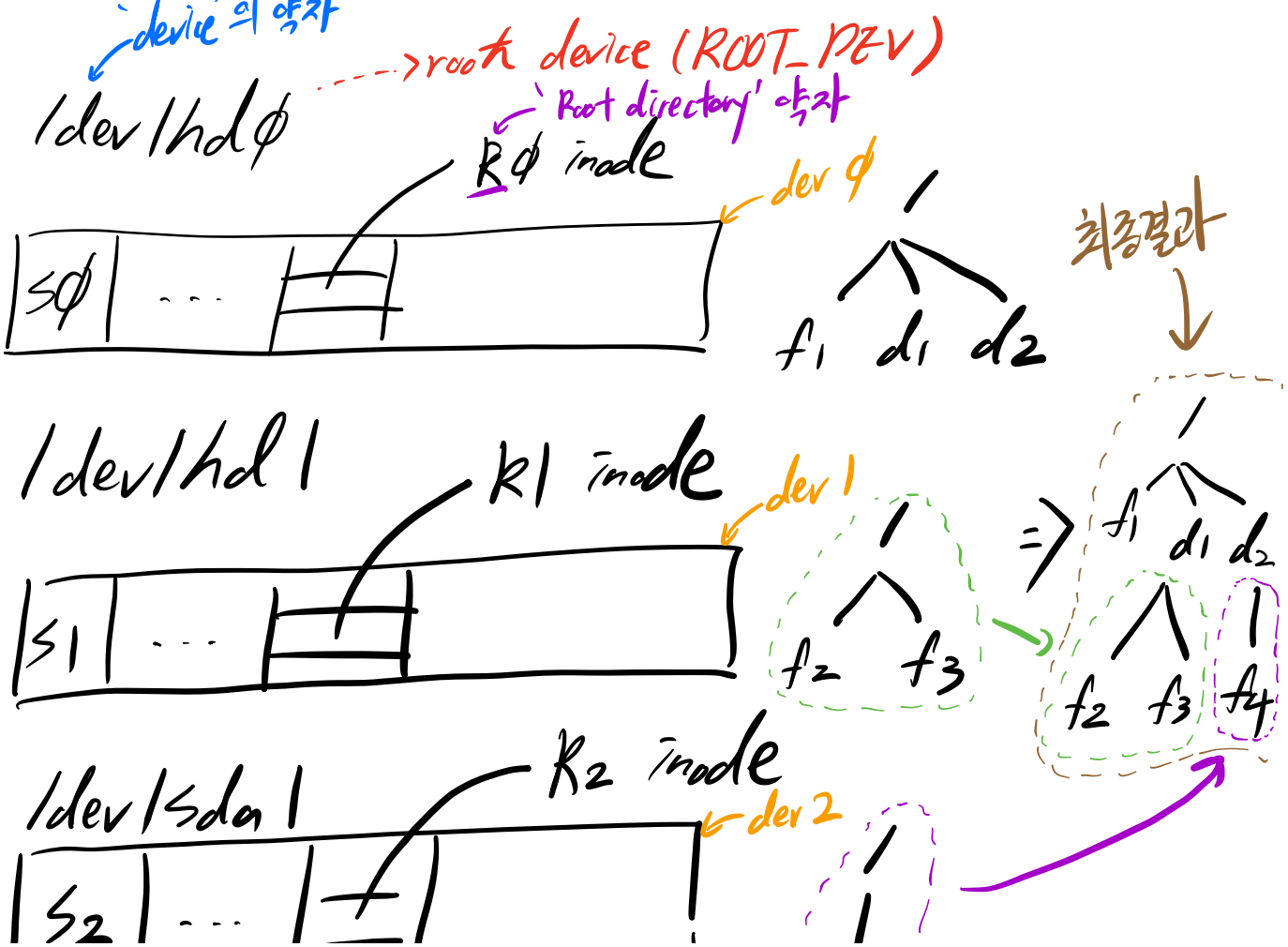
· On memory file system

- on disk fs : disk에 내포된 file system
- on mem fs : memory에 caching된 disk file system.
  (superblock, inode number, group descriptor 들과 같은 자주 사용되어지는 data들만 memory로 caching 된다.)

