CS Study

File System - 2

2025. 09. 29

Jiwon Son

목차

FS 리뷰

File System(:FS) 정의

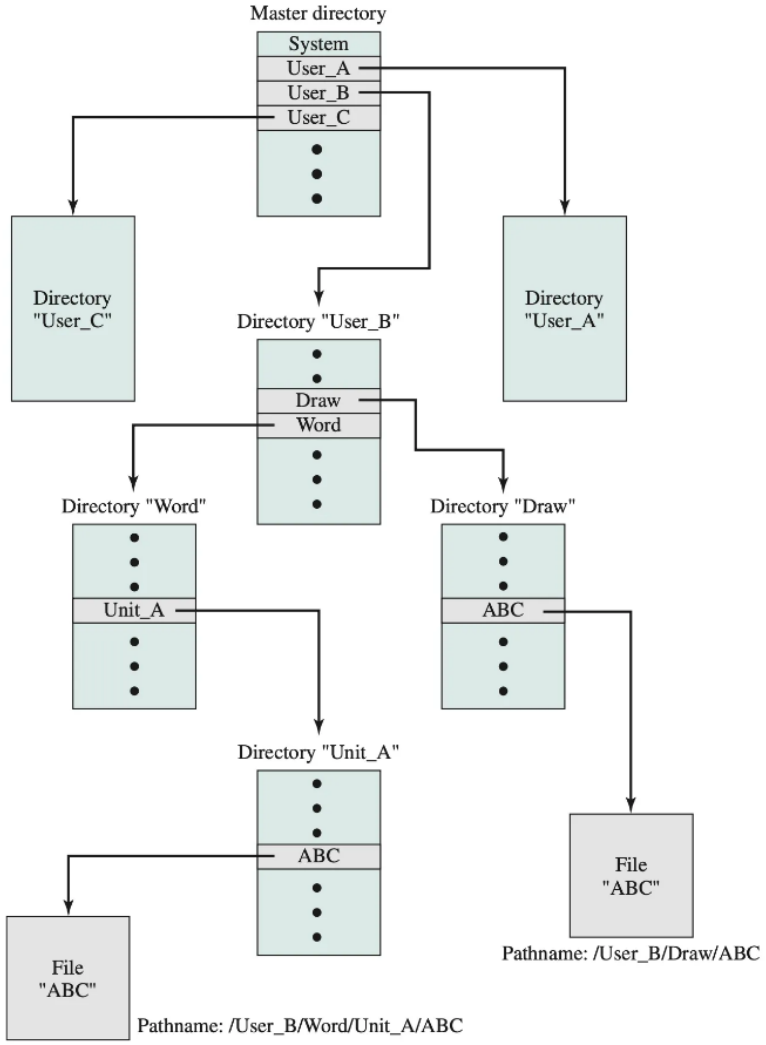
- OS가 데이터를 저장/조작/보호/복구 등 기능을 제공하는 소프트웨어 계층
- =데이터와 메타데이터를 활용해 파일을 관리하는 시스템

동작 방식

- Superblock에 모든 파일 시스템 관련 내용을 저장
- File System이 마운트되면 OS는 Superblock을 읽어들임
- 이후 파일에 관련된 모든 동작은 Superblock을 기반으로 동작

