EliceAcademy

핵심 운영체제

04 파일 시스템

목차

운영체제가 파일을
관리하는 방법을
알아봅시다

01 파일과 파일 시스템

02 디스크 할당

03 링크

04 마운트

01

파일과 파일 시스템

파일 (file)이란?

컴퓨터에서 정보를 담는 논리적인 저장 단위

- 파일은 비휘발성 기억장치에 저장되어 있다.

파일 메타데이터 (file’s metadata)

파일을 관리하기 위한 정보들

- 파일 이름
- 유형
- 저장된 위치
- 파일 크기
- 권한
- 생성된 시간, 변경된 시간
- ...

디렉토리 (directory)

파일에 대한 여러 정보를 가지고 있는 특수한 파일

- 해당 디렉토리에 속한 파일의 이름과 속성들을 포함하고 있음
- 검색, 생성, 삭제 등의 기능을 제공

파일 시스템 (file system)

파일들을 쉽게 접근할 수 있도록 관리하는 체제

- 파일 시스템은 크게 디스크, 네트워크, 그리고 특수 용도의 파일 시스템으로 나뉨
- 운영 체제는 각각 파일 시스템을 가지고 있음

02

디스크 할당

EliceAcademy

순차 접근 저장장치

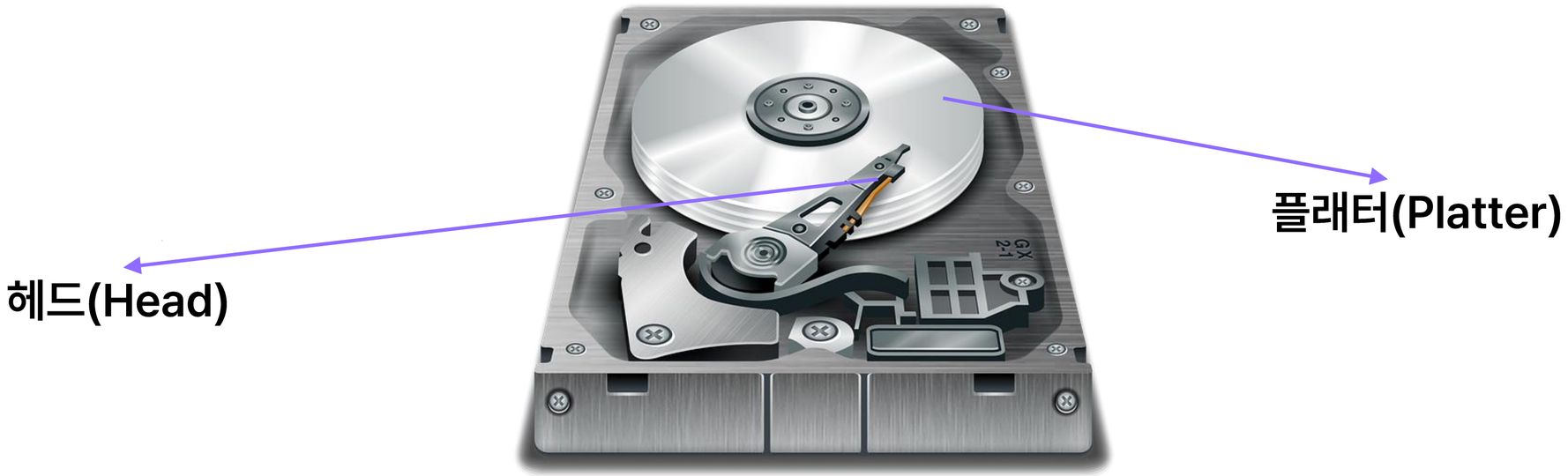
- 앞에서부터 순서대로 읽어야 하는 저장 장치
- 원하는 지점을 찾기 어렵고 빨리 감기와 되감기 기능이 필요하다

