

Ext3

Filesystem Seminar

2009/05/27

이경식

목차

- ExT₃
 - ExT₃
- FileSystem 분석
 - Lay-out
 - Super Block
 - Group Descriptor Table
 - Inode
 - Timestamp
 - Root Directory 찾기
 - /bin/l_s 찾기
- 추가사항
 - Symbolic Link
 - Mount
 - File 삭제
 - Journaling

Ext3

Ext3

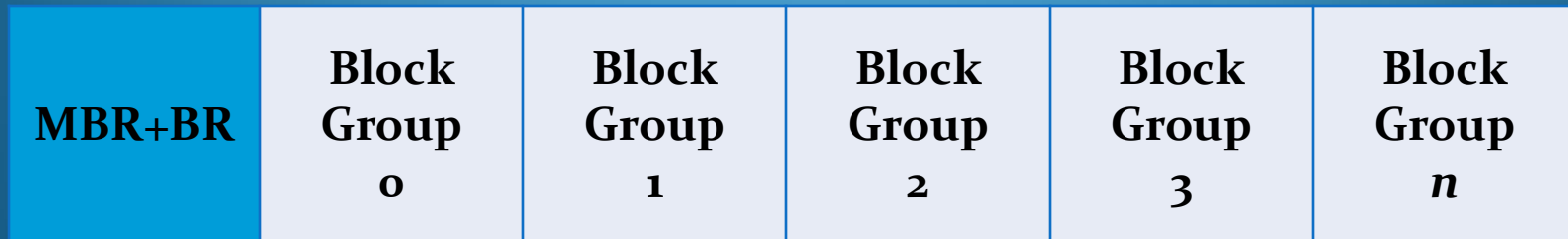
- Extended FileSystem 3
 - Linux Kernel 2.4.15판에 처음으로 등장
- Advantage
 - Ext2에서 자료 손실없이 Ext3로 이동가능
 - Journaling
 - Htree(Btree의 고급판)
- Disadvantage
 - No defragmentation
 - No checksumming in journal

File System 분석

- Lay-Out
 - Block Group
 - Backup Super Block
- Super Block
- Group Descriptor Table
- Inode
- Timestamp
- Root directory 찾기
- /bin/ls 찾기

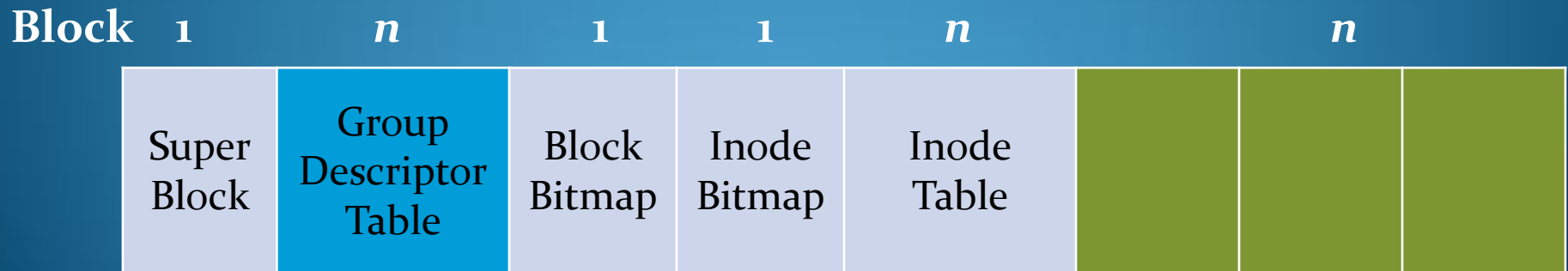
Lay-Out

- Boot Sector + Block Group
 - Block - 데이터 저장 단위(FAT의 Cluster)
 - 1,2,4KB 지정가능
- 중요 데이터는 여러 군데에 저장을 하는 구조를 가짐
- 블록 그룹들은 모두 같은 개수의 블록 수를 가짐
 - 그룹 내부에는 Meta-Data, File Data 등이 기록



Block Group

- Block Group은 Block들의 모임
 - 각 블록 내부에는 파일 시스템 정보 및 데이터를 저장
 - OS는 같은 파일의 데이터 블록은 같은 블록 그룹에 저장하려는 성질을 가짐
 - 파일 단편화 발생의 감소



Backup SuperBlock

- 블록사이즈에 따라 Backup SuperBlock의 위치 변경

Block size (byte)	Offset (Block)
1024	8193
2048	16384
4096	32768

- Backup Superblock은 하나가 아니며, 다수의 백업블록을 갖는다.
 - mke2fs로 파일 시스템 생성 시 생성
 - Offset :
32768 → 98304 → 163840 → 229376 → 294912 → 819200 ...

Super Block

- 주요 설정 정보들을 기록
- 1024byte의 size를 가짐.
- 블록의 크기는 Super Block에 정의
 - 슈퍼 블록은 블록 사이즈를 모르는 상태에서도 접근 가능해야 함
 - 블록 그룹의 첫 번째 블록에 위치
- Super Block앞에는 2 Sector의 reserved 영역이 존재
 - Boot Code가 위치

Super Block

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00000400	A0	B1	05	00	3D	C4	16	00	69	23	01	00	6B	56	0D	00	± =Ä i# kV
00000410	B5	FB	03	00	00	00	00	00	02	00	00	00	02	00	00	00	μû
00000420	00	80	00	00	00	80	00	00	B0	1F	00	00	2D	6D	12	4A	ı ĩ ° -m J
00000430	2D	6D	12	4A	05	00	15	00	53	EF	01	00	01	00	00	00	-m J Si
00000440	49	10	0D	4A	00	4E	ED	00	00	00	00	00	01	00	00	00	I J Ní
00000450	00	00	00	00	0B	00	00	00	00	01	00	00	3C	00	00	00	<
00000460	06	00	00	00	03	00	00	00	8E	99	9C	99	D1	7B	4F	47	ııııÑ{OG
00000470	8E	C9	1B	7D	DE	8F	F4	E7	00	00	00	00	00	00	00	00	ıÉ }Pıôç
00000480	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
00000490	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000004A0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000004B0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000004C0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	6C	01	1
000004D0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
000004E0	08	00	00	00	00	00	00	00	00	00	00	00	BF	BC	61	D9	ıkaÜ
000004F0	8B	85	4C	FF	B9	B6	BD	E0	08	2A	4E	28	01	01	00	00	ıııLy¹kà *N(
00000500	00	00	00	00	00	00	00	00	49	10	0D	4A	FD	81	0A	00	I Jýı

Super Block

- Inode 정보
 - Inode count
 - Free inode count
 - Inodes per Group
 - First non-reserved Inode
 - Inode size
 - Structure size

