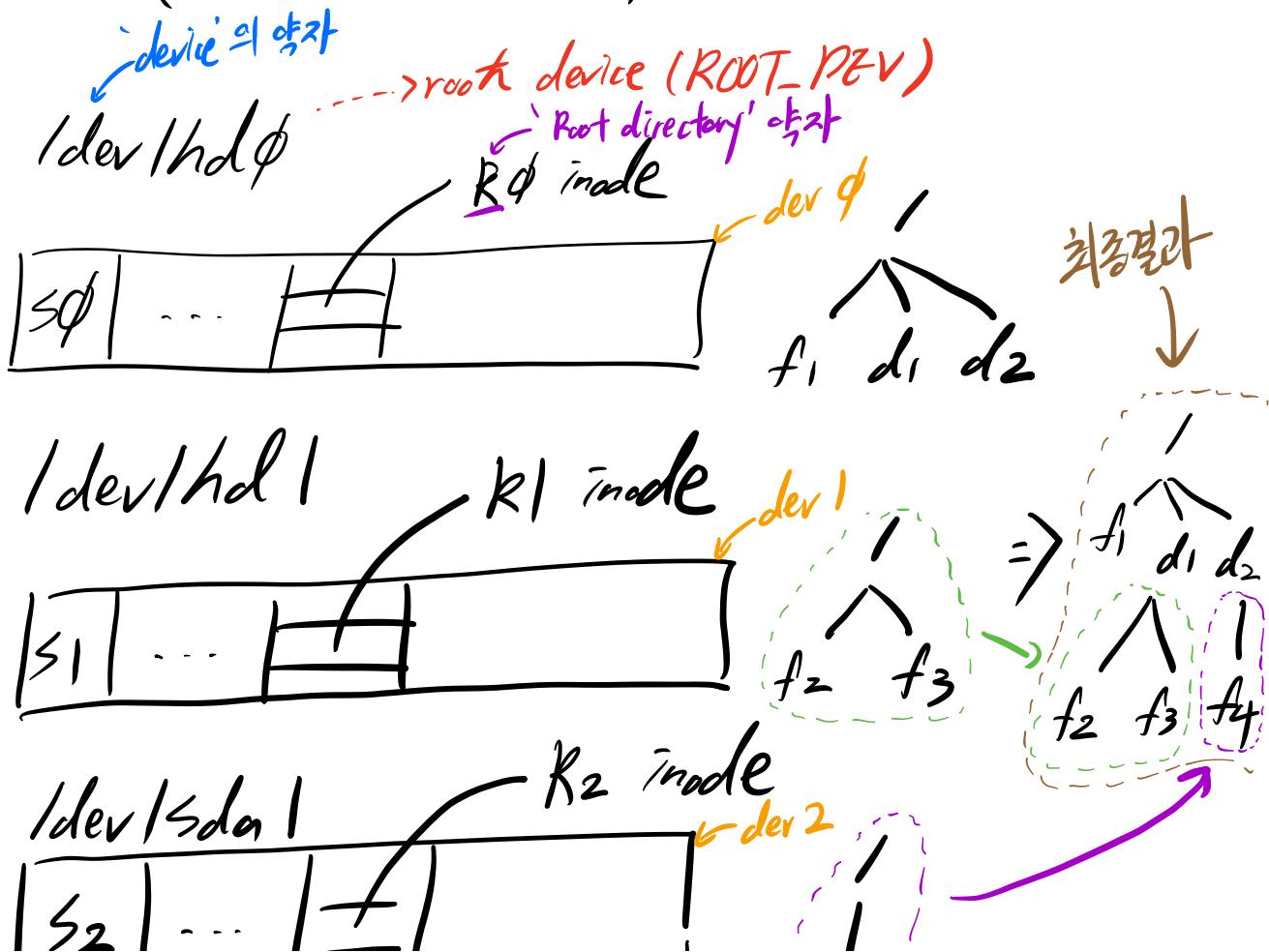


On memory file system

- on disk fs : disk on memory file system
- (on mem fs : memory on caching된 disk file system.
- multiple disk (Superblock, inode number, group descriptor  
을 갖는 각각 사용되는 어려운 데이터는 메모리로 caching된다.)  
disk slow
- solution (caching mounting)
- VFS (Virtual File System)



