8.파일시스템과 파일 복구
(파일시스템 개론)
-요약 -

박 종 혁 교수
UCS Lab
Tel: 970-6702
Email: jhpark1@snut.ac.kr

개요

- 학습 목표
  - 디지털 포렌식에서 기본 단위는 파일의 추출과 분석을 통해 이루어지므로, 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

파일 시스템의 이해

- 파일시스템 이란?
  - 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

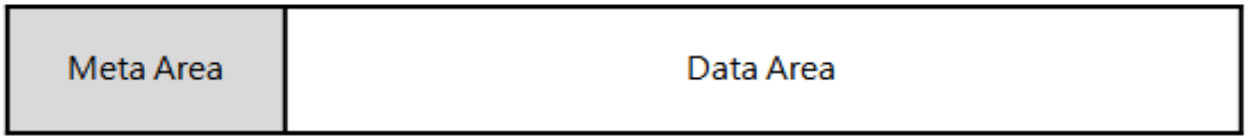
파일 시스템의 이해 – 파일 시스템의 종류

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 – 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 방식이 새롭게 대두

파일 시스템의 이해 – 주소 지정 방식

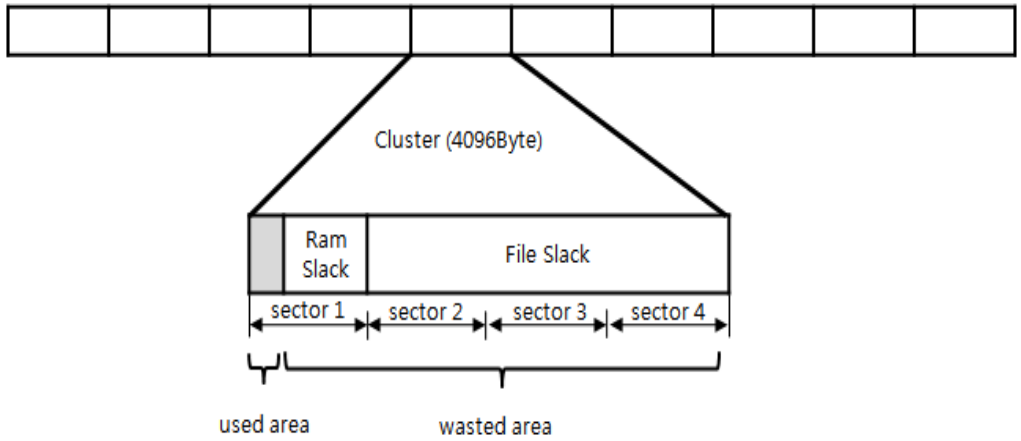
- Addressing (주소 지정 방식)

  ✓ 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

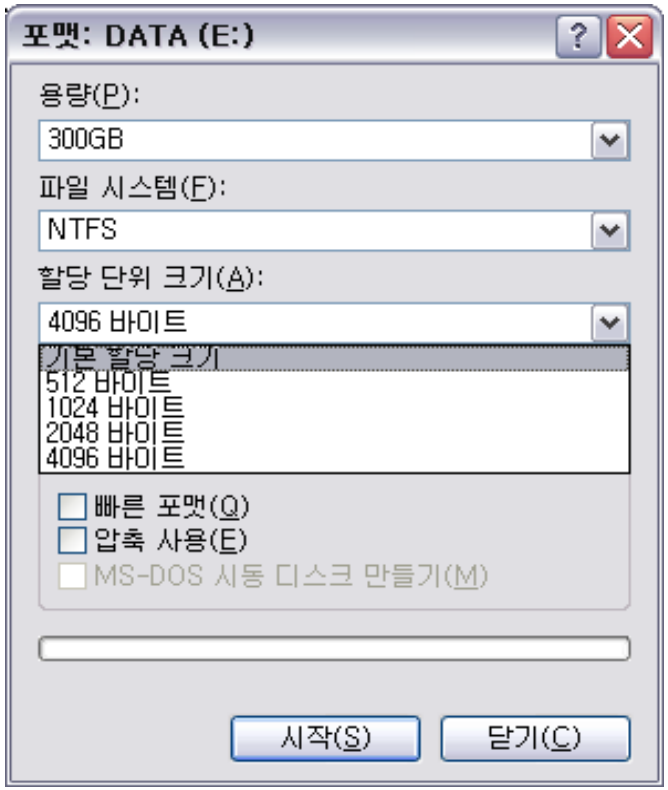
- Cluster (클러스터)
  - ✓ 클러스터 = 여러 개의 섹터(하드디스크의 물리적 최소 단위)를 묶은 단위
  - ✓ 섹터단위로 입출력 처리하면 시간이 오래 걸리므로 여러 개의 섹터를 묶어 한번에 처리



