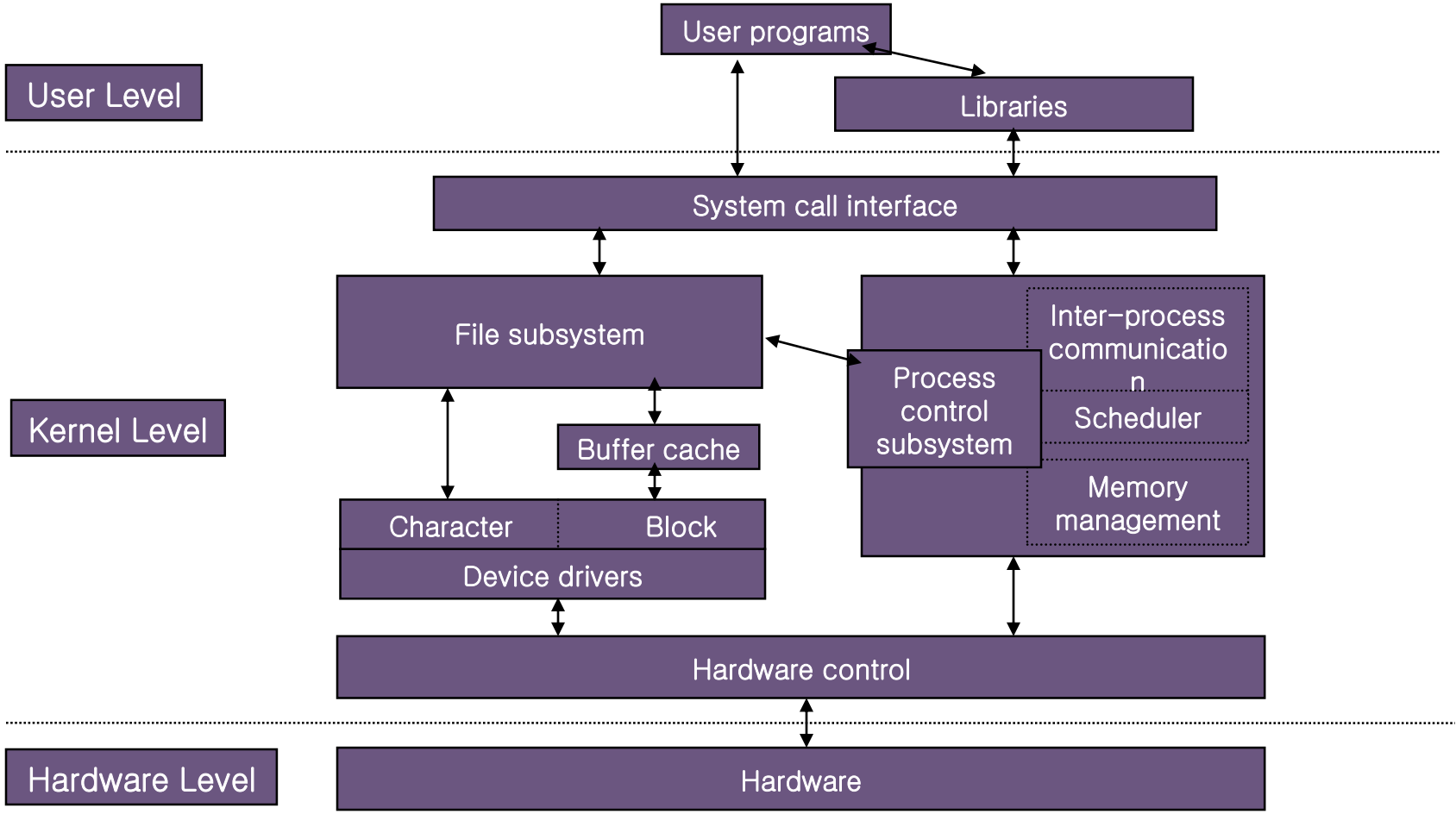
4장 파일 입출력
5장 파일시스템 구조

5.1 파일 시스템 구현

Kernel의 구조


User programs
User Level
Libraries
System call interface
File subsystem
Inter-process communicatio n
Process control subsystem
Scheduler
Kernel Level
Buffer cache
Memory management
Character
Block
Device drivers
Hardware control
Hardware Level
Hardware