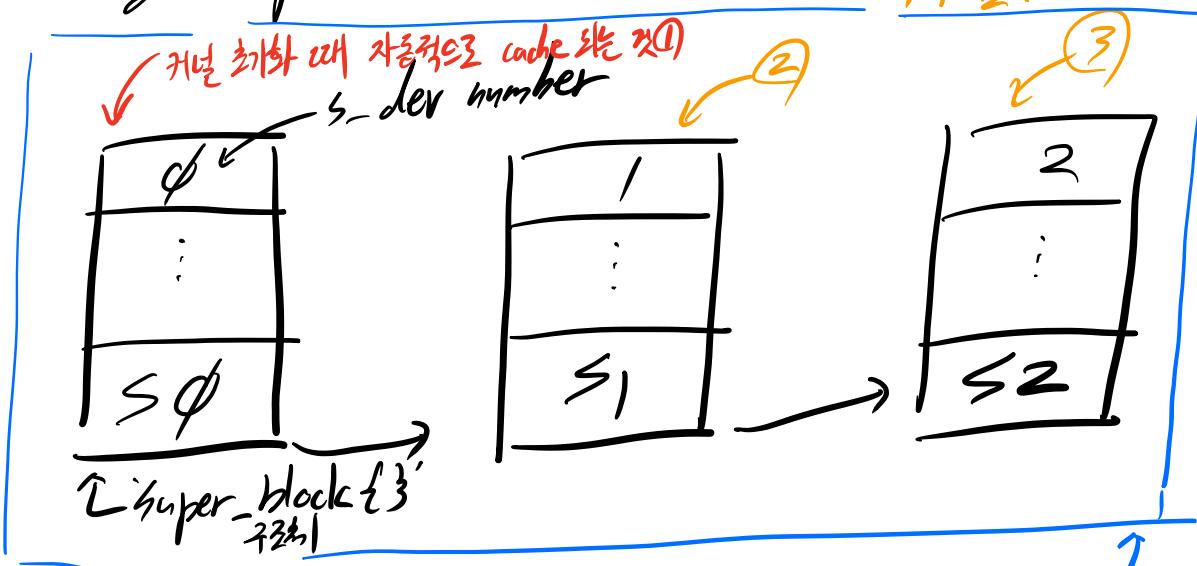


f4 특정 파일을 블록 단위로 차운 장치로 copy하는 과정.

### Memory<sup>3</sup> on-disk file system 정보를 caching 하는 과정

→ 즉, mounting 될 때 메모리상에서  
교환되는 모습을. -X ②③④ - dev1, dev2가  
mount 될 때 추가됨.