Ext2/Ext3/Ext4 Superblock, Base Offset: 400

Offset	Title	Value
400	Inode count	373152
404	Block count	1492029
408	Reserved blocks	74601
40C	Free blocks	874091
410	Free Inode	261045
414	First data block	0
418	Block size (bytes)	2
41C	Fragment size (bytes)	2
420	Blocks per group	32768
424	Fragments per block	32768
428	Inodes per group	8112
42C	Last mount time	2009-05-19 08:26:21
430	Last write time	2009-05-19 08:26:21
434	Mount count	5
436	Maximal mount count	21
438	Magic signature	53 EF
43A	File system revision	1
43C	Behavior with reserved blocks	1

Ext2/Ext3/Ext4 Superblock, Base Offset: 400

Group ID for reserved blocks: 0

Extended Superblock Section

454	First non-reserved Inode	11
458	Inode size	256
45A	This superblock's block group	0

Super Block

- Block 정보
 - Block count
 - Reserved block count
 - Free block
 - Block size
 - Blocks per group
 - Block group number
- Mount /time 정보
 - Last mount time
 - FileSystem mount time
 - Last write time
 - SuperBlock
 - Mount count
 - Maximal mount count

Ext2/Ext3/Ext4 Superblock, Base Offset: 400

Offset	Title	Value
400	Inode coun	373152
404	Block coun	1492029
408	Reserved b	74601
40C	Free block	874091
410	Free Inode	261045
414	First data b	0
418	Block size (2
41C	Fragment s	2
420	Blocks per	32768
424	Fragments	32768
428	Inodes per	8112
42C	Last mount	2009-05-19 08:26:21
430	Last write ti	2009-05-19 08:26:21
434	Mount cour	5
436	Maximal m	21
438	Magic sign:	53 EF
43A	File system	1
43C	Behavior w	1

Ext2/Ext3/Ext4 Superblock, Base Offset: 400

432 Group id for reserved block 0

Extended Superblock Section

454	First non-reserved Inode	11
458	Inode size	256
45A	This superblock's block gr	0

Super Block

- 기타
 - Version – 파일시스템의 버전정보
 - Creator OS – 파일시스템을 생성한 운영체제
 - State – 파일시스템의 상태(에러/정상)
 - Magic Signature – Super Block인지 확인 (0xEF53)
 - Journal – 저널링 관련 정보

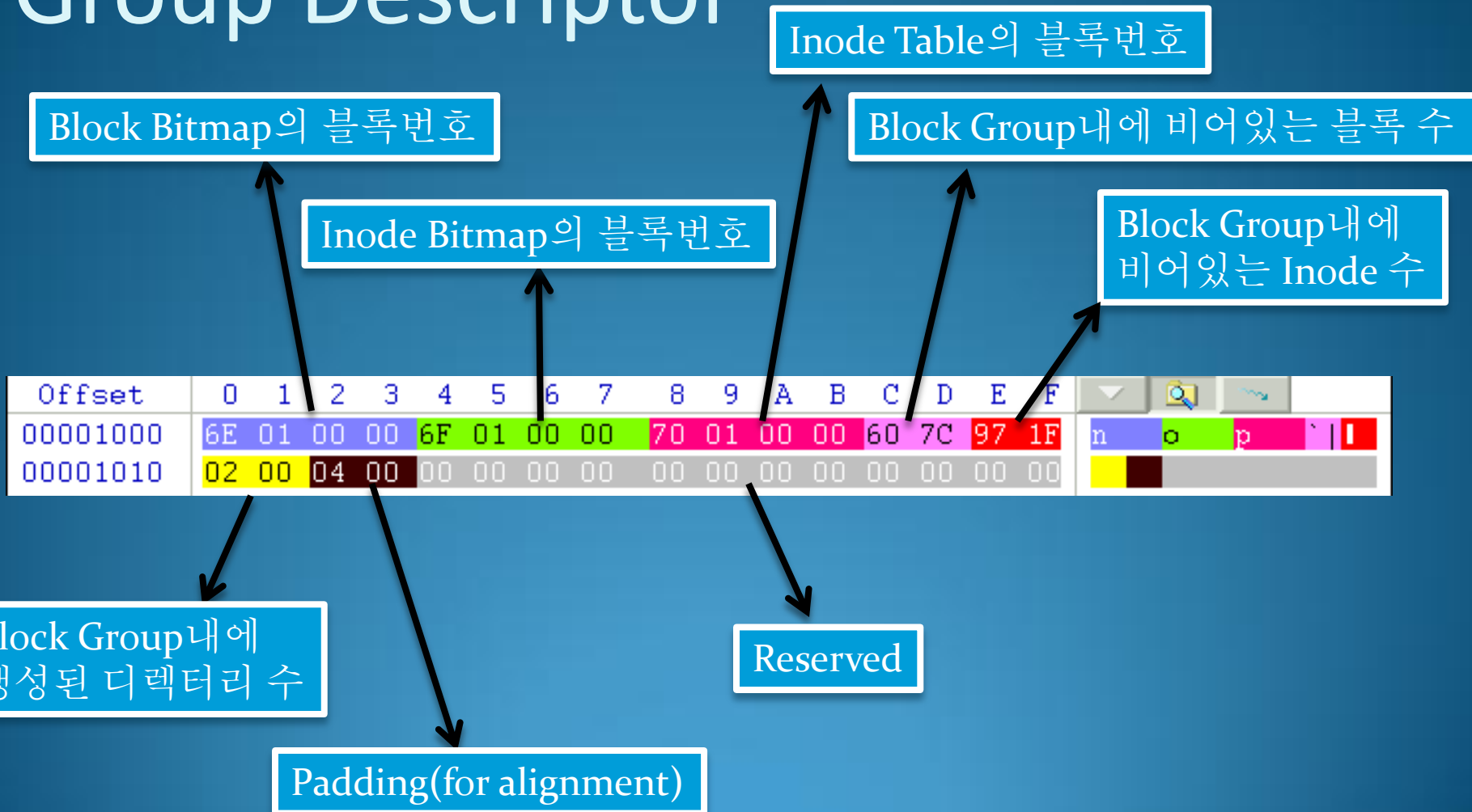
Group Descriptor Table

- Super Block 바로 다음에 위치
- 32byte 크기의 Group Descriptor들로 구성
- Descriptor는 각 블록그룹에 대한 정보를 가짐
- 크기가 가변적
 - Block Bitmap
 - Inode Bitmap
 - First Inode Table Block Number
- 그룹 내부의 블록, Inode, 빈디렉터리 개수를 가짐
- Backup Superblock과 동일한 Block Group에 Backup