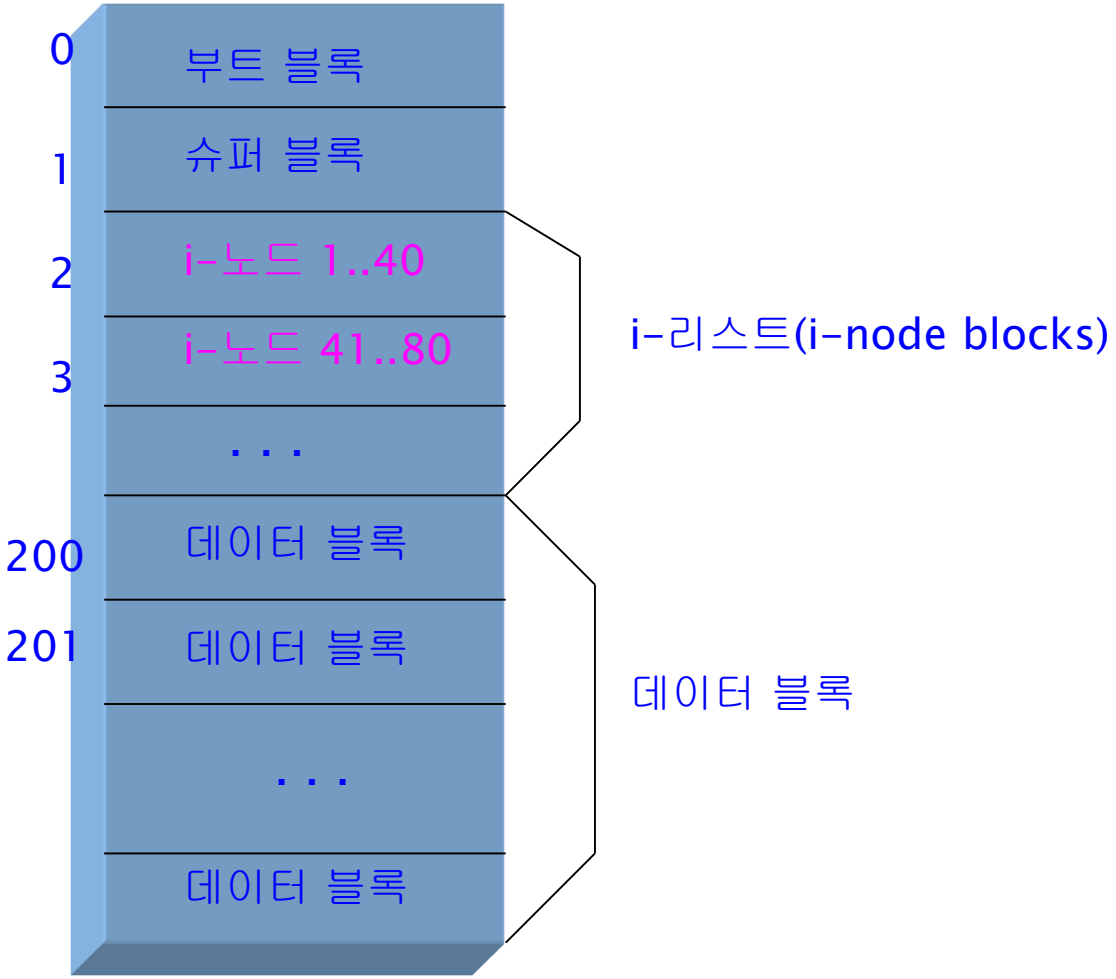
파일 서브시스템

- 기능
  - 파일 관리, 파일 공간 할당, free space 관리
  - 파일에 대한 Access Control, File I/O

- File Subsystem과 I/O Sunsystem과의 관계
  - Block I/O
    - 블록 I/O 장치의 데이터는 버퍼링 메카니즘에 의해 파일 서브시스템에게 제공됨
    - 버퍼링 메카니즘은 블록 I/O 장치 드라이버와 상호 작용하여 커널로(혹은 커널로부터) 데이터 전송
  - raw(character) I/O
    - 문자 I/O 장치(tty 등)는 버퍼링 메카니즘을 사용하지 않음

- System calls
  - open, close, read, write, stat, chown, chmod etc.

파일 시스템 구조

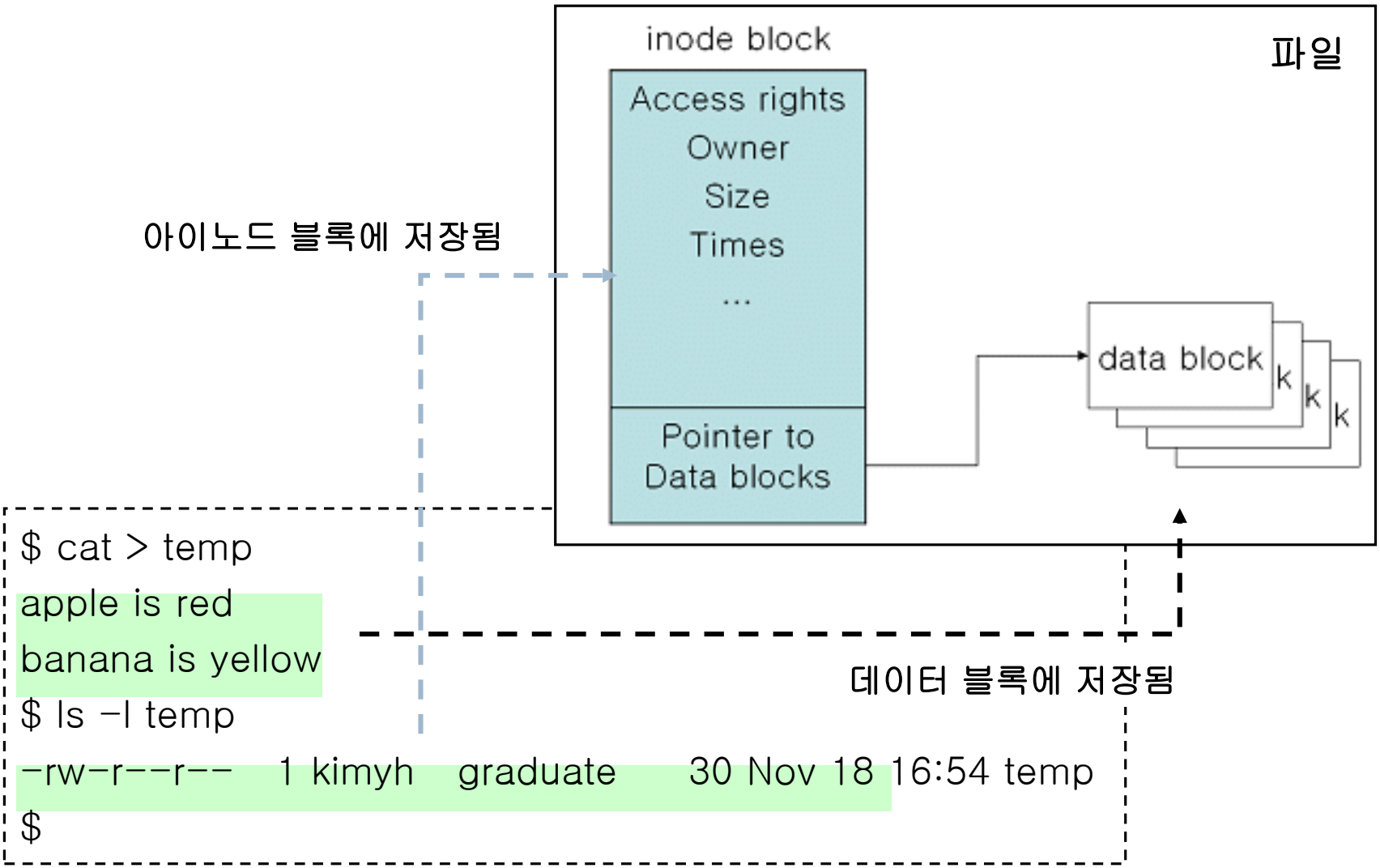

0
부트 블록
1
슈퍼 블록
2
i-노드 1..40
3
i-노드 41..80
. . .
200
데이터 블록
201
데이터 블록
. . .
데이터 블록
i-리스트(i-node blocks)
데이터 블록