직접 접근 저장장치

- 헤드를 옮겨 원하는 부분부터 접근이 가능
- 모든 데이터를 읽지 않고도 바로 원하는 곳에 접근이 가능

EliceAcademy

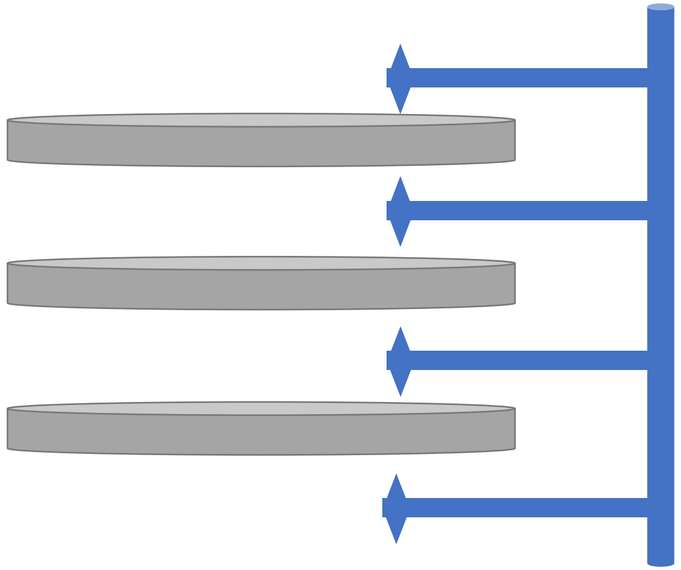
하드 디스크

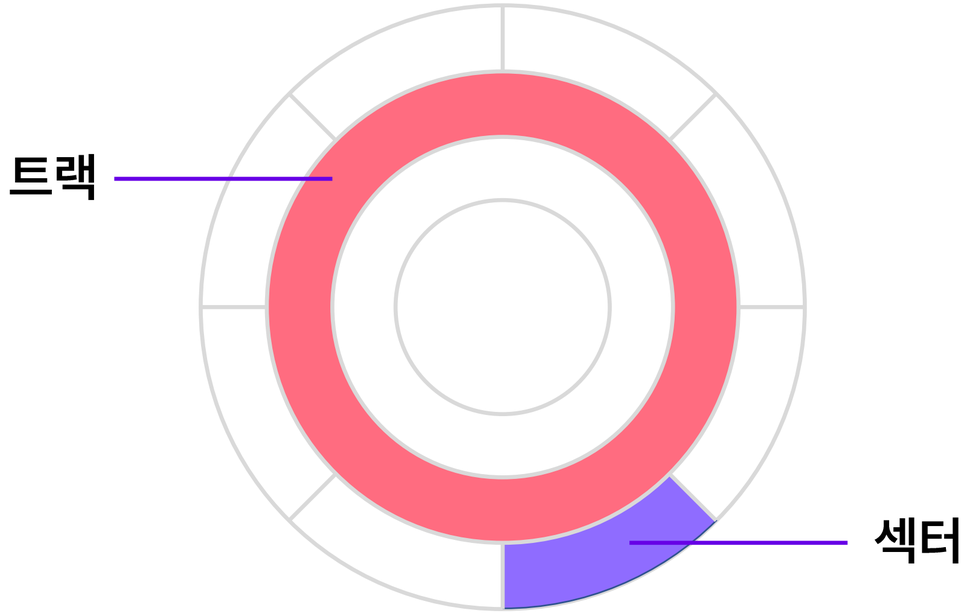


- 플래터 사이사이에 정보를 읽고 쓸 수 있는 헤드가 들어가 있음
- 플래터가 회전하고 헤드가 정보를 읽어오는 방식

EliceAcademy

하드 디스크





- **트랙(Track):** 플래터에서 띠 모양의 영역
- **섹터(Sector):** 하드 드라이브의 최소 기억 단위

EliceAcademy

하드 디스크와 SSD

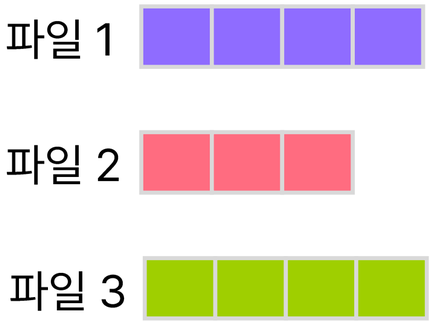
- 하드 디스크와 다르게 물리적인 기계 장치가 없이 전기 장치로만 구성되어 있음
- 모터와 같은 기계 장치가 없어서 하드 디스크에 비해서 훨씬 빠름

파일 할당

하드 디스크

0	1	2	3	4	5
6	7	8	9	10	11
11	12	13	14	15	16
17	18	19	20	21	22
23	24	25	26	27	28
29	30	31	32	33	34
35	36	37	38	39	40
41	42	43	44	45	46

저장할 파일



- 하드 디스크는 블록이라는 단위로 나누어 번호를 붙임
- 파일을 저장하는 방법들이 여러가지가 있음

연속 할당 (Contiguous Allocation)

0	1	2	3	4	5
6	7	8	9	10	11
11	12	13	14	15	16
17	18	19	20	21	22
23	24	25	26	27	28
29	30	31	32	33	34
35	36	37	38	39	40
41	42	43	44	45	46

- 앞에서부터 차례대로 데이터를 입력하는 방식

- 헤드의 이동을 최소화하여 입출력 속도가 빠름
- 순차 접근과 직접 접근이 모두 가능

- 할당과 삭제를 반복할 경우 중간에 구멍이 생기는 외부 단편화가 발생
- 파일이 계속 커지는 경우에는 매우 부적절한 방법

연결리스트(Linked List)

데이터와 다음을 가리키는 포인터로 구성되어 한 줄로 연결된 자료구조

- 길이가 계속 변화하는 데이터에 적합한 자료구조
- 데이터를 읽고 포인터를 보고 다음 노드로 이동하는 작업을 반복하여 데이터를 읽음

연결리스트(Linked List)

화살표
데이터와 다음을 가리키는 포인터로 구성되어 한 줄로 연결된 자료구조

- 길이가 계속 변화하는 데이터에 적합한 자료구조
- 데이터를 읽고 포인터를 보고 다음 노드로 이동하는 작업을 반복하여 데이터를 읽음

연결리스트(Linked List)

아래를 보세요
↓

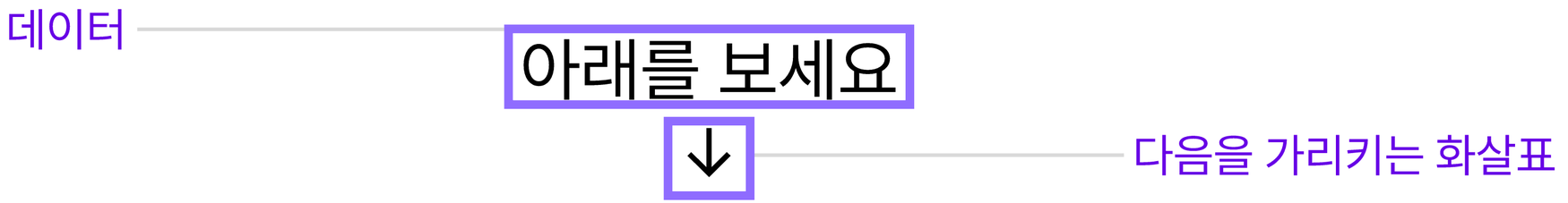
엘리스 최고

↑
위를 보세요

오른쪽을 보세요 →

EliceAcademy

연결리스트(Linked List)

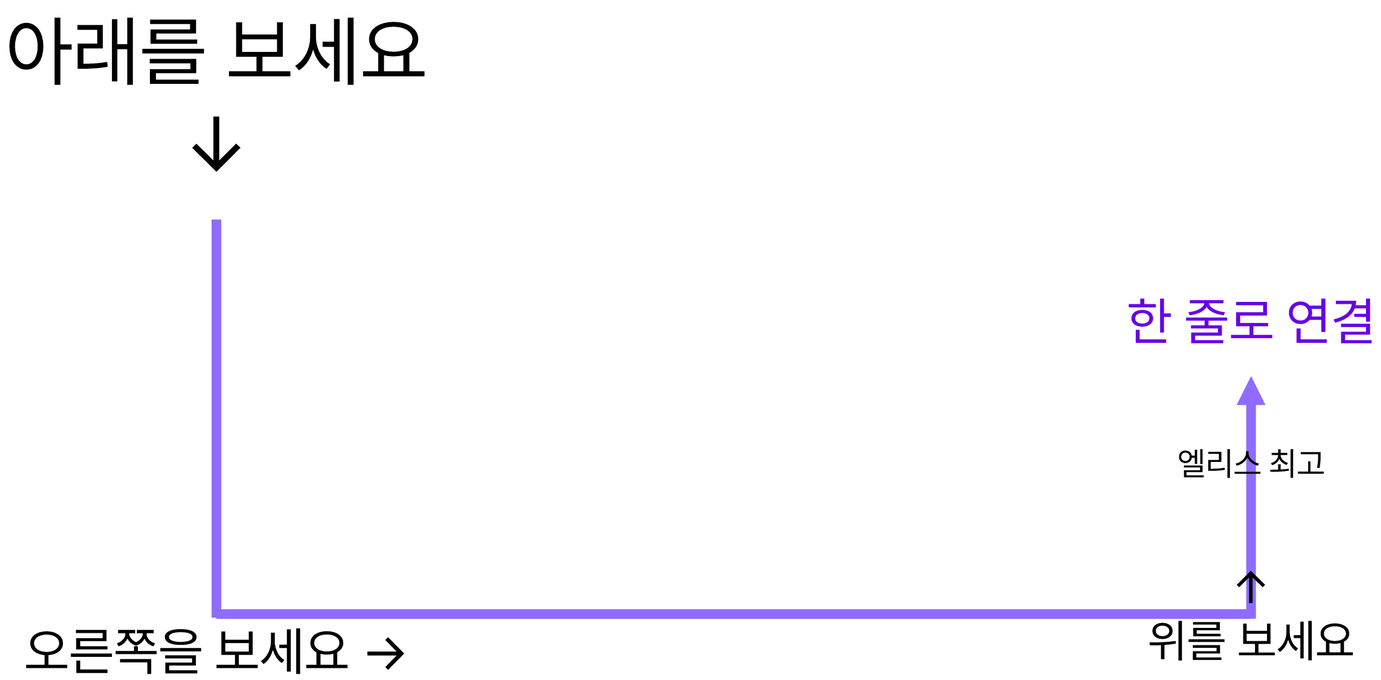


엘리스 최고

↑
위를 보세요

오른쪽을 보세요 →

연결리스트(Linked List)