Group Descriptor Table

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
00001000	6E	01	00	00	6F	01	00	00	70	01	00	00	60	7C	97	1F	n o p '
00001010	02	00	04	00	00	00	00	00	00	00	00	00	00	00	00	00	
00001020	6E	81	00	00	6F	81	00	00	70	81	00	00	93	70	8F	1F	n o p p
00001030	02	00	04	00	00	00	00	00	00	00	00	00	00	00	00	00	
00001040	00	00	01	00	01	00	01	00	02	00	01	00	61	7D	AA	1F	a } a
00001050	01	00	04	00	00	00	00	00	00	00	00	00	00	00	00	00	
00001060	6E	81	01	00	6F	81	01	00	70	81	01	00	95	04	81	14	n o p
00001070	0A	00	04	00	00	00	00	00	00	00	00	00	00	00	00	00	
00001080	00	00	02	00	01	00	02	00	02	00	02	00	00	00	DE	1A	p
00001090	DC	00	04	00	00	00	00	00	00	00	00	00	00	00	00	00	Ü
000010A0	6E	81	02	00	6F	81	02	00	70	81	02	00	DB	23	BC	13	n o p Ü#¼
000010B0	E6	02	04	00	00	00	00	00	00	00	00	00	00	00	00	00	æ
000010C0	00	00	03	00	01	00	03	00	02	00	03	00	C4	4C	4B	1B	ÄLK
000010D0	54	00	04	00	00	00	00	00	00	00	00	00	00	00	00	00	T
000010E0	6E	81	03	00	6F	81	03	00	70	81	03	00	C4	04	14	1B	n o p Ä
000010F0	20	01	04	00	00	00	00	00	00	00	00	00	00	00	00	00	

Group Descriptor



Inode

- Block과 더불어 가장 기본이 되는 단위
 - 모든 파일이나 디렉터리는 각기 하나의 Inode가 할당
- 하나의 Inode는 128/256byte의 크기를 가진다.
 - Super Block에 정의
 - 보통 EXT2는 128byte, EXT3는 256byte을 가짐

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
00170000	00	00	00	00	00	00	00	00	4A	10	0D	4A	4A	10	0D	4A
00170010	4A	10	0D	4A	00	00	00	00	00	00	00	00	00	00	00	00
00170020	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00170030	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00170040	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00170050	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00170060	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00170070	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00170080	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Inode

- Reserved Inode List
 - 여러 용도의 예약된 Inode가 존재.

Reserved Inode	Number	Comment
EXT3_BAD_INO	1	사용 불가능한 데이터 블록의 Inode
EXT3_ROOT_INO	2	루트 디렉터리의 Inode
EXT3_BOOT_LOADER_INO	5	부트로더의 Inode(사용되지 x)
EXT3_UNDER_DIR_INO	6	삭제 불가 디렉터리의 Inode
EXT3_GOOD_OLD_FIRST_INO	11	예약되지 않은 첫 번째 Inode

Inode

- File Mode
 - Unix : chmod
- Uid
 - Owner ID 하위 16bit
- Size in bytes
 - 파일 크기(byte 단위)
- Access time
 - 최종 접근 시간
- Change time
 - Inode정보의 변경 시간

00	00	00	00	00	00	00	00	00	00	4A	10	0D	4A	4A	10	0D	4A
4A	10	0D	4A	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Inode

- Modification time
 - 마지막 변경 시간
- Deletion time
 - 삭제된 시간
- Group ID
 - GID의 하위 16비트
- Link count
 - Hard Link Count
- Block Count
 - 데이터 저장에 필요한 블록 수

00	00	00	00	00	00	00	00	00	00	4A	10	0D	4A	4A	10	0D	4A
4A	10	0D	4A	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

Timestamp

- A-Time(Access Time)
 - When the contents of a file or directory is read
- A-Time is updated..
 - When process reads the contents
 - When file is copied
 - And file is moved to a new volume

Timestamp

- M-Time(Modified Time)
 - When the content of a file or directory changes
 - File - file content changes
 - Directory – file is created or deleted inside of it
 - The move is to the same volume
 - M-Time does not change
 - When a file is copied(inc network) – M-Time is updated

Timestamp

- C-Time(Change Time)
 - when permissions or ownership of a file are changed
 - When the content of a file or directory changes
 - If a file is moved then the C-time of the file will be updated
- D-Time(Deleted Time)
 - It is set only when a file has been deleted
 - It is cleared when the inode is allocated
 - Deleted file have the M-,C-times equal to the D-time

Timestamp

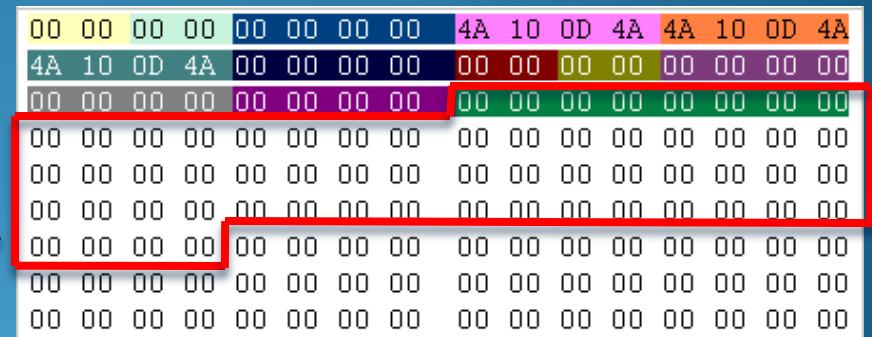
- Unix Time
 - Default Unix Time : 1970-01-01 00:00:00
- Ex) file is created
 - File : M-,A-,C- times are updated to the time of creation
 - D- time is set to 0
 - Parents Directory : M-,C- times are updated

Timestamp

- Ex) file is copied
 - Original parents directory have an updated A-time
 - Destination file has new M-,A-,C- times
 - Destination parents Directory have an updated M-,C-
- Ex) the move is to the new volume
 - Original – file was deleted, Destination – was created
 - Source inode will be unallocated
 - M-,A-,C-,D- times will updated

Inode

- Flags
 - File flag, 파일 변경 불가 등
- OS Description 1
 - 사용되지 않음
- Block point
 - 데이터 블록을 가리키는 포인터 배열