파일 시스템 구조

- 부트 블록(Boot block)
  - 파일 시스템 시작부에 위치하고 보통 첫 번째 섹터를 차지
  - 부트스트랩 코드가 저장되는 블록
- 슈퍼 블록(Super block)
  - 전체 파일 시스템에 대한 정보를 저장
    - 총 블록 수, 사용 가능한 i-노드 개수, 사용 가능한 블록 비트 맵, 블록의 크기, 사용 중인 블록 수, 사용 가능한 블록 수 등
- i-리스트(i-list)
  - 각 파일을 나타내는 모든 i-노드들의 리스트
  - 한 블록은 약 40개 정도의 i-노드를 포함

    모든 파일은 반드시 i-노드 블록을 하나 가지고 있다.
- 데이터 블록(Data block)
  - 파일의 내용(데이터)을 저장하기 위한 블록들

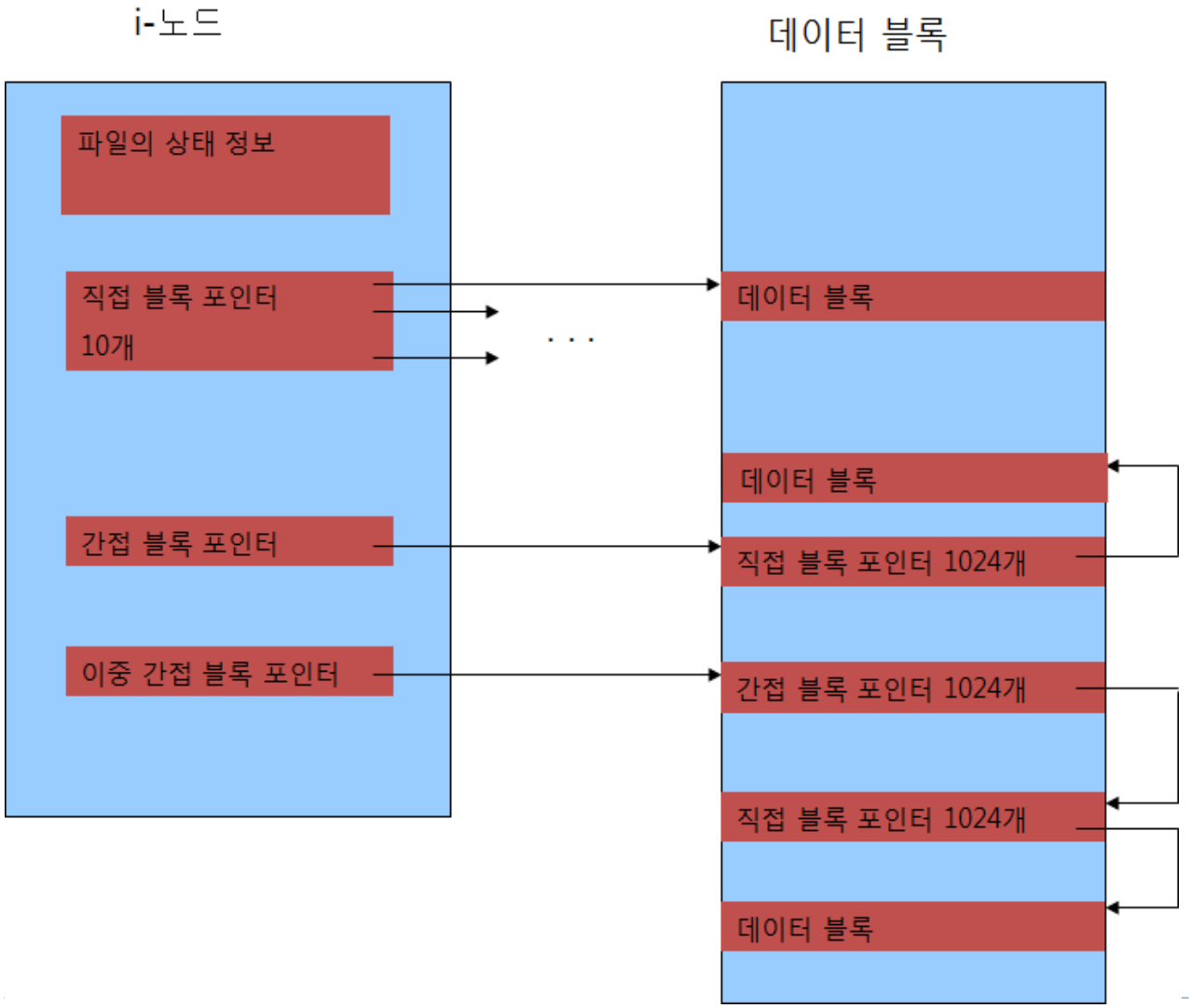
파일 시스템의 구조


inode block
파일
Access rights
Owner
Size
Times
...
Pointer to
Data blocks
data block
k
k
k
아이노드 블록에 저장됨
$ cat > temp
apple is red
banana is yellow
$ ls -l temp
-rw-r--r-- 1 kimyh graduate 30 Nov 18 16:54 temp
$
데이터 블록에 저장됨

