last written, modified, accessed time이 변경된다.
8. 마지막으로 각 작업에 대해 \$LogFile과 ₩Extend₩UsnJrnl 파일에 로그 정보가 기록되고 Quotas 기능을 사용 중일 경우 해당 사용자의 할당량을 관리하는 ₩Extend₩Quota 정보가 수정된다.

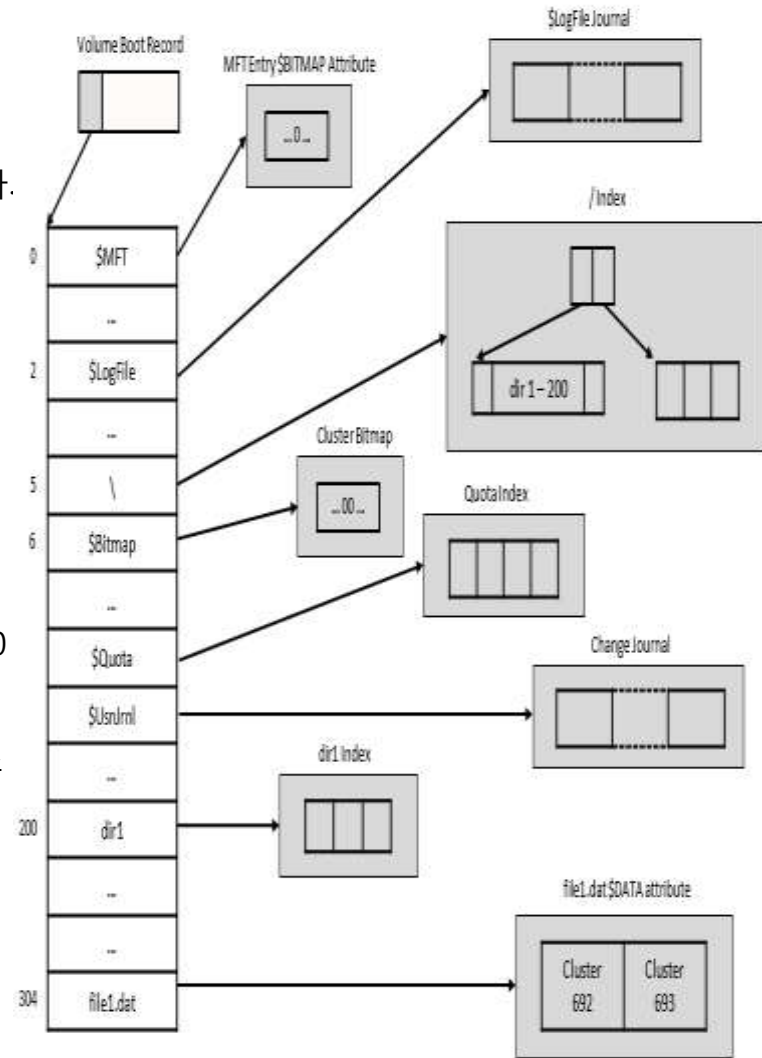


₩dir1₩file1.dat 파일 할당 시 NTFS 파일 시스템 변화

NTFS – 파일의 삭제로 인한 변화 요소

₩dir1 디렉터리는 이미 생성되어 있고 클러스터 크기는 2,048 바이트, 파일 크기는 4,000 바이트이라 하자.

1. VBR의 BPB 정보에 MFT 시작위치 정보를 얻어 MFT 시작위치로 이동한다.
2. \$MFT(MFT Entry 0) 파일을 읽어 전체 MFT 구조를 파악한다.
3. 루트 디렉터리(MFT Entry 5) 파일의 \$INDEX_ROOT와 \$INDEX_ALLOCATION 속성에서 ₩dir1 Entry 위치를 탐색한다.
4. 루트 디렉터리(MFT Entry 5) 파일의 \$INDEX_ROOT와 \$INDEX_ALLOCATION 속성에서 ₩dir1 Entry 위치를 탐색한다.
5. MFT Entry 200번에서 file1.dat와 관련된 인덱스 엔트리를 삭제한다. 이 경우 디렉터리의 last written, modified, accessed time이 변경된다.
6. \$MFT 파일의 \$BITMAP 속성에서 삭제된 MFT Entry에 해당하는 비트를 0으로 세트한다.
7. \$Bitmap 파일에서 삭제 파일의 \$DATA 속성 내용으로 할당되었던 클러스터에 관련 비트를 0으로 세트한다.
8. 마지막으로 각 작업에 대해 \$LogFile과 ₩UsnJrnl 파일에 로그 정보가 기록되고 Quotas 기능을 사용 중일 경우 해당 사용자의 할당량을 관리하는 ₩Extend₩Quota 정보가 수정된다.