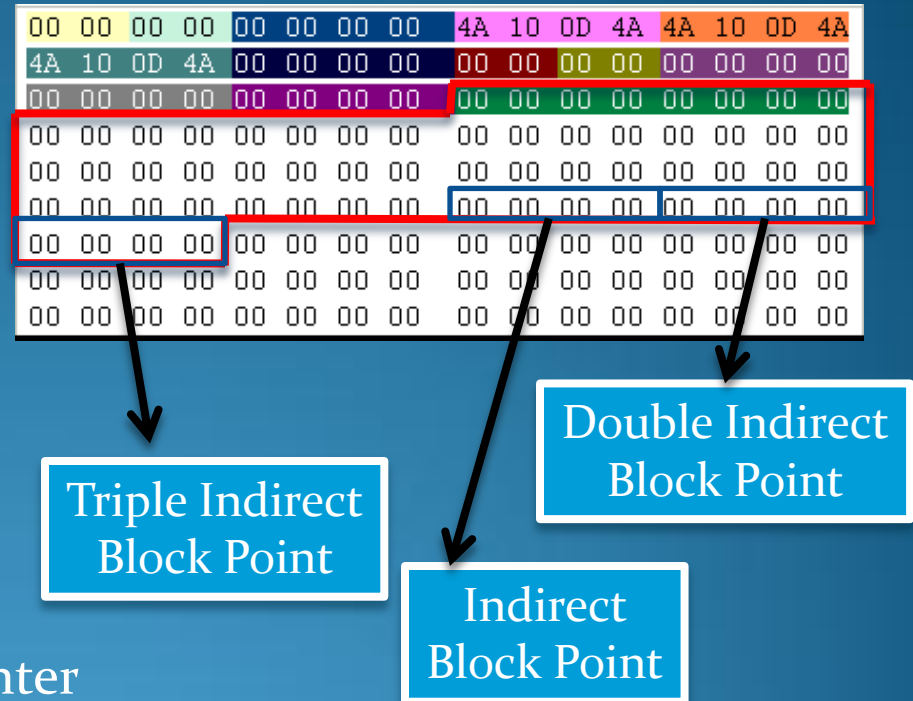


00	00	00	00	00	00	00	00	00	00	4A	10	0D	4A	4A	10	0D	4A
4A	10	0D	4A	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00
00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00

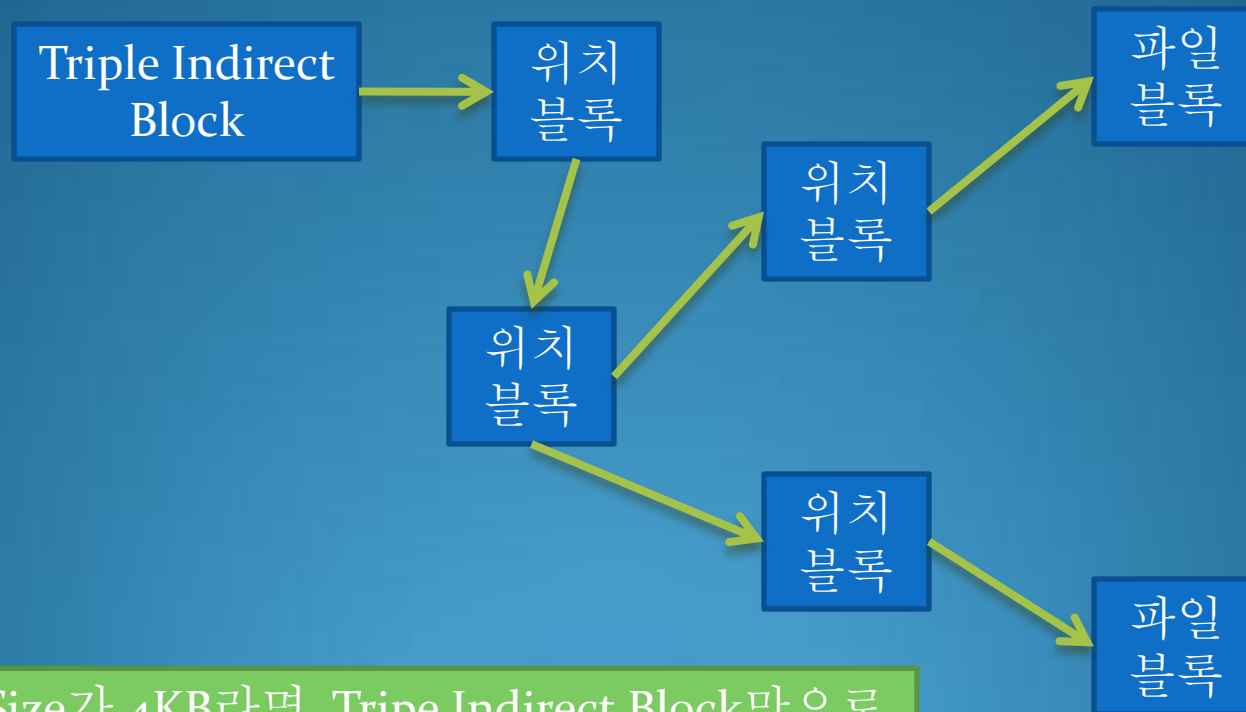
Inode

- Block Point는 int단위
 - 총 15개의 Block Point

- 각 블록포인트의 역할
 - Block[0~11] = Data Block Pointer
 - Block[12] = Indirect Block Pointer
 - Block[13] = Double Indirect Block Pointer
 - Block[14] = Triple Indirect Block Pointer



Inode(Block Point)



※ Block Size가 4KB라면, Triple Indirect Block만으로도 한 파일을 $1024 * 1024 * 1024 * 4KB = 4TB$ 까지 저장가능

Root Directory찾기

- Reserved Inode List 정보 → Inode index 2에 접근
- 해당 Block으로 이동해 보자
 - $0x1000 * 0x036B = 0x036B000$

00170100	ED 41 00 00 00 10 00 00	F2 16 0D 4A DF 16 0D 4A
00170110	DF 16 0D 4A 00 00 00 00	00 00 14 00 08 00 00 00
00170120	00 00 00 00 00 00 00 00	6B 03 00 00 00 00 00 00
00170130	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00170140	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00170150	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00170160	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00
00170170	00 00 00 00 00 00 00 00	00 00 00 00 00 00 00 00

Root Directory 찾기

- Root Directory의 정보

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0036B000	02	00	00	00	0C	00	01	02	2E	00	00	00	02	00	00	00	.
0036B010	0C	00	02	02	2E	2E	00	00	0B	00	00	00	14	00	0A	02	..
0036B020	6C	6F	73	74	2B	66	6F	75	6E	64	00	00	61	35	04	00	lost+found a5
0036B030	0C	00	03	02	76	61	72	00	71	94	04	00	0C	00	03	02	var q
0036B040	65	74	63	00	61	3A	02	00	10	00	05	02	6D	65	64	69	etc a: medi
0036B050	61	00	00	00	0C	00	00	00	10	00	05	07	63	64	72	6F	a cdro
0036B060	6D	00	00	00	21	B9	02	00	0C	00	03	02	62	69	6E	00	m !' bin
0036B070	B1	1F	00	00	0C	00	04	02	62	6F	6F	74	41	7C	01	00	± bootA
0036B080	0C	00	03	02	64	65	76	00	21	B4	04	00	0C	00	04	02	dev !'
0036B090	68	6F	6D	65	11	55	04	00	0C	00	03	02	6C	69	62	00	home U lib
0036B0A0	81	FD	00	00	0C	00	03	02	6D	6E	74	00	61	3F	00	00	!ý mnt a?
0036B0B0	0C	00	03	02	6F	70	74	00	A1	B6	03	00	0C	00	04	02	opt i
0036B0C0	70	72	6F	63	91	57	03	00	0C	00	04	02	72	6F	6F	74	proc'W root
0036B0D0	A1	BB	01	00	0C	00	04	02	73	62	69	6E	91	5C	01	00	i» sbin'\
0036B0E0	0C	00	03	02	73	72	76	00	D1	D8	02	00	0C	00	03	02	srv Ñ0
0036B0F0	73	79	73	00	91	52	05	00	0C	00	03	02	74	6D	70	00	sys 'R tmp
0036B100	11	5F	00	00	0C	00	03	02	75	73	72	00	0D	00	00	00	- usr
0036B110	14	00	0A	07	69	6E	69	74	72	64	2E	69	6D	67	00	00	initrd.img
0036B120	0E	00	00	00	E0	0E	07	07	76	6D	6C	69	6E	75	7A	00	à vmlinuz
0036B130	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	
0036B140	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