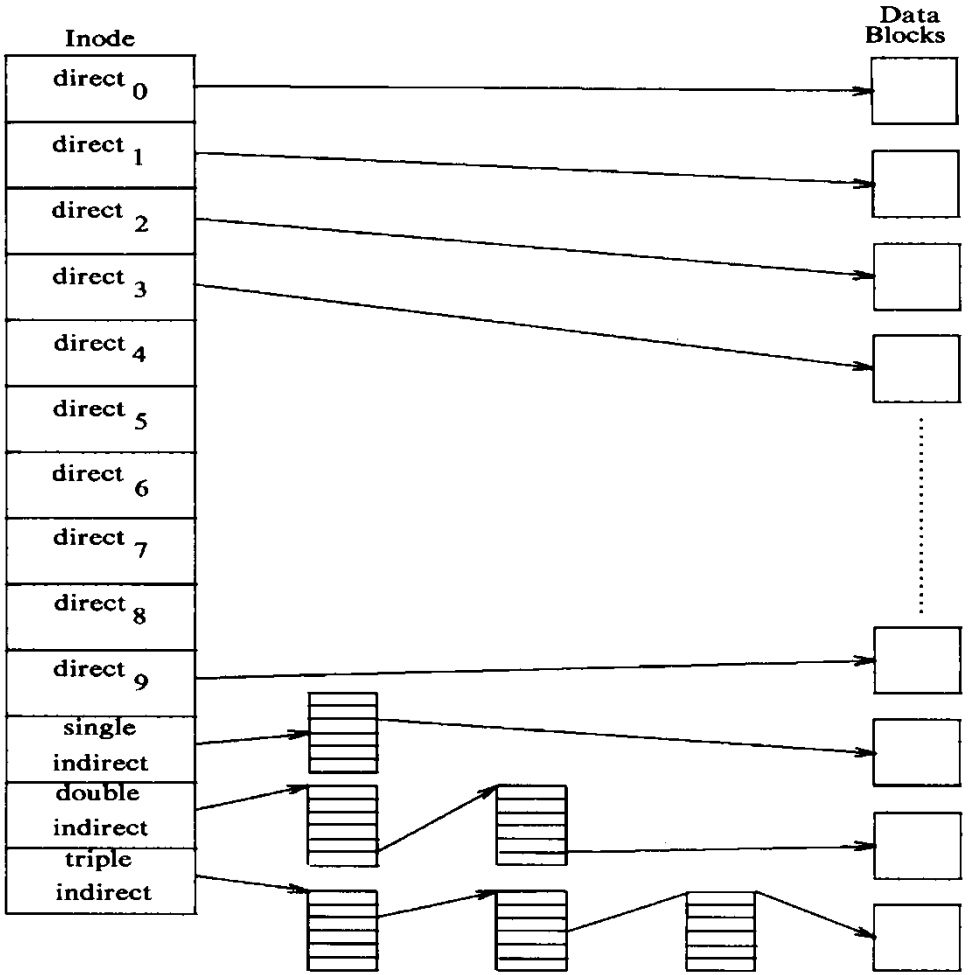
i-노드(i-node)

- 한 파일은 하나의 i-노드를 갖는다.
- 파일에 대한 모든 정보를 가지고 있음
  - 소유자(individual owner/group owner)
  - 파일 타입: 일반 파일, 디렉터리, 블록 장치, 문자 장치 등
  - 파일 크기
  - 사용권한(owner/group/other)
  - 파일 소유자 및 그룹
  - 접근 및 갱신 시간
  - the number of links in the file
  - 데이터 블록에 대한 포인터(주소) 등