₩dir1₩file1.dat 파일 삭제 시 NTFS 파일 시스템 변화

디지털 포렌식 관점에서의 파일 시스템 분석

● 파일 시스템 상의 삭제 파일 복구(Deleted Files Analysis)

- 삭제한 파일은 다른 파일보다 우선 분석
- Directory Entry나 MFT Entry가 덮여 쓰이지 않았다면 복구 가능

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

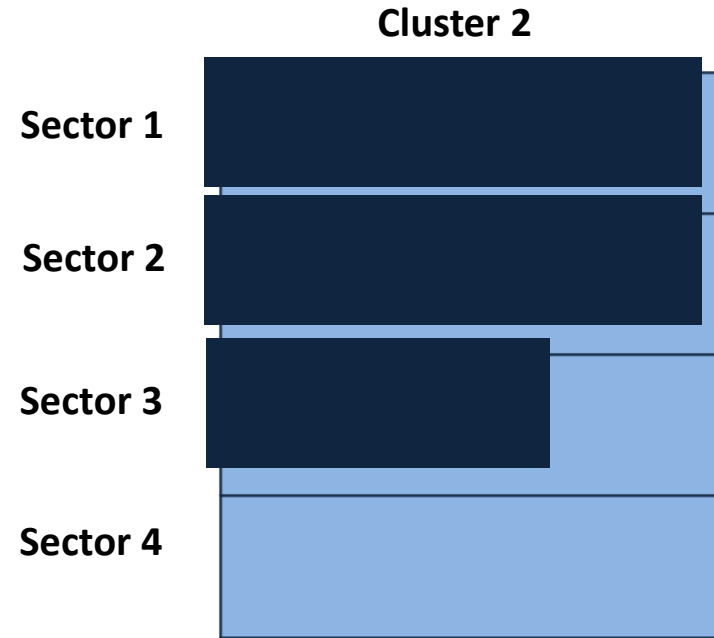
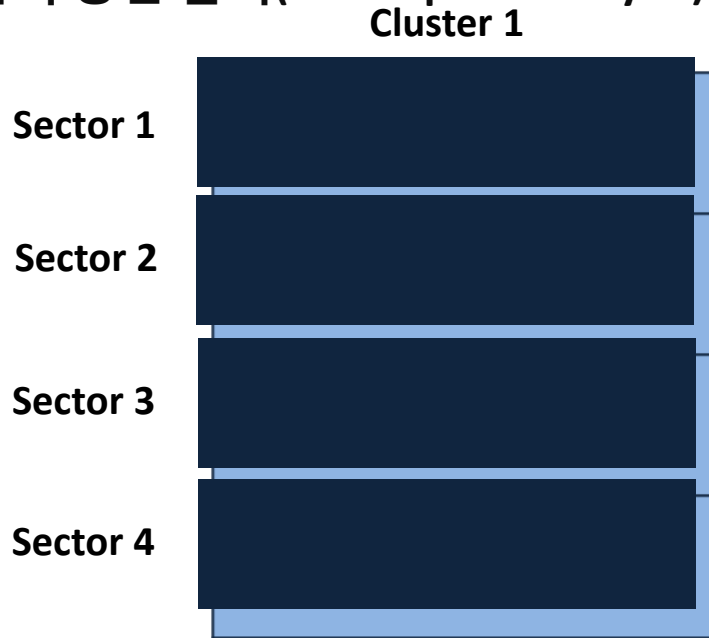
디지털 포렌식 관점에서의 파일 시스템 분석

● 비 할당 클러스터 분석(Unallocated Clusters Analysis)

- ✓ 비 할당 클러스터 : 메타정보를 통해 접근할 수 없는 클러스터
- ✓ 대용량의 하드디스크 사용으로 많은 양의 공간이 비 할당 영역일 가능성
- ✓ 비 할당된 클러스터는 이전 데이터가 남아 있을 가능성
 - 포맷하기 이전의 데이터
 - 포맷한 후 할당되었다가 삭제된 데이터
- ✓ 비 할당 클러스터 판별
 - FAT : FAT 영역에서 0x00 값을 갖는 클러스터
 - NTFS : MFT Entry 6번의 \$Bitmap 파일로부터 할당되지 않은 클러스터를 조사
클러스터 비트맵에서 0x00 값을 갖는 클러스터

디지털 포렌식 관점에서의 파일 시스템 분석

● 슬랙 공간 분석(Slack Space Analysis)



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

디지털 포렌식 관점에서의 파일 시스템 분석

시간 정보 분석(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)

이름	설명
Created Time	파일이 생성된 시간
Created Date	파일이 생성된 날짜
Accessed Date	마지막으로 파일 내용에 접근한 날짜
Written Time	마지막으로 파일 내용을 수정한 시간
Written Date	마지막으로 파일 내용을 수정한 날짜