- multiple disk
- disk slow

- solution (caching, mounting)

- VFS (Virtual File System)

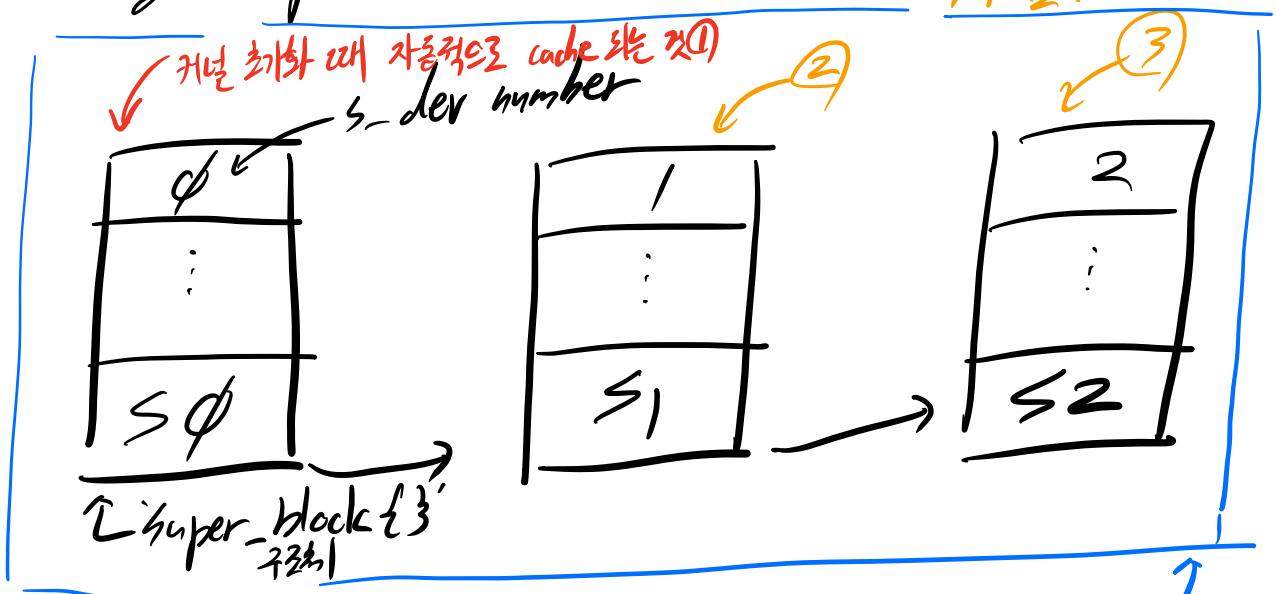


f4 특정 저장장치 내 들어있는 데이터를 속도가 좀 더 빠른 저장장치로 copy 하는 과정.

Memory로 on-disk file system 정보를 caching 하는 과정

=> 즉, mounting 될 때 메모리상에서 캐싱되는 모습을. -X ②③④ - dev 1, dev 2가 mount 될 때 추가됨.