  - ✓ 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

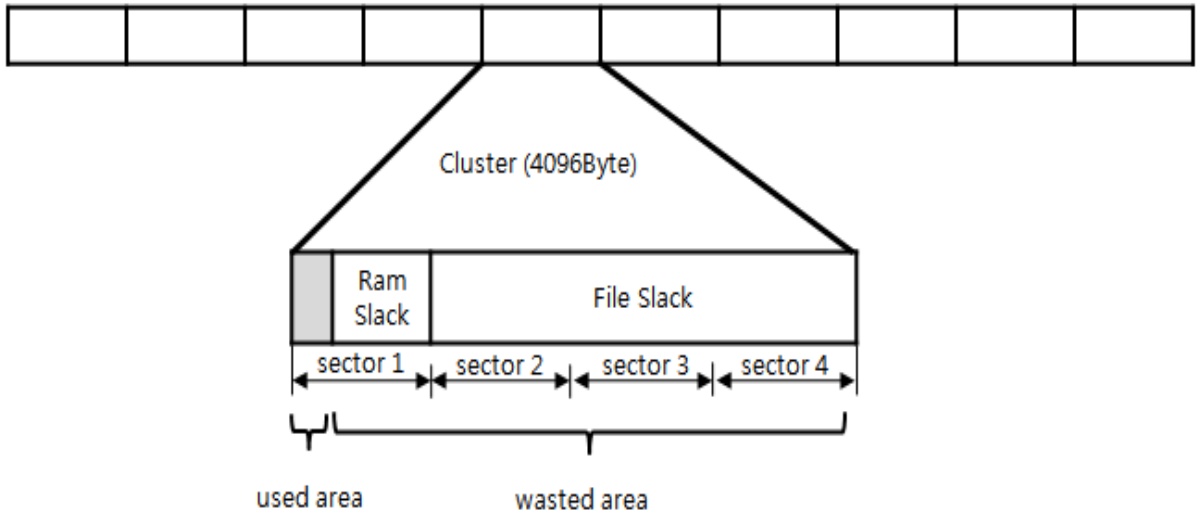
파일 시스템의 이해 – 슬랙 공간

- 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

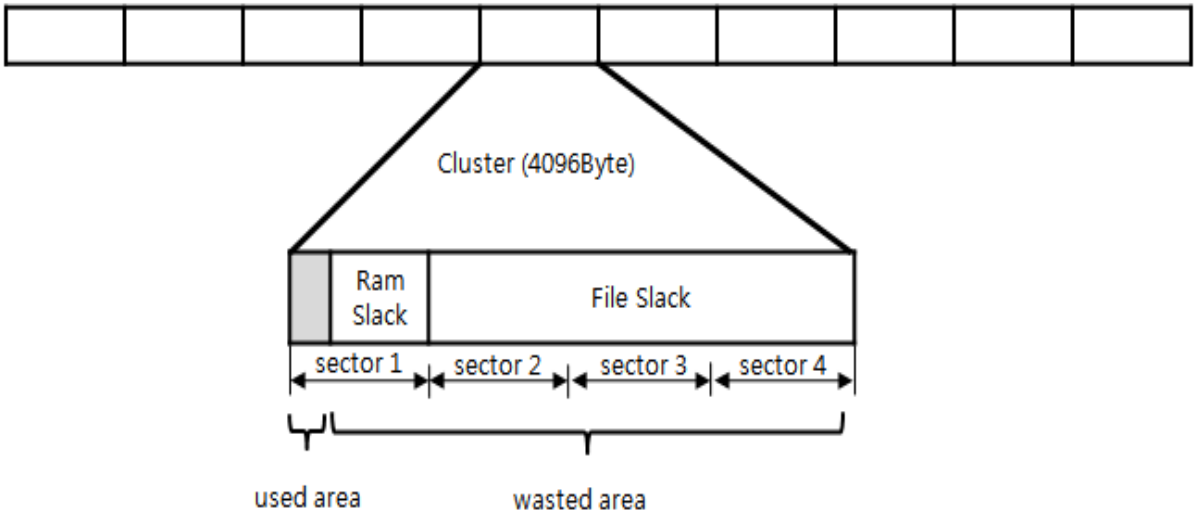
파일 시스템의 이해 – 슬랙 공간

- 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)

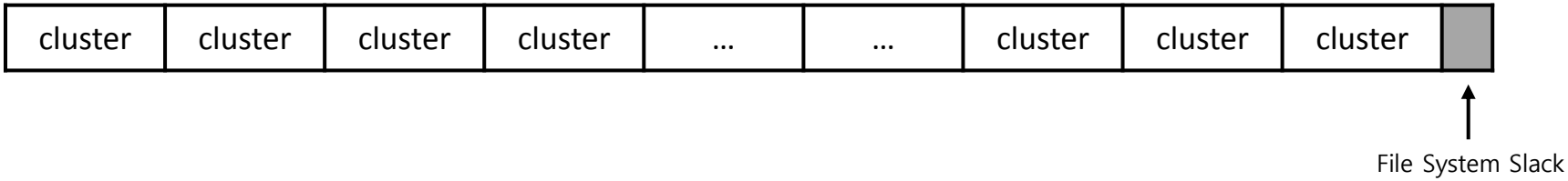


파일 시스템의 이해 – 슬랙 공간

- Slack Space (File System Slack & Volume Slack)

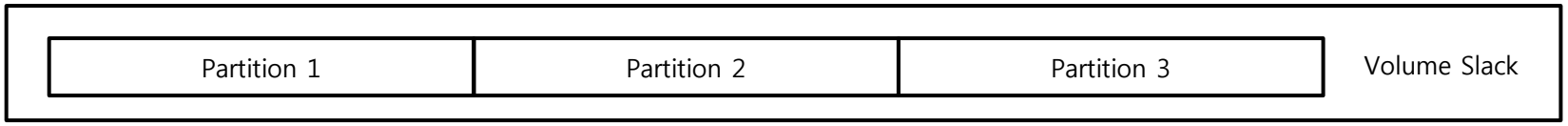
✓ File System Slack

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



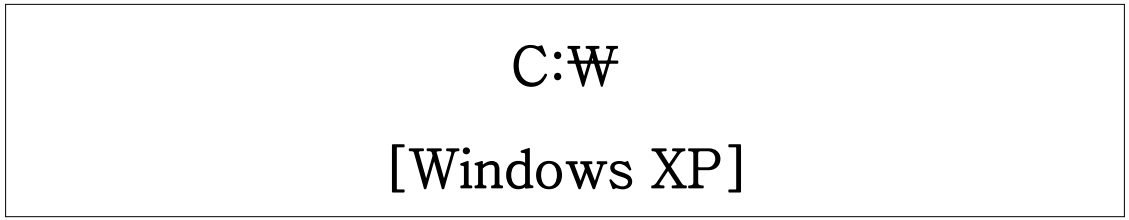
✓ Volume Slack

- 전체 볼륨 크기와 할당된 파티션 크기의 차이로 인해 발생되는 공간

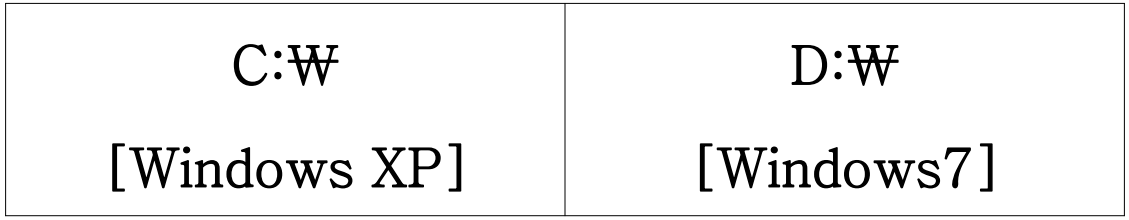


파티션과 MBR – 파티션

- **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:₩ [Windows XP]