Root Directory 찾기

- 파일의 길이는 총 255바이트까지 가능

현재 디렉터리 엔트리 크기

디렉터리 엔트리의
하위 디렉터리/파일을 가리킴

파일명 길이

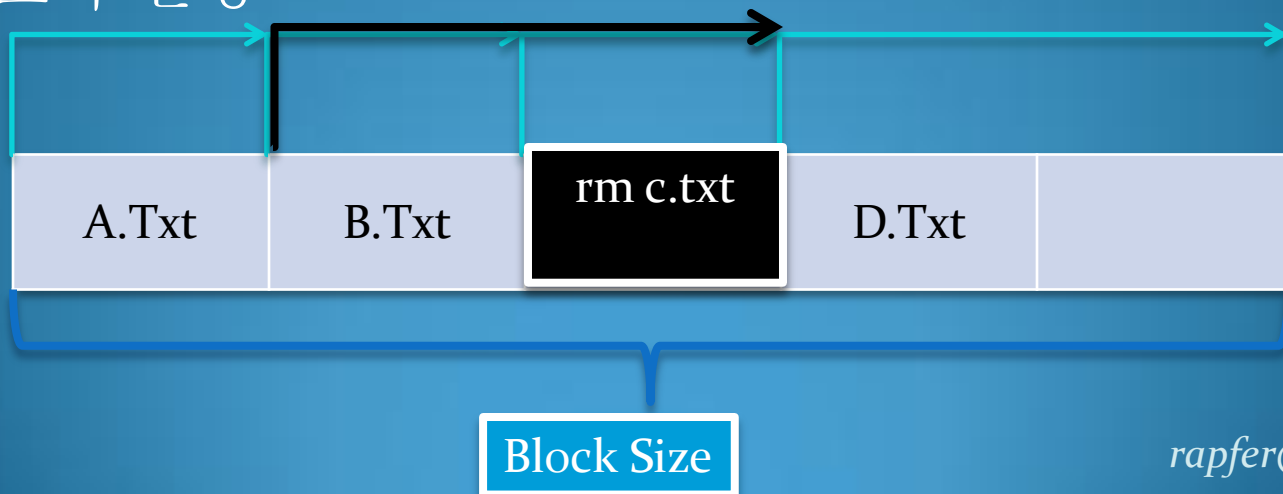
파일 타입

파일명

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
0036B000	02	00	00	00	0C	00	01	02	2E	00	00	00	02	00	00	00	..
0036B010	0C	00	02	02	2E	2E	00	00	0B	00	00	00	14	00	0A	02	lost+found a5
0036B020	6C	6F	73	74	2B	66	6F	75	6E	64	00	00	61	35	04	00	var q
0036B030	0C	00	03	02	76	61	72	00	71	94	04	00	0C	00	03	02	etc a: medi
0036B040	65	74	63	00	61	3A	02	00	10	00	05	02	6D	65	64	69	a cdro
0036B050	61	00	00	00	0C	00	00	00	10	00	05	07	63	64	72	6F	m !' bin
0036B060	6D	00	00	00	21	B9	02	00	0C	00	03	02	62	69	6E	00	± bootA
0036B070	B1	1F	00	00	0C	00	04	02	62	6F	6F	74	41	7C	01	00	dev !'
0036B080	0C	00	03	02	64	65	76	00	21	B4	04	00	0C	00	04	02	home U lib
0036B090	68	6F	6D	65	11	55	04	00	0C	00	03	02	6C	69	62	00	!ý mnt a?
0036B0A0	81	FD	00	00	0C	00	03	02	6D	6E	74	00	61	3F	00	00	

Root Directory 찾기

- Directory Entry
 - Root의 경우 기본적으로 Directory
 - Directory는 Directory Entry들을 가짐.
- Ex) File is deleted
 - 이전 Directory Entry의 Length를 삭제된 파일까지 포함 되도록 변경



Root Directory 찾기

•파일타입 옵션

Type	Value	Comment
EXT3_FT_UNKNOWN	0	Unknown Type
EXT3_FT_REG_FILE	1	Regular File
EXT3_FT_DIR	2	Directory
EXT3_FT_CHRDEV	3	Character Device
EXT3_FT_BLKDEV	4	Block Device
EXT3_FT_FIFO	5	Named Pipe
EXT3_FT SOCK	6	Socket
EXT3_FT_SYMLINK	7	Symbolic Link

/bin/lS 찾기

- Root directory 상에서 /bin/lS에 접근해보자.

해당 파일에 대한 inode 위치

파일명

0036B060	6D 00 00 00	21 B9 02 00	0C 00 03 02	62 69 6E 00	m	!	bin
0036B070	B1 1F 00 00	0C 00 04 02	62 6F 6F 74	41 7C 01 00	±		bootA
0036B080	0C 00 03 02	64 65 76 00	21 B4 04 00	0C 00 04 02			dev !'

※주의 : 디렉터리 또한 파일로 취급한다.

- 해당 정보를 기반으로 inode에 접근

/bin/ls 찾기

- 앞의 슈퍼블록 데이터를 기반으로 어느 그룹인지 산출
 - $0x0002B921 / 8112_{(10)} = 22 \rightarrow 22$ 번째 블록그룹
 - 22번 다음 블록이므로,
 - $\rightarrow 23$ 번 블록그룹에 위치

420	Blocks per	32768
424	Fragments	32768
428	Inodes per	8112
42C	Last mount	2009-05-19 08:26:21

- 그룹 내의 inode 위치
 - $(\text{Inode Count}-1) / \text{Inode_Per_Group}$
 - $(0x0002B921-1) \% 8112_{(10)} = 0$
- 23번째 블록그룹의 인덱스 0(첫 번째 엔트리) 접근

/bin/ls 찾기

- $32768(\text{Block per Group}) * 0x1000 \rightarrow \text{Block Group Size}$
- $\text{Block Group Size} * 22$
 - $= 0x8000000 * 22$
 - $= 0xB0000000$
- 이 주소는 23번째 블록그룹의 주소
 - GDT \rightarrow InodeTable : $0xB0002$ Block
 - $0xB0002000$ 이 Inode Table의 위치

