Ch10, 11, 12. File system and Storage (Selected Part) (완)

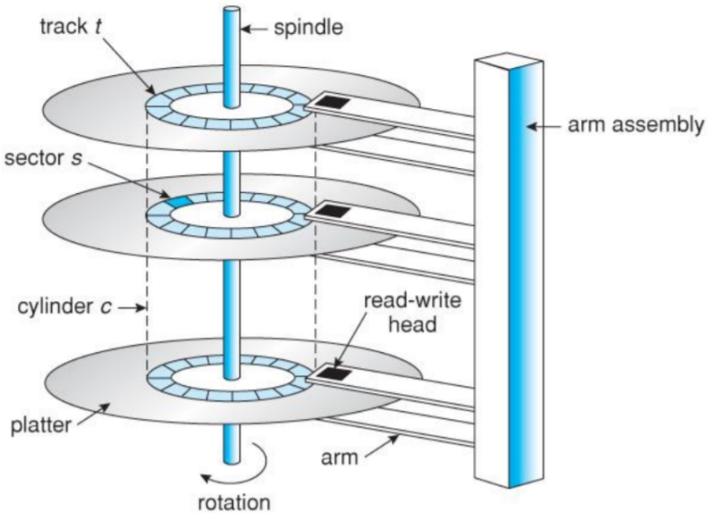
Created	@Jun 26, 2020 5:15 PM
Tags	운영체제

https://s3-us-west-2.amazonaws.com/secure.notion-static.com/600cdbe2-e91c-4a2e-918e-63c7e48a5f73/CH10._File_System.pdf

File Concepts

- **File : 바이트들의 배열.**
  - **Attributes : 메타데이터.** 파일 이름, id, 타입, 위치, 크기, 권한, 시간, 날짜 등
- **File operations**
  - create, write, read, reposition, ..
- **File access method**
  - 파일 읽어들이는 패턴에 관한 것.
  - **Sequental** access
  - **Direct** access: Random access

Disks and Partitions



디스크는 기계적으로 Sector 단위로 동작한다. (Sector: 512bytes)

512바이트 미만의 데이터는 읽을 수 없다.

- **Seek time** : 암이 그 트랙으로 이동하는 시간
- **Rotational delay** : 원하는 섹터가 헤더 밑에 올 때까지 디스크 회전에 걸리는 시간
- **Transfer time** : 데이터 읽어서 전송하는 시간

File System

- **파일 시스템 : 파일과 블록의 매핑을 관리하는 것.** 파일에 속하는 블록의 개수와 위치를 기록해놓는 것.
  - 파일 시스템은 **블록 단위**로 작동한다.
  - **블록 : 4KB**. (페이지, 프레임 모두 4KB!)
    - 섹터는 디스크를 하드웨어적으로 읽을 때 이야기.
    - **논리적으로** 봤을 때, 디스크는 블록의 연속이다.
  - **특정 파일 하나는 여러 블록으로 이루어져 있는데, 블록이 연속적일 필요는 없다.**
  - 그래서, **파일과 블록의 매핑**을 관리해주어야 한다. → **File System.**
  - ext2, ext3, ext4 등이 있다.
- **파일 시스템 구조**
  - **Boot block** : 부팅할 때 필요

- **Super block** : i-node table의 크기는 얼마인지, Data block의 크기는 얼마인지.
- **i-node table**
- **Data block**

File System Implementation

- **On-disk Information** : 디스크에 저장되는 정보들
  - **Boot control block**
  - **Volume control block** (=super block)
  - **Directory structure**
  - **FCB**
    - FCB (File Control Block) : 리눅스에서 inode - 한 파일의 메타데이터들을 기록해둔 것.
- **메모리에 저장되는 정보들 → OS에서 관리**
  - **Mount table**
  - In-memory **directory-structure cache**
  - (system-wide/per-process) **open-file table**
    - 파일 오픈한 기록(open-file table)은 메모리에 저장된다.
  - **In-memory FCB**
    - 메모리에 저장되는 FCB들도 있다.

FCB

리눅스에서는 inode라 부르는 그것. 파일 하나당 inode 하나.

파일에 관련된 모든 정보가 inode에 저장된다.

inode에는 해당 파일의 Data block의 위치가 저장된다.

inode를 먼저 보고 해당 위치의 Data block으로 가서 데이터를 읽는 것.