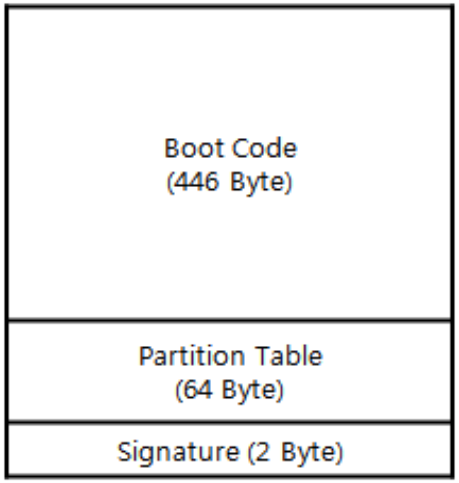
단일 파티션에서의 Boot Record

MBR	BR	C:₩ [Windows XP]	BR	D:₩ [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)

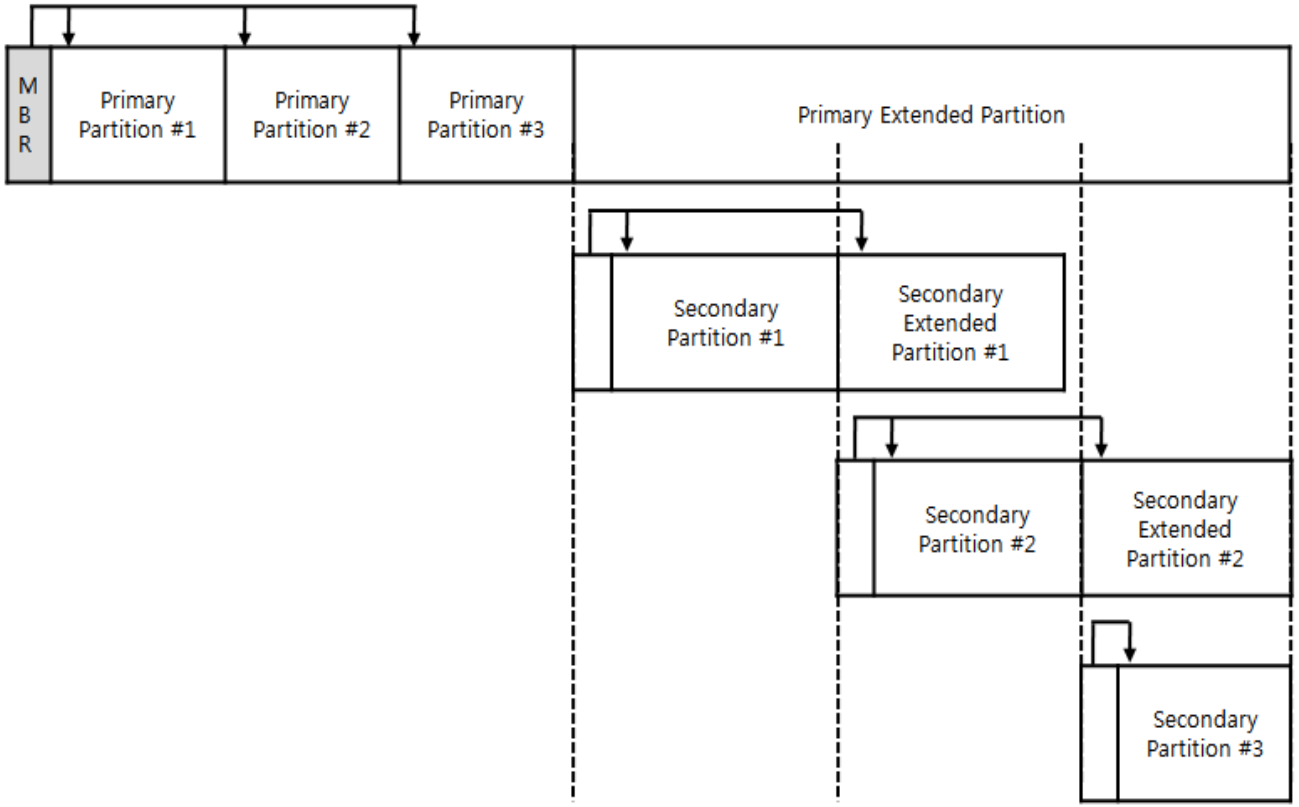
- 파티션 테이블 영역
  - 16 바이트씩 총 4개의 파티션 정보가 저장
  - 첫 번째 값인 부트 플래그(Bootable Flag)는 해당 파티션이 부팅 가능한 파티션인지를 나타내며, 부팅 가능한 파티션일 경우 해당 부트 플래그의 값이 0x80
  - MBR의 부트 코드는 파티션 테이블을 검색하여 부트 플래그 값이 0x80 값을 갖는 파티션의 부트 섹터 위치로 점프하는 역할을 수행
  - 포렌식 조사 시, 보이는 파티션 영역 및 디스크 전 영역을 조사, 필요에 따라 MBR영역 직접 해석
    - MBR영역과 실제 파일 시스템이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

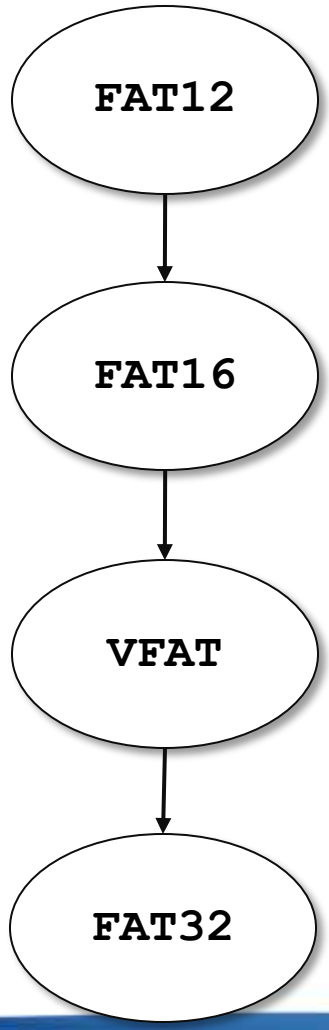
- FAT (File Allocation Table) History

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} = 4,096개의 클러스터를 표현, 하지만 미리 예약된 12개의 클러스터가 존재하기 때문에 최대 4,084개의 클러스터를 표현