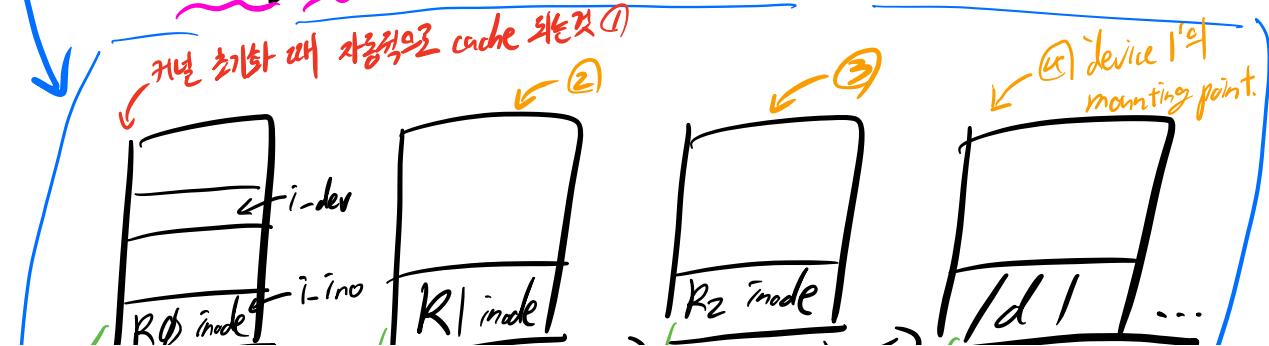
#### 1. caching superblock

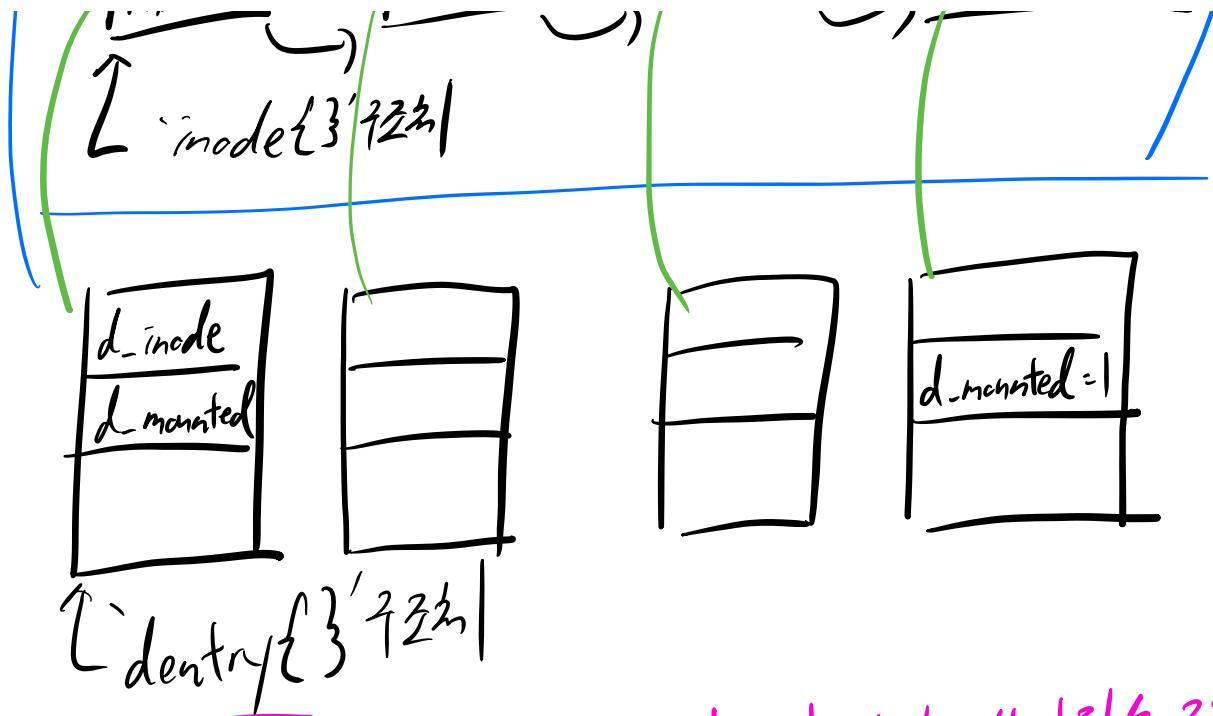


linked list'를 형성함.  
해당 linked list는  
'inode\_in\_use' pointers  
pointed됨.  
(2) 링크드 리스트  
현재 사용되는 file의  
inode를 caching  
하는 과정.

linked list'를  
형성함. 해당 linked list는  
super-blocks' pointers로 pointed  
됨.

#### 2. caching inode (inode number) caching(2)





① caching 된 디렉토리 흐름 | 각각 부여되는 구조

② dentry는 기본적으로 디렉토리 내의 파일, 즉 파일  
(및 하위 디렉토리) 이름을 저장한다.

③ 해당 디렉토리 파일의 inode number를 저장한다.

④ 해당 디렉토리가 mounting point라면,  
d\_mounted는 1 이상이다. ( $\Rightarrow d_{-}mounted > 0$ )

<커널 초기화 때, root device를 mount하는 과정>

start-kernel → kernel-init →

prepare-namespace → ① mount-root()

[root device의  
file system을 caching  
한다.]