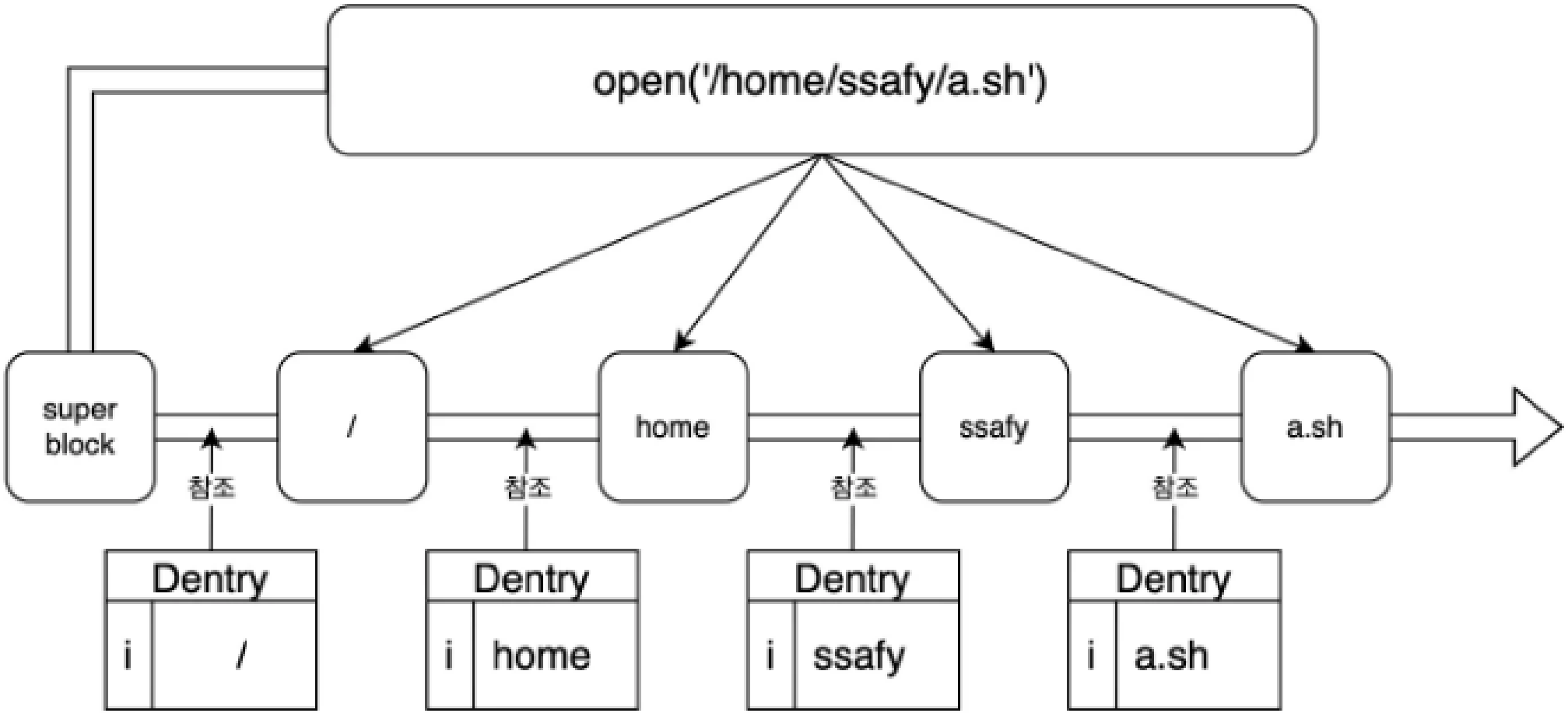
구조

- Tree 형태로 구성됨
  - 현실과 비슷한 방식으로, 직관성 보장
  - 리눅스 기본 철학을 따름
    - “Everything is a file”
    - 단일 namespace에서 다양한 자원을 동일 인터페이스로 사용



FS 리뷰

◆ 파일 참조 과정


open('/home/ssafy/a.sh')
super block
/
home
ssafy
a.sh
참조
참조
참조
참조
Dentry
i
/
Dentry
i
home
Dentry
i
ssafy
Dentry
i
a.sh