420	Blocks per	32768
424	Fragments	32768
428	Inodes per	8112

B0002000	ED 41 00 00 00 10 00 00 7B EE 08 49 5C 17 0D 4A	iA	{i I\ J
B0002010	5C 17 0D 4A 00 00 00 00 00 00 02 00 08 00 00 00	\ J	
B0002020	00 00 00 00 00 00 00 00 00 08 0B 00 00 00 00 00		
B0002030	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
B0002040	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
B0002050	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		
B0002060	00 00 00 00 BC 53 D4 F9 00 00 00 00 00 00 00 00		4S0ù
B0002070	00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00		

Data Block의 위치

/bin/ls 찾기

- 해당 데이터 블록으로 접근
 - $0x1000 * 0xB0800 = 0xB0800000$


Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
B0800000	21	B9	02	00	0C	00	01	02	2E	00	00	00	02	00	00	00	!
B0800010	54	00	02	02	2E	2E	00	00	22	B9	02	00	48	00	0A	01	T .. "1 H
B0800020	61	72	63	68	64	65	74	65	63	74	00	00	23	B9	02	00	archdetect #1
B0800030	18	00	0D	01	61	75	74	6F	70	61	72	74	69	74	69	6F	autopartitio
B0800040	6E	00	00	00	24	B9	02	00	1C	00	12	01	61	75	74	6F	n \$1 auto
B0800050	70	61	72	74	69	74	69	6F	6E	2D	6C	6F	6F	70	00	00	partition-loop
B0800060	25	B9	02	00	0C	00	04	01	62	61	73	68	26	B9	02	00	%1 bash&1
B0800070	10	00	07	01	62	75	6E	7A	69	70	32	00	27	B9	02	00	bunzip2 '1
B0800080	10	00	05	01	62	7A	63	61	74	00	00	00	28	B9	02	00	bzcat (1
B0800090	10	00	05	07	62	7A	63	6D	70	00	00	00	29	B9	02	00	bzcmp)1
B08000A0	10	00	06	01	62	7A	64	69	66	66	00	00	2A	B9	02	00	bzdiff *1
B08000B0	10	00	07	07	62	7A	65	67	72	65	70	00	2B	B9	02	00	bzegrep +1

- 그림과 같이 /bin 디렉터리의 내용이 확인 가능
- 해당 구조를 읽어 들이면서 ls를 찾는다.

/bin/lS 찾기

- Length를 통해 한 디렉터리 엔트리 씩 접근을 수행

/bin/lS의 디렉터리 엔트리



B08003D0	5C B9 02 00 10 00 05 01 6C 6F 67 69 6E 00 00 00	\ ¹	login
B08003E0	5D B9 02 00 0C 00 02 01 6C 73 00 00 5E B9 02 00] ¹	ls ^{^1}
B08003F0	20 00 05 01 6C 73 6D 6F 64 00 00 00 5F B9 02 00		lsmod ₋ ¹

- 이제 inode로 접근하여 파일의 이미지를 확인하자.
 - 앞의 방법과 동일하게 접근
 - $0x2B95D / 8112_{(10)} = 22.007xx \rightarrow 23$ 번째 블록에 존재
 - $(0x2B95D - 1) \% 8112_{(10)} = 60 \rightarrow$ inode table의 60번째
 - $60 * 256\text{byte} = 15360\text{byte} = 0x3C00 \rightarrow 0xB0002000 + 0x3C00$
 - 최종 주소 : $0xB0005C00$

/bin/lis 찾기

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
B0005C00	ED	81	00	00	D8	77	01	00	29	16	0D	4A	7D	10	0D	4A	í 0w) J} J
B0005C10	FD	34	64	48	00	00	00	00	00	00	01	00	C8	00	00	00	ý4dH È
B0005C20	00	00	00	00	00	00	00	00	37	0C	0B	00	38	0C	0B	00	7 8
B0005C30	39	0C	0B	00	3A	0C	0B	00	3B	0C	0B	00	3C	0C	0B	00	9 : ; <
B0005C40	3D	0C	0B	00	3E	0C	0B	00	3F	0C	0B	00	40	0C	0B	00	= > ? @
B0005C50	41	0C	0B	00	42	0C	0B	00	43	0C	0B	00	00	00	00	00	A B C
B0005C60	00	00	00	00	07	54	D4	F9	00	00	00	00	00	00	00	00	Tôù
B0005C70	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	

- 해당 Block으로 이동(색칠된 부분)
- 순차적으로 정리되었기 때문에 맨 앞 블록으로 접근

/bin/lS (Success!!)