이름	설명
Creation Time	파일이 생성된 시간
Modified Time	마지막으로 파일 내용이 수정된 시간
MFT Modified Time	MFT 내용이 마지막으로 수정된 시간
Accessed Time	마지막으로 파일 내용에 접근한 시간

디지털 포렌식 관점에서의 파일 시스템 분석

• 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

디지털 포렌식 관점에서의 파일 시스템 분석

● 부트 코드 분석

- ✓ MBR 부트 코드 : 파티션 테이블 읽어 부팅 가능한 파티션의 부트 섹터 호출 역할
- ✓ 부트 섹터의 부트 코드 : 파일 시스템의 BPB 활용하여 부트 로더 호출 역할
- ✓ 분석 방법
 - ✓ MBR : 부트 코드 해석하여 부팅 가능한 파티션의 시작위치로 점프하는지 확인
 - ✓ 부트 섹터 : 부트 코드를 해석하여 정상적으로 부트 로더를 로드 하는지 확인

디지털 포렌식 관점에서의 파일 시스템 분석

• 미사용 영역 분석

- ✓ 미래를 위해 예약해 둔 영역, 불필요하게 생성된 영역
- ✓ 기본적으로 참조하는 영역이 아니므로 쉽게 파악 어려움

- ✓ 파일 시스템 별 미사용 영역
 - ✓ FAT : MBR과 예약 영역 사이 / 예약 영역에서 사용하지 않는 섹터(0,1,2,6,7,8번 제외) / FSINFO 구조체 섹터(예약 영역의 1, 7번 섹터)에서 사용되지 않는 영역
 - ✓ NTFS : VBR에서 부트 섹터를 제외한 나머지 섹터 / 미래를 위해 예약해 둔 MFT Entry 12 ~ 15번 영역

디지털 포렌식 관점에서의 파일 시스템 분석

• 은닉 파일 분석

- ✓ 파일 시스템에서 숨긴 속성을 가진 파일을 분류해 분석하는 방안 필요
- ✓ 파일 시스템 별 숨긴 속성 확인 방법
 - ✓ FAT : 파일의 Directory Entry 항목 중 오프셋 11의 Attribute가 0x02값인 것 조사
 - ✓ NTFS : 파일의 MFT Entry에서 \$STANDARD_INFORMATION 속성의 오프셋 32 ~ 35 Flags가 0x0002인 것 조사

디지털 포렌식 관점에서의 파일 시스템 분석

• 암호 파일 분석

✓ NTFS는 EFS에 의해 파일 시스템 수준에서 암호화 기능 제공

✓ NTFS에서 암호화 속성을 확인하는 방법

파일의 MFT Entry에서 \$STANDARD_INFORMATION 속성의 오프셋 32 ~ 35 Flags가 0x4000인 것 조사

디지털 포렌식 관점에서의 파일 시스템 분석

● ADS 파일 분석

- ✓ NTFS는 하나의 파일이 두 개 이상의 데이터 속성을 가질 수 있는 ADS를 지원
- ✓ 운영체제 통해 확인할 수 없으므로 데이터 은닉 목적으로 이용될 가능성 존재
- ✓ NTFS에서 ADS 파일을 확인하는 방법
전체 MFT Entry를 대상으로 \$DATA 속성을 두 개 이상 가지는 MFT Entry를 조사

디지털 포렌식 관점에서의 파일 시스템 분석

● 로그 정보 분석

✓ 사건 발생 시점의 파일 시스템 변경 사항들을 조합하여 용의자의 행위를 파악

✓ NTFS 파일 시스템의 변경 사항 기록되는 파일 조사

MFT Entry 2번인 \$LogFile과 MFT Entry 11번인 \$Extend 파일에 포함된 \$Extend\UsnJrnl
파일 조사

디지털 포렌식 관점에서의 파일 시스템 분석

• \$Boot 파일 분석

- ✓ NTFS의 MFT Entry 7번인 \$Boot 파일의 \$DATA 속성에서 부트 섹터의 위치 정보,
부트 코드가 저장 됨
- ✓ 부팅 용도로 사용되지 않는 NTFS에서의 \$DATA 속성에 데이터 은닉 가능성