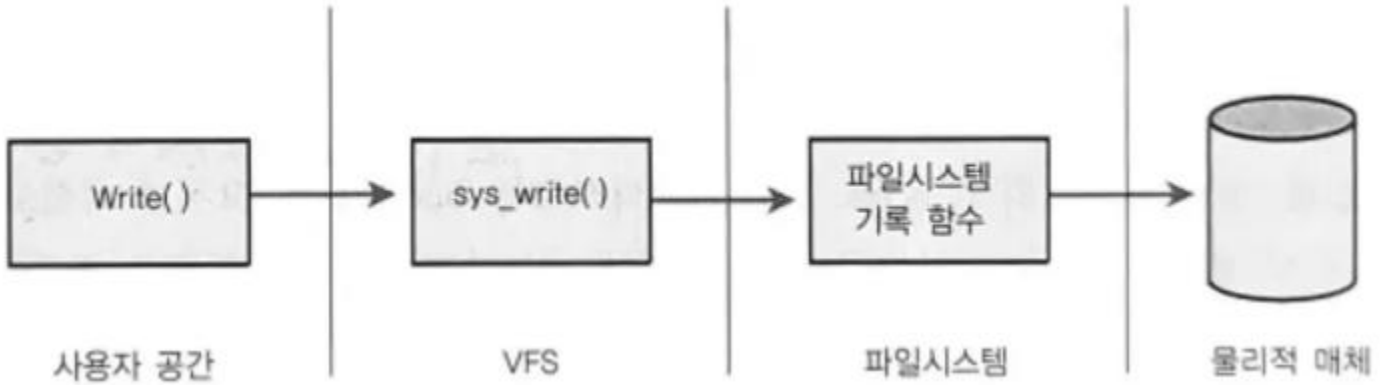
FS 리뷰

◆ Virtual File System(:VFS) 정의

- 커널 안에서 여러 종류의 파일 시스템을 추상화해서, 공통 인터페이스로 제공하는 계층
- 사용자는 파일 시스템의 구조/방식/구현 방법에 관계없이 모두 동일 메소드로 사용 가능

◆ 역할

- VFS -> User(=process)
  - POSIX 표준 System call을 동일하게 제공
    - POSIX(Portable Operating System Interface) : UNIX-like OS의 공통 API를 정리한 국제 표준
- VFS -> File System
  - 각 파일 시스템의 구체적인 동작을 파일 연산 함수 포인터 테이블로 연결



FS의 4대 객체 - File

정의

- 논리적으로 연관된 데이터의 집합

특성

- File 안에는 단순히 데이터 바이트 스트림만 존재
  - 바이트 스트림 = 2bit 나열
  - 파일에 관련된 모든 메타데이터는 별도 블록에 저장됨
    - 권한, 소유자, 크기, 위치, 타임스탬프 등 : inode에 기록
    - Inode 번호와 파일 이름 : Directory Entry에 기록

Everything is a file

- 리눅스는 모든 관리대상을 file로 간주

FS의 4대 객체 - inode

◆ 정의

- 파일에 대한 메타데이터 정보를 표현하는 데이터 구조(index node의 준말)
- 파일의 이름과 내용을 제외한 거의 모든 정보가 담김

◆ 주요 요소

- 파일 권한, 생성/수정시간, 파일 크기 등 메타데이터
  - Mode, owners, timestamp, size, etc.
- 데이터가 저장되어있는 주소
  - Direct, indirect(single, double, triple)
- 기타 정보
  - Block count : 실제 할당된 블록 수
  - Reference count : 파일이 참조되는 곳의 개수
  - Flags : 특수동작 플래그
    - E.g.,
      - I : immutable = 삭제불가
      - D : no dump = 백업제외
  - Generation number : 재사용 판단 번호