실제 리눅스 상의
파일 이미지

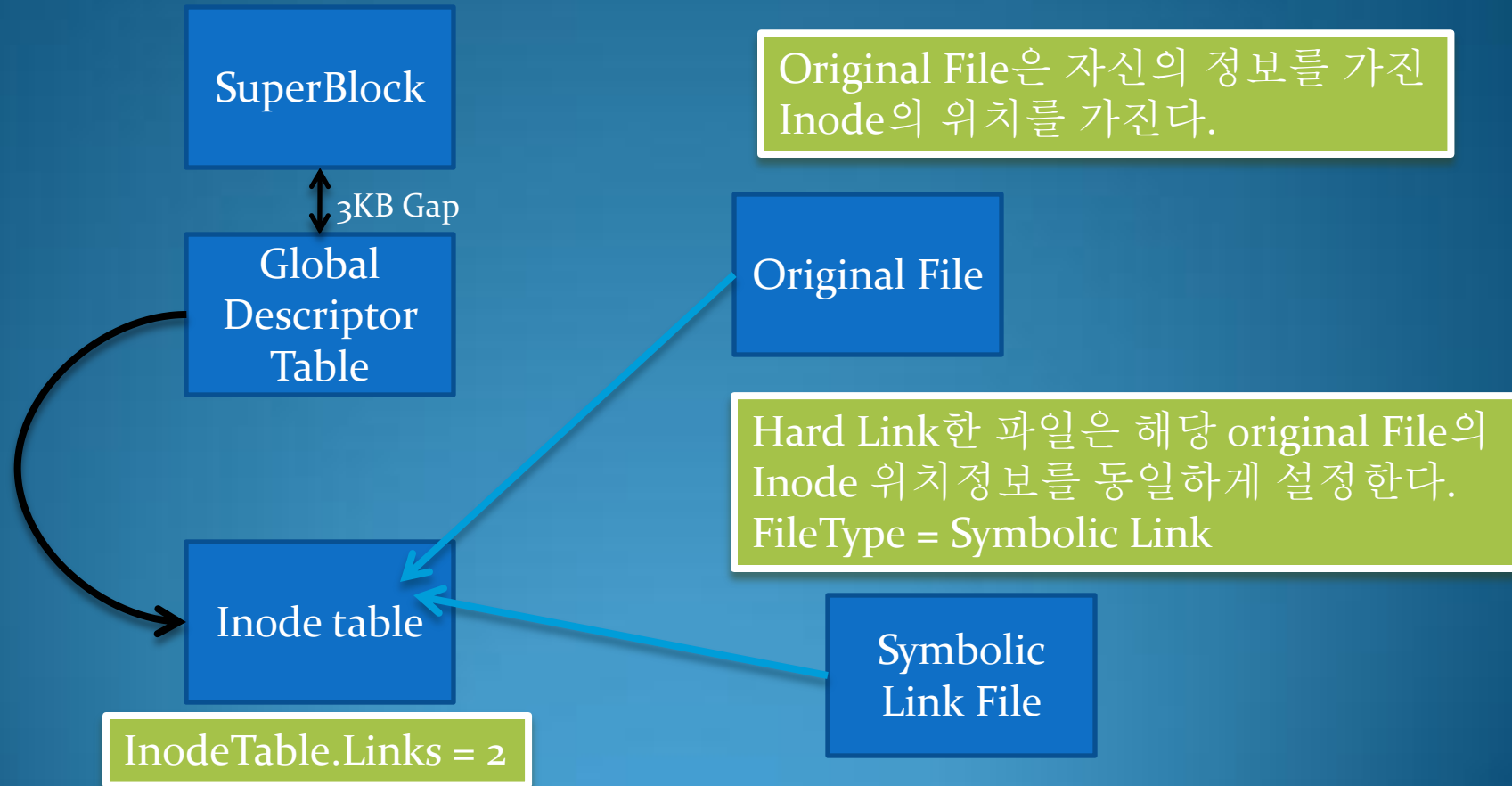
```
00000000 7F 45 4C 46 01 01 01 00 00 00 00 00 00 00 00 00 .ELF.....
00000010 02 00 03 00 01 00 00 00 20 9B 04 08 34 00 00 00 .....4....
00000020 78 73 01 00 00 00 00 00 34 00 20 00 09 00 28 00 xs.....4. ...
00000030 1C 00 1B 00 06 00 00 00 34 00 00 00 34 80 04 08 .....4...4...
00000040 34 80 04 08 20 01 00 00 20 01 00 00 05 00 00 00 4.....T...T...
00000050 04 00 00 00 03 00 00 00 54 01 00 00 54 81 04 08 .....T.....
00000060 54 81 04 08 13 00 00 00 13 00 00 00 04 00 00 00 T.....
00000070 01 00 00 00 01 00 00 00 00 00 00 00 00 80 04 08 .....n...n...
00000080 00 80 04 08 B4 6E 01 00 B4 6E 01 00 05 00 00 00 .....n...n...
00000090 00 10 00 00 01 00 00 00 F0 6E 01 00 F0 FE 05 08 .....n...
000000A0 F0 FE 05 08 A0 03 00 00 1C 08 00 00 06 00 00 00 .....o...
000000B0 00 10 00 00 02 00 00 00 04 6F 01 00 04 FF 05 08 .....o...
000000C0 04 FF 05 08 E8 00 00 00 E8 00 00 00 06 00 00 00 .....h...h...
000000D0 04 00 00 00 04 00 00 00 68 01 00 00 68 81 04 08 .....h...h...
000000E0 68 81 04 08 20 00 00 00 20 00 00 00 04 00 00 00 h.....
000000F0 04 00 00 00 50 E5 74 64 EC 6D 01 00 EC ED 05 08 ....P.td.m....
00000100 EC ED 05 08 2C 00 00 00 2C 00 00 00 04 00 00 00 .....Q.td.....
00000110 04 00 00 00 51 E5 74 64 00 00 00 00 00 00 00 00 .....R.td.n....
00000120 00 00 00 00 00 00 00 00 00 00 00 00 06 00 00 00 .....
00000130 04 00 00 00 52 E5 74 64 F0 6E 01 00 F0 FE 05 08 .....
-%% ls --0x0/0x177D8-----
```

Offset	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F	
BOC37000	7F	45	4C	46	01	01	01	00	00	00	00	00	00	00	00	00	ELF
BOC37010	02	00	03	00	01	00	00	00	20	9B	04	08	34	00	00	00	4
BOC37020	78	73	01	00	00	00	00	00	34	00	20	00	09	00	28	00	(
BOC37030	1C	00	1B	00	06	00	00	00	34	00	00	00	34	80	04	08	4
BOC37040	34	80	04	08	20	01	00	00	20	01	00	00	05	00	00	00	4
BOC37050	04	00	00	00	03	00	00	00	54	01	00	00	54	81	04	08	T
BOC37060	54	81	04	08	13	00	00	00	13	00	00	00	04	00	00	00	T
BOC37070	01	00	00	00	01	00	00	00	00	00	00	00	00	80	04	08	
BOC37080	00	80	04	08	B4	6E	01	00	B4	6E	01	00	05	00	00	00	'n
BOC37090	00	10	00	00	01	00	00	00	F0	6E	01	00	F0	FE	05	08	ân
BOC370A0	F0	FE	05	08	A0	03	00	00	1C	08	00	00	06	00	00	00	âp
BOC370B0	00	10	00	00	02	00	00	00	04	6F	01	00	04	FF	05	08	o
BOC370C0	04	FF	05	08	E8	00	00	00	E8	00	00	00	06	00	00	00	ÿ
BOC370D0	04	00	00	00	04	00	00	00	68	01	00	00	68	81	04	08	h
BOC370E0	68	81	04	08	20	00	00	00	20	00	00	00	04	00	00	00	h
BOC370F0	04	00	00	00	50	E5	74	64	EC	6D	01	00	EC	ED	05	08	Pâtdim
BOC37100	EC	ED	05	08	2C	00	00	00	2C	00	00	00	04	00	00	00	ii
BOC37110	04	00	00	00	51	E5	74	64	00	00	00	00	00	00	00	00	Qâtd
BOC37120	00	00	00	00	00	00	00	00	00	00	00	00	06	00	00	00	