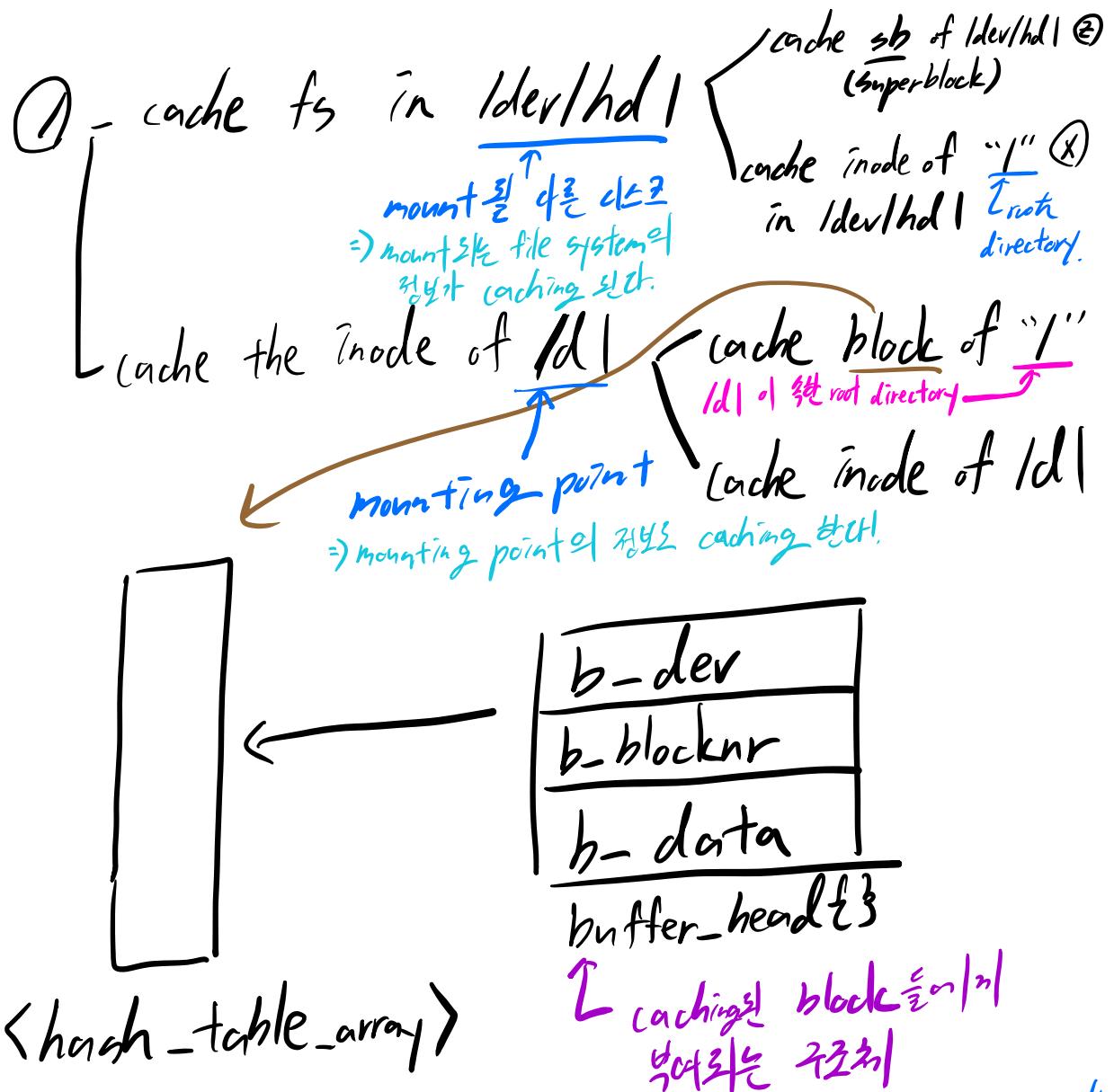
① mount 명령어 실행.

--- ② mount /dev/hd1 /d1

③ mount /dev/fd0 /d2

• mount -o loop /dev/hd1 /d1

mount 명령어를 시작하자.  
mount를 따른 대로 dk2  
→ mounting point.



