Computer Science Study

Operating System

Scheduling

Subject 1. Basic

Subject 2. Algorithm

File System

Subject 3. Concept

Subject 4. Directory

Subject 5. Design

Subject 6. Hierarchy

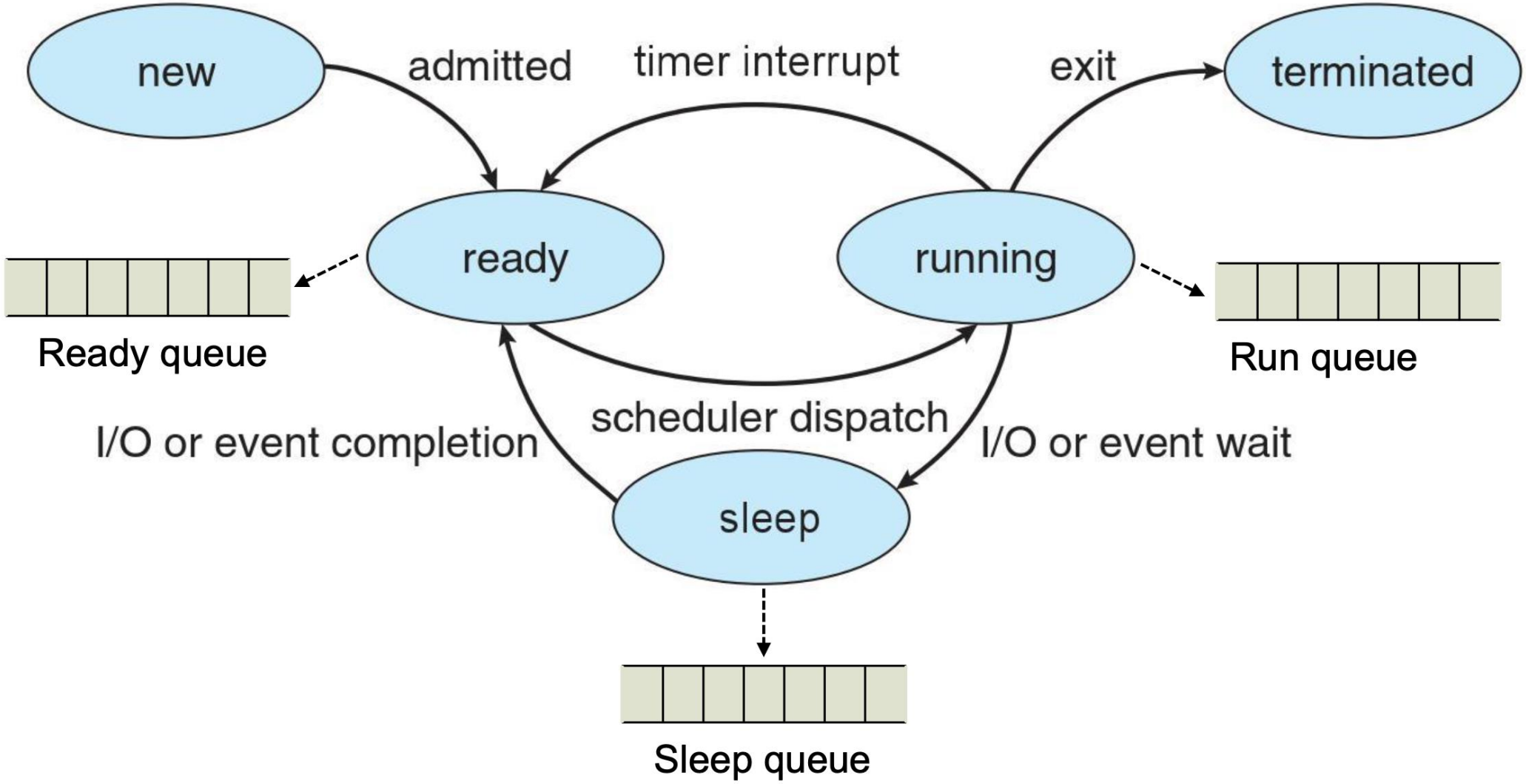
홍승택

2024. 11. 11. 18시

subject 1.

Scheduling Basic

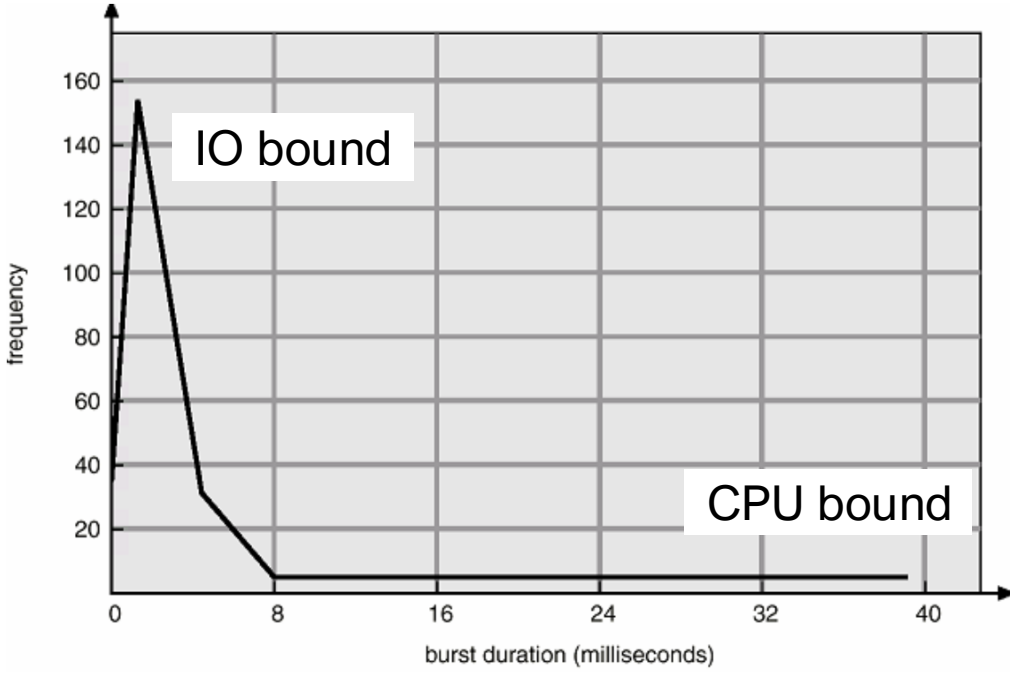
Process State



Process Cycle

- CPU Burst Time: CPU 할당 후 입출력 요구시까지의 시간
- **IO-bound vs. CPU-bound**

- 보통 우리가 사용하는 컴퓨터에서는 I/O-bound 작업이 많다.
  - 그나마 흔한 CPU-bound 작업은 유튜브 시청 정도



Scheduling Criteria

CPU utilization	전체 시스템 시간 중 CPU가 작업을 처리하는 시간의 비율(not idle)
Throughput	CPU가 단위 시간당 처리하는 프로세스의 개수
Response time	프로세스가 입출력을 시작해서 첫 결과가 나오는데 까지 걸리는 시간
Waiting time	프로세스가 Ready Queue 내에서 대기하는 시간의 총합
Turnaround time	프로세스가 시작해서 끝날 때까지 걸리는 시간

시스템의 용도에 따른 요구사항이 달라짐

- CPU utilization 추구 : 슈퍼 컴퓨터
- Response time 추구 : 개인용 컴퓨터

스케줄링의 종류

비선점형 스케줄링 (Non-preemptive Scheduling)

- 프로세스가 CPU를 자진 반납할 때까지 실행 보장

선점형 스케줄링 (Preemptive Scheduling)

- 타임퀀텀 소진, 인터럽트 등의 이유로 강제로 CPU 회수할 수 있음

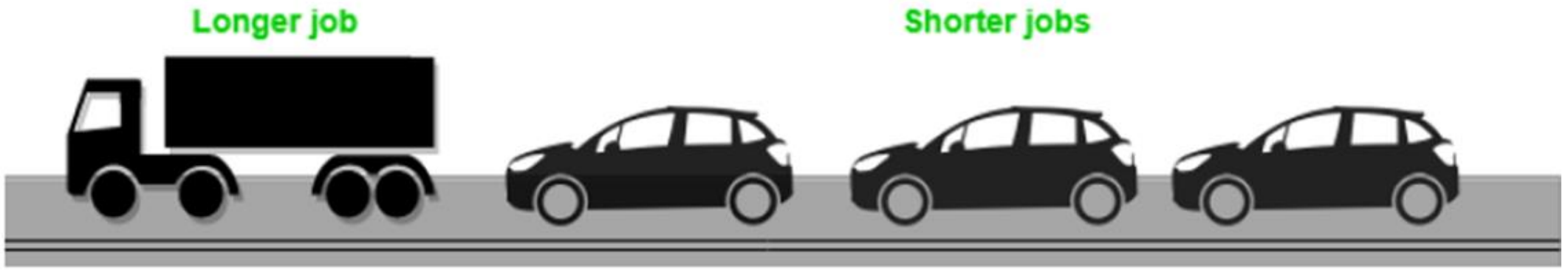
subject 2.