필자가 찾은
파일 이미지

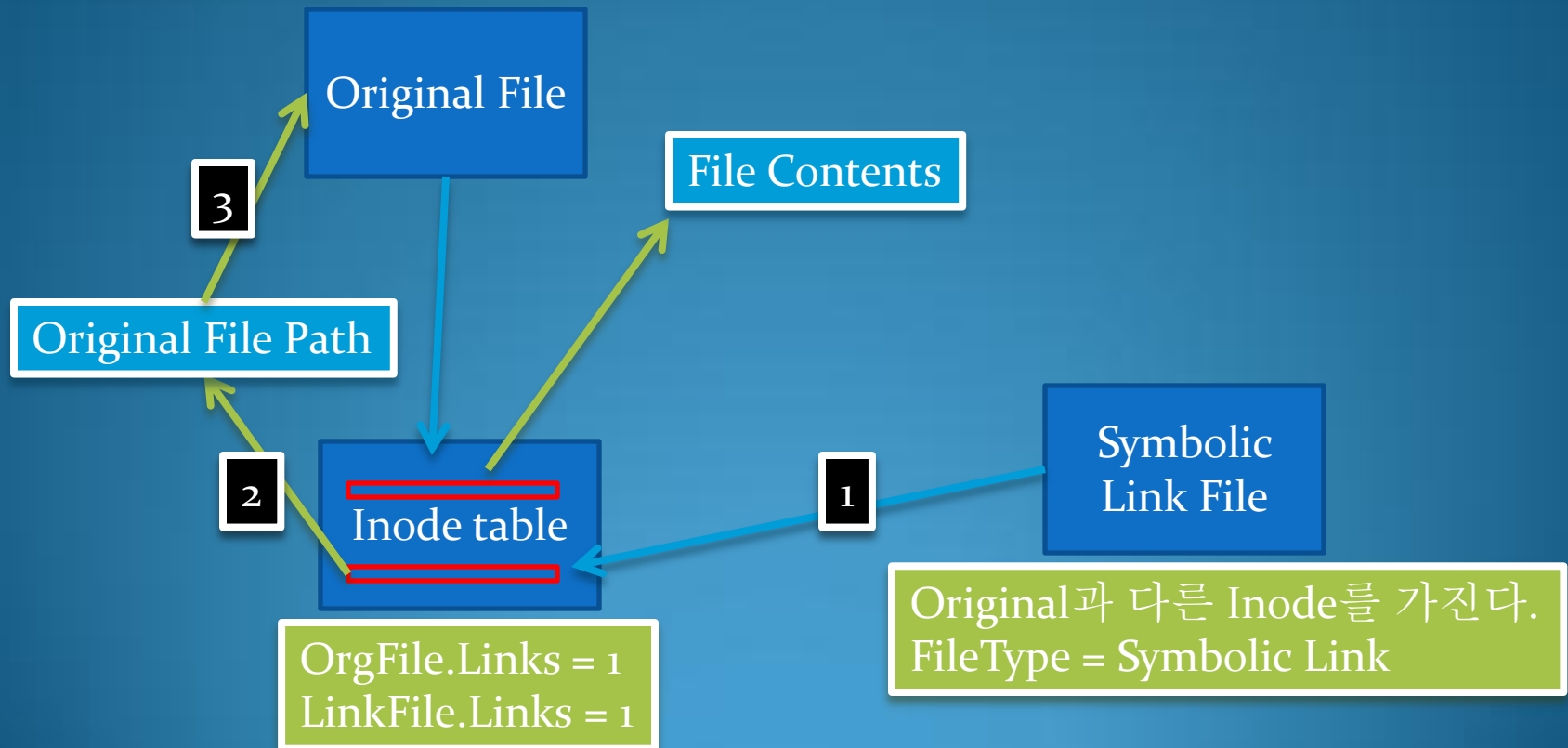
추가사항

Hard Link



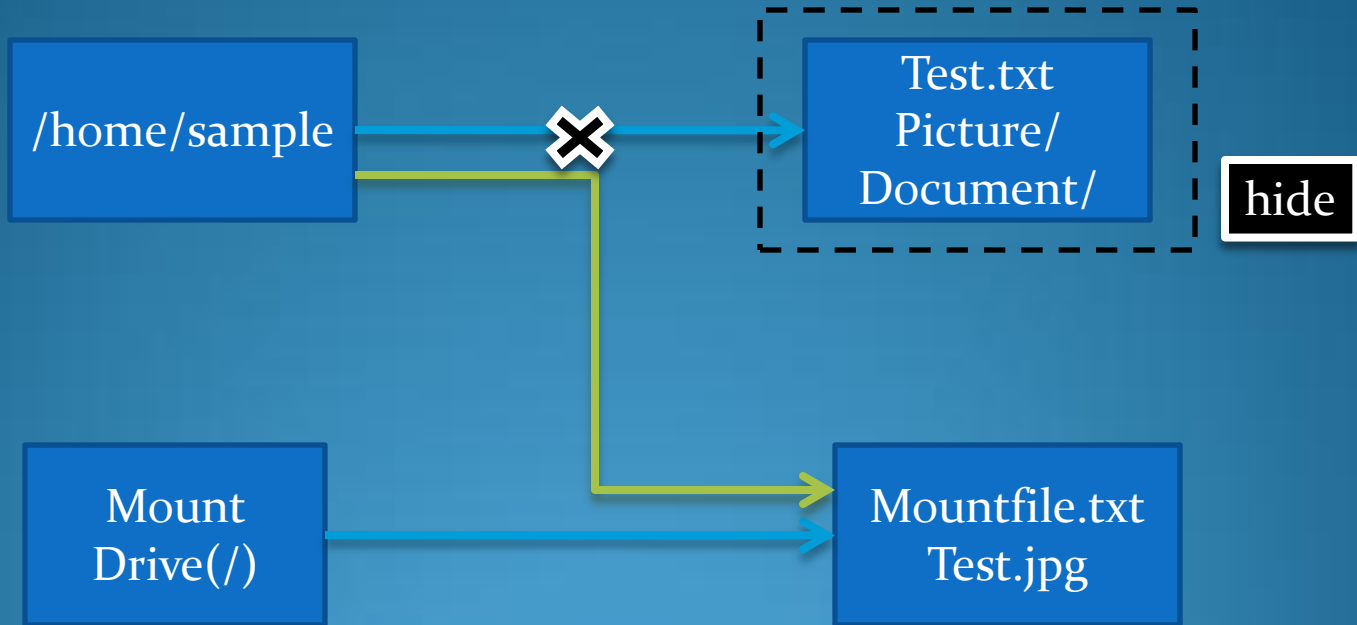
Soft Link

Original File은 자신의 정보를 가진 Inode의 위치를 가진다.



Mount

- /home/sample에 다른 디바이스를 mount 시..



파일 삭제

- Ext2의 경우 파일 삭제 시,
 - 해당 Directory Entry의 inode정보 삭제
 - Block Bitmap, Inode Bitmap의 해당 Bit Clear
 - Super Block의 해당 데이터 갱신
- EXT3의 경우
 1. Directory Entry의 inode정보를 기준으로 inode구조체 접근
 2. 해당 inode 구조체 내의 Block[x]을 전부 0으로 초기화
 3. Block Bitmap, Inode Bitmap의 해당 Bit Clear
 4. Super Block의 해당 데이터 갱신

Journaling

- Ext3는 3단계의 저널링을 지원
 - Journal(Risk : Low)
 - FileSystem의 Metadata와 File Contents를 Journal에 기록
 - Commit 시 Journal 내용을 디스크에 기록
 - Ordered(Risk : Medium)
 - Metadata만 Journal에 기록
 - Commit 시 Metadata와 Contents를 디스크에 기록
 - Linux Default Setting
 - Writeback(Risk : High)
 - Metadata만 Journal에 기록,
 - File Contents는 Commit 전/후에 실시

Question?