Reading/Writing a File

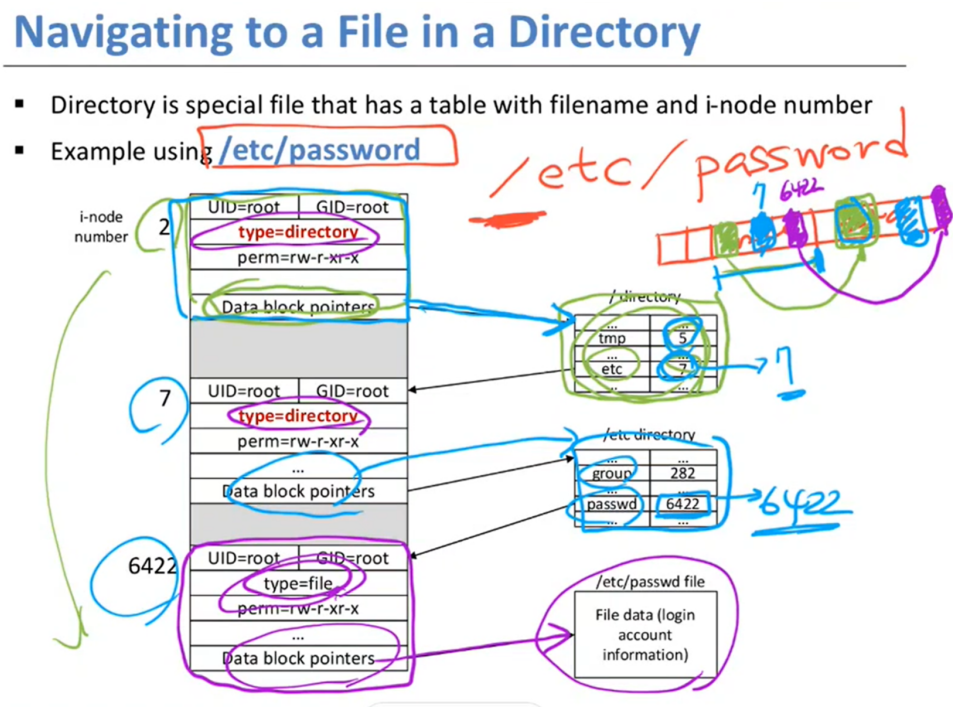
파일을 사용하기 전에 open()해야 한다. 커널 내부적으로 어떤 일이 일어나는가?

> 💡 **system-wide open-file table (swoft)** : 시스템에서 현재 열어놓은 모든 파일의 리스트
> **per-process open-file table (ppoft)** : 프로세스별로 열어놓은 파일의 리스트

> 💡 ppoft와 swoft의 가장 큰 차이?
> → **swoft에는 file pointer(=file offset)가 있다.** 현재 열려있는 파일에서 읽기/쓰기 하는 현재 위치가 어디인가. 파일 내에서의 현재 위치.

- **open()**
  - **swoft**에서 해당 파일(에 대한 정보)을 찾는다.
    - 해당 파일을 찾았으면, 이 entry를 가리키는 엔트리를 **ppoft**에 하나 추가.
    - 해당 파일이 없으면, **swoft**에 엔트리를 하나 만든다.
      - secondary storage에 가서 파일을 찾아와서, 알아서 swoft에 엔트리를 하나 만든다.
      - swoft에 엔트리 만들고 똑같이 ppoft에 이를 가리키는 엔트리 하나 만든다.
    - 이의 **file descriptor(= ppoft의 엔트리의 위치. index)가 리턴**된다.

Navigating to a File in a Directory



파일 경로를 주면, 어떻게 찾아가는가? 예를 들어, /etc/password

- /(루트 디렉토리)를 먼저 찾아간다.
  - 파티션 내부의 **inode table에서 해당 디렉토리 파일의 inode block**을 찾는다.
  - 이 inode가 가리키는 **데이터 블록의 위치로 가서**, 데이터 블록을 본다.
  - **디렉토리 파일**의 경우, **데이터 블록에 폴더 내부의 {파일, inode 번호} 쌍의 목록이 저장**된다.
  - 이 목록을 쭉 보고, 다음 디렉토리의 inode 번호를 얻어서 또 inode table에서 거기로 가서, ...
  - ... 이를 반복해서, 파일 inode까지 가서 파일 데이터 블록까지 가서 데이터 읽어오는 것

Virtual File Systems

컴퓨터에는 여러 파일 시스템이 존재할 수 있다. 구현이 모두 제각각이다. 어떻게 같이 쓰냐?