디지털 포렌식 관점에서의 파일 시스템 분석

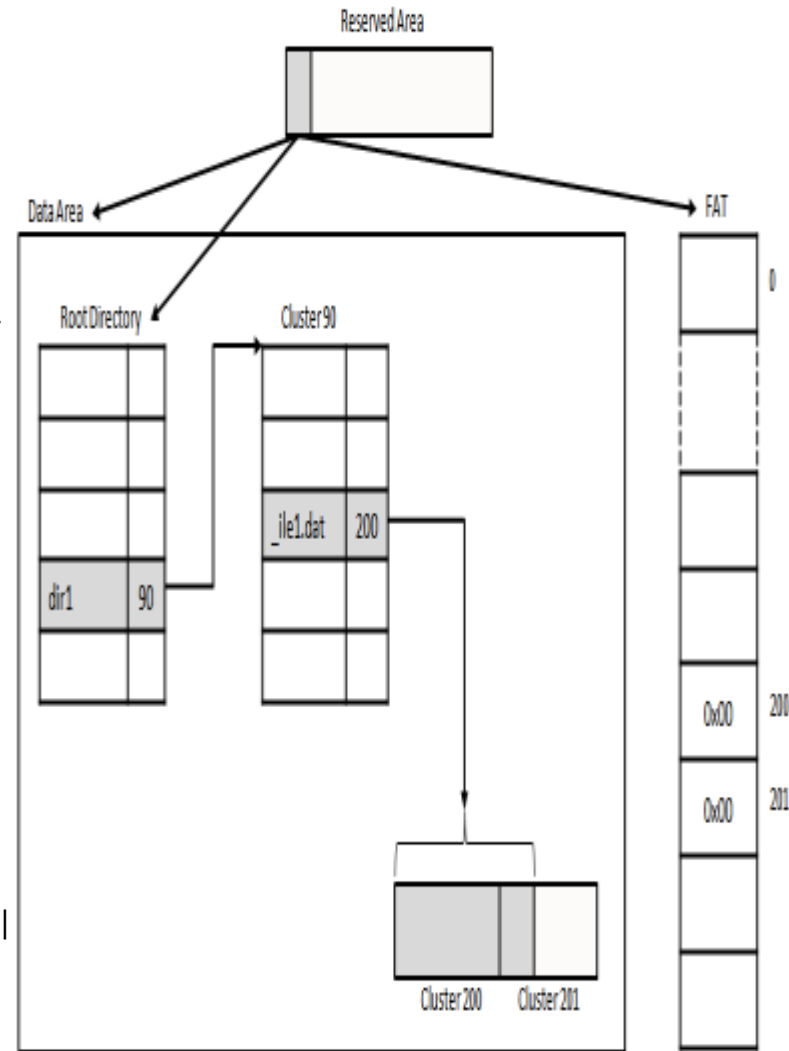
• \$BadClus 파일 분석

- ✓ NTFS의 MFT Entry 8번인 \$BadClus 파일은 배드 섹터가 발생한 클러스터 관리
- ✓ 정상적인 클러스터를 \$BadClus에 등록한 후 해당 영역에 데이터를 은닉 가능성

파일 복구 - 파일 시스템 상의 파일 복구

FAT 파일 시스템은 파일의 메타정보를 유지하기 위해 FAT 영역과 디렉터리 엔트리를 사용한다. 파일이 삭제될 경우 FAT 영역에서는 파일에 할당되었던 클러스터에 대응되는 FAT Entry가 0x00으로 초기화된다. 그리고 해당 파일의 디렉터리 엔트리의 오프셋 0x00의 값이 삭제를 나타내는 0xE5 값으로 변경된다. 이 경우 파일에 할당되었던 클러스터에는 파일의 내용이 그대로 남아 있게 된다. 따라서 해당 클러스터가 새로운 파일에 사용되지 않는다면 삭제 표시된 디렉터리 엔트리에 파일 크기와 시작 클러스터 정보를 통해 비교적 쉽게 파일을 복구할 수 있다. 단, 디렉터리 엔트리에는 시작 클러스터에 대한 정보만 기록되어 있기 때문에 파일이 여러 클러스터에 조각나 기록된 경우에는 FAT 영역에서 클러스터 체인을 확인할 수 없어 완벽하게 복구하기가 어렵다.

1. 부트 섹터의 BPB에서 데이터 영역, 루트 디렉터리, FAT 영역의 위치, 클러스터 크기를 얻어온다.
2. wdir1 디렉터리의 정보를 얻기 위해 루트 디렉터리에서 wdir1에 해당하는 디렉터리 엔트리를 검색한다. 그 결과 wdir1 디렉터리 정보를 가지는 시작 클러스터가 90번임을 확인했다.
3. 클러스터 90번에서 삭제된 file1.dat 파일의 디렉터리 엔트리를 찾는다. 삭제 표시를 위해 파일 이름의 첫 바이트인 0x66('f') 대신 0xE5 값이 기록되어 있다.
4. 삭제된 file1.dat 파일의 디렉터리 엔트리에서 파일의 이름, 확장자, 크기, 시작 클러스터의 위치를 확인한다.
5. 시작 클러스터에서부터 파일 크기만큼 데이터를 획득한 후 저장할 위치에 디렉터리 엔트리에서 확인한 파일 이름과 확장자로 파일을 저장한다.