연결리스트(Linked List)

아래를 보세요
↓

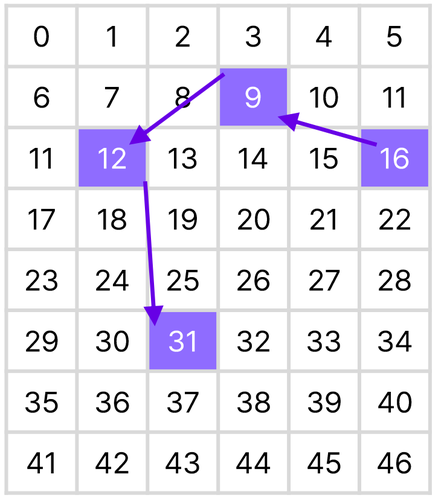
엘리스 최고

← 왼쪽을 보세요

오른쪽을 보세요 →

↑
위를 보세요

연결 할당 (Linked Allocation)



- 연결리스트와 같은 방식으로 파일을 할당
- 데이터와 주소를 저장하는 공간으로 나눠서 사용

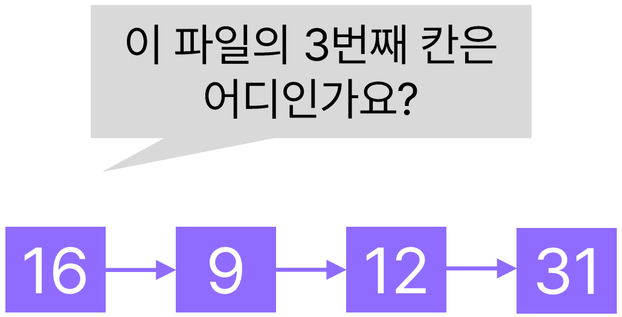
- 외부 단편화 문제가 없다.

- 헤드가 여러 번 움직여야 한다.
- 직접 접근이 불가능하다.
- 한 칸이 손상되면 나머지 모든 데이터를 읽을 수 없다.

EliceAcademy

연결 할당의 단점

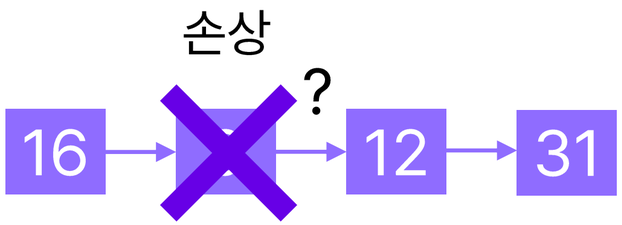
직접 접근이 불가능



- 시작 위치를 알아도 화살표를 따라가지 않으면 다음 위치를 알 수 없음
- 한 파일은 무조건 처음부터 접근해야 함

연결 할당의 단점

한 칸이 손상되면 나머지 모든 데이터를 읽을 수 없음



- 화살표가 손상되면 다음으로 읽을 곳이 어디인지 알 수 없음
- 중간에 연결이 손상될 경우 이후 모든 데이터에 접근할 방법이 없음

FAT(File Allocation Table) 방식

화살표들을 모아서 하나의 표로 저장해두자

1	
...	
9	12
...	
12	31
...	
16	9
...	
31	-1
...	

FAT

0	1	2	3	4	5
6	7	8	9	10	11
11	12	13	14	15	16
17	18	19	20	21	22
23	24	25	26	27	28
29	30	31	32	33	34
35	36	37	38	39	40
41	42	43	44	45	46

- 각 칸이 다음은 어디로 연결되는지 FAT라는 테이블에 기록
- FAT만 읽어 두면 원하는 칸부터 접근하는 직접 접근도 가능
- 손상되어도 FAT를 보면 다음 칸이 어디인지 알 수 있음

FAT(File Allocation Table) 방식의 한계

FAT크기에 따라서 최대 파일 크기와 파티션의 크기가 제한됨

1	
...	
9	12
...	
12	31
...	
16	9
...	
31	-1
...	

FAT

0	1	2	3	4	5
6	7	8	9	10	11
11	12	13	14	15	16
17	18	19	20	21	22
23	24	25	26	27	28
29	30	31	32	33	34
35	36	37	38	39	40
41	42	43	44	45	46

- (FAT에 기록할 수 있는 칸의 수) * (한 칸의 크기) 보다 큰 저장 장치에서는 사용할 수 없음

- 칸 번호의 자릿수를 늘림: 표가 커질수록 검색하는 시간이 증가
- 칸의 크기를 늘림: 작은 파일도 큰 칸에 저장하면서 단편화 발생

EliceAcademy

FAT(File Allocation Table)방식의 발전

FAT에서 주소를 표시하는 비트 수를 변경하면서 발전

1	
...	
9	12
...	
12	31
...	
16	9
...	
31	-1
...	

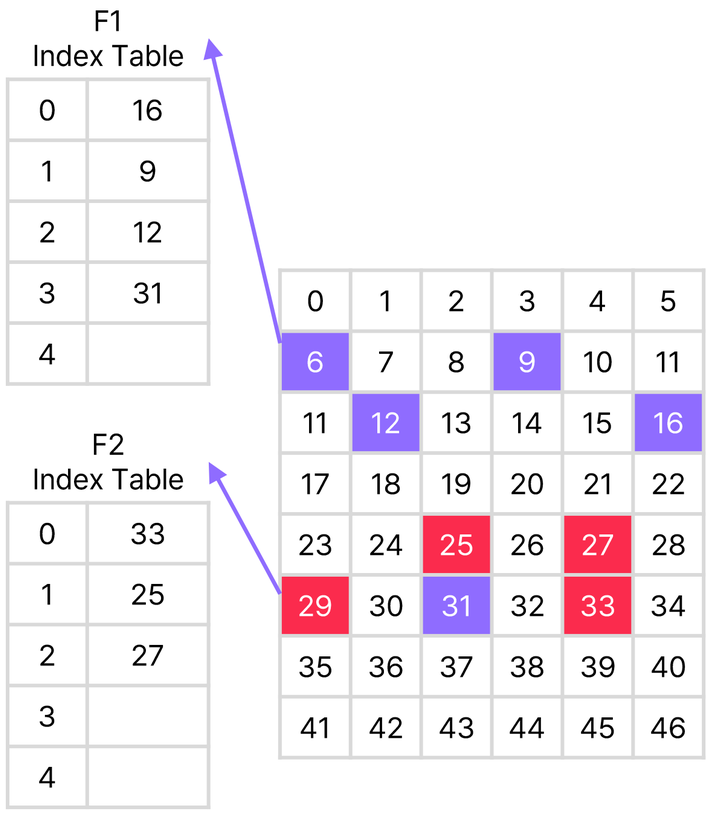
FAT

0	1	2	3	4	5
6	7	8	**9**	10	11
11	**12**	13	14	15	**16**
17	18	19	20	21	22
23	24	25	26	27	28
29	30	**31**	32	33	34
35	36	37	38	39	40
41	42	43	44	45	46

- 칸의 번호를 몇 비트로 기록할 것인지에 따라 버전이 달라짐
- FAT16, FAT32, exFAT가 그 예시

- 표가 아닌 트리의 형태로 저장하는 NFTS 방식도 등장

색인 할당 (Indexed Allocation)



- 번호들을 순서대로 한 칸에 기록
- 번호들이 기록된 칸을 인덱스 블록이라고 부름

- 인덱스 블록을 읽으면 직접 접근이 가능
- 외부 단편화 문제가 발생하지 않는다.