1. caching superblock

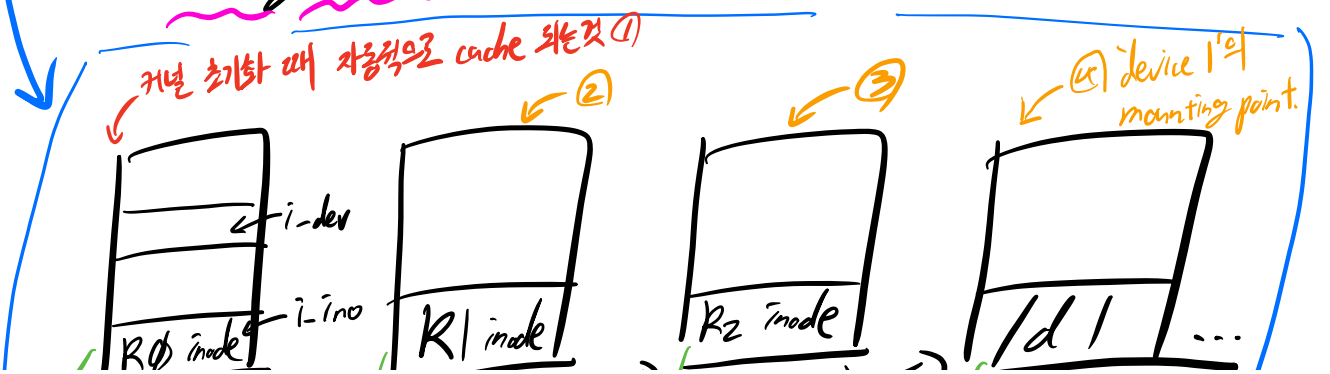


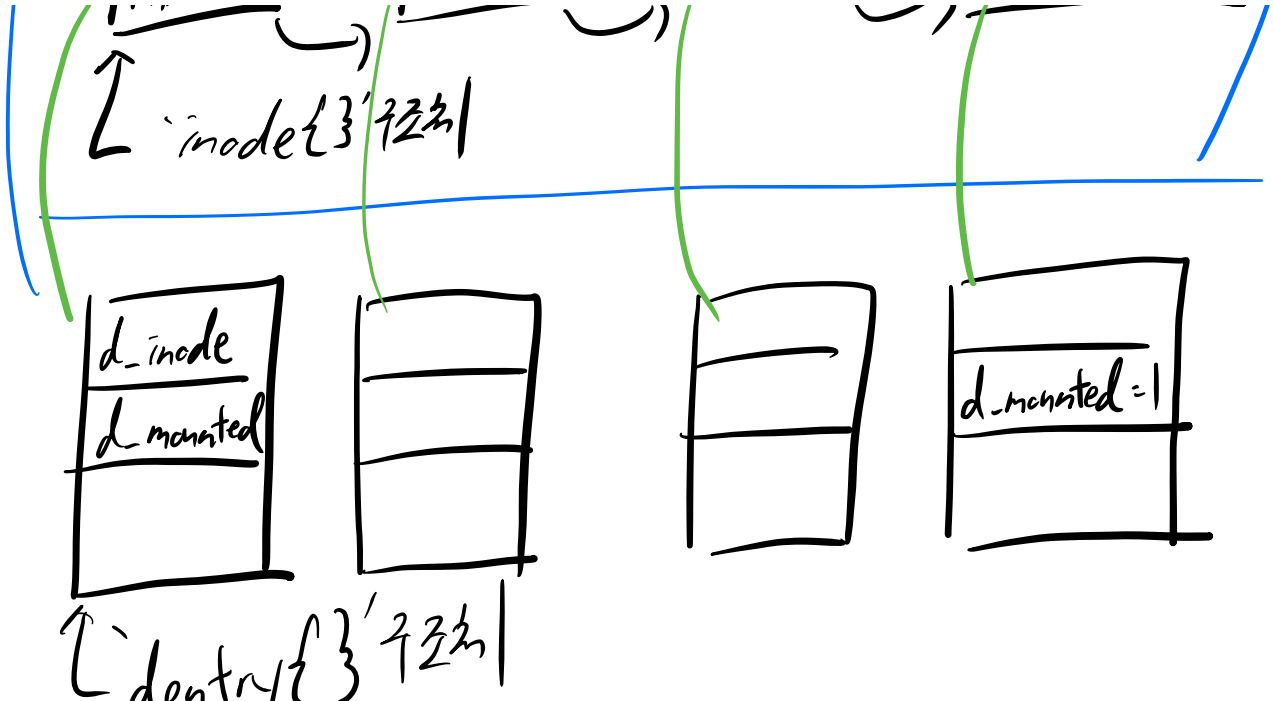
'linked list'를 형성함. 해당 linked list는 'super_blocks' 포인터로 pointed 됨.

'linked list'를 형성함. 해당 linked list는 'inode_in_use' 포인터로 pointed 됨.

<리눅스 내, 현재 사용되는 file의 inode들만 caching 한다.>

2. caching inode (inode number가 caching 될)





- caching된 디렉토리들이 각각 복여되는 구조체
  - ② dentry는 기본적으로 디렉토리 내의 오소를, 즉 파일 (및 하위 디렉토리) 이름을 저장한다.
  - ③ 해당 디렉토리 파일의 inode number도 저장한다.
  - ④ 해당 디렉토리가 mounting point라면, d_mounted는 1 이상이다. (⇒ d_mounted > 0)

< 커널 초기화 때, root device가 mount되는 과정 >

start_kernel → kernel_init →

prepare_namespace → ① mount_root()

↑ <root device의 file system을 caching 한다.>

㉮ mount 명령어 실행.