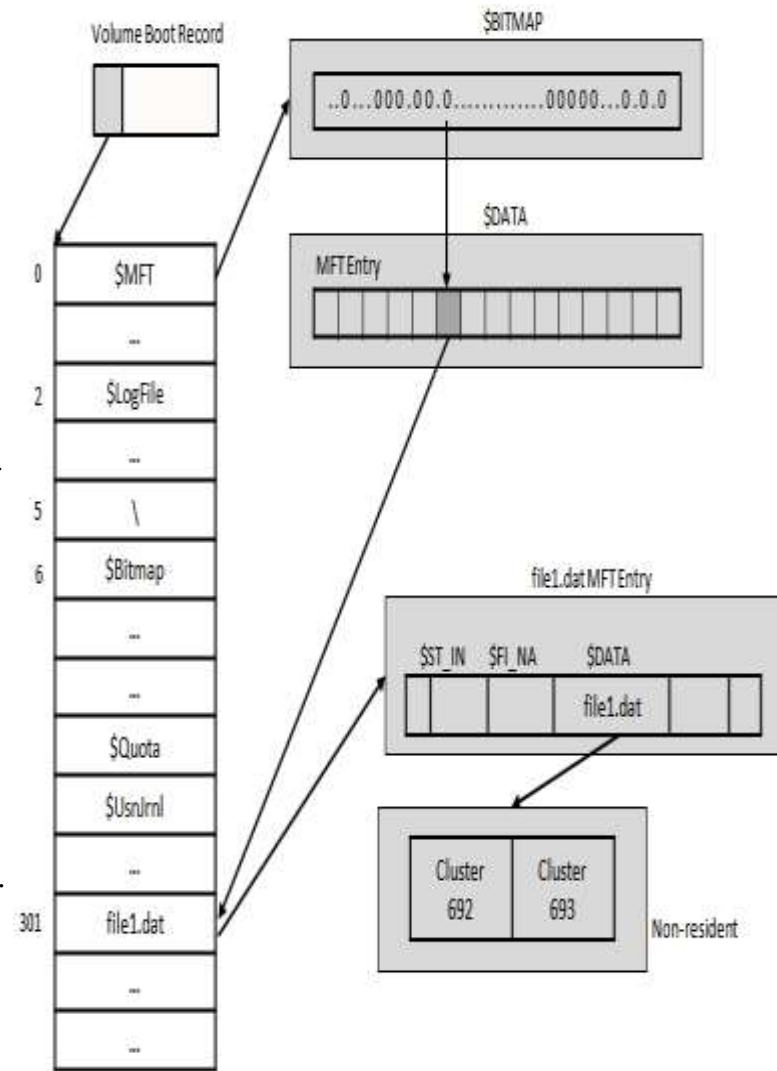


FAT 파일 시스템에서 삭제된 wdir1wfile1.dat 파일 복구

파일 복구 - 파일 시스템 상의 파일 복구

NTFS에서는 파일의 메타정보를 유지하기 위해 MFT Entry, MFT Entry의 할당상태를 표시하기 위한 \$MFT(MFT Entry 0) 파일의 \$BITMAP 속성, 클러스터의 할당 상태를 표시하기 위한 \$Bitmap(MFT Entry 6) 파일의 \$DATA 속성을 사용한다. 파일이 삭제될 경우 \$MFT 파일의 \$BITMAP 속성에서 해당 파일이 사용했던 MFT Entry 비트가 0으로 세트되고 \$Bitmap 파일의 \$DATA 속성에서 해당 파일에 할당되었던 클러스터 비트가 0으로 세트된다. 결국 파일의 MFT Entry와 실제 파일에 할당되었던 클러스터의 내용은 변경되지 않는다. 따라서 해당 파일의 MFT Entry와 할당되었던 클러스터 내용이 새로운 파일의 정보로 덮여 쓰여지지 않았다면 비교적 쉽게 파일을 복구할 수 있다. NTFS에서는 FAT 파일 시스템과 다르게 파일의 내용이 Resident 속성일 경우 MFT Entry 상에 저장되므로 완벽하게 복구할 수 있다. 또한 Non-resident 속성의 경우에도 Cluster Runs 정보를 통해 완벽하게 복구할 수 있다.

1. VBR의 BPB 정보에 MFT 시작위치 정보를 얻어 MFT 시작위치로 이동한다.
2. \$MFT 파일의 \$BITMAP 속성에서 현재 사용 중이지 않은 MFT Entry(0x00 값을 갖는) 정보를 얻어온다.
3. \$MFT 파일의 \$DATA 속성에서 0x00 값을 갖는 MFT Entry를 대상으로 \$FILE_NAME 속성의 파일 이름이 "file1.dat"를 가지는 MFT Entry를 찾는다. 그 결과 MFT Entry 301번이 해당 파일의 MFT Entry 임이 확인되었다.
4. MFT Entry 301번의 \$FILE_NAME 속성에서 파일 크기를 확인한다. 그리고 \$DATA 속성을 확인한 결과 Non-resident 속성임이 확인되었다. 따라서 Cluster Runs 정보를 기반으로 파일 크기만큼 데이터를 획득한다. 획득한 데이터를 저장할 위치에 앞서 확인한 파일 이름으로 파일을 저장한다.