1980년대 초 (QDOS)
플로피디스크용으로 처음 개발

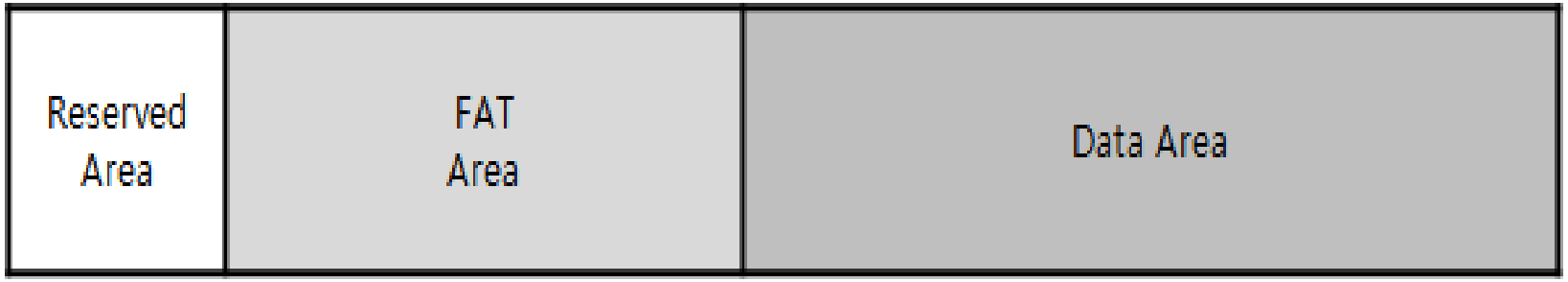
1980년대 말
하드디스크를 지원하기 위해 개발

1995년
FAT의 성능 향상, 긴 파일 이름이 가능해짐

1996 년
VFAT을 확장, 고용량 하드디스크 지원

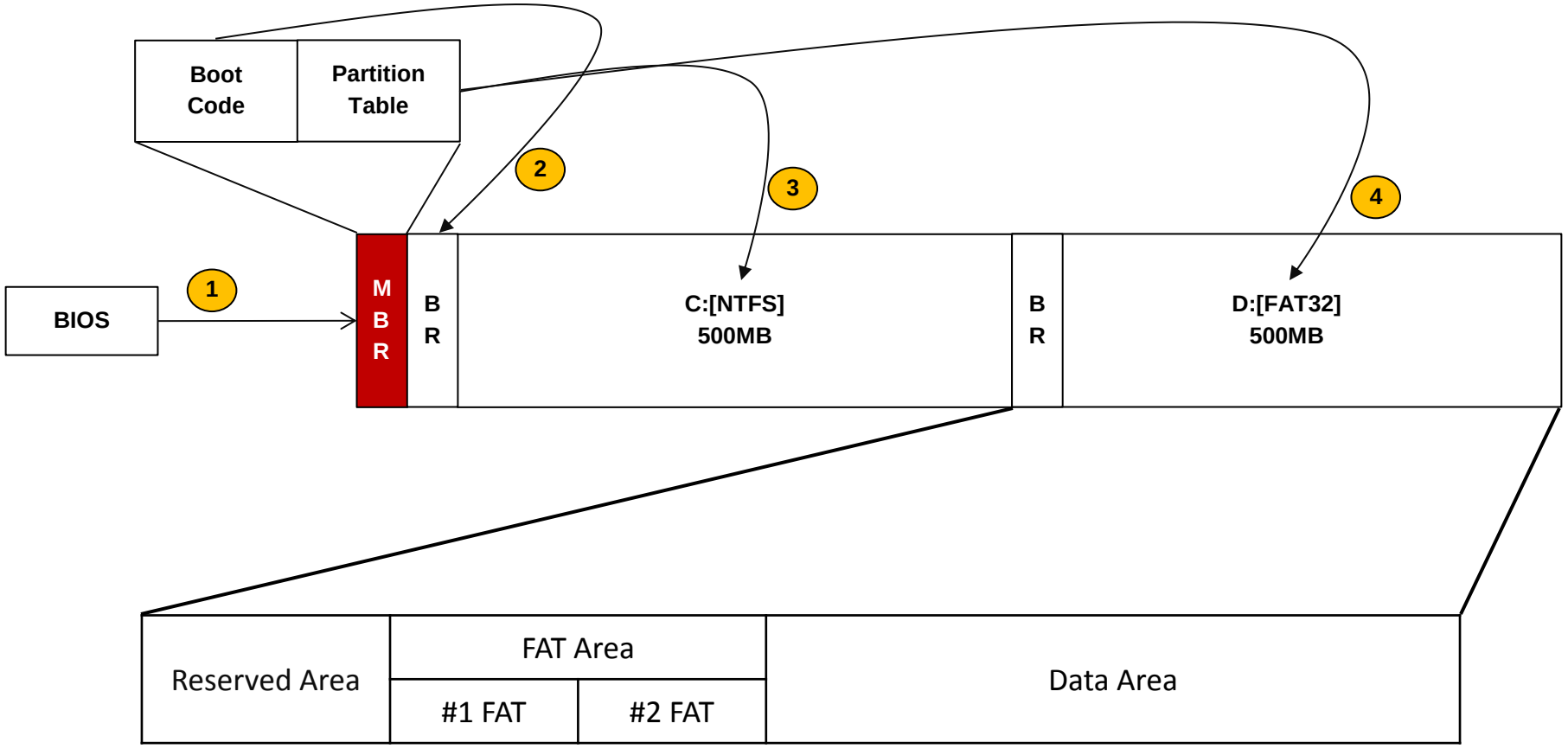
FAT 파일 시스템 – FAT 파일 시스템 소개

- FAT 파일 시스템의 구조
  - ✓ 예약 영역 / FAT 영역 / 데이터 영역



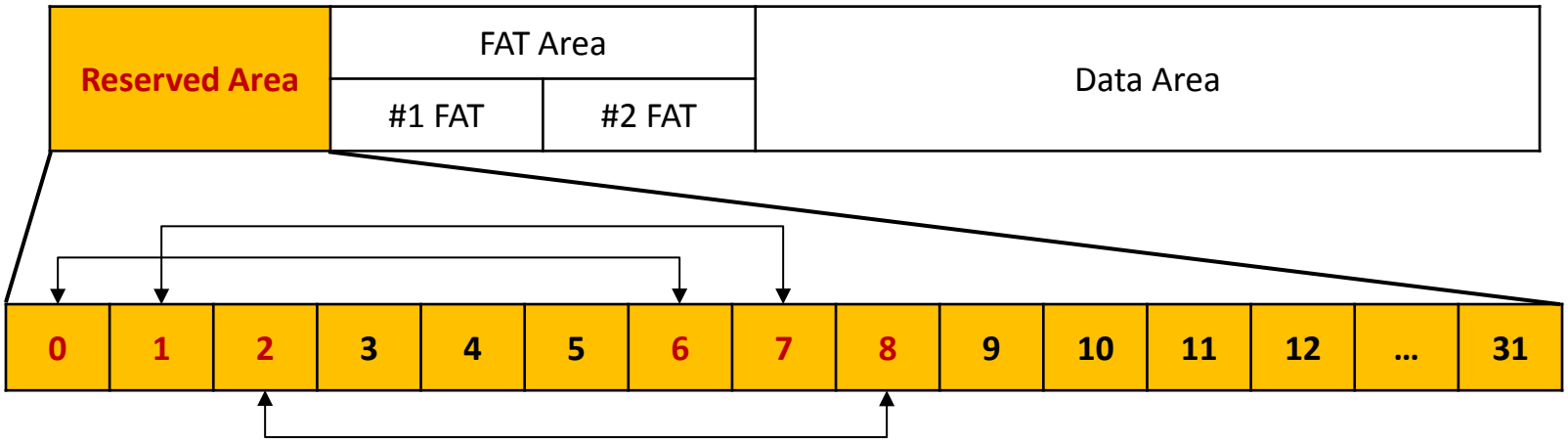
FAT 파일 시스템 – FAT 파일 시스템 소개

- FAT (File Allocation Table) Layout



FAT 파일 시스템 – 예약 영역

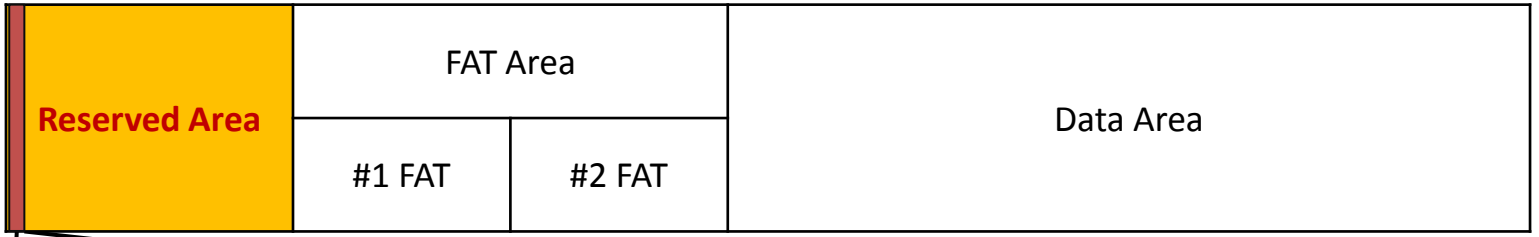
- Structure – Reserved Area

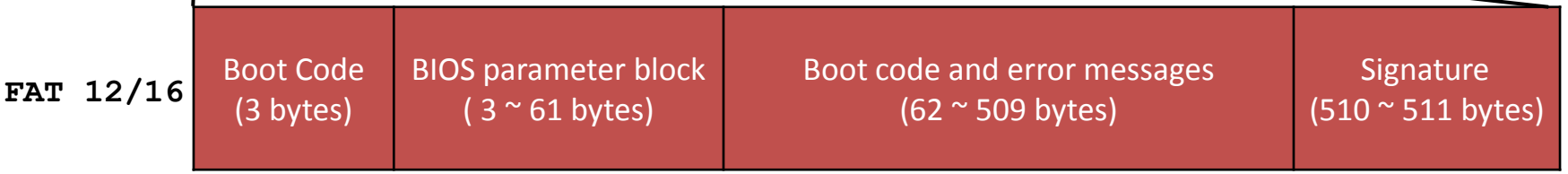


- ✓ FAT32 – 예약영역 중 6개의 섹터만 사용, 나머지는 만약을 대비해 예약해 둔 것
- ✓ 예약 영역의 크기는 가변적 이지만 0, 1, 2, 6, 7, 8번 섹터는 미리 정해져 있음
  - 0, 6 : Volume Boot Sector (0번은 부트섹터로 사용, 만약을 대비해 6번에 백업)