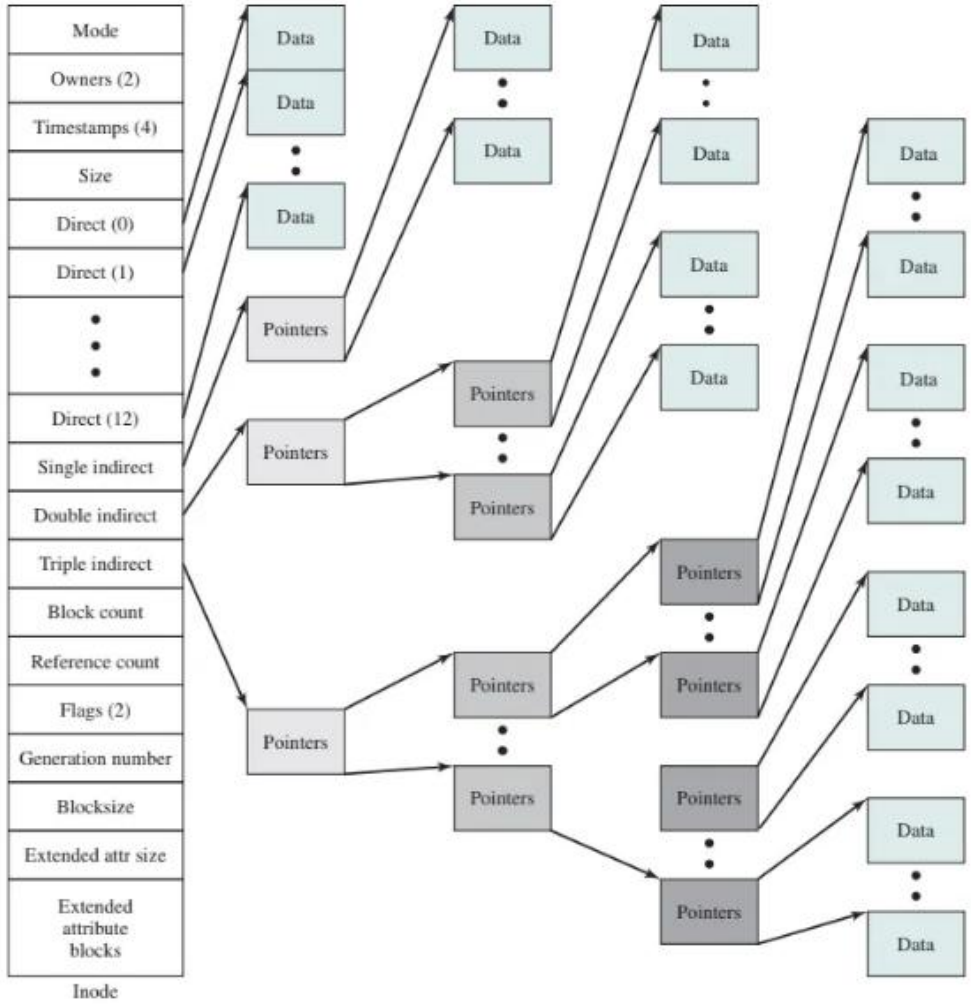


Figure 12.15 Structure of FreeBSD Inode and File

FS의 4대 객체 - inode

- **Ubuntu에서 inode 정보 확인하기**
  - 사진은 사용중인 FS, 마운트 위치 단위로 확보중인 inode의 최대 개수, 사용량 확인

```
ssafy@ssafy-VirtualBox:~/250924$ df -i
파일 시스템      Inodes  IUsed   IFree IUse% 마운트위치
tmpfs            501221   1059  500162     1% /run
/dev/sda3       3244032 238936 3005096     8% /
tmpfs            501221      1  501220     1% /dev/shm
tmpfs            501221      4  501217     1% /run/lock
/dev/sda2             0      0       0      - /boot/efi
tmpfs            100244    147  100097     1% /run/user/1000
```

FS의 4대 객체 – inode (참고)

◆ Indirect block 사용 의의

- Inode에서 지원하는 최대 파일 용량보다 약 8300만배 많은 데이터 저장 가능
  - 최대 4TB, 최소 48kb

- 총 용량
  - $C = (D + N + N^2 + N^3) * B$
    - C : 총 가용 용량
    - D : direct 개수
    - B : 한 블록당 용량
    - N : B / P
      - P : 포인터 크기

가정

고전형 포인터 방식(12개 direct + single/double/triple indirect)