... ② mount /dev/hd1 /d1

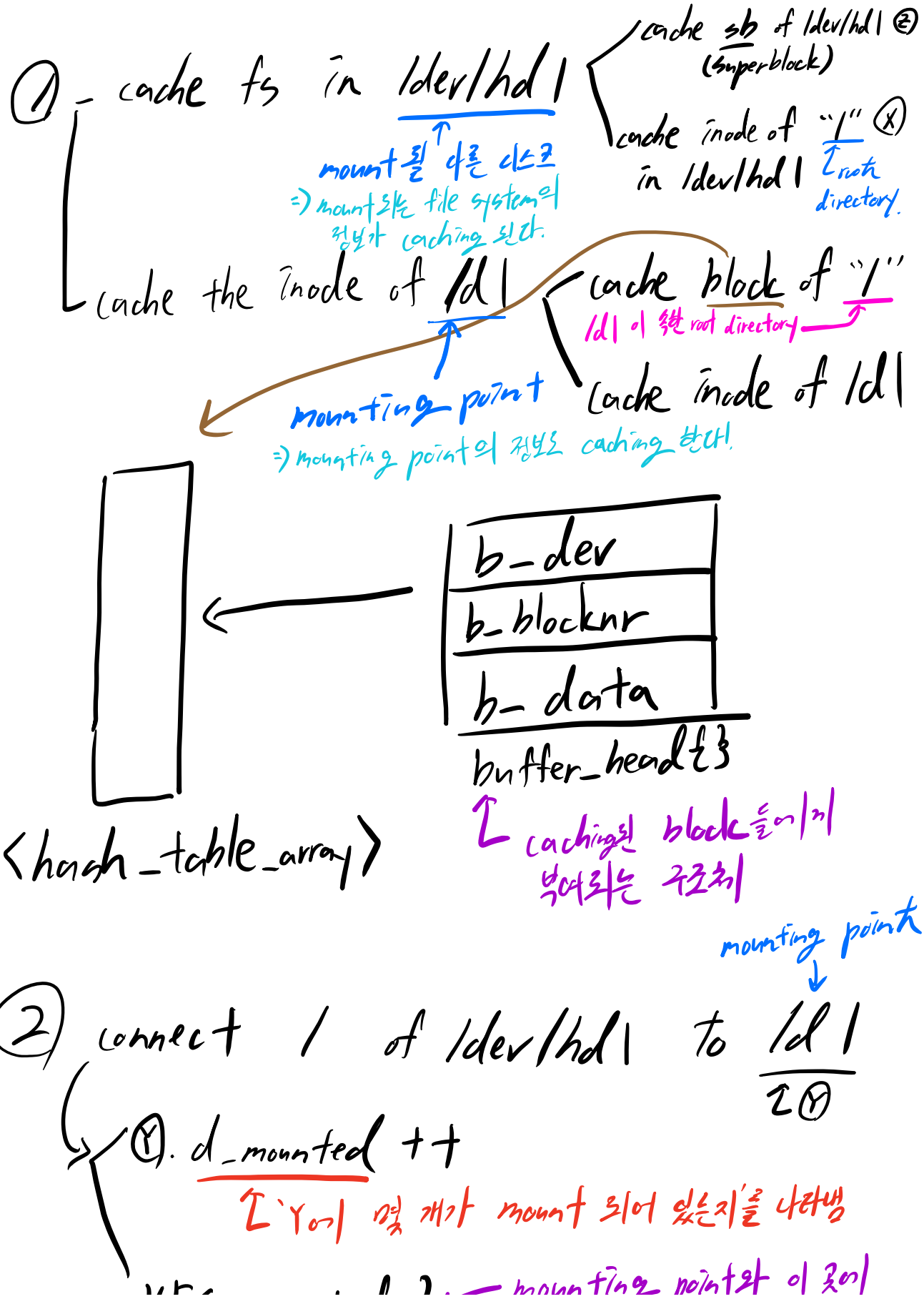
③ mount /dev/fd0 /d2