Scheduling Algorithms

First-Come, First-Serve (FCFS)

프로세스 도착 순으로 CPU 배정

Convoy Effect



Shortest Job First (SJF)

- 다음 CPU burst 시간이 가장 짧은 프로세스에 CPU를 먼저 할당한다.
- **최소의 평균 대기 시간을 제공**

- 이전 버스트 타임들을 통해 다음 버스트 시간을 예측

Exponential Average
t_n이 실제 n번째 CPU 버스트 타임이고, τ_{n+1}이 예상 값일 때

$$\tau_{n+1} = \alpha t_n + (1-\alpha)\tau_n$$
$$= \alpha t_n + (1-\alpha)\alpha t_{n-1} + \cdots + (1-\alpha)^j \alpha t_{n-j} + \cdots + (1-\alpha)^{n+1} t_0$$

Priority Scheduling

- 프로세스 Priority에 따라 CPU를 할당한다
- 일반적으로 CPU-bound 프로세스보다 IO-bound 프로세스에게 높은 우선순위를 부여

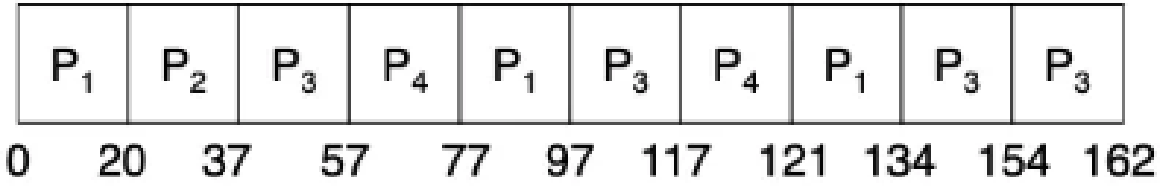
- 문제점: **Starvation**
  - 낮은 priority를 가진 프로세스는 전혀 수행되지 않을 수 있다.
- 해결 방법: **Aging**
  - 기다리는 시간에 따라 프로세스 우선순위를 증가

Round Robin (RR)

- CPU를 시간 단위(time quantum)로 할당한다.
- 선점형 스케줄링 방식
- Time quantum을 수행을 한 프로세스는 Ready 큐의 끝으로 들어가 다시 할당을 기다림.
- 성능
  - q 가 클 경우 : FCFS
  - q 가 작을 경우 : 문맥전환(Context Switching)에 필요한 시간보다 낮다면 효율이 매우 떨어짐.

Process	Burst Time
P_1	53
P_2	17
P_3	68
P_4	24

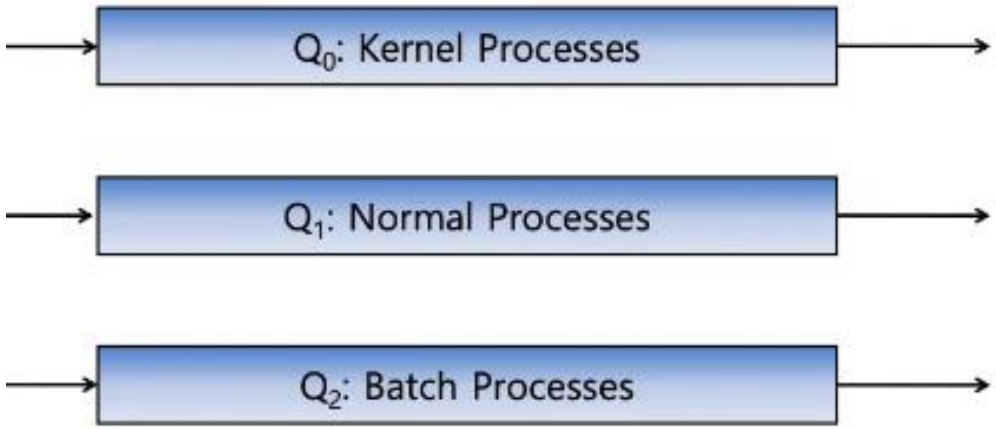
Time quantum = 20



Multilevel Queue

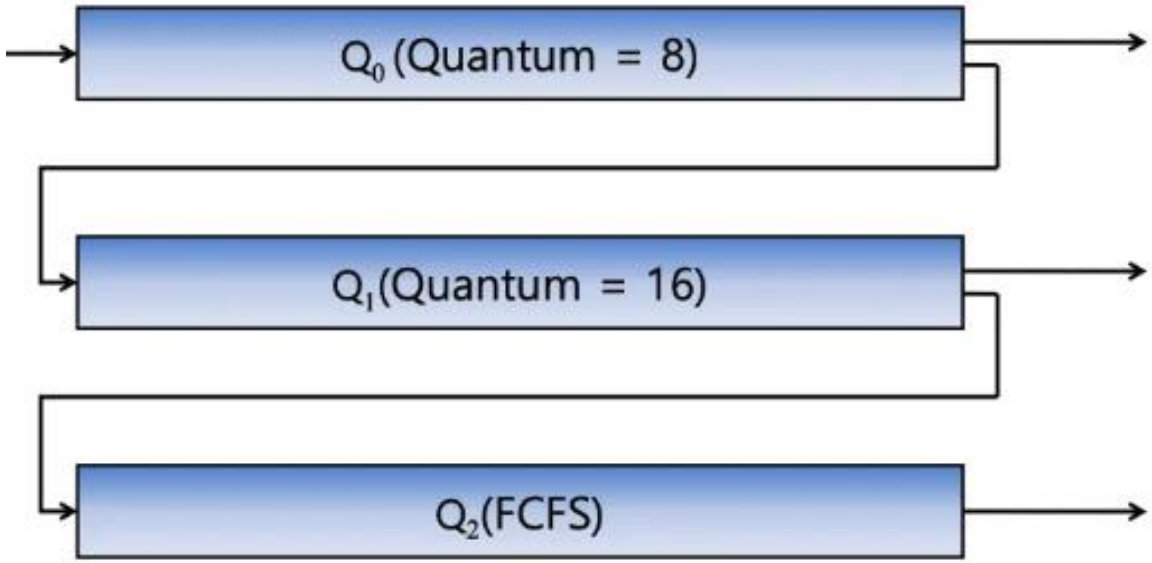
- Ready 큐를 여러 개 만들어 각각에 대해 다른 우선순위와 스케줄링 알고리즘을 사용하는 기법
- 예시
  - Foreground 큐
    - Interactive한 동작이 필요한 프로세스를 위한 큐. (Round Robin)
  - Background 큐
    - CPU 연산 작업을 주로 수행하는 프로세스를 위한 큐. (FCFS)

- 문제점
  - 상위 큐에 프로세스가 계속 있으면 하위 큐에 기아 현상 발생



Multilevel Feedback Queue

- Multilevel Queue + 동적인 프로세스 우선순위 변화
- Design choices
  - 큐의 개수
  - 각 큐마다의 스케줄링 기법
  - 언제 프로세스를 한 단계 높은 큐로 옮길 것인가
  - 언제 프로세스를 한 단계 낮은 큐로 옮길 것인가