  - 1, 7 : File System Information(FSINFO) Structure
    (7번에 백업, FSINFO구조체는 운영체제에게 비 할당 클러스터의 첫 위치와 전체 비 할당 클러스터의 수를 알려줌으로써 저장할 데이터를 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)

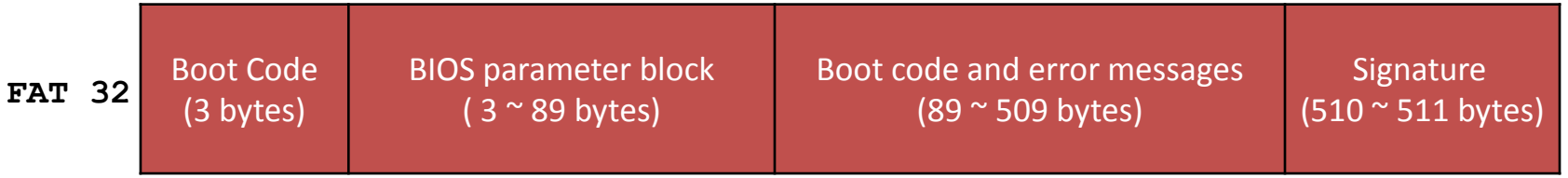
FAT 파일 시스템 – 예약 영역

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





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



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			

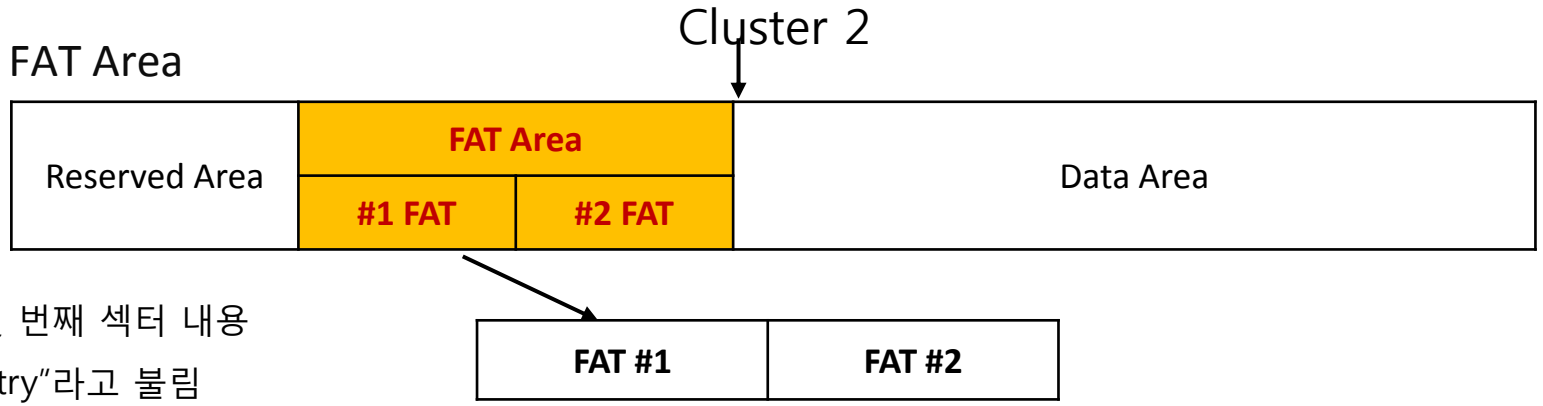
FAT 파일 시스템 – FAT 영역

- 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



- 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 2번이므로 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
	... ...			