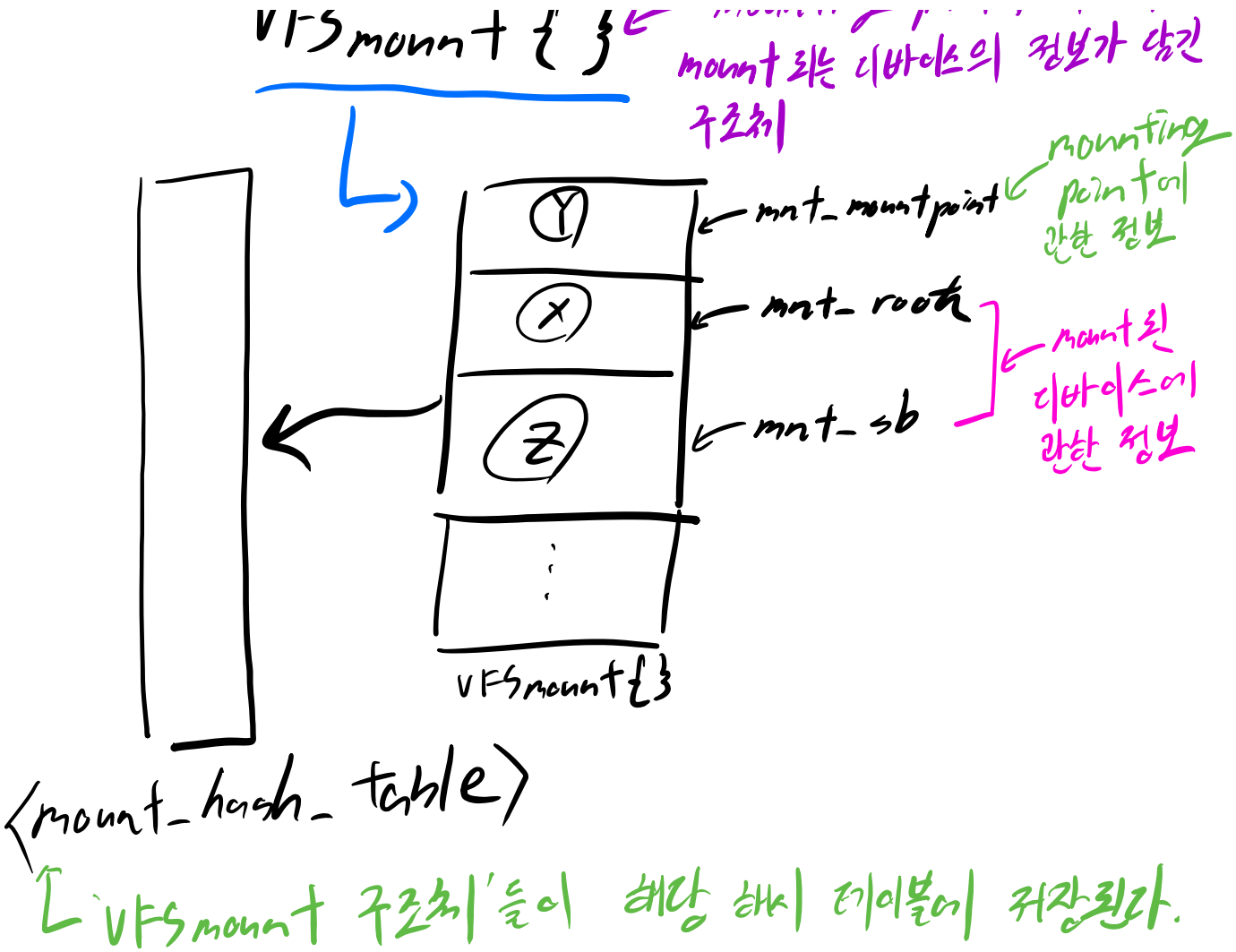
mount을 시작하라.

mount할 대상 디스크

mounting point.