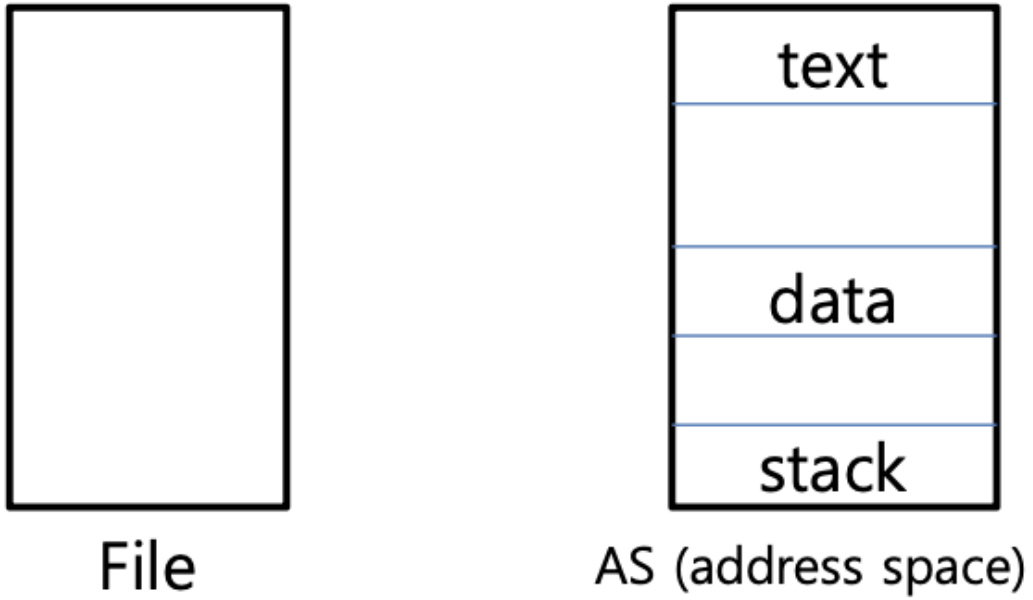


subject 3.

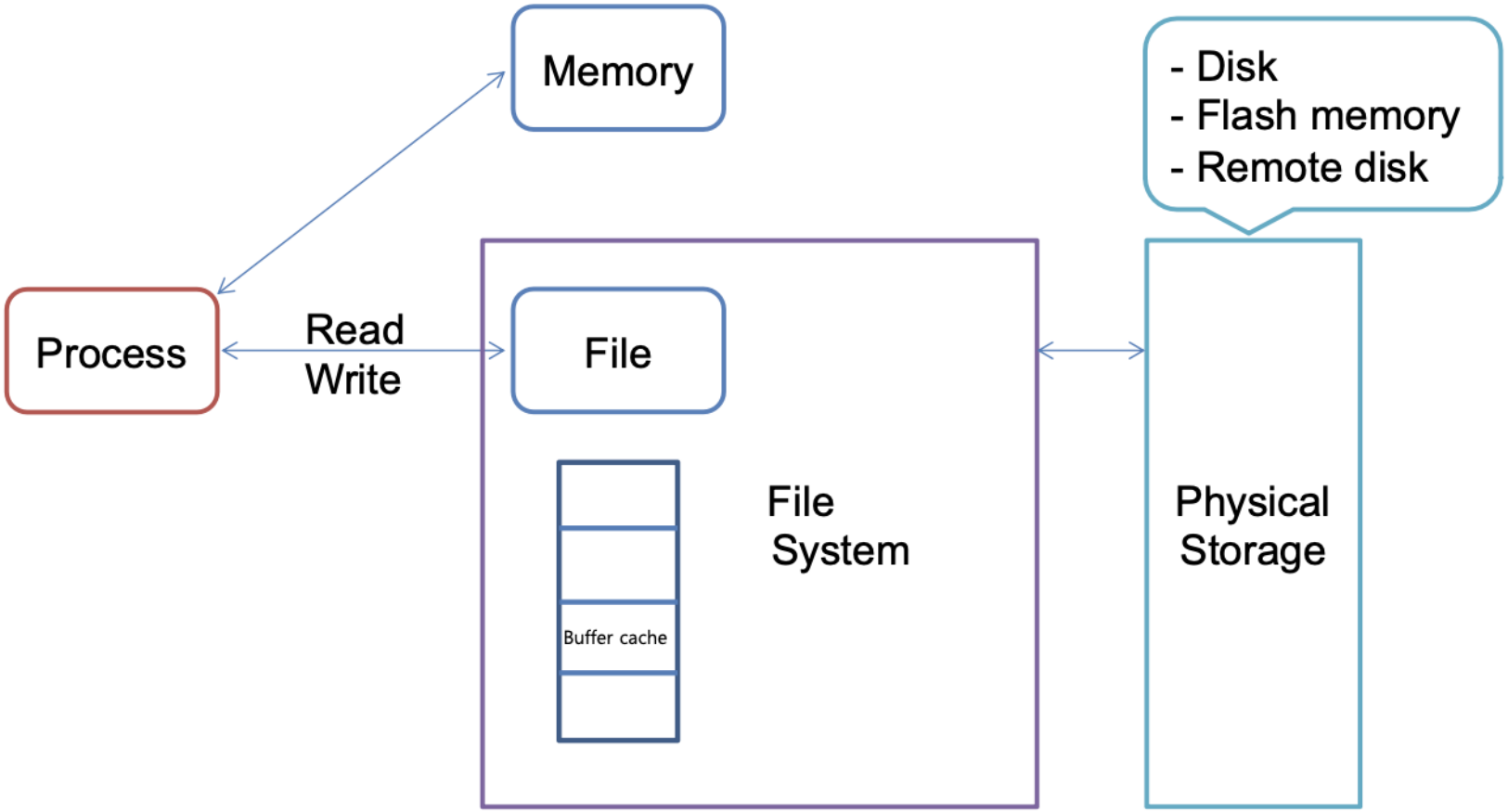
File system concept

File as abstraction

- 정의: Linear array of bytes
- Linear: 데이터 배치가 i, i+1, i+2처럼 연속적
- Array: 중간이 비어 있지 않음 (not sparse)
  - 비교: 주소공간은 sparse
- Byte addressable: 접근 단위



Process and File, File system



Metadata

Field	Description
Name	사람이 읽을 수 있는 기호화된 이름
Type	운영체제에서 지원하기 위한 정보 (executable, text, library, archive, mp4, etc.)
Location	저장장치에 기록되어 있는 블록 위치
Size	파일의 크기
Protection	파일에 대한 read, write, execution 허가 정보

그 외 metadata들

- Time, date, user info.
- **inode**(index node): 파일 위치
- directory

포렌식이 가능한 이유 : Metadata만 삭제

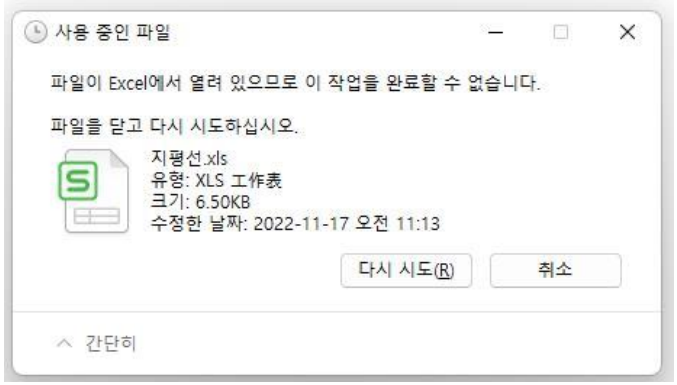
Metadata and Caching

- Metadata가 저장되는 곳 : Disk
- Caching하는 이유
  - Metadata를 접근하고, 거기서 data 위치를 읽어서 접근한다 (디스크 접근 2번 필요)
  - **Metadata는 엄청 자주 접근한다.** (여러 번 디스크를 접근해야 함)