FAT32에서 FAT영역의 구조

FAT 파일 시스템 – FAT 영역

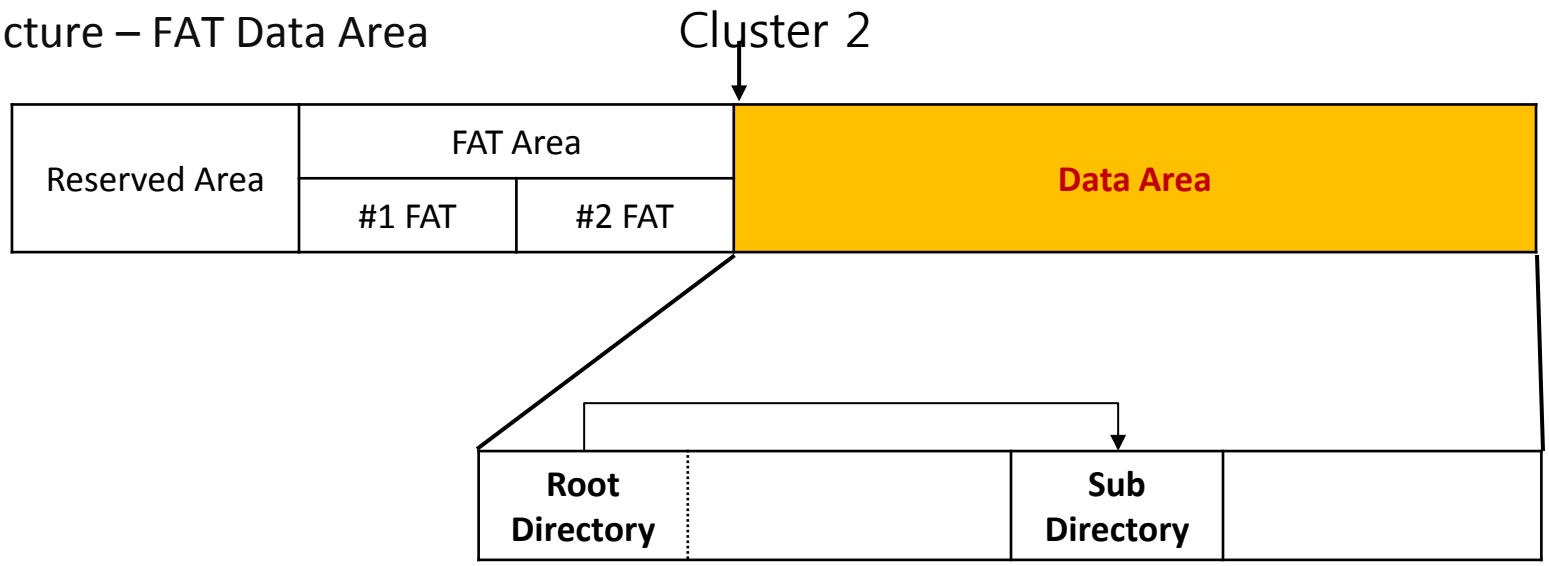
- Structure – FAT Area (Entry Type)
  - 비할당 상태일 경우, 0x00의 값을 가진다. 따라서 운영체제는 새로운 파일 및 디렉터리를 저장하고자 할 경우 FAT 영역에서 FAT Entry 값이 0x00인 클러스터를 찾아 할당한다.

  - 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 – 데이터 영역

- Structure – FAT Data Area



✓ 최상위 루트 디렉터리가 가장 중요

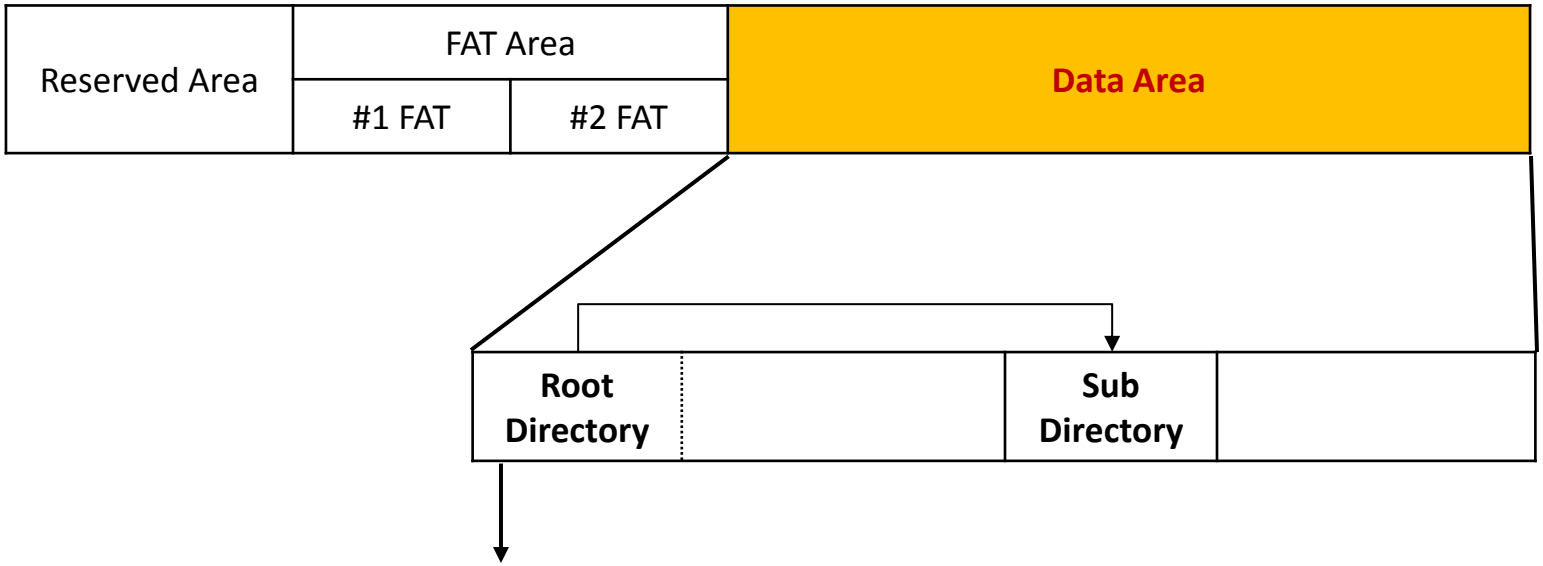
- FAT12/16 : FAT Entry 2번, FAT32 : 어느 곳에나 (기본적으로는 FAT Entry 2번)