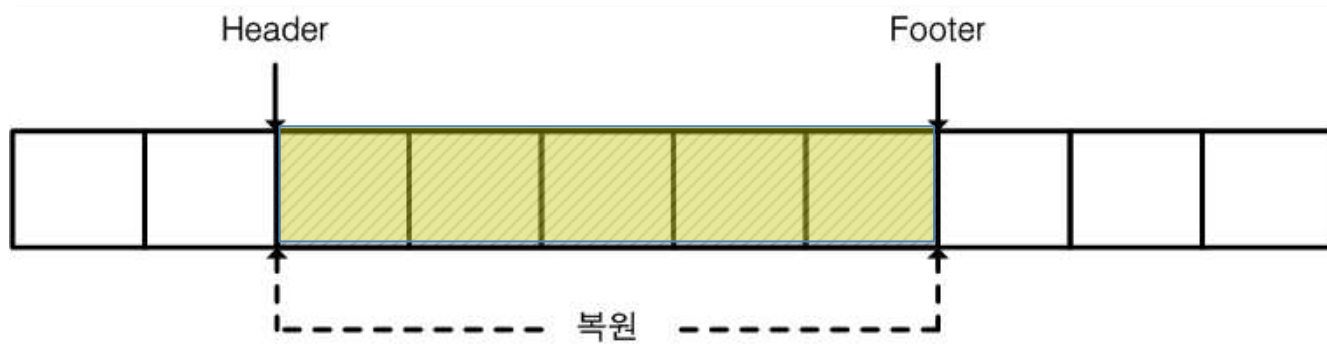


NTFS 파일 시스템에서 삭제된 wdir1wfile1.dat 파일 복구

파일 복구 - 파일 시스템 상의 파일 복구

파일 시스템 정보를 얻을 수 없는 경우의 복구

- 파일 시스템에서 얻을 수 있는 정보 없이 '**파일 자체 정보**' 기반 복구
 - 즉, 파일의 고유한 특성이 있는 파일만 복구 가능
- 연속적으로 존재하는 파일에 대한 복구는 대부분 가능, 조각난 경우는 어려움
- 추출된 파일이 올바른 파일이라는 보장이 없음
- 많은 시간이 소요됨



파일 복구 - 파일 시스템 상의 파일 복구

파일이 연속적이지 않고 조각난 경우

- 조각난 파일이 생성되는 이유
 - 파일을 저장할 충분한 연속 공간이 없을 경우
 - 기존 파일에 데이터가 추가될 때, 파일의 후반부 영역에 할당되지 않은 영역의 크기가 충분하지 않은 경우
- 파일 포맷의 특성을 이용한 여러 복구 방안이 연구 중
- Pattern Recognition, 통계 분석 등

파일 복구 - 파일 카빙

- ✓ 파일 카빙 기법 : 저장 매체의 비할당 영역으로부터 파일을 복구하는 기법
 - 연속적인 카빙 기법
 - 파일 내용이 저장 매체의 연속된 공간에 저장된 경우 수행
 - 비연속적인 카빙 기법
 - 파일의 내용이 저장 매체의 여러 부분에 조각나 저장된 경우 수행

파일 복구 - 파일 카빙

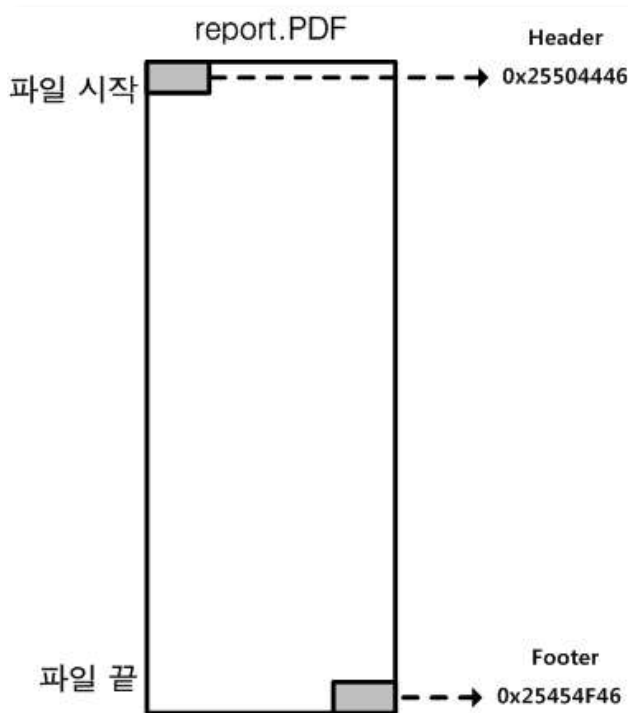
• 시그니처 기반 카빙

- ✓ 파일 포맷별로 존재하는 고유한 시그니처를 이용하는 방법
- ✓ 헤더와 푸터 시그니처가 모두 존재하는 파일의 경우 두 시그니처 사이의 데이터가 파일의 내용