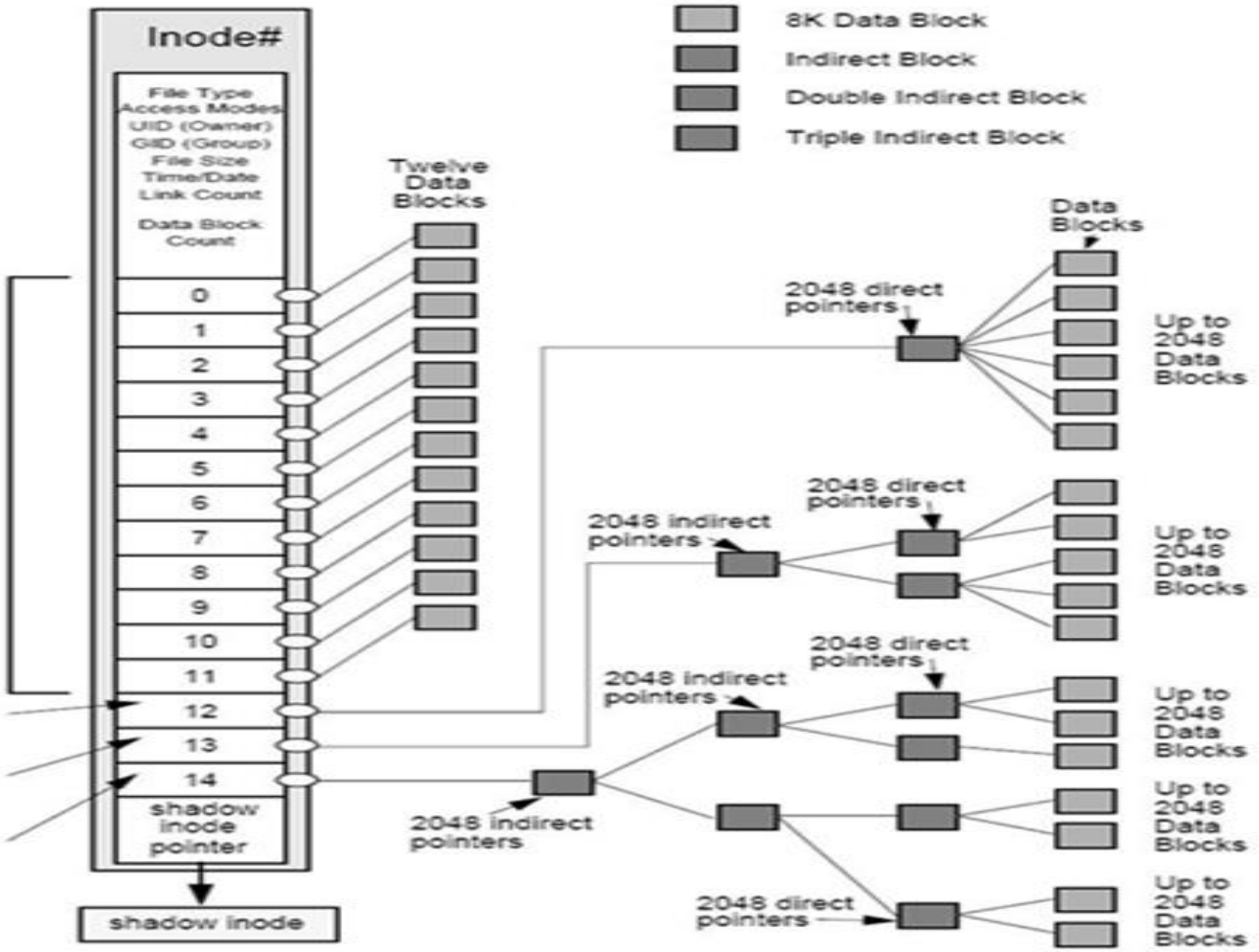
블록 포인터


i-노드
데이터 블록
파일의 상태 정보
직접 블록 포인터
10개
간접 블록 포인터
이중 간접 블록 포인터
데이터 블록
데이터 블록
직접 블록 포인터 1024개
간접 블록 포인터 1024개
직접 블록 포인터 1024개
데이터 블록

데이터 블록에 대한 포인터



Linux File System(ext2, ext3)

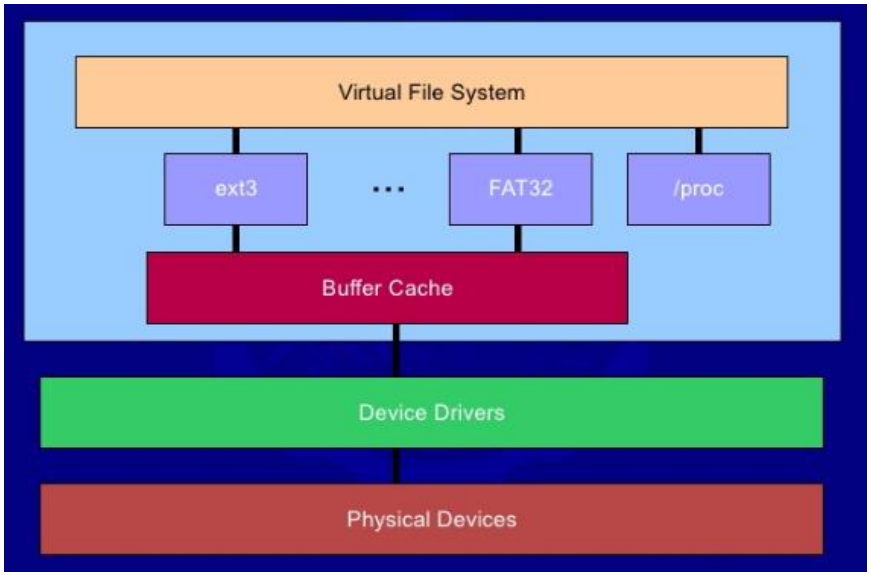


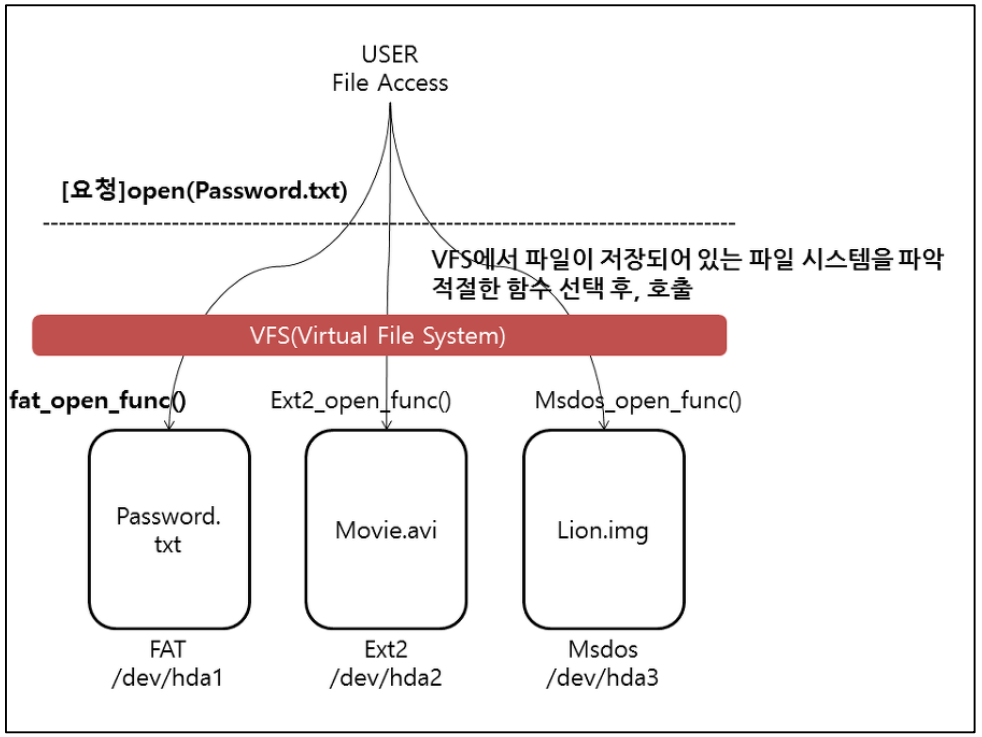
df –h | -i | -T

```
[hansung@localhost ~]$ df -h
Filesystem            Size  Used Avail Use% Mounted on
/dev/mapper/VolGroup-lv_root
                      6.5G  3.4G  2.9G  54% /
tmpfs                 504M   80K  504M   1% /dev/shm
/dev/sda1             477M   35M  418M   8% /boot
[hansung@localhost ~]$ df -i
Filesystem            Inodes  IUsed  IFree IUse% Mounted on
/dev/mapper/VolGroup-lv_root
                      440640 103754 336886   24% /
tmpfs                 128809      5 128804    1% /dev/shm
/dev/sda1             128016     38 127978    1% /boot
[hansung@localhost ~]$ df -T
Filesystem            Type  1K-blocks    Used Available Use% Mounted on
/dev/mapper/VolGroup-lv_root
                      ext4    6795192 3472344   2971004  54% /
tmpfs                 tmpfs    515236     300    514936   1% /dev/shm
/dev/sda1             ext4     487652   34825    427227   8% /boot
[hansung@localhost ~]$
```