블록 크기 B = 4096 byte

포인터 크기 P = 4byte

한 포인터 블록이 가리킬 수 있는 데이터 블록 수 N

$$N = B/P = 4096/4 = 1024$$

단계별 수용 용량

1. Direct

$$12blocks * 4KB = 48KB$$

2. Single Indirect

→ 포인터 블록 1개가 1024개의 데이터 블록의 시작점을 가리킴

$$1024 * 4KB = 4MB$$

3. Double Indirect

→ 첫 번째 포인터 블록이 1024개의 두 번째 포인터 블록을 가리키고, 두 번째 포인터 블록은 다시 1024개의 데이터 블록을 가리킴

$$1024 * 1024 * 4KB = 4GB$$

4. Triple Indirect

→위 Double Indirect에서 가리키는 과정이 1번 더 추가됨

$$1024 * 1024 * 1024 * 4KB = 4TB$$

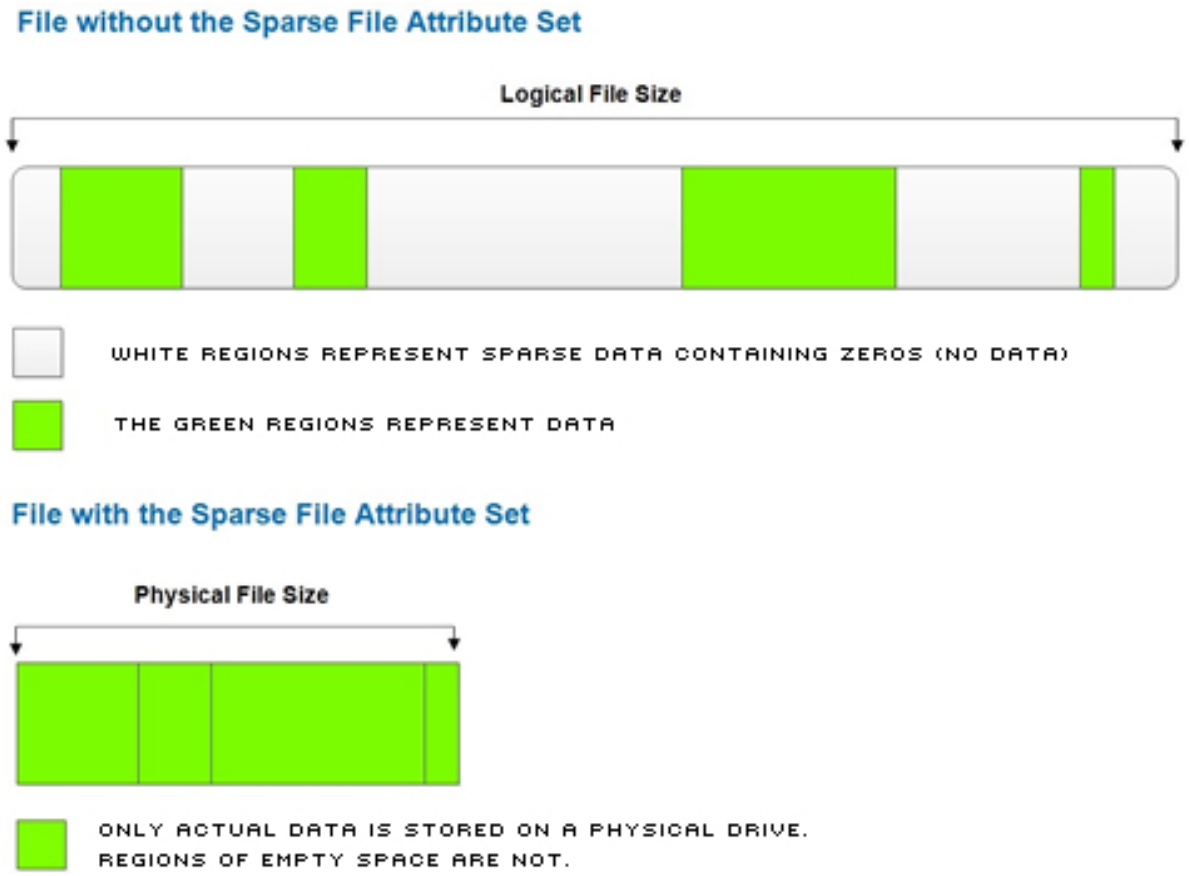
총합(이론상 데이터 최대 지원 용량)