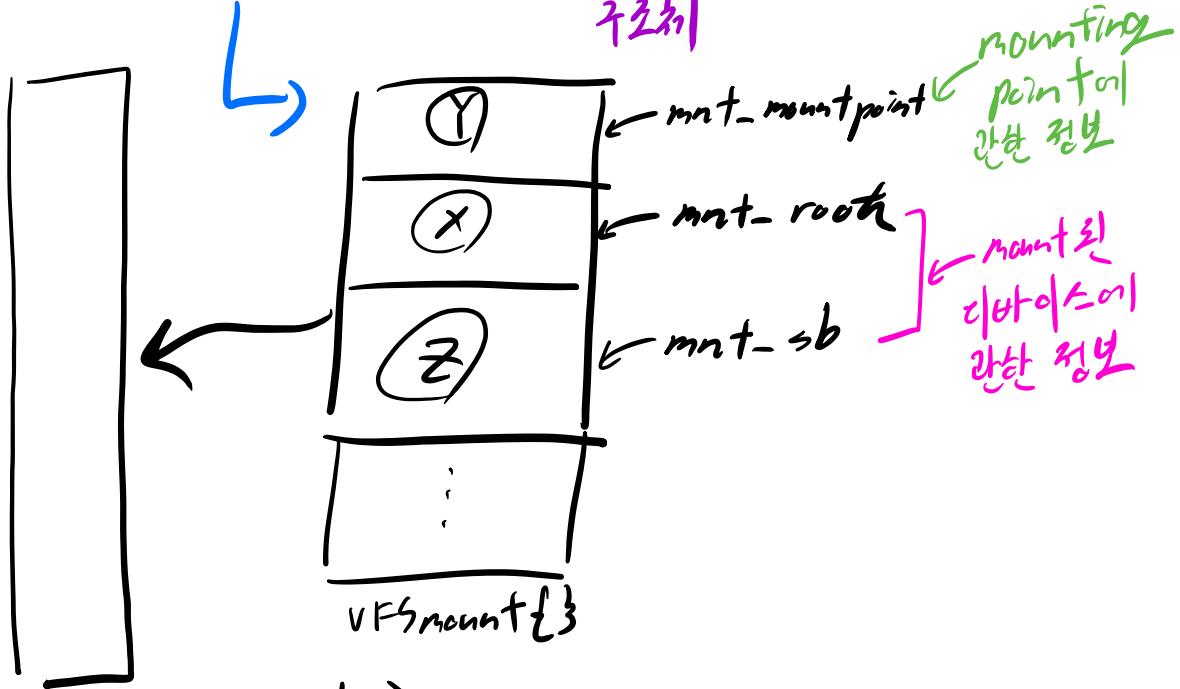
② connect / of /dev/hd1 to ld1 ② ③

①. d\_mounted ++  
 [Yon] 및 mount slot 있는지를 체크

... . 1 2 . — mounting point of 2nd

VFSmount { }

mount 되는 디바이스의 정보가 담긴  
구조체



mount\_table

[VFSmount 구조체]들이 해당 해시 테이블에 저장된다.

## process & file system

## file descriptor

- fd table
  - file table
  - inode table
  - open, read, write, close, seek, dup, link
  - chroot, chdir