✓ 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

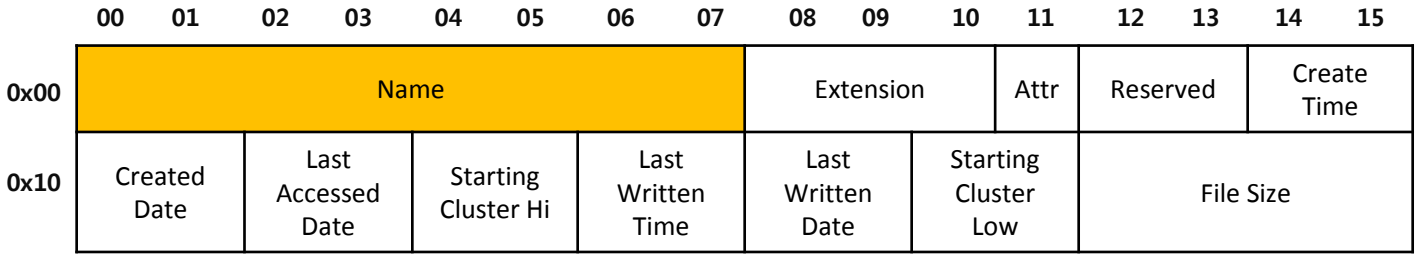


	00	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15
0x00	Name								Extension			Attr	Reserved		Create Time	
0x10	Created Date			Last Accessed Date		Starting Cluster Hi		Last Written Time	Last Written Date		Starting Cluster Low		File Size			

Directory Entry구조

FAT 파일 시스템 – 데이터 영역

- Structure – FAT Data Area (Directory Entry – Name)

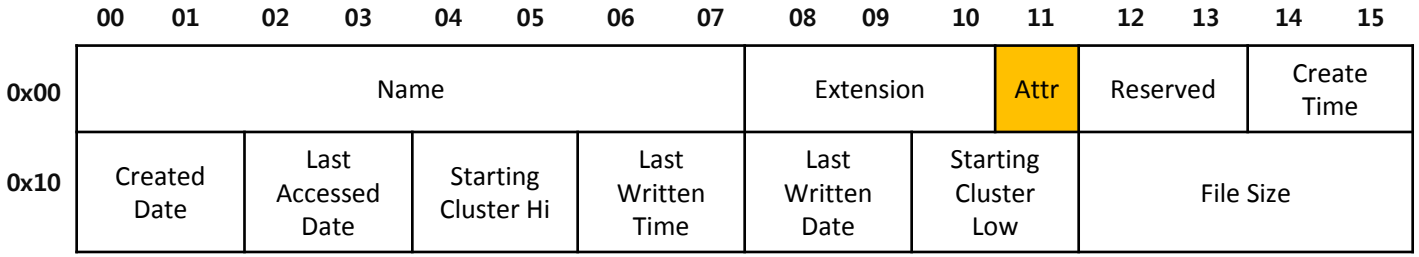


- ✓ 첫 번째 바이트가 0xE5 값을 가진다면 해당 엔트리는 삭제되었음을 의미
  - ➢ 첫 번째 바이트 값만 변경되고 나머지 값은 초기화하지 않기 때문에 파일의 정보 및 내용을 복구할 수 있음
- ✓ 첫 번째 바이트가 0x00의 값을 가진다면 해당 엔트리는 사용되지 않는 엔트리
  - ➢ 이후에 엔트리가 존재하지 않는다는 것을 의미하므로 더 이상 검색할 필요가 없음

- ✓ 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 – Attribute)



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

- **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**

특 징	설 명
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 파일이름, 소유자, 시간 정보 등을 스트림 통해 표현, 데이터도 하나의 데이터 스트림으로 표현 추가된ADS는 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 – NTFS 구조

- **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 동적으롤 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 포멧된 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 – MFT(Master File Table)

- **MFT Entry**

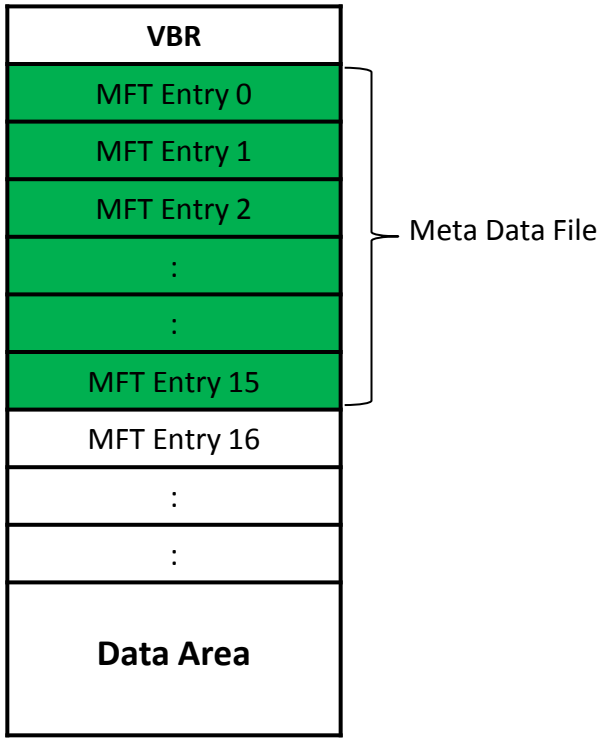
NTFS 구조

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
- ✓ 사용자에 의해 파일이 생성될 때마다 새로운 MFT Entry가 할당되어 해당 파일의 정보를 유지 및 관리

NTFS – MFT(Master File Table)