- 이 표를 저장하기 위해 파일마다 한 칸을 사용해야 한다
- 한 칸에 들어갈 수 없을 정도로 큰 파일은 저장할 수 없다

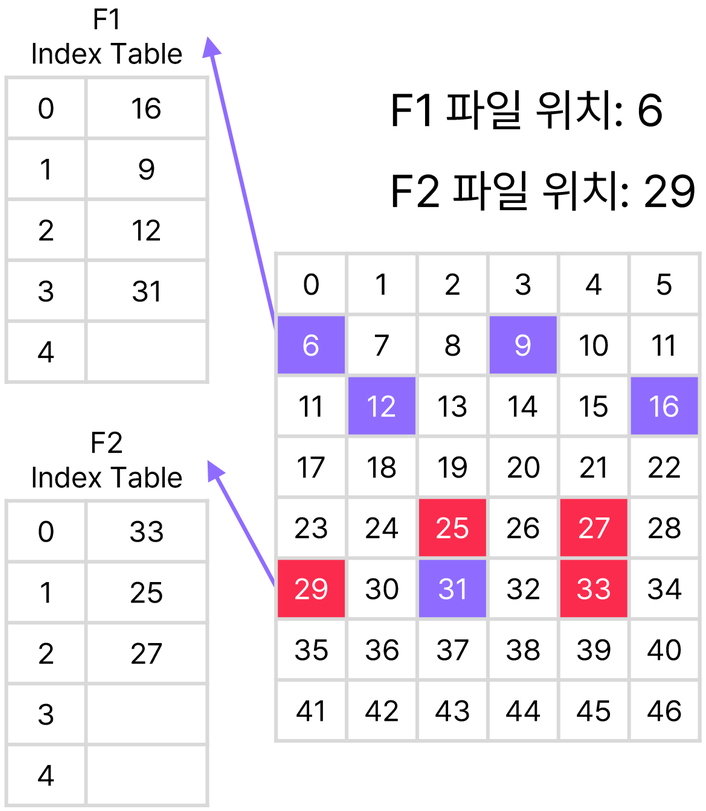
EliceAcademy

03

링크

EliceAcademy

색인 할당 (Indexed Allocation)



- 번호들을 순서대로 한 칸에 기록
- 번호들이 기록된 칸을 인덱스 블록이라고 부름
- 파일이 인덱스 테이블의 위치를 가리킴

리눅스의 inode (index-node)

F1 Inode 12351

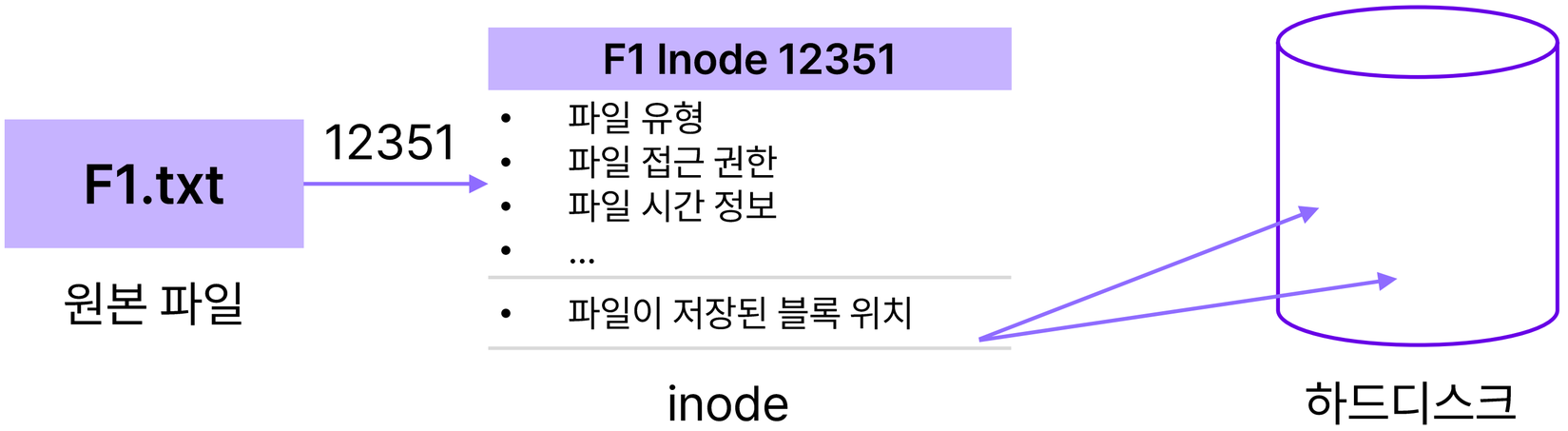
- 파일 유형
- 파일 접근 권한
- 파일 시간 정보
- ...
- 파일이 저장된 블록 위치

F2 Inode 52231

- 파일 유형
- 파일 접근 권한
- 파일 시간 정보
- ...
- 파일이 저장된 블록 위치

- 파일에 대한 모든 정보가 기록
- 각 노드에 번호를 부여함
- 모든 파일과 디렉토리는, 고유한 inode를 가짐

리눅스의 inode (index-node)