위 모든 포인터를 전부 다 사용한다면, 1 ~ 4의 총합으로 구할 수 있음.

$$4TB + 4GB + 4MB + 48KB$$

FS의 4대 객체 – inode (참고2)

- **Block count는 어차피 할당 용량만큼 주어지는 것 아닌가?**
  - 그럴수도있고, 아닐수도있음
  - LINUX는 희소 파일을 지원해서 효율적인 프로비저닝을 지원함
    - 희소 파일(Sparse File) : 내용은 커보이지만, 실제로는 비어있어서 디스크 블록을 차지하지 않는 파일
    - 프로비저닝(Provisioning) : 사용자 요구에 맞춰 필요한 자원을 준비하고 할당하는 과정
  - 겉보기엔 1GB여도, 거의 모든 부분이 0이고 실제 데이터를 저장하기 위해 필요한 만큼만 물리 공간 사용
  - 용례
    - 가상머신 디스크
    - DBMS 로그 파일
    - 과학 계산 / 시뮬레이션



FS의 4대 객체 – inode (참고2)

- 희소 파일 구경

```
ssafy@ssafy-VirtualBox:~/workspace$ truncate -s 1G sparse.img
ssafy@ssafy-VirtualBox:~/workspace$ ls -lh sparse.img
-rw-rw-r-- 1 ssafy ssafy 1.0G  9월 29 15:34 sparse.img
ssafy@ssafy-VirtualBox:~/workspace$ du -h sparse.img
0       sparse.img
ssafy@ssafy-VirtualBox:~/workspace$ echo "hello" > sparse.img
ssafy@ssafy-VirtualBox:~/workspace$ du -h sparse.img
4.0K    sparse.img
```

FS의 4대 객체 - dentry

◆ 정의

- 파일 이름과 inode 번호를 매핑하는 항목
  - 파일엔 데이터 스트림만, inode엔 메타데이터만 있음
  - 이 둘을 매핑해줘야 inode가 파일을 참조할 수 있음

- 사진 설명
  - 위는 매핑 정보를 진단할 수 있는 명령어($ ls –li)
    - 가장 좌측이 inode 번호, 가장 우측이 파일명
  - 아래는 파일단위 매핑정보 체크 명령어($ stat (파일명))