파일 복구 - 파일 카빙

파일의 Header와 Footer 정보

- 일부 파일은 파일의 시작과 끝을 알 수 있는 고유한 Header와 Footer를 가짐
- PDF, GIF, PNG, JPG, ALZ, ZIP, RAR, MPG ...



```
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
00000000 45 50 44 46 20 31 2E 34 0A 25 C3 A4 C3 BC C3 B6 25 1.4.%
00000010 C3 9F 0A 32 20 30 30 6F 62 6A 0A 3C 3C 2F 4C 65
00000020 6E 67 74 68 20 33 20 30 20 52 2F 46 69 6C 74 65
00000030 73 2F 46 6C 61 74 65 44 65 63 6F 64 65 3E 3C 0A
00000040 73 74 72 65 61 60 0A 79 3C 95 56 4B 68 00 30 10
00000050 6E EF AF 00 89 60 AE 66 F4 E2 41 18 76 90 00 43
00000060 6F 81 85 1E 4A 6F 6D 0A 43 5A 6B 2E FD FB 9D 87
00000070 64 CB 0E 37 A4 04 84 BA 34 CF 6F BE 18 D9 76 80
00000080 FE 1E FE 18 6B 6E 6E 73 2E 0D AE EB 40 1B 02 ED
00000090 5F BE 9B CF 1F CC EF 83 ED 42 DF A3 33 66 83 43
000000A0 6F F9 17 30 FB C1 BC FC 30 40 AF 6A 6F 0E 45 FF
000000B0 99 0C 61 36 FD BA 86 6C CA 55 95 7E 36 3F 0F 4F
000000C0 1F C4 29 FF 91 90 F3 ED ED 1D 45 E2 87 2E 9A 08
000000D0 37 F3 F1 0A 06 00 0C 9E BE 64 06 23 66 88 63 91 7
```

```
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
00000020 20 6E 20 0A 30 30 30 31 39 37 34 35 39 20 30 n_0000197453 0
00000030 30 30 30 20 6E 20 0A 74 72 61 69 6C 65 72 0A 0000 n_trailer_
00000040 30 3C 2F 53 89 7A 65 20 31 30 34 2F 52 6F BF 74 <</Size 104/Root
00000050 20 31 30 32 20 30 20 52 0A 2F 49 44 20 5B 20 3C 30 39
00000060 30 33 20 30 20 52 0A 2F 49 44 20 5B 20 3C 30 39
00000070 43 34 42 43 43 36 43 42 33 39 32 39 30 37 33 34
00000080 34 36 39 43 34 42 36 38 37 42 38 43 32 31 3C 0A
00000090 3C 30 39 43 34 42 43 43 36 43 42 33 39 32 38 30
000000A0 37 33 34 34 36 39 43 34 42 36 38 37 42 38 43 32
000000B0 31 3E 20 6D 0A 2F 44 6F 63 43 68 65 63 6B 73 75
000000C0 60 20 2F 38 43 38 36 37 44 38 43 45 36 43 30 31
000000D0 32 43 42 35 37 30 36 38 41 45 30 36 30 34 41 34
000000E0 38 46 38 0A 3E 0A 73 74 61 72 74 38 72 65 68
000000F0 0A 31 38 37 36 34 35 0A 39 25 48 4F 48 0A
,197646,83207
```