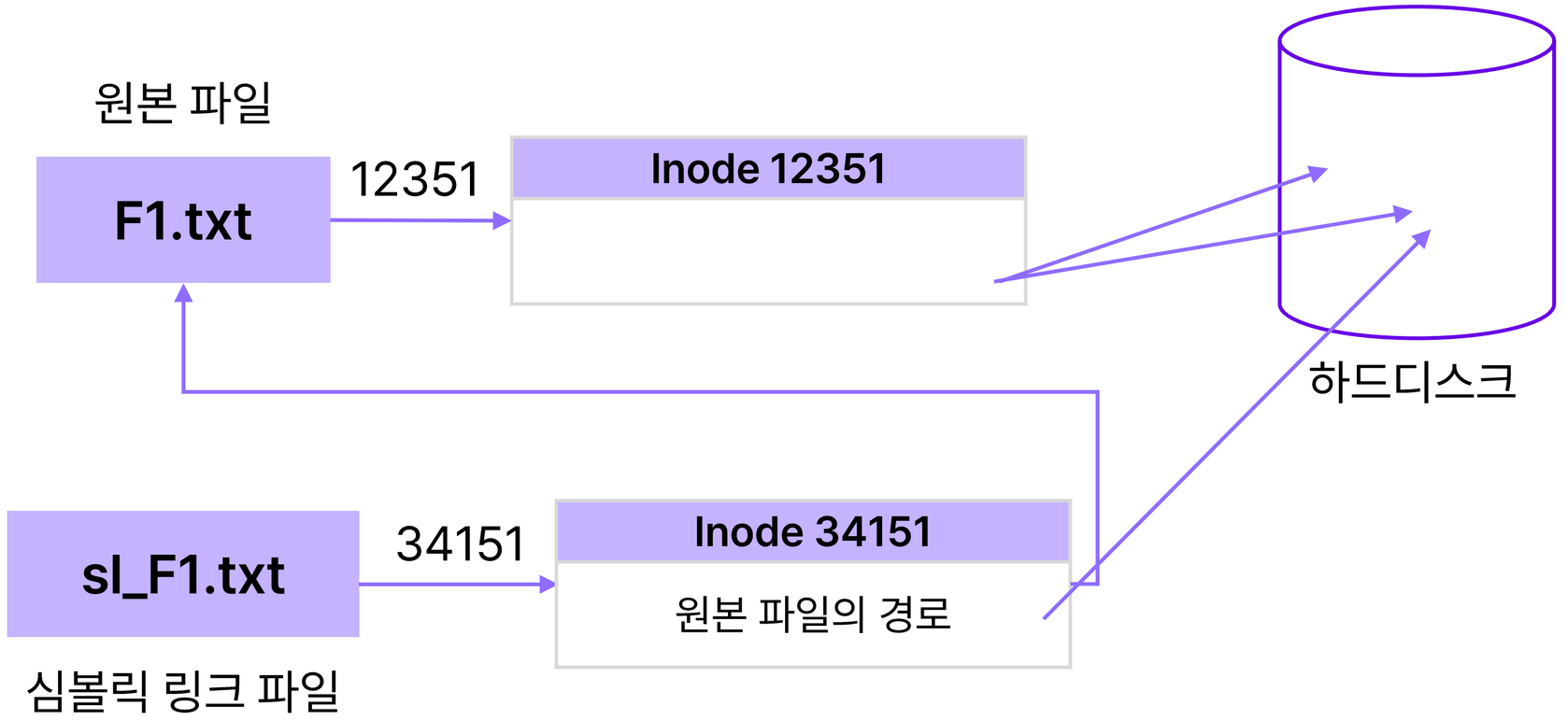
바로가기

- 바로가기를 삭제해도 원본 파일이 삭제되지 않음
- 바로가기를 실행하는 것은 해당 파일을 실행하는 것과 동일

심볼릭 링크와 하드링크

리눅스의 바로가기 기능

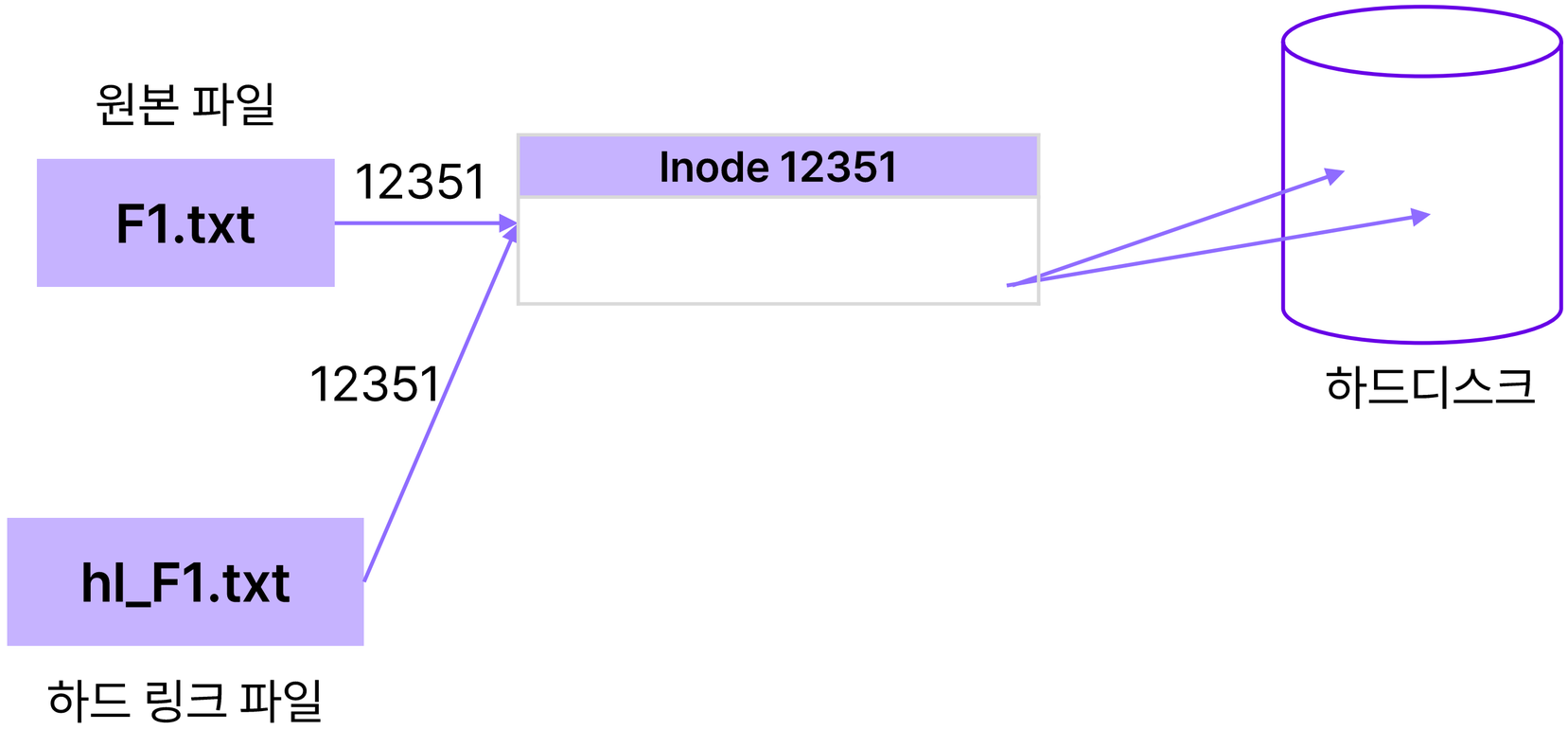
심볼릭 링크 (Symbolic link)