◆ 요약

- 디렉토리는 결국 파일명과 inode 매핑 테이블
  - 각 테이블 내 원소가 dentry인것

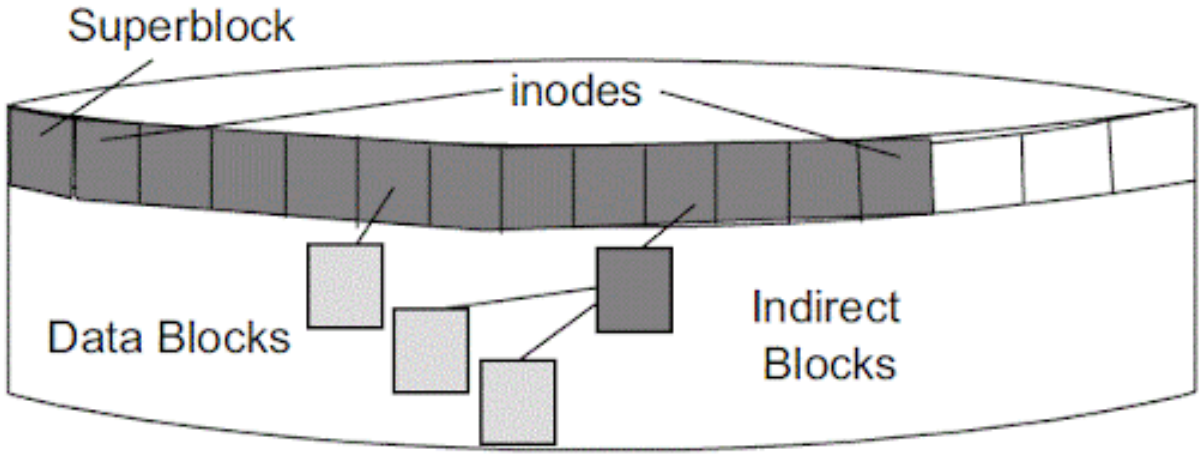
```
ssafy@ssafy-VirtualBox:~/250924$ ls -li
합계 28
1050124 -rw-rw-r-- 1 ssafy ssafy  108  9월 24 08:57 a.sh
1710267 drwxrwxr-x 2 ssafy ssafy 4096  9월 24 10:07 backup
1710471 -rwxrwxr-x 1 ssafy ssafy  117  9월 24 10:51 backup.sh
1710284 drwxrwxr-x 3 ssafy ssafy 4096  9월 24 10:33 home
1710280 -rw-rw-r-- 1 ssafy ssafy   78  9월 24 09:04 mem.sh
1710281 -rw-rw-r-- 1 ssafy ssafy  130  9월 24 09:52 run.sh
1710282 -rw-rw-r-- 1 ssafy ssafy  300  9월 24 09:37 secret.sh
```

```
ssafy@ssafy-VirtualBox:~/250924$ echo "Hello World" > hello.txt
ssafy@ssafy-VirtualBox:~/250924$ ls -l hello.txt
-rw-rw-r-- 1 ssafy ssafy 12  9월 24 17:20 hello.txt
ssafy@ssafy-VirtualBox:~/250924$ stat hello.txt
  파일: hello.txt
  크기: 12              블록: 8          입출력 블록: 4096   일반 파일
Device: 803h/2051d      Inode: 1705904     Links: 1
접근: (0664/-rw-rw-r--)  UID: ( 1000/   ssafy)   GID: ( 1000/   ssafy)
접근: 2025-09-24 17:20:52.387951175 +0900
수정: 2025-09-24 17:20:52.387951175 +0900
변경: 2025-09-24 17:20:52.387951175 +0900
생성: 2025-09-24 17:20:52.387951175 +0900
ssafy@ssafy-VirtualBox:~/250924$
```

FS의 4대 객체 - superblock

◆ 정의

- 파일시스템 전체에 대한 메타데이터 블록
  - 블록 크기와 개수, inode 테이블, 상태정보 획득



struct super_block

Type	Field	Description
struct list_head	s_list	Superblock linked list
dev_t	s_dev	Device identifier
u_long	s_blocksize	Block size in bytes
u_char	s_dirt	Dirty (modified) flag
struct super_operations *	s_op	Superblock methods
struct semaphore	s_lock	Superblock semaphore
struct list_head	s_inodes	List of all inodes
struct list_head	s_io	Inodes waiting for write
struct list_head	s_files	List of file objects

더 알아보기

- 일관성 유지 정책
  - Journaling
  - Copy-on-Write
- 복구 정책
  - Snapshot
  - Clone
- 디스크 사용량 제한
  - Quota
  - ACL(Access Control List)
- 분산 저장 정책
  - Consensus Algorithm(합의 알고리즘)
    - Paxos
    - Practical Byzantine Fault Tolerance
    - Zyzzyva
- 성능 최적화
  - Buffer cache
    - Read ahead, Write back
  - Delegation(권한 위임)
- 내결함성(Fault Tolerance)
  - RAID
  - Erasure Coding
  - Replication
  - Striping
- 백업 알고리즘(HA / DR)
  - Clustering
  - Active – Passive
  - Active - Active

EOF