전부 m[33(0)] caching  
되어 있을

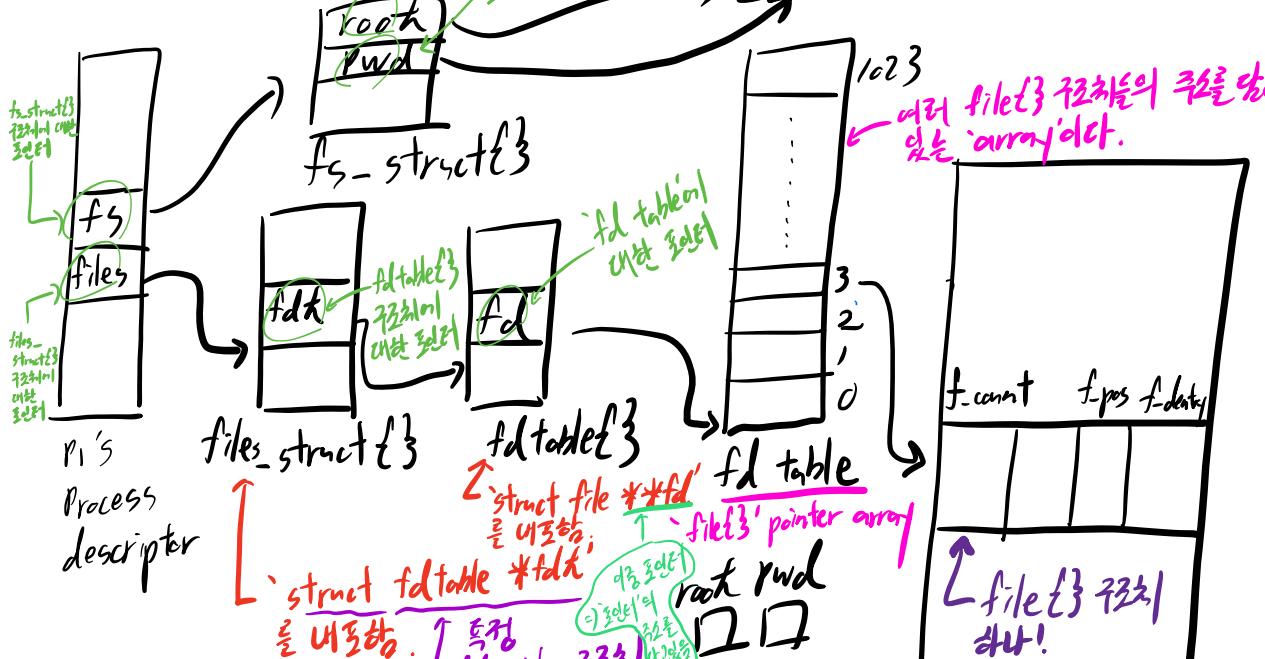
전부 미리 캐싱되어 있을

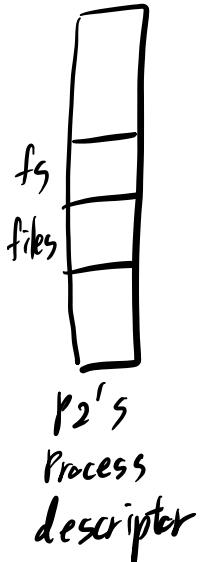
- absolute path
- relative path

<root directory> dentry 3.2.2

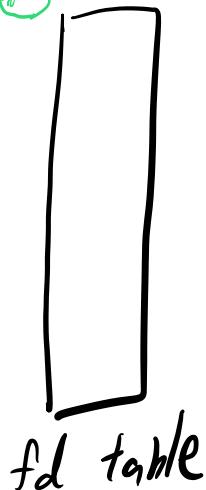
현재 작업중인 directory의  
directory 구조를 찾고 있다.

1023 ← 여러 file[3] 구조체들의 주소를 담고  
 있는 array이다.





L fd table T271  
 번수의 주소값이  
 'fd' 링버 번수에  
 저장됨.



file table  
 (linked list of  
 'file{}')

It is for  
 each opened file  
 · 열려진 파일에 대한  
 정보가 들어있음.