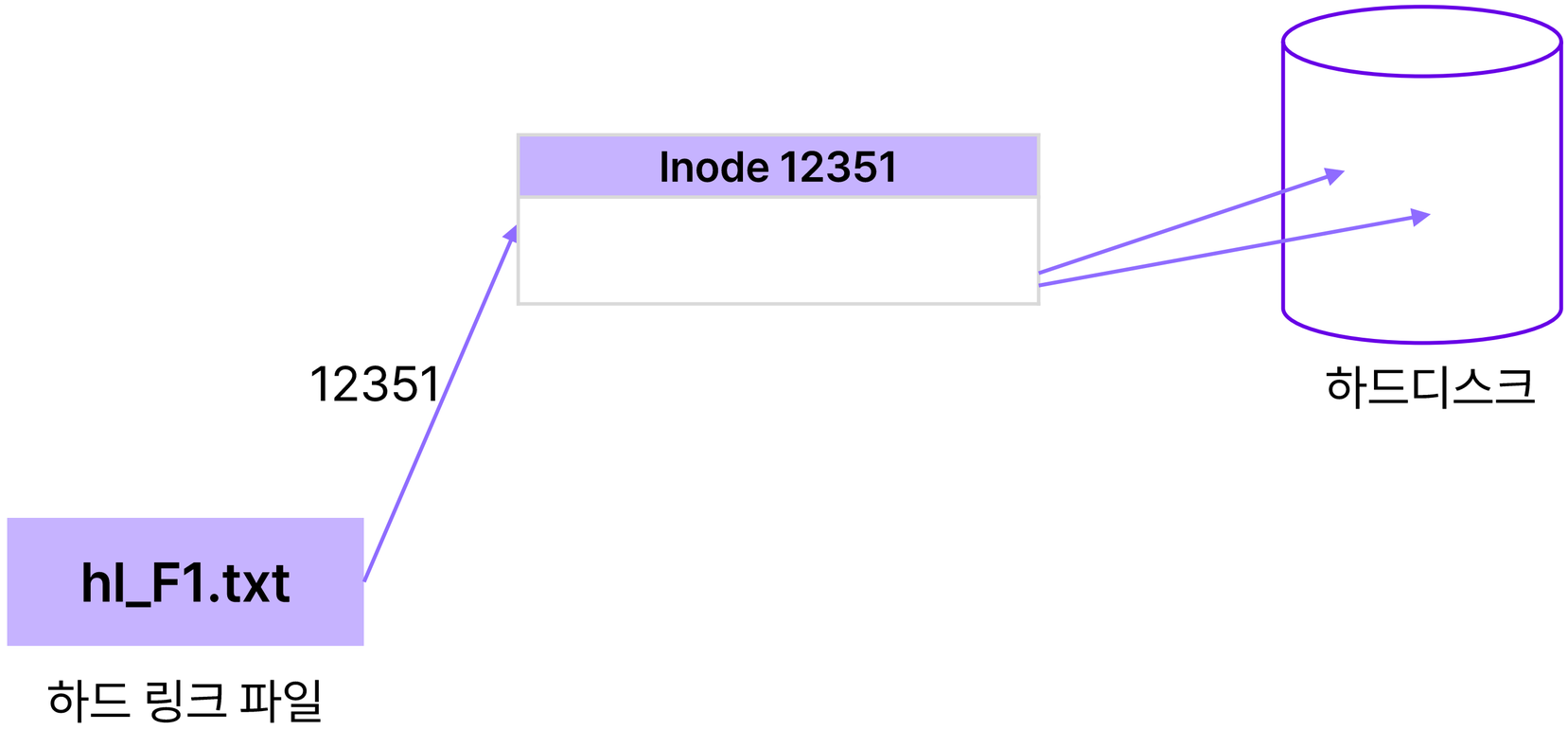
심볼릭 링크 (Symbolic link)의 특징

- 원본파일이 삭제되거나 경로가 변하면 링크가 끊어짐
- 링크 파일을 수정하면 원본 파일의 내용이 동일하게 수정됨
- 링크 파일도 별도의 inode를 가짐

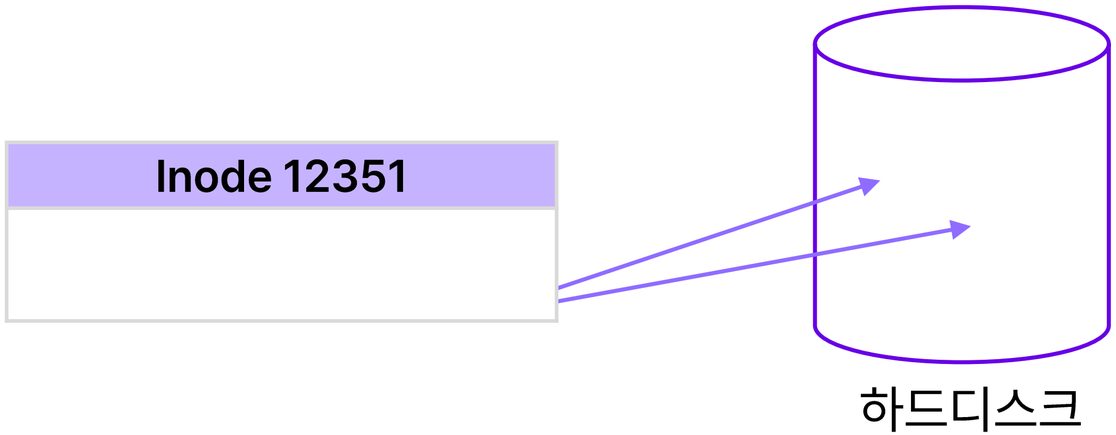
하드 링크 (Hard link)



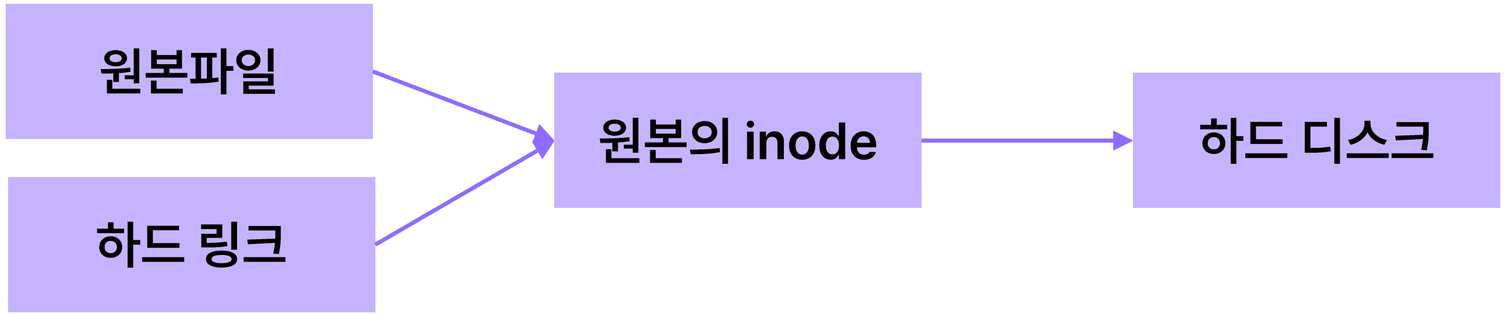
하드 링크 (Hard link)



하드 링크 (Hard link)



하드 링크 (Hard link)의 특징



- 원본 파일을 이동하거나 삭제해도 하드 링크 파일은 그대로 사용할 수 있음
- 링크 파일을 수정하면 원본 파일도 수정됨
- 링크가 원본파일의 inode를 공유함

링크 생성하기

- 심볼릭 링크 생성

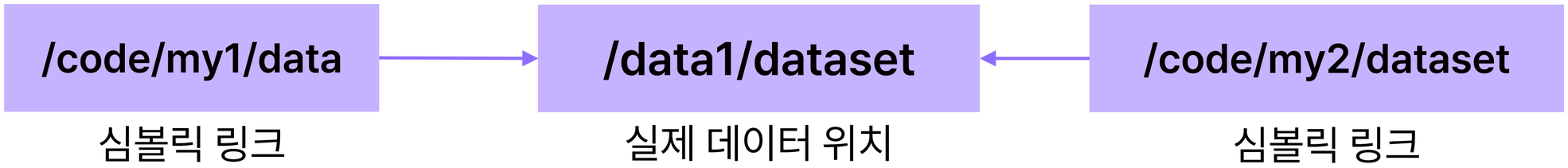
```
ln -s [원본파일] [심볼릭 링크 파일]
```

- 하드 링크 생성

```
ln [원본파일] [하드 링크 파일]
```

링크의 활용방법

- 대량의 데이터를 복사하는 대신 사용



- 대량의 데이터를 같은 폴더에 복사하는 대신 링크를 이용해 비슷한 편의성을 얻을 수 있음
- 여러 위치에 같은 데이터가 저장된 것처럼 사용할 수 있지만, 하드 디스크 용량은 한 번만 차지하고 있음
- 수정하면 원본도 같이 수정되는 점을 유의하면서 사용해야 함

EliceAcademy

04

마운트

마운트(Mount)란?