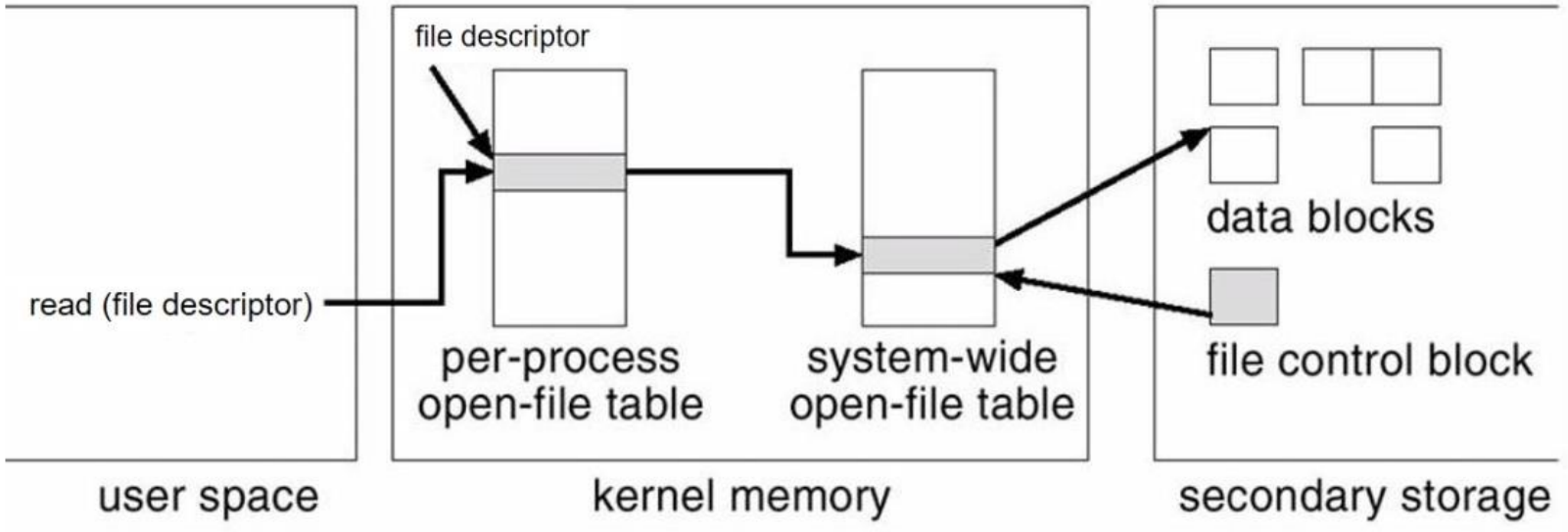
- Asynchronous update : Inconsistency problem
  - 캐시되어 있는 메모리에 수정이 일어나고,
    아직 디스크에는 저장이 안된 상태에서 전원이 나간다면?
  - **Journaling file system**
  - Journal에 메타 데이터 변경점을 임시 저장

Open-file table

- **Per-process table**: 각 프로세스마다 유지하는 state
  - file pointer
  - file descriptor(FD)은 Per process table의 인덱스이다.
- **System-wide table**: process independent information (공통 테이블)
  - access date, file location, open count



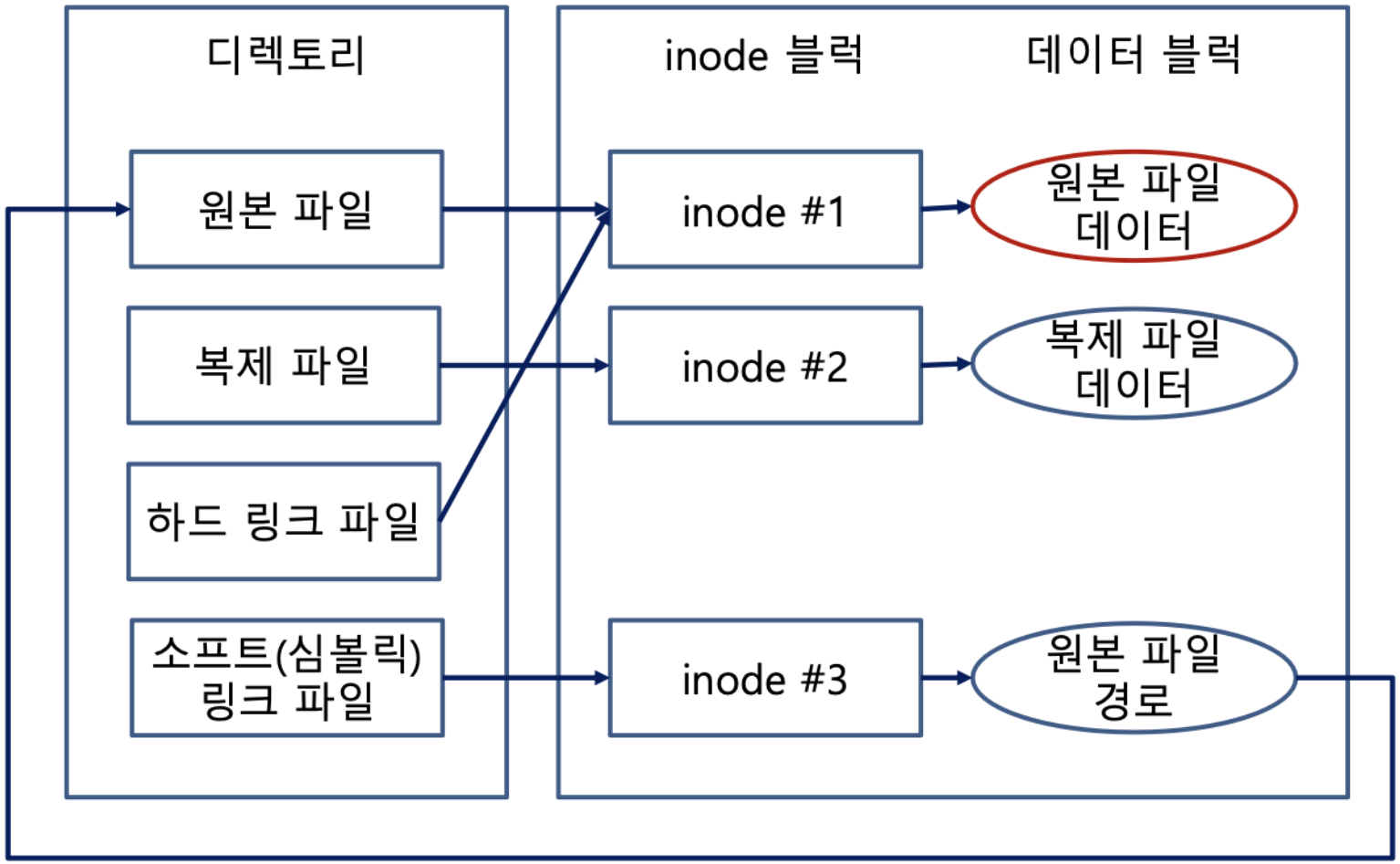
- Open: open count++, Close: open count--