- **Master File Table**



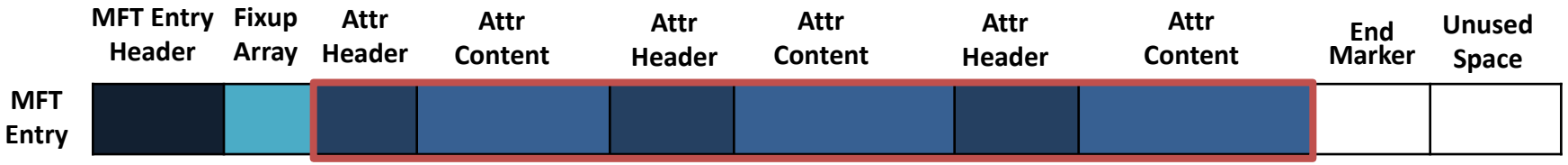
- ✓ MFT는 파일 시스템의 여러 영역에 조각나서 존재 가능하므로, $MFT파일이 MFT 전체 정보를 유지 관리 하는 역할 수행
- ✓ 각 파일들 분석 위해 전체 MFT 정보를 획득이 선행되어야 함

예약된 MFT Entry

Entry 번호	Entry 이름	설 명
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 -)

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

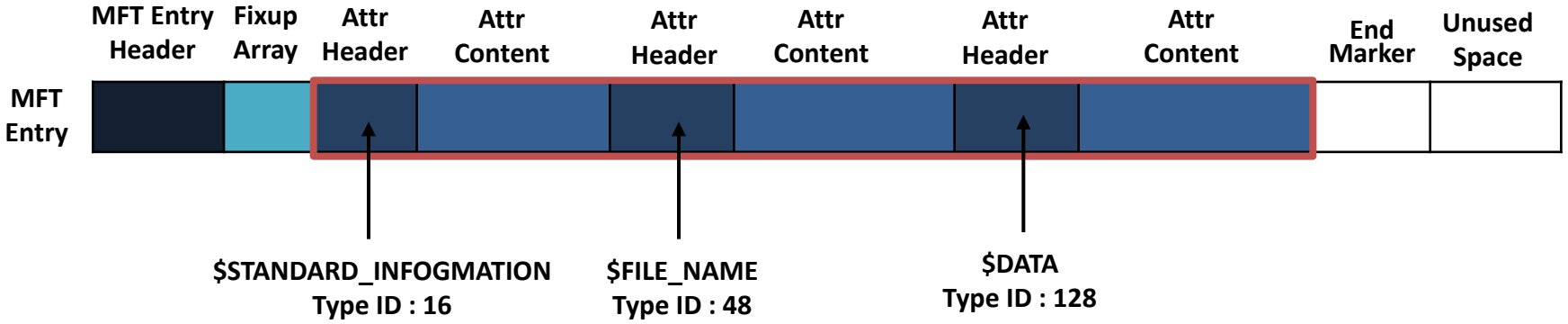
NTFS – MFT(Master File Table)

- **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_ONFORMATION 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

*flags : 0x0002 → 은닉
0x4000 → 암호화

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

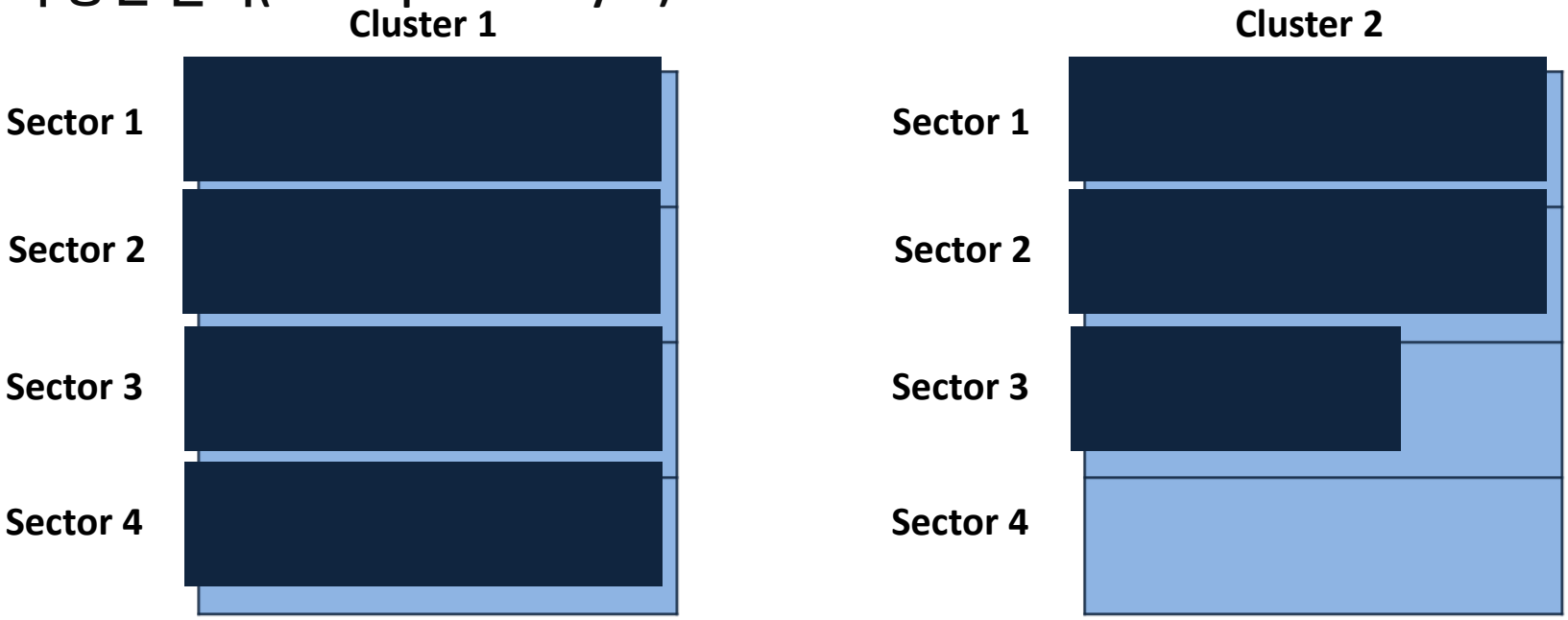
- FAT : Root Directory부터 삭제된 Directory Entry 검색(value : 0xE5, offset : 0x00)
- NTFS : $MFT 내의 MFT Entry bitmap에서 0x00 값을 가지는 MFT Entry 조사

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 :** 마찬가지
- ✓ 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)

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

  - ✓ NTFS : 해당 파일의 속성 ($STANDARD_INFORMATION, $FILE_NAME)

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 $DATA속성에 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

Q & A