물리적인 장치를 특정 디렉토리에 연결하는 것

- 새로운 물리 장치를 사용하기 위해서는 반드시 마운트를 해야함
- 일부 운영체제에서는 자동으로 마운트를 실행함

마운트가 필요한 이유

마운트를 해야 운영체제가 저장 장치를 인식하여 접근이 가능

- 저장 장치의 파일 시스템에 접근하려면 먼저 인식시키는 작업이 필요
- 운영체제가 저장 장치를 인식하도록 하는 작업이 마운트

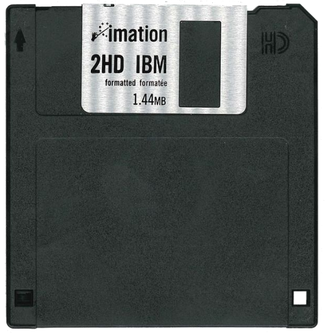
마운트의 대상

대부분의 저장장치

하드 디스크

USB 메모리

CD, DVD



플로피 디스크

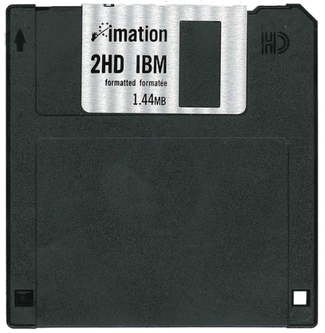
왜 지금까지 몰랐을까

대부분 자동으로 마운트를 해주기 때문

하드 디스크

USB 메모리

CD, DVD



플로피 디스크

윈도우에서 마운트

드라이브에 각 장치를 자동으로 마운트

C:드라이브

하드 디스크

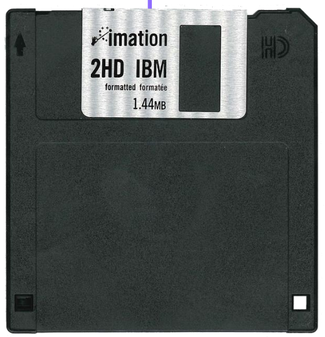
D:드라이브

USB 메모리

E:드라이브

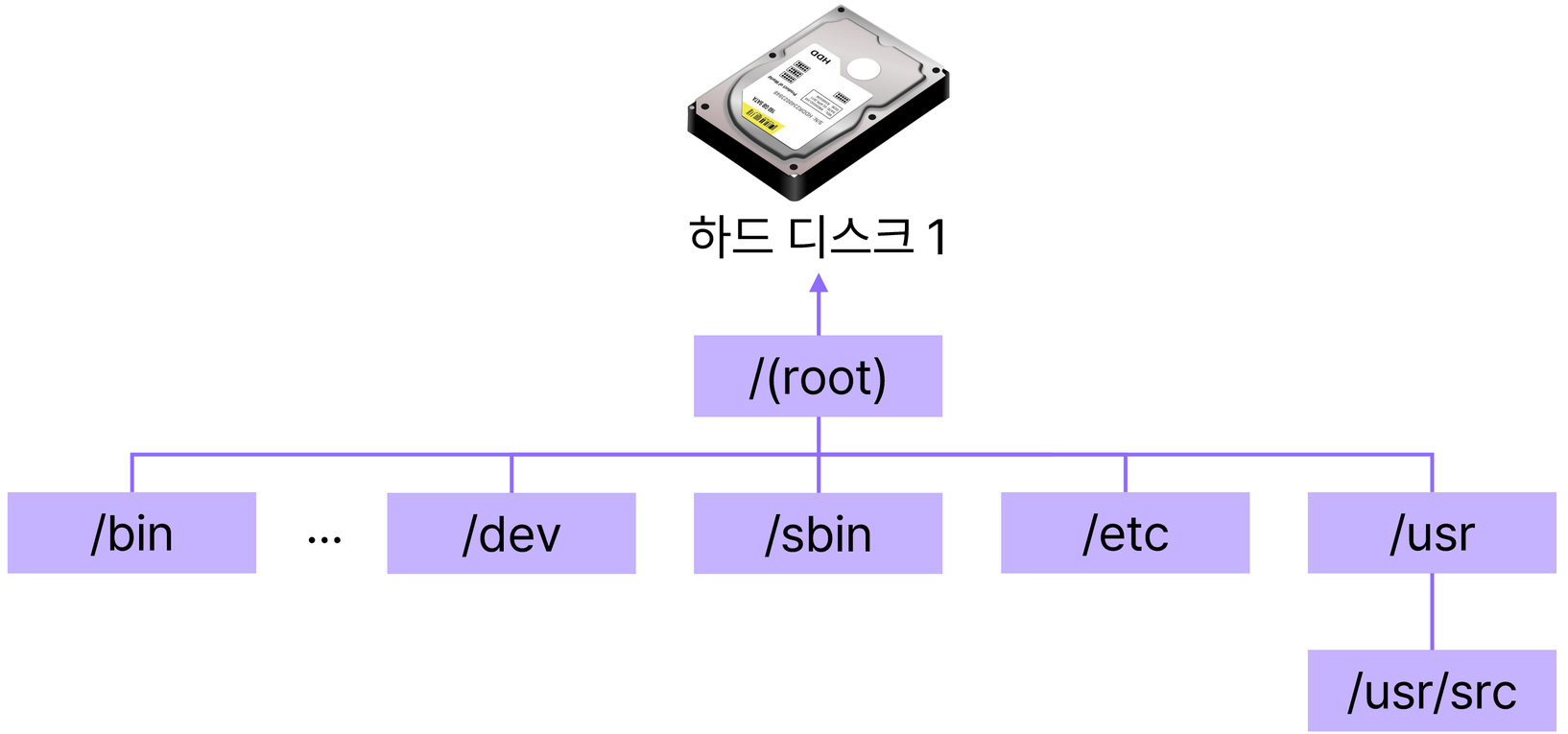
CD, DVD

A:드라이브

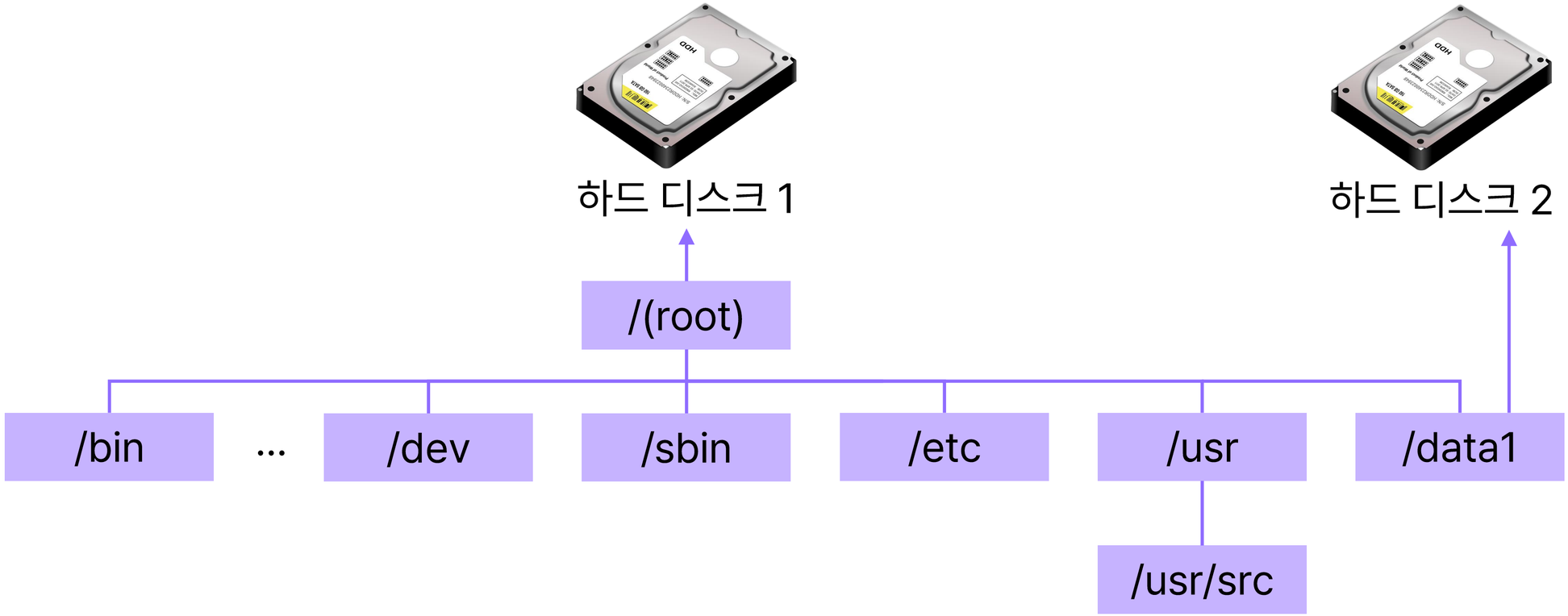


플로피 디스크

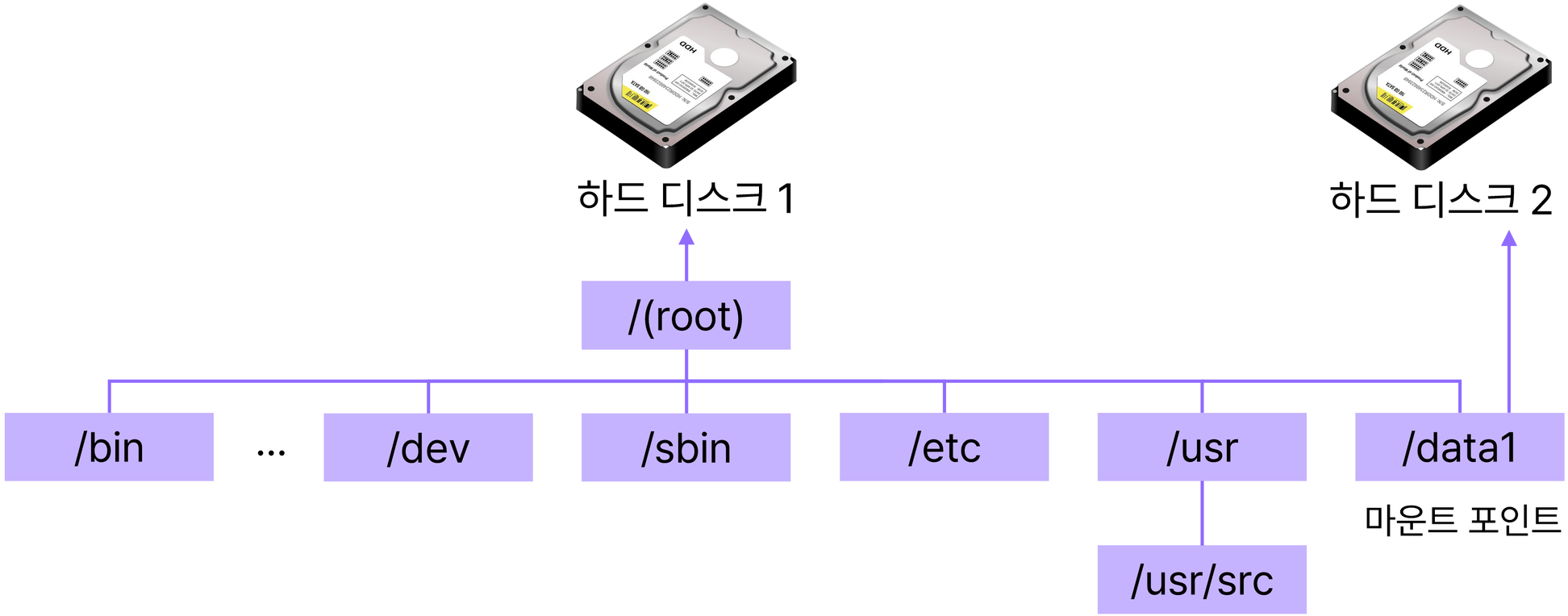
리눅스에서 마운트



리눅스에서 마운트



리눅스에서 마운트



파일 시스템의 목록 확인하기

```
lsblk -p
```

- 파일 시스템의 목록을 확인하는 명령어

EliceAcademy

파일 시스템의 목록 확인하기

```
NAME                  MAJ:MIN RM  SIZE RO TYPE MOUNTPOINT