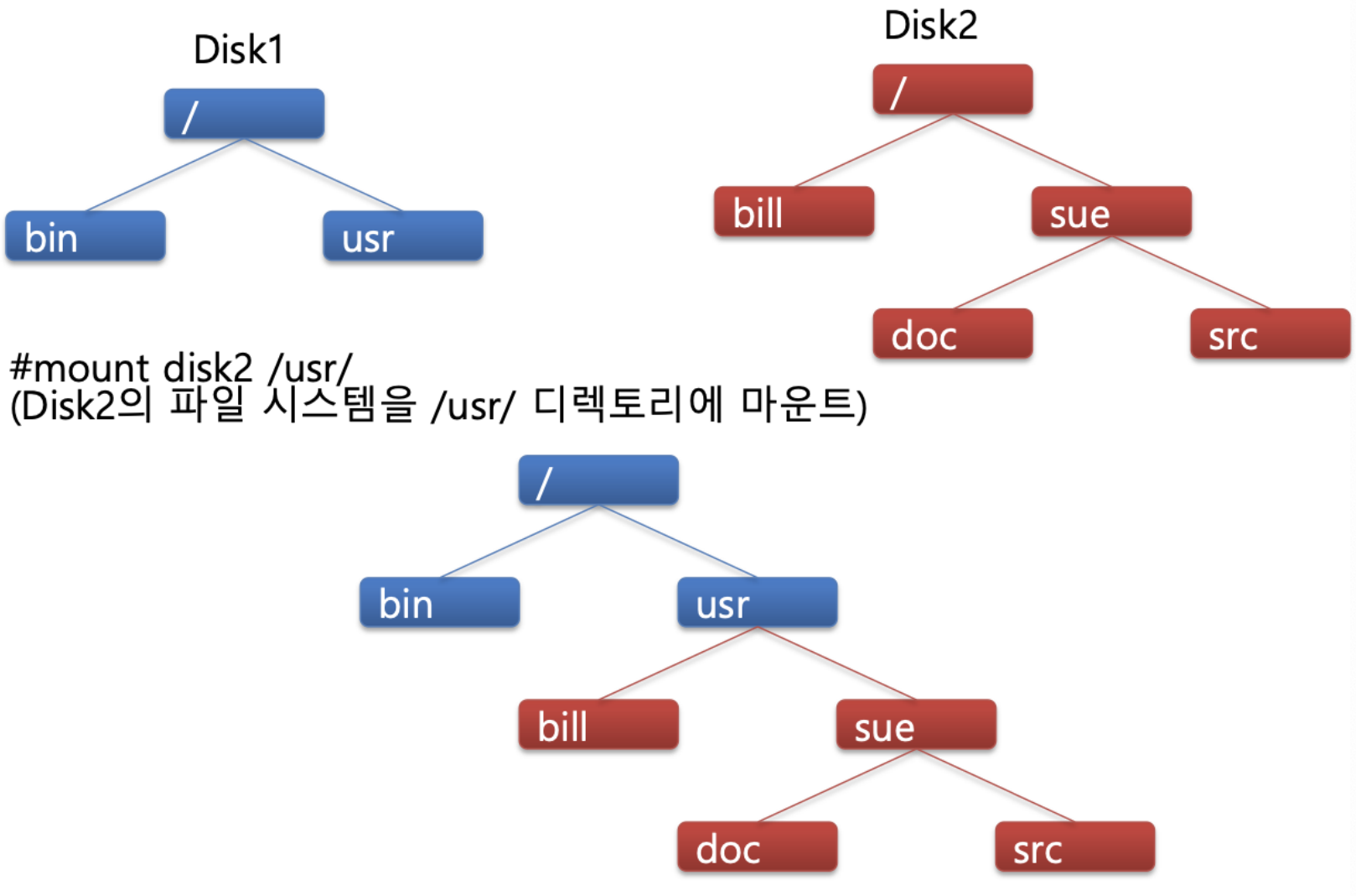
subject 4.

Directory

Hard link, Soft link



Mount


Disk1
/
bin
usr
Disk2
/
bill
sue
doc
src
#mount disk2 /usr/
(Disk2의 파일 시스템을 /usr/ 디렉토리에 마운트)
/
bin
usr
bill
sue
doc
src

subject 5.

File System Design

File implement

File의 내용을 담고 있는 data block이 저장되는 디스크상 위치 정보를 결정하는 것과 이에 대한 관리 방법

- Contiguous Allocation
- Linked List Allocation
- Linked List Allocation Using Index
- inode

Contiguous Allocation

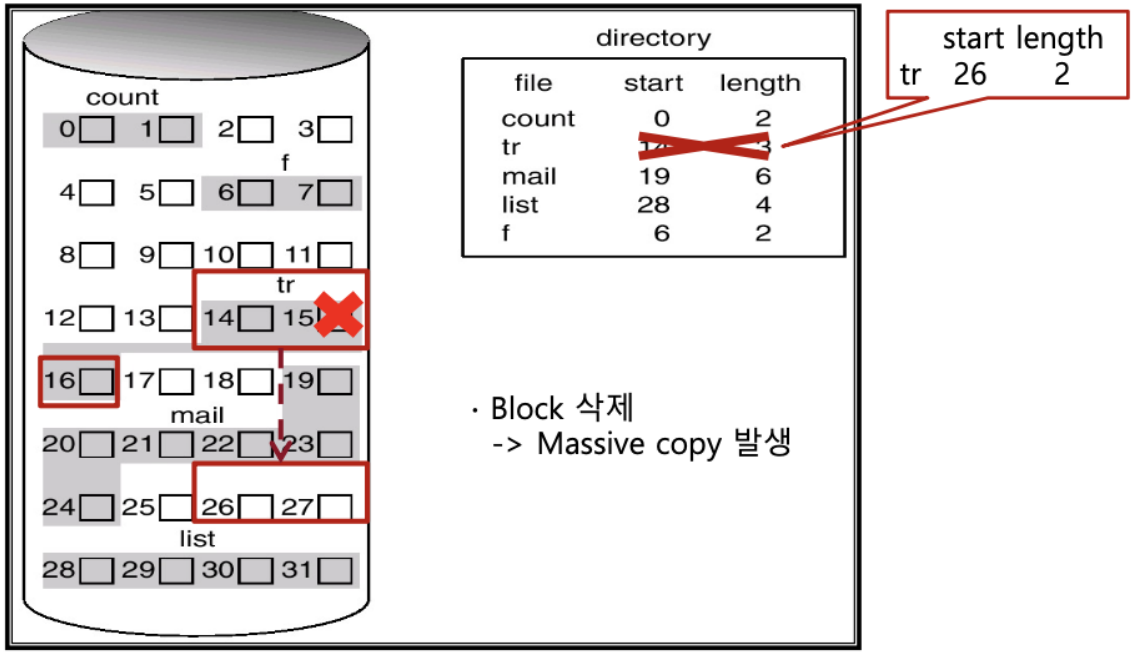
파일을 물리적으로 연속된 disk block에 저장

장점

- 구현이 간단함
- 물리적으로 연속된 공간에 있으므로 전체 파일을 한번에 읽어 들일 경우 성능이 매우 뛰어남

단점

- 파일은 반드시 한번에 끝까지 기록 되어야 한다.
- 확장을 위하여 파일의 끝에 예비용 block을 남겨둘 경우, disk의 공간을 낭비하게 됨



Linked List Allocation

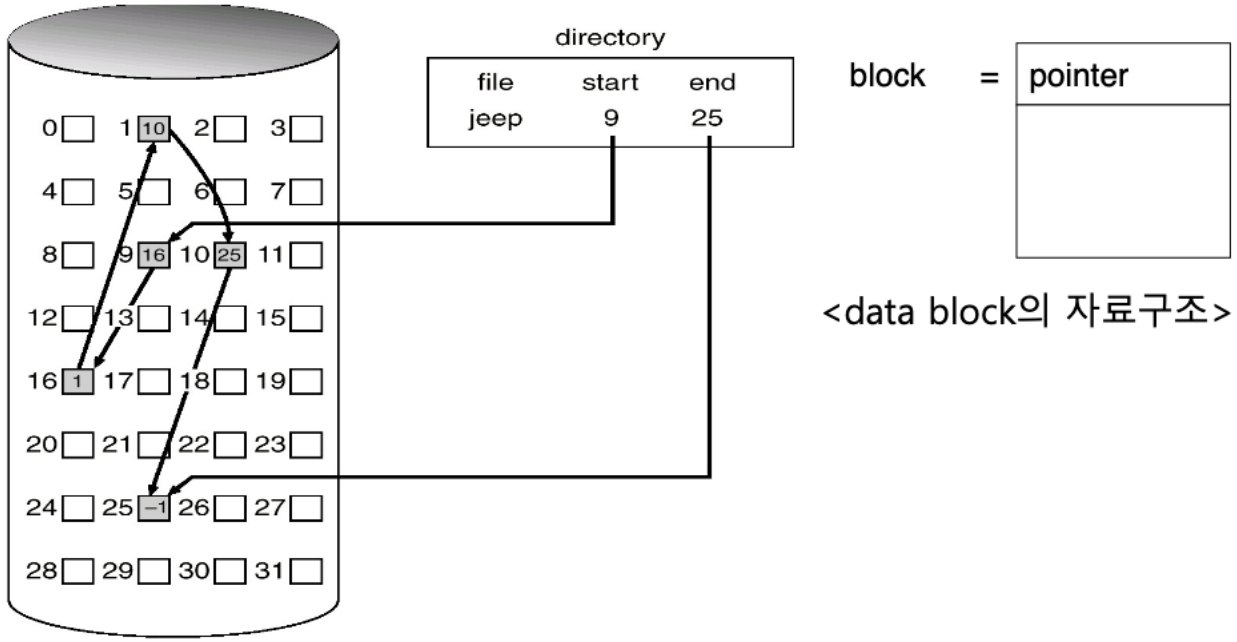
Disk의 block을 linked list로 구현하여 file의 data를 저장하도록 함