파일시스템 종류 및 Virtual File System

ext2, ext3, ext4, mimix, xfs, hpfs 등





https://richong.tistory.com/197

파일관련 정보

- $ ls –l 명령을 실행했을 때 보여지는 파일의 정보

-	rw-r--r--	1	kimyh	gradua te	30	Nov 18 16:54	temp
파일 유형	접근권한	하드링크수	소유주 이름	그룹 이름	파일 크기	수정 날짜	파일명

← 아이노드 블록에 저장 →

디렉터리 파일의 데이터 블록에 저장

```
$ cat > temp
apple is red
banana is yellow
$ ls -l temp
-rw-r--r--   1 kimyh   graduate      30 Nov 18 16:54 temp
$
```

디렉터리와 경로명

- **Directory**
  - 파일의 목록을 저장하기 위한 특수한 형태의 파일이다.
  - 디렉터리 파일이라고 부르기도 함
  - 디렉터리 파일의 데이터 블록에 파일명이 목록으로 저장되어 있다.
- **Directory entry**
  - 디렉터리 파일의 목록을 항(entry)이라고 한다.
  - 모든 디렉터리는 항상 두 개의 항을 가지고 있다.
    - 자기 자신을 나타내는 항 (.)
    - 부모 디렉터리를 나타내는 항 (..)

```
$ ls -la
drwxr-xr-x    2 kimyh    graduate     4096 Nov 18 17:39 .
drwxr-xr-x    3 kimyh    graduate     4096 Nov 18 17:39 ..
$
```

Directory와 경로명

- **아이노드 블록 번호**
  - 모든 디렉터리 항은 가리키는 파일의 아이노드 블록 번호를 가지고 있다.
    - $ ls –i 로 확인할 수 있음

```
$ ls -lai
2845303 drwxr-xr-x    2 kimyh    graduate      4096 Nov 18 17:44 .
3139591 drwxr-xr-x    3 kimyh    graduate      4096 Nov 18 17:39 ..
2845304 -rw-r--r--    1 kimyh    graduate        13 Nov 18 17:44 file
$
```

아이노드 블록 번호

Directory와 경로명

- **ls 명령이 수행되는 과정**

  ls 명령은 지정한 디렉터리의 디렉터리 항을 출력한다.

  그 과정을 다음과 같다.

1. 현재 디렉터리 파일을 open하여 데이터 블록에서 항을 하나 읽는다.
2. 읽어 들인 디렉터리 항으로부터 아이노드 번호와 파일 이름을 구한다.
3. 아이노드 번호로 아이노드 블록을 지정하여 필요한 정보를 가져온다.
4. 정보를 출력한다.
5. 현재 디렉터리의 데이터 블록에서 다음 항을 읽고 2번에서 4번의 과정을 반복한다.

Directory와 경로명

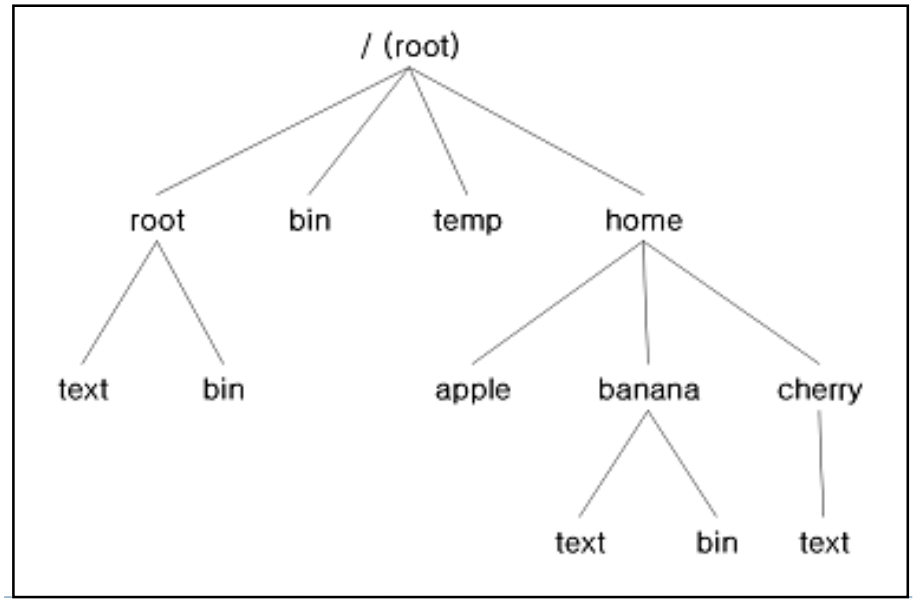
- **디렉터리 파일의 논리적인 구조**

```
$ ls -lai
119271 drwxr-xr-x   2 kimyh    graduate      512 Nov 18 17:55 ./
 15552 drwxr-xr-x   5 kimyh    graduate      512 Nov 18 17:54 ../
119272 -rw-r--r--   1 kimyh    graduate       13 Nov 18 17:55 file
119273 -rw-r--r--   1 kimyh    graduate       14 Nov 18 17:55 text
$
```