/dev/sda                8:0    0   20G  0 disk
|-/dev/sda1             8:1    0 15.8G  0 part /mnt/stateful_p
|-/dev/sda2             8:2    0   16M  0 part
|-/dev/sda3             8:3    0    2G  0 part
| `-/dev/mapper/vroot 253:0    0  1.9G  1 dm   /
|-/dev/sda4             8:4    0   16M  0 part
|-/dev/sda5             8:5    0    2G  0 part
|-/dev/sda6             8:6    0  512B  0 part
|-/dev/sda7             8:7    0  512B  0 part
|-/dev/sda8             8:8    0   16M  0 part /usr/share/oem
|-/dev/sda9             8:9    0  512B  0 part
|-/dev/sda10            8:10   0  512B  0 part
|-/dev/sda11            8:11   0    8M  0 part
`-/dev/sda12            8:12   0   32M  0 part
```

- NAME: 장치 이름
- TYPE: 장치 유형
  - Disk: 디스크
  - Part: 파티션
- MOUNTPOINT: 마운트된 위치

마운트 하기

```
sudo mount [옵션] [장치] [디렉토리]
```

- 장치: 장치의 이름
- 디렉토리: 이 장치를 연결할 디렉토리 이름
- 옵션에 -t 를 추가하면 파일 시스템의 유형을 입력(예: ext4)