

# 10. File and File System

## • File

- "A named collection of related information"
- 일반적으로 비휘발성의 보조 기억 장치에 저장
- 운영체제는 다양한 저장 장치를 file 이라는 동일한 논리적 단위로 볼 수 있게 해줌
- Operation
  - Create, read, write, reposition(seek), delete, open, close 등

파일을 접근하는 위치를  
수정해 줄 때  
파일의 meta 정보를  
매번 바꿔 쓰는 일

## • File attribute (혹은 파일의 meta data)

- 파일 자체의 내용이 아니라 파일을 관리하기 위한 각종 정보들
- 파일 이름, 유형, 저장된 위치, 파일 사이즈
- 접근 순서(읽기, 쓰기, 실행), 시간(생성, 변경, 사용), 소유자 등

## • File System

Software

- 운영체제에서 파일을 관리하는 부분
- 파일 및 파일의 메타데이터, 디렉토리 정보 등을 관리
- 파일의 저장방법 결정
- 파일 보호 등

# Directory and Logical Disk

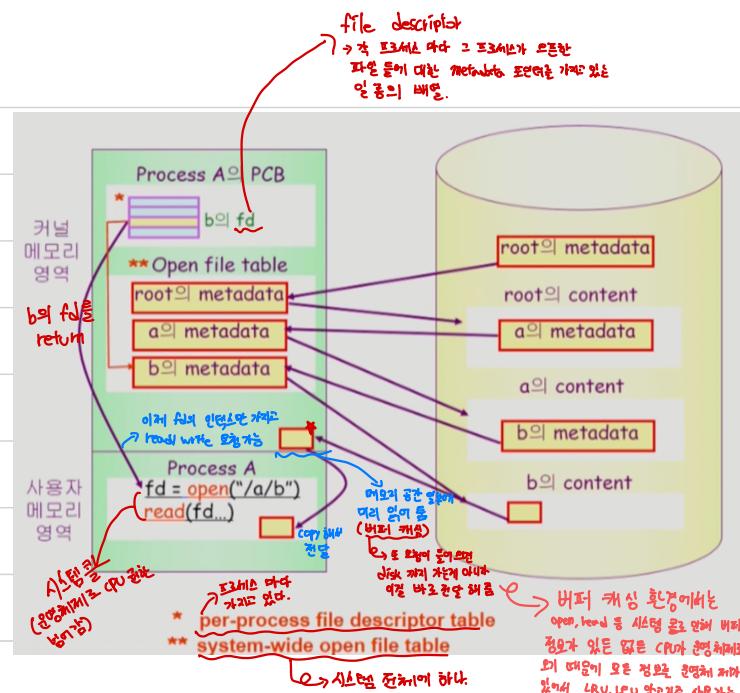
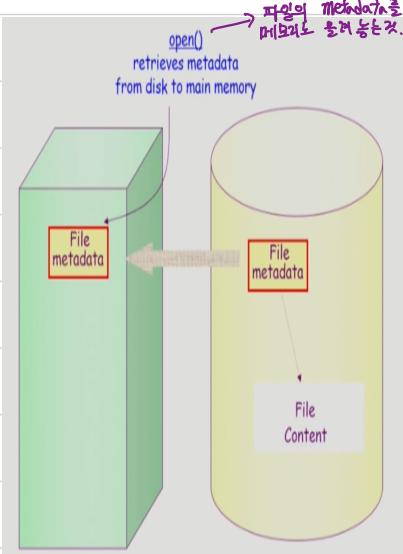
## • Directory

- 파일의 메타데이터 중 일부를 포함하고 있는 일종의 특별한 파일
- 그 디렉토리에 속한 파일 이름 및 파일 attribute 들
- Operation
  - Search for a file, create a file, delete a file
  - list a directory, rename a file, traverse the file system  
전체 탐색

## • Partition (= Logical Disk)

- 하나의 (물리적) 디스크 안에 여러 파티션을 두는게 일반적
- 여러개의 물리적인 디스크를 하나의 파티션으로 구성하기도 함
- (물리적) 디스크를 파티션으로 구성한 뒤 각각의 파티션에  
file system을 할거나 swapping 등 다른 용도로 사용할 수 있음

# open( )



## • open("a/b/c")

v 디스크로부터 파일 C의 메타데이터를 메모리로 가지고 옴

v 이를 위하여 directory path를 search

• 루트 디렉토리 "/"를 open하고 그 안에서 파일 "a"의 위치 획득

• 파일 "a"를 open한 후 read 하여 그 안에서 파일 "b"의 위치 획득

• 파일 "b"를 open하는 동안 read 하여 그 안에서 파일 "c"의 위치 획득

• 파일 "c"를 open 한다.

v Directory path의 search가 너무 많은 시간 소요

• open을 read/write와 별도로 두는 이유임.