장점

- File의 data block은 disk의 어디든지 위치할 수 있다.
- Contiguous allocation의 방법과 달리 공간의 낭비가 없음

단점

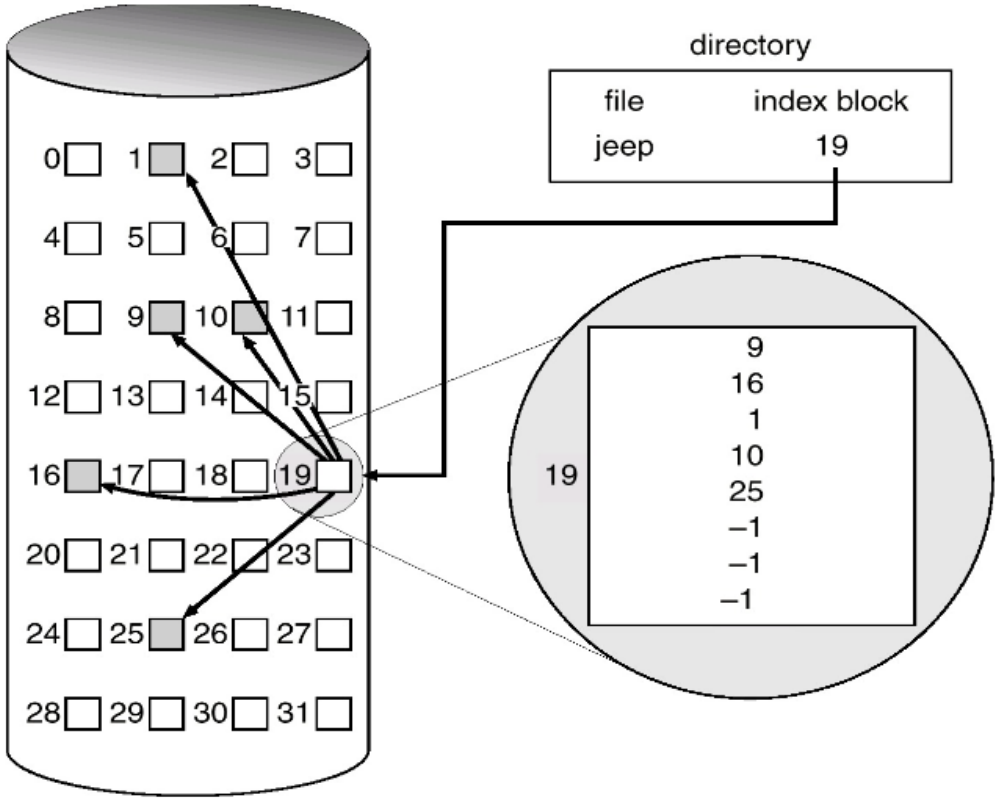
- Random access가 불가능
- 다음 data block에 대한 pointer 저장 필요



<data block의 자료구조>

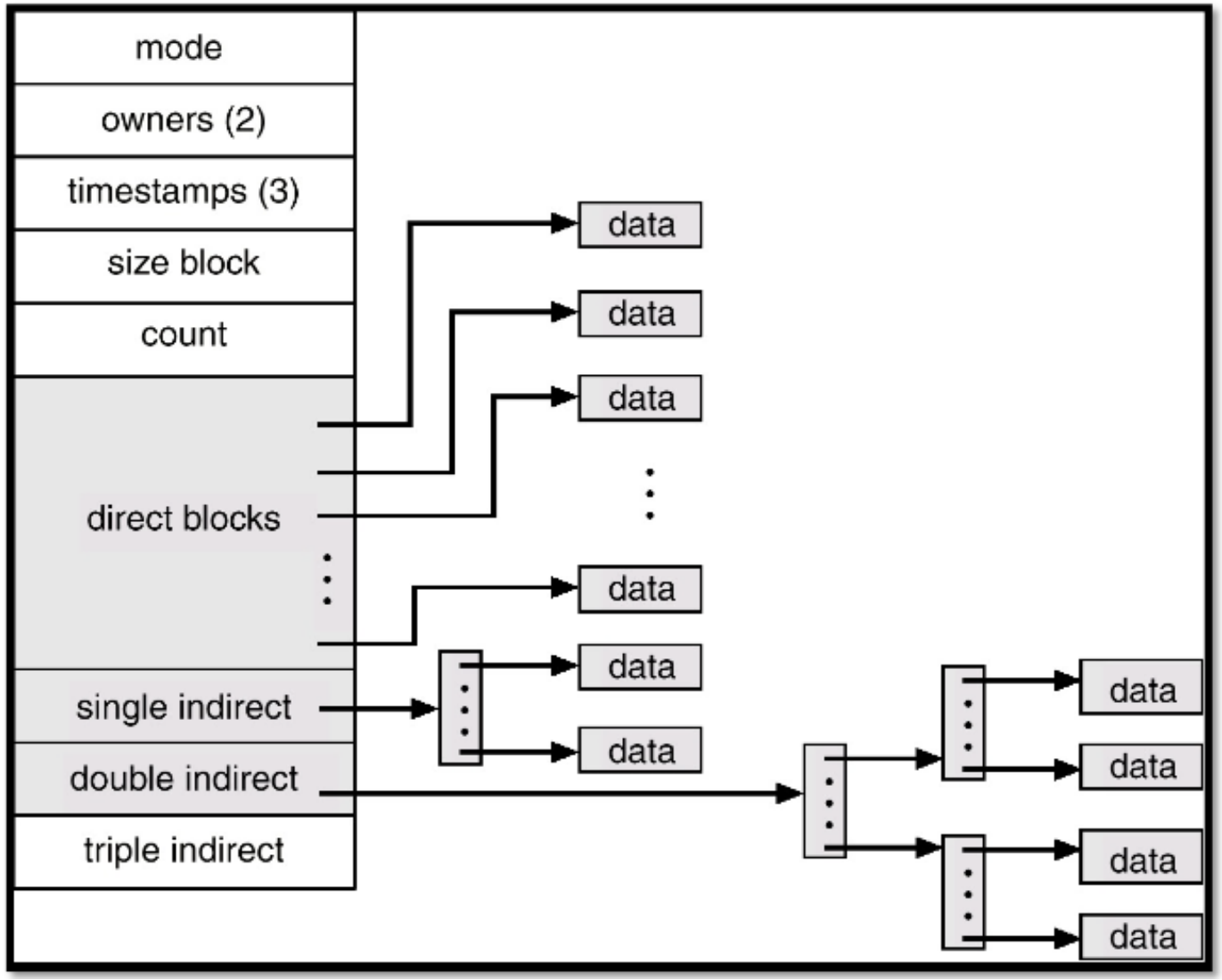
Linked List Allocation Using Index

- File의 data block의 위치를 별도의 block에 모아둠
- File의 data block 중 하나를 index block이라 하여,
  모든 data block의 위치를 index block에서 알 수 있음