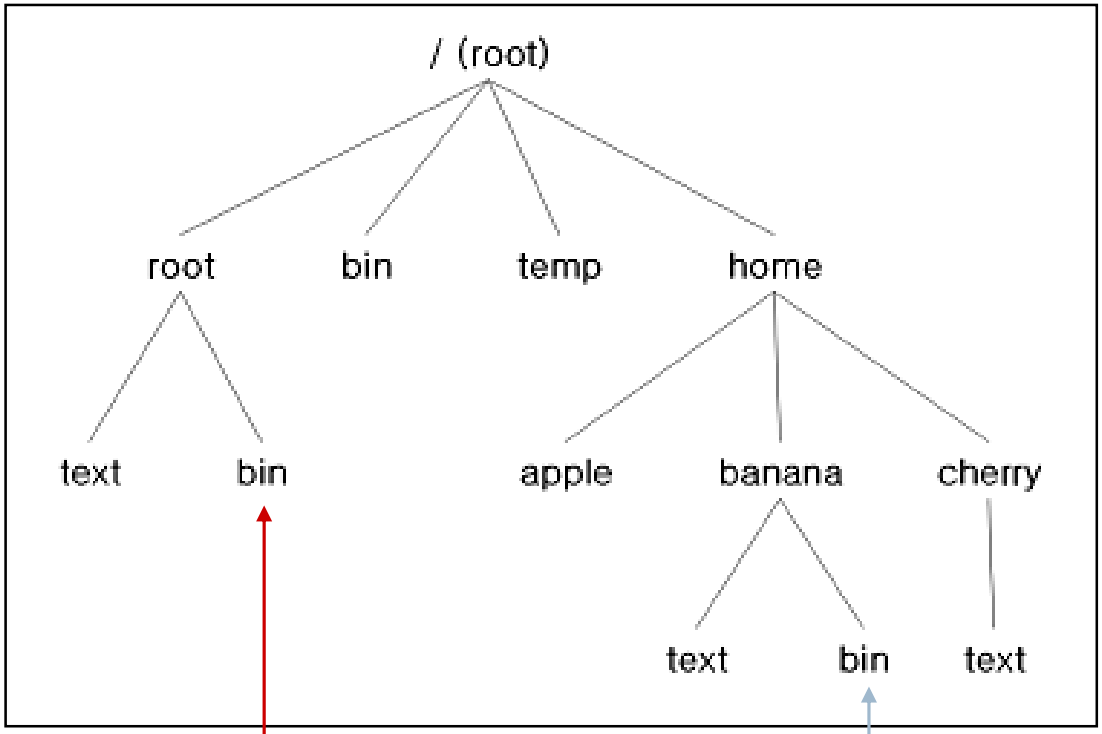
119271	.	₩0			
15552	.	.	₩0		
119272	f	i	l	e	₩0
119273	t	e	x	t	₩0

Directory와 경로명

- **계층 구조**
  - 리눅스의 파일 시스템에는 많은 수의 디렉터리와 파일이 존재한다.
  - 하나의 디렉터리 안에는 또 다른 디렉터리나 파일이 존재한다.
  - 모든 디렉터리와 파일은 유일하게 존재하는 루트 디렉터리를 시작으로 트리(tree) 모양의 계층적인 구조를 이루고 있다.



절대 경로와 상대 경로



절대 경로

/home/apple

현재 디렉터리에 상관없이 항상 동일한 대상을 가리킴

상대 경로

./bin

현재 디렉터리에 따라 가리키는 대상이 달라짐

현재 디렉터리가 banana

현재 디렉터리가 root

※상대 경로에서 "."는 현재 디렉터리, ".."는 부모 디렉터리를 의미한다.
banana 디렉터리의 ".."는 home 디렉터리이다.

새로운 파일의 생성

- **새로운 파일을 생성하는 과정**

```
$ cat > file2
apple is red
^D
$ ls -li
2845304 -rw-r--r--    1 kimyh    graduate        13 Nov 18 17:44 file
2845305 -rw-r--r--    1 kimyh    graduate        13 Nov 18 20:10 file2
$
```

1. **디렉토리 파일을 open해서 새롭게 생성할 디렉터리에 동일한 이름의 항이 존재하는지 확인한다.**
2. **비어있는 아이노드 블록 하나를 할당 받는다.**
3. **파일이 저장할 데이터의 크기에 따라 데이터 블록을 할당 받는다.**
4. **데이타블록 기록 및 저장 -> Inode 블록에 정보저장 -> 디렉토리 파일에 파일이름과 inode번호 저장**

파일 입출력 구현

- 파일 입출력 구현을 위한 커널 내 자료구조
  - 파일 디스크립터 배열(Fd array)
  - 열린 파일 테이블(Open File Table)
  - 동적 i-노드 테이블(Active i-node table)

File Descriptor

- 실행중인 프로그램과 하나의 파일 사이에 연결된 개방 상태
- 음수가 아닌 정수형 값으로 시스템이 결정
- 프로그램 작성 시 실제 값이 무엇인지 알 필요 없음
- 파일 개방이 실패하면 -1이 됨
- 커널에 의해서 관리