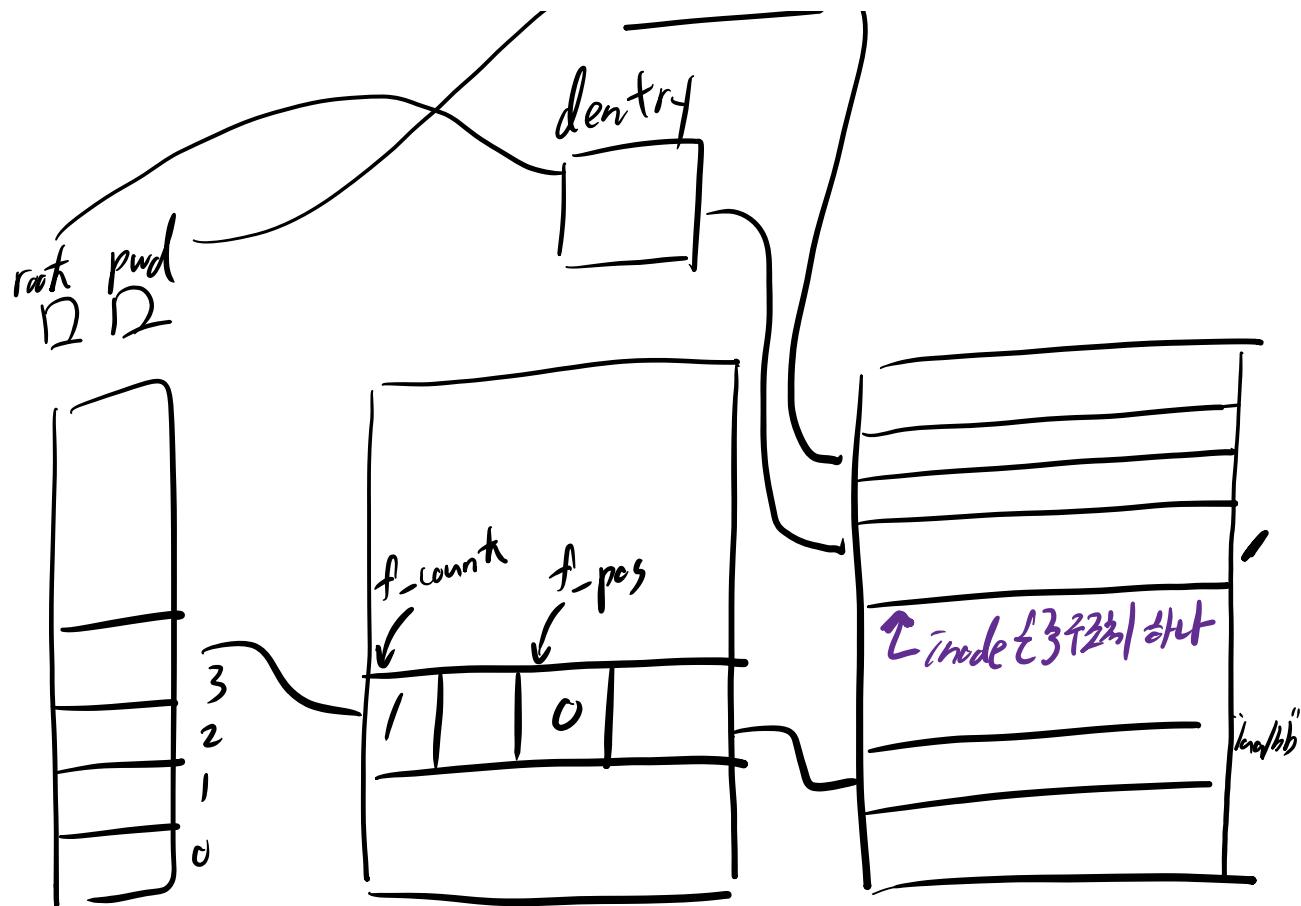
Struct file {

- f\_list
- f\_dentry
- f\_op
- f\_pos
- f\_count

:

}





fd table      file table  
 (linked list `file{}`)      `inode-in-use`  
 (linked list `inode{}`)

`i_ino, i_order, i_count`  
 맴버 변수를 가지고 있다.

< X 각 프로세스마다 root, pwd, fd table 을  
 가지고 있다!! >

< 특정 프로세스가  
 열고 있는 file에 대한  
`file{}` 구조체의 주소를  
 가지고 있는 array임! >

P1:

•  $x = \text{open}("aa/bb", \dots)$  ??

- ① find inode of 'aa/bb'
- ② cache it to the Inode into memory
- ③ allocate file{} in file table
- ④ find an empty place in fd table and allocate that empty place.
- ⑤ chaining
- ⑥ return fd

•  $x = \text{read}(y, \dots)$

- ① go to Inode y

- ② read  $n$  bytes
- ③ store in buf
- ④ increase f-pos