inode

File에 대한 data block index를 계층 형태로 관리하는 방법



inode 용량

- Block size: 4KB, Number of direct blocks: 12, Pointer size: 4B

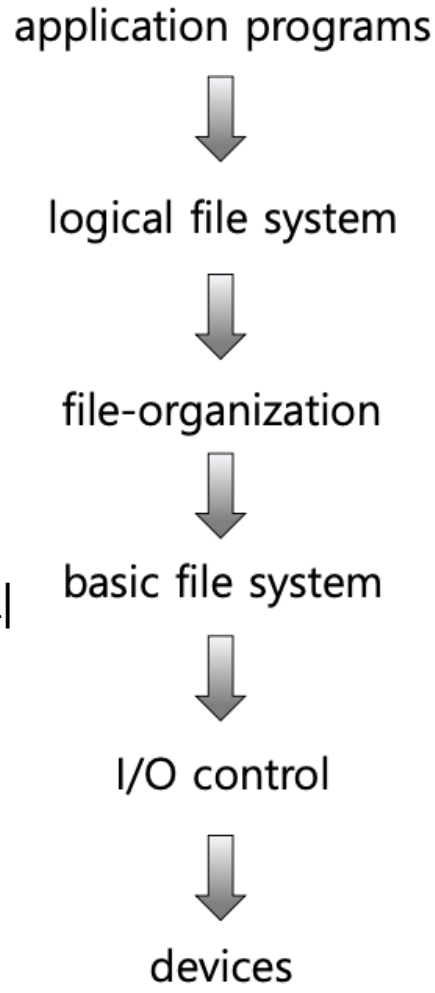
- Direct Blocks
  - 4KB x 12 = 48KB
- Single Indirect
  - 4KB x 1024 = 4MB
- Double Indirect
  - 4KB x 1024 x 1024 = 4GB
- Triple Indirect
  - 4KB x 1024 x 1024 x 1024 = 4TB

subject 6.

File System Hierarchy

File System Hierarchy

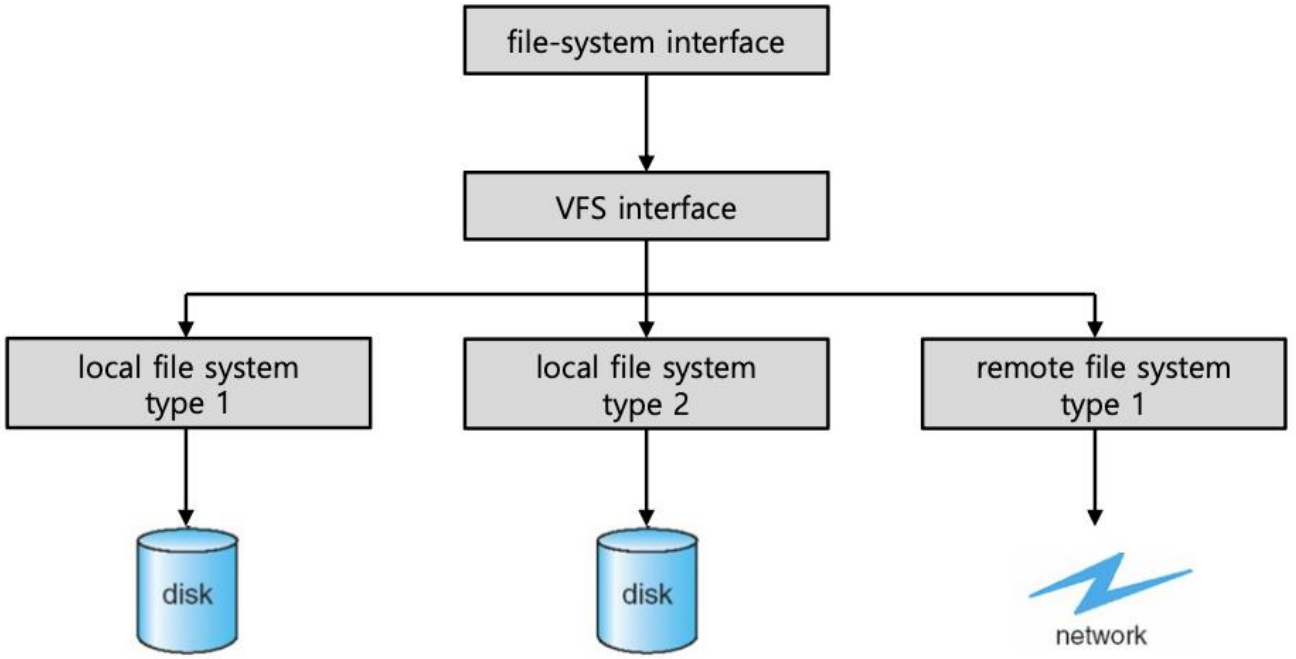
- 파일 시스템은 일반적으로 여러 개의 계층으로 나뉘어져 구성
  - 계층 구조를 사용하여 파일 시스템의 코드의 중복을 최소화 됨
  - 다수의 파일 시스템에 의해 I/O control 계층이 재사용 됨



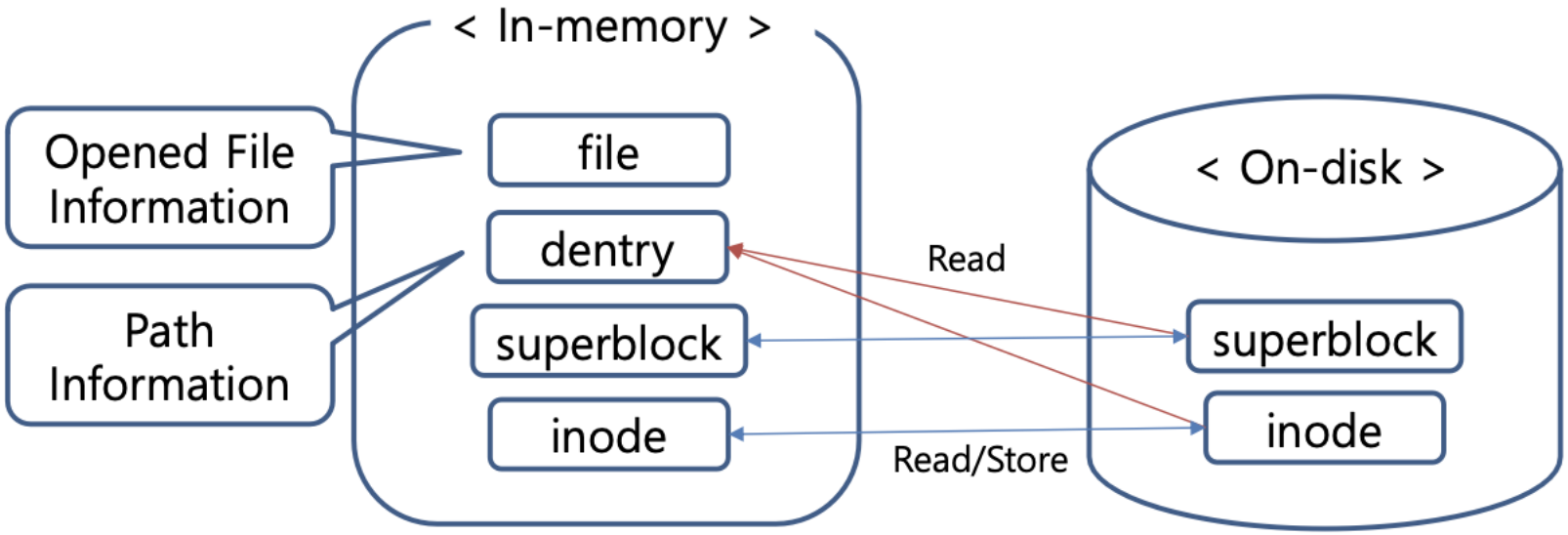
- Logical file system
  - 파일 시스템 메타 데이터 관리
- file-organization module
  - 파일의 논리 블록 주소를 물리 블록 주소로 변환
  - 각 파일의 논리 블록은 0부터 N까지 번호가 주어지며, 데이터를 실제 저장하는 물리 블록은 저장장치 주소임
- Basic file system
  - 장치 드라이버에게 저장장치의 물리 블록을 읽고 쓰도록 명령을 내림
- I/O control
  - 장치 드라이버가 저장장치 하드웨어에 맞게 명령어 전달