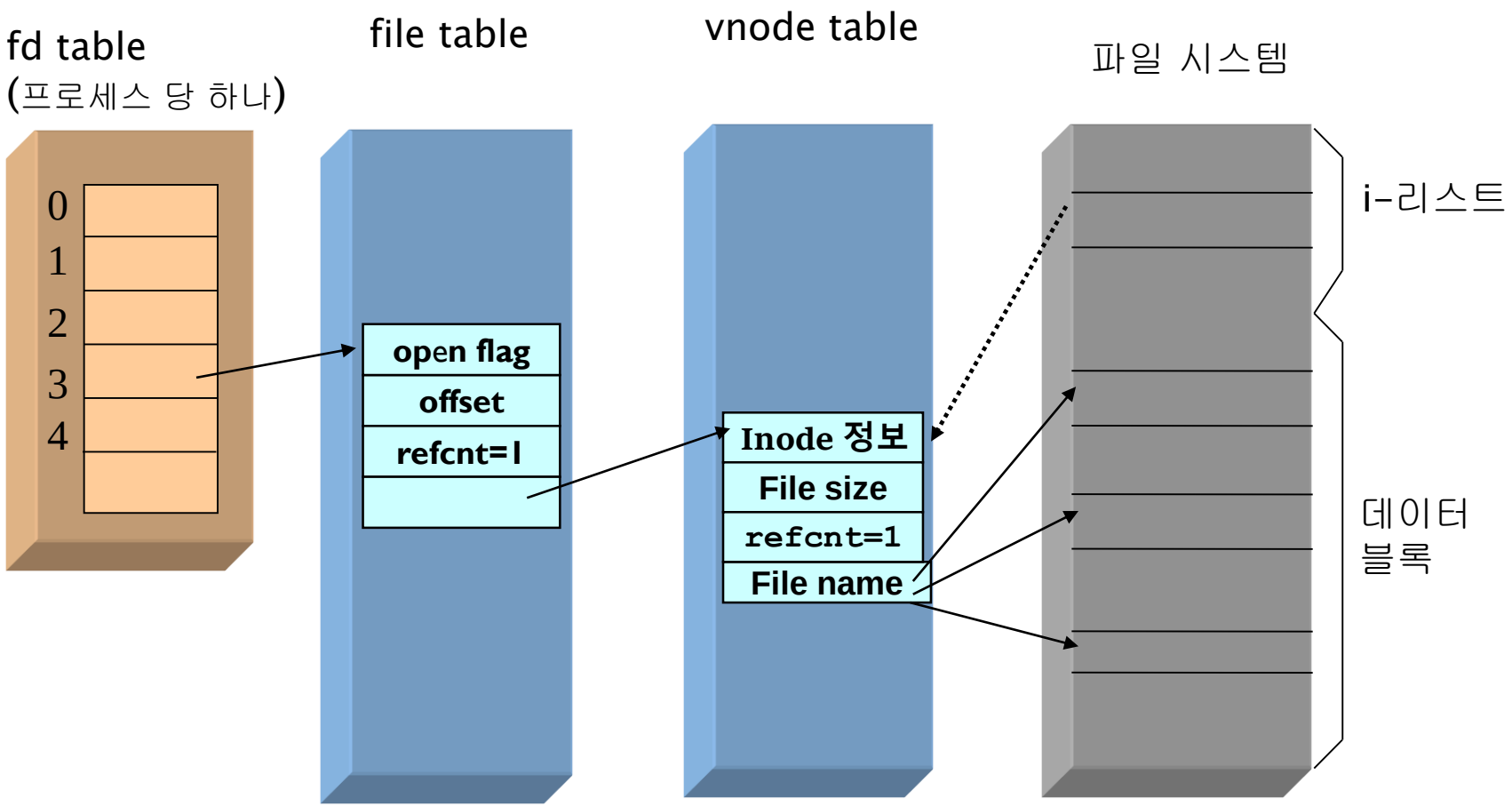
- 하나의 프로그램은 동시에 여러 개의 파일을 개방할 수 있다.
- 여러 개의 프로그램이 동시에 하나의 파일을 개방할 수 있다.
- 어떤 경우든 커널에 의해서 각 개방상태가 유일하게 식별되어 관리된다.

read/write pointer

- 개방된 파일 내에서 읽기 작업이나 쓰기 작업을 수행할 바이트 단위의 위치
- 특정 위치를 기준으로 한 상대적인 위치를 의미
  - 그래서 오프셋(offset)이라고 한다.
- 파일을 개방한 직후에 읽기/쓰기 포인터는 0
  - 파일의 첫 번째 바이트를 가리킨다.
  - 파일의 내용을 읽거나 파일에 새로운 데이터를 작성하면 그 만큼 증가한다.
- 파일 기술자마다 하나씩 존재
  - 서로 다른 프로그램이 동일한 파일을 개방해도 파일 기술자가 다르기 때문에 마찬가지로 서로 다른 읽기/쓰기 포인터를 가진다.
  - 즉, 서로의 작업이 상대에게 영향을 주지 않는다.

파일을 위한 커널 자료 구조

- fd = open("file", O_RDONLY);



파일 디스크립터 테이블(Fd Table)

- 프로세스 당 하나씩 갖는다.
- 파일 디스크립터 배열
  - 열린 파일 테이블의 엔트리를 가리킨다.
- 파일 디스크립터
  - 파일 디스크립터 배열의 인덱스
  - 열린 파일을 나타내는 번호
- FD Array의 index number
  - 최대 1024
    > getconf OPEN_MAX
  - 0, 1 and 2 are reserved for standard input, output, error respectively.

Global open file table

- 파일 테이블 (file table)
  - 커널 자료구조
  - **열려진 모든 파일 목록**
  - 열려진 파일 → 파일 테이블의 항목

- 파일 테이블 항목 (file table entry)
  - 파일 상태 플래그
    (read, write, append, sync, nonblocking,…)
  - 파일의 현재 위치 (current file offset)
  - i-node에 대한 포인터
  - (reference counter)

Vnode table

- Open 된 파일들의 i-node를 저장하는 테이블
  - 각 파일 시스템의 특화된 Inode 정보, 파일 이름, 사이즈 등을 커널 자료구조에서 관리
- Inode - 하드 디스크에 저장되어 있는 파일에 대한 자료구조
  - 한 파일에 하나의 i-node
  - 하나의 파일에 대한 정보 저장
    - 소유자, 크기
    - 파일이 위치한 장치
    - 파일 내용 디스크 블럭에 대한 포인터
- (reference counter)

파일을 위한 커널 자료 구조

- fd = open("file", O_RDONLY); //두 번 open