→ **VFS**!

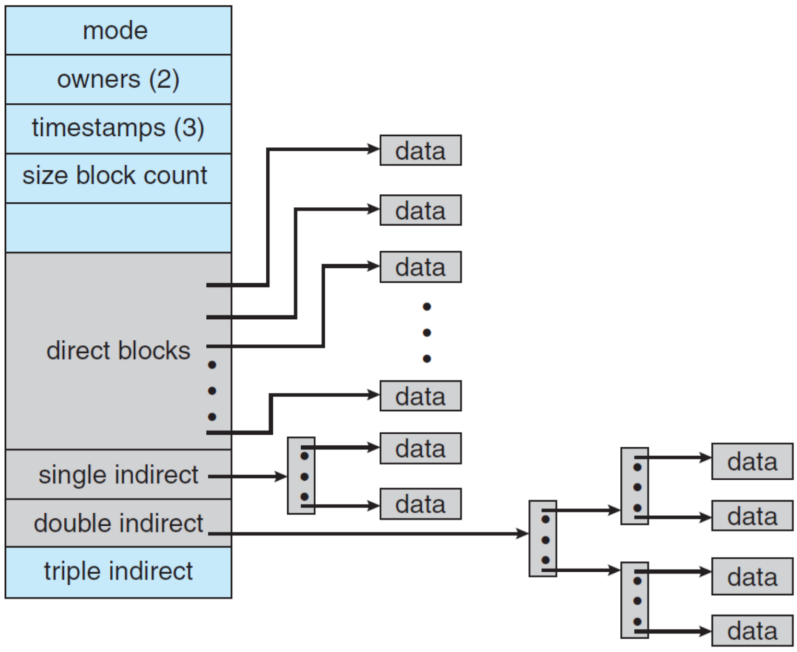
VFS : 모든 파일 시스템들의 추상화 레이어. 기준이 되는 표준 파일 시스템. 파일 시스템의 규격을 제시.

- 서로 다른 실제 파일 시스템들이 VFS 밑에 서브트리로 붙어 있다. mount되어 있다.

- **VFS는 File system interface와 각 파일 시스템들의 중간자 역할**을 한다.
  - **File system interface** : 우리가 파일 시스템을 쓰기 위한 모든 것들. open, write, ... 사용자가 직접 접하는 부분.
  - VFS는 요청이 들어오면 각 파일 시스템들의 알맞는 함수를 호출하고, 결과를 받아 유저에게 되돌려준다.
- **VFS의 역할**
  - **인터페이스와 구현을 분리한다.**
    - 사용자 입장에서의 인터페이스와, 그 뒷부분의 구현 부분은 분리되어야 한다. 사용자 입장에서는 일관적인 인터페이스를 제공한다.
  - **네트워크 상에서 각각의 모든 파일들의 이름(경로)이 unique하도록 만들어준다.**
    - **vnode** : 하나의 FS 내에서는 중복된 path가 없지만, **네트워크에서는 여러 파일 시스템이 붙어 있는 경우, 동일한 경로의 서로 다른 파일이 존재할 수 있다. → 네트워크 정보까지 더해서 파일 이름(경로)를 정한다.**

Combined Scheme - inode에서의 Data block 구조

Remind: inode 블록 하나는 4KB. 그냥 모든 블록은 4KB.



inode에는 Data block에 대한 포인터들이 있다.

- **direct block** : 특정 Data block의 번호를 바로 가리킴.
- **single indirect block** : 포인터를 따라가면 4KB짜리 inode block이 또 하나 나오는데, 이 블록에는 여러 Data block에 대한 포인터들이 있다.
- **double indirect block, triple indirect block**도 똑같음. 몇 번 거쳐가는가?

→ 이를 통해, inode에서는 큰 사이즈의 파일도 지원할 수 있는 것!

문제: direct, single, double, triple 각각 몇 개일때, 지원할 수 있는 최대 파일 크기는?