Virtual File System

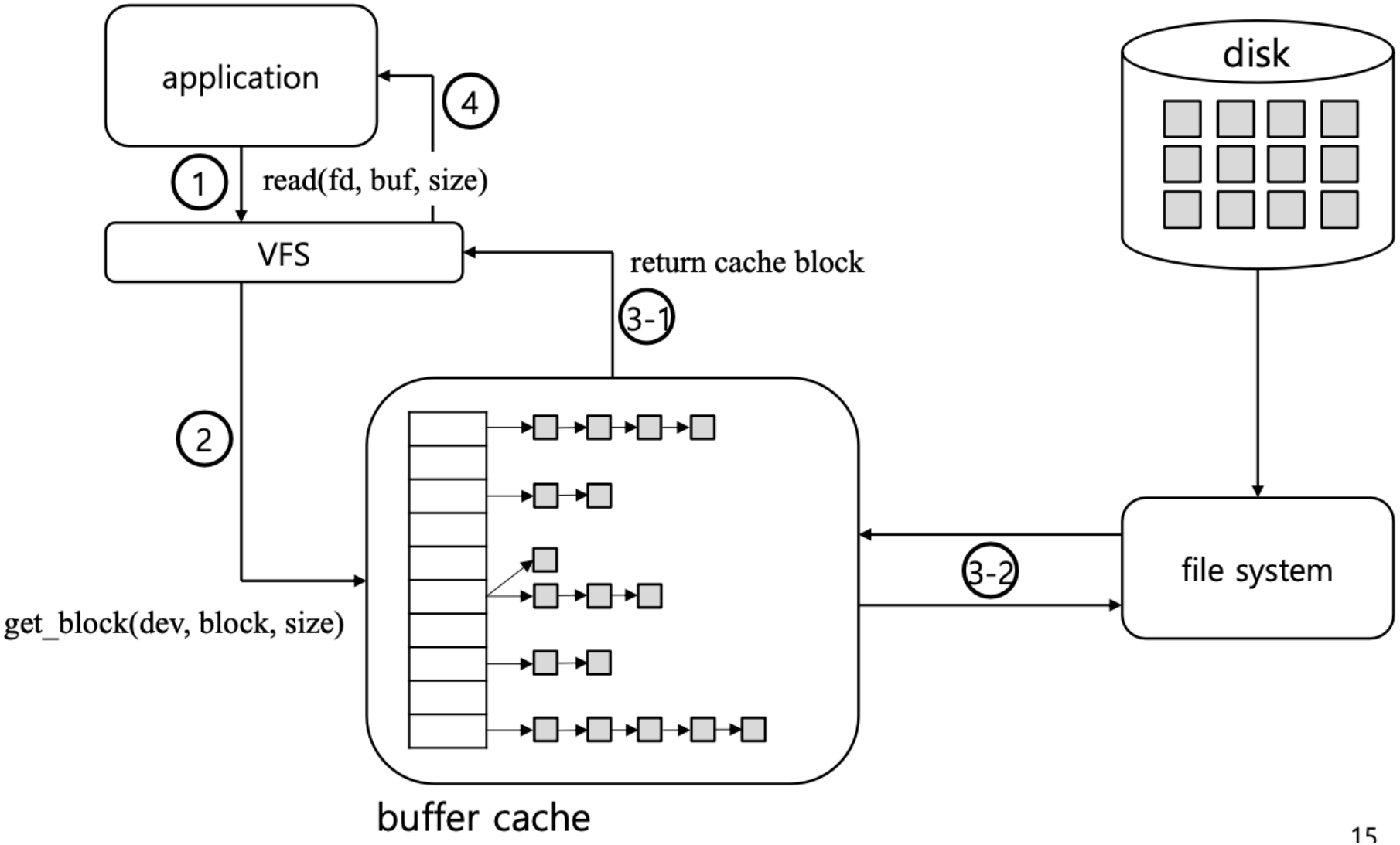
- 다양한 logical file system들을 abstract 함
  - Disk용 FS과 flash FS, CD-ROM용 FS들을 **하나의 VFS로 abstract함**
- VFS는 여러 개의 file system들을 사용할 수 있도록 같은 system call interface (API)를 제공
  - API를 통해서 VFS는 여러 file system을 uniform하게 사용
  - 장점: VFS를 통해서 시스템에 여러 개의 다른 file system이 사용되더라도 마치 1개의 파일 시스템만 있는 것처럼 프로그램 가능



File System Structure

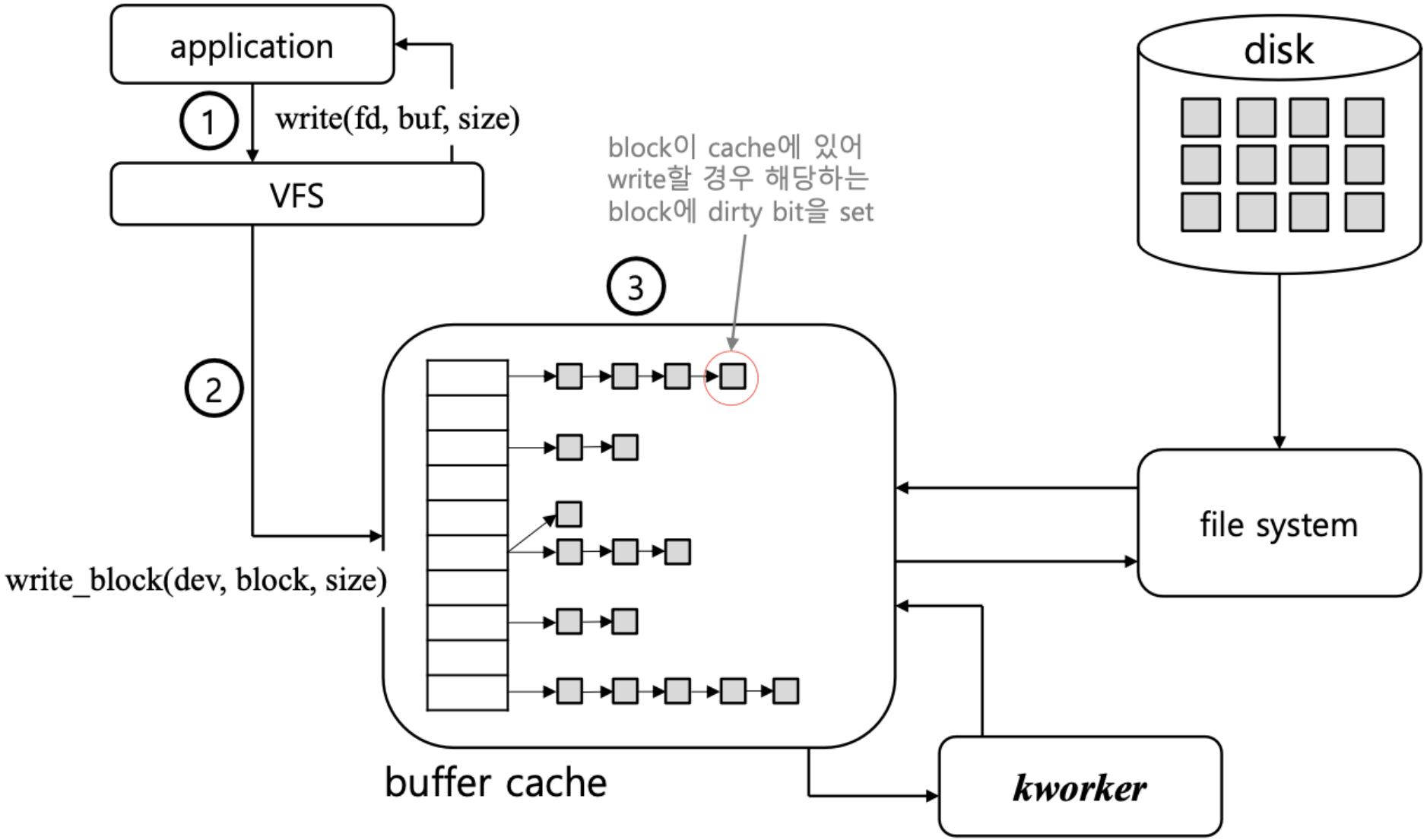


Read System Call



15

Write System Call



Reference

System programming(korea university, 유혁)

Operating System Concepts (10/E, Silberschatz)

Computer Systems A Programmer's Perspective (3/E, Randal E. Bryan)

Computer Organization And Design (6/E, David A. Patterson)