• 한번 open한 파일은 read/write 시 directory search 불필요

## v Open file table

• 현재 open된 파일들의 메타데이터 보관소 (in memory)

• 디스크의 메타 데이터 보다 빠르게 정보 추가

• open 한 프로세스의 수

- file offset: 파일 어느 위치 접근 풍선지 표시 (별도 테이블 필요)

## v File descriptor (file handle, file control block)

• Open file table에 대한 위치 정보 (프로세스 별)

file descriptor

→ 각 프로세스마다 그 프로세스가 오픈한 파일에 대한 file descriptor를 가지고 있는 일종의 배열.

버퍼 캐싱 환경에서는  
open, read 등 시스템 호출 단위 버퍼 캐싱  
정우과 같은 경우 CPU가 경쟁해도 낭비  
와 메모리 오른 정우와 함께 쪼개 알고  
있어서 LRU, LFU 알고리를 사용 가능.  
 Paging 단위로 메모리 대로 쪼개...!

# File Protection

- 각 파일에 대해 누군가 어떤 유형의 접근(read/write/execution)을 허락할 것인가?

## Access control 방법

### Access control Matrix

	file 1	file 2	file 3
User 1	rw	rw	r
User 2	rw	r	r
User 3	r		

ACL (Access Control List)  
linked list

Capability  
linked list

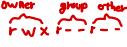
행렬 관 관리 → linked list로 관리

- Access control list: 파일 별로 누군가에게 어떤 접근 권한이 있는지 표시

- Capability: 사용자 별로 자신이 접근 권한을 가진 파일 및 해당 권한 표시  
→ 어떤 방법보다  
하나만 사용하면 됨

but, 이렇게 해도 overhead 큼.  
그래서 일반 운영 체제에서 Grouping 사용

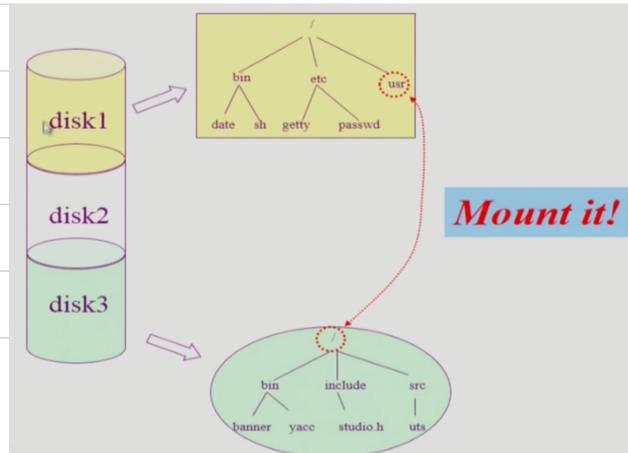
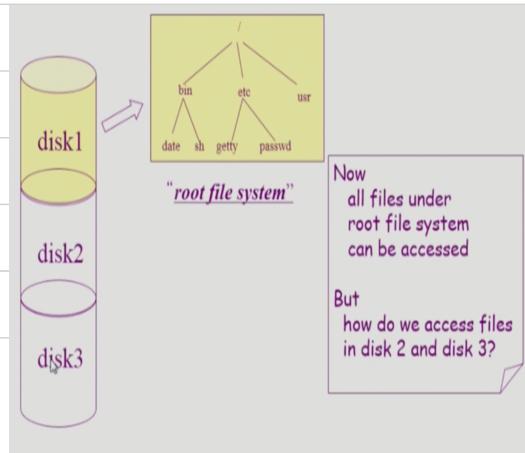
### Grouping

- 전체 USER를 owner, group, public의 세 그룹으로 구분
- 각 파일에 대해서 세 그룹의 접근 권한 (rwx)을 비트셋으로 표시
- (예) UNIX   
총 9비트

### Password

- 각 파일마다 password를 두는 방법 (디렉토리 파일에 두는 방법도 가능)
- 모든 접근 권한이 대해 하나의 password: all-or-nothing
- 접근 권한 별 password: 암기 문제, 관리 문제

# File System의 Mounting



# Access Methods

- 시스템이 제공하는 파일 정보의 접근 방식

- ▼ 순차 접근 (sequential access)

- 커서트 레코드를 사용하는 방식처럼 접근
    - 일제히 쓰면 애너테는 자동적으로 증가

- ▼ 직접 접근 (direct access, random access)

- LP 레코드 번호 같이 접근 하도록 함
    - 파일을 구성하는 레코드를 임의의 순서로 접근 할 수 있음.