파일 복구 - 파일 카빙

◉ 램 슬랙 카빙

- ✓ 푸터 시그니처 이후에 램 슬랙이 존재
- ✓ 시그니처 기반 카빙 기법에서 푸터 시그니처와 함께 확인하여 많은 오탐 줄일 수 있음

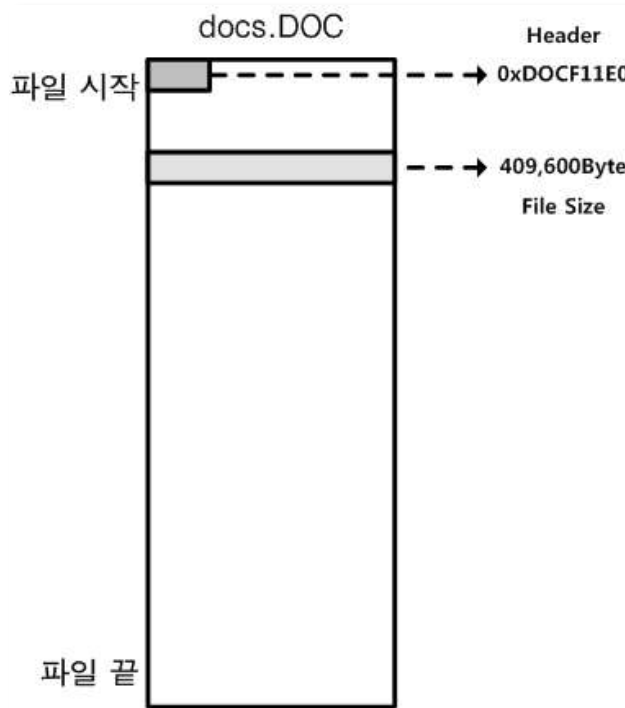
○ 파일 구조체 카빙

- ✓ 푸터 시그니처가 존재하지 않거나 파일 포맷 내부에 여러 개의 시그니처가 존재하는 경우
효과적인 방법으로 파일의 구조를 분석하여 카빙 하는 기법
 - 파일 크기 획득 방법
 - 파일 구조 검증 방법

파일 복구 - 파일 카빙(파일 크기 획득 방법)

파일의 Header와 File size 정보

- 일부 파일은 파일의 시작과 크기를 알 수 있는 정보를 포함하고 있음
- DOC, ODT, ODS, BMP, AVI, ASF, WAV ...



```
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
00000000 00 CF 11 E0 A1 B1 1A E1 00 00 00 00 00 00 00 00
00000010 00 00 00 00 00 00 00 3E 00 08 00 FE FF 08 00
00000020 06 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00000030 4C 14 00 00 00 00 00 00 10 00 00 4F 14 00 00
00000040 01 00 00 00 FE FF FF FF 00 00 00 54 14 00 00
00000050 55 14 00 00 56 14 00 00 57 14 00 00 58 14 00 00
00000060 59 14 00 00 5A 14 00 00 5B 14 00 00 5C 14 00 00
00000070 5D 14 00 00 5E 14 00 00 5F 14 00 00 60 14 00 00
00000080 61 14 00 00 62 14 00 00 63 14 00 00 64 14 00 00
00000090 65 14 00 00 66 14 00 00 67 14 00 00 68 14 00 00
000000A0 69 14 00 00 6A 14 00 00 6B 14 00 00 6C 14 00 00
000000B0 6D 14 00 00 6E 14 00 00 6F 14 00 00 70 14 00 00
000000C0 71 14 00 00 72 14 00 00 73 14 00 00 74 14 00 00
000000D0 75 14 00 00 76 14 00 00 77 14 00 00 78 14 00 00
```

```
0 1 2 3 4 5 6 7 8 9 A B C D E F 0123456789ABCDEF
00000040 4C 14 00 00 00 00 00 00 10 00 00 4F 14 00 00
00000050 01 00 00 00 FE FF FF FF 00 00 00 54 14 00 00
00000060 55 14 00 00 56 14 00 00 57 14 00 00 58 14 00 00
00000070 59 14 00 00 5A 14 00 00 5B 14 00 00 5C 14 00 00
00000080 5D 14 00 00 5E 14 00 00 5F 14 00 00 60 14 00 00
00000090 61 14 00 00 62 14 00 00 63 14 00 00 64 14 00 00
000000A0 65 14 00 00 66 14 00 00 67 14 00 00 68 14 00 00
000000B0 69 14 00 00 6A 14 00 00 6B 14 00 00 6C 14 00 00
000000C0 6D 14 00 00 6E 14 00 00 6F 14 00 00 70 14 00 00
000000D0 71 14 00 00 72 14 00 00 73 14 00 00 74 14 00 00
000000E0 75 14 00 00 76 14 00 00 77 14 00 00 78 14 00 00
000000F0 79 14 00 00 7A 14 00 00 7B 14 00 00 7C 14 00 00
00000100 FF FF FF FF FF FF FF FF FF FF FF FF FF FF FF
```


파일 복구 - 파일 카빙(파일 구조 검증 방법)

- ✓ 데이터 표현 위한 고유한 계층 구조를 검증하여 카빙하는 방법
- ✓ 문서 파일과 같이 빈번한 수정이 이루어지는 경우 사용

파일 구조 검증 방법을 적용할 수 있는 파일 포맷

파일 포맷	시그니처	
	헤더 (Hex)	푸터 (Hex)
ZIP	50 4B 03 04	50 4B 05 06
ALZ	41 4C 5A 01	43 4C 5A 02
RAR	52 61 72 21 1A 07	3D 7B 00 40 07 00
Compound	D0 CF 11 E0 A1 B1 1A E1	-

Q & A