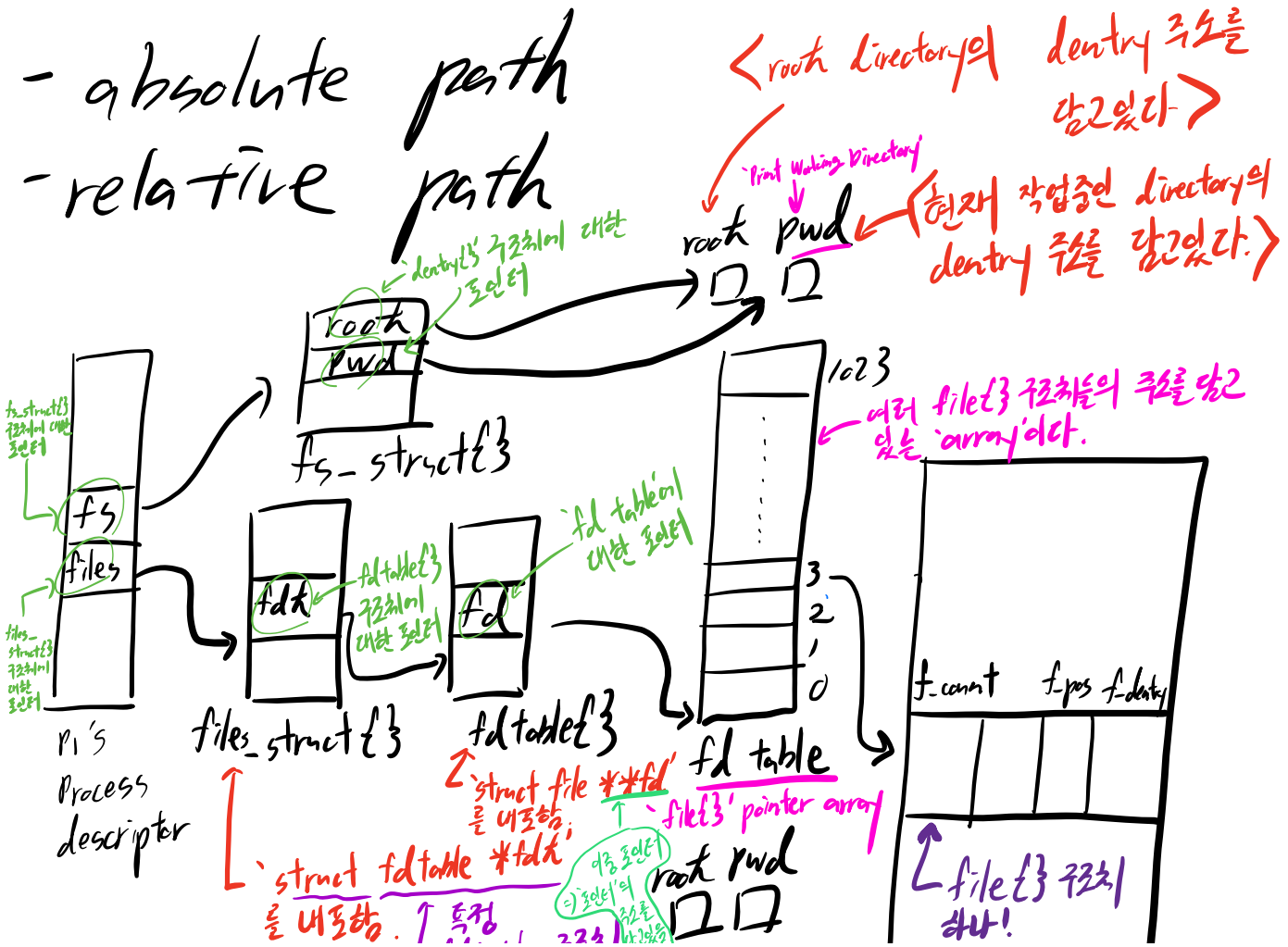
. mount -o loop /dev/hd1 /d1

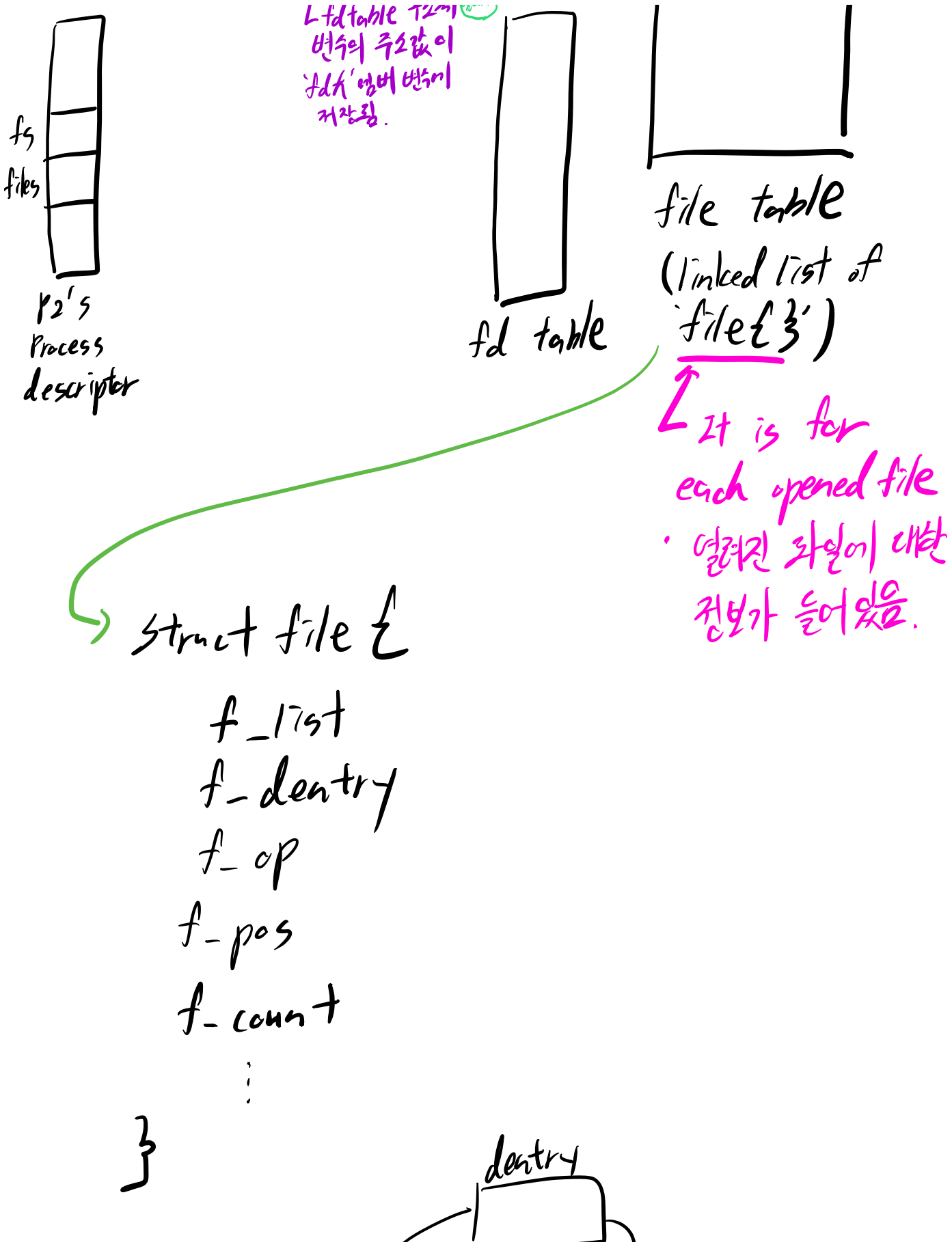

① cache fs in /dev/hd1
cache sb of /dev/hd1 ②
(superblock)
cache inode of "/" ⓧ
in /dev/hd1
↑ root directory
↑ mount될 대상 디스크
=) mount되는 file system의 정보가 caching 된다.
cache the inode of /d1
cache block of "/"
/d1 의 root directory
cache inode of /d1
mounting point
=) mounting point의 정보도 caching 됨.
b_dev
b_blocknr
b_data
buffer_head {3
<hash_table_array>
↑ caching된 block들을 관리
하기 위한 구조체
mounting point
② connect / of /dev/hd1 to /d1
ⓨ
ⓨ. d_mounted ++
↑ 'ㅛ'이 몇 개가 mount 되어 있는지를 나타냄
— mounting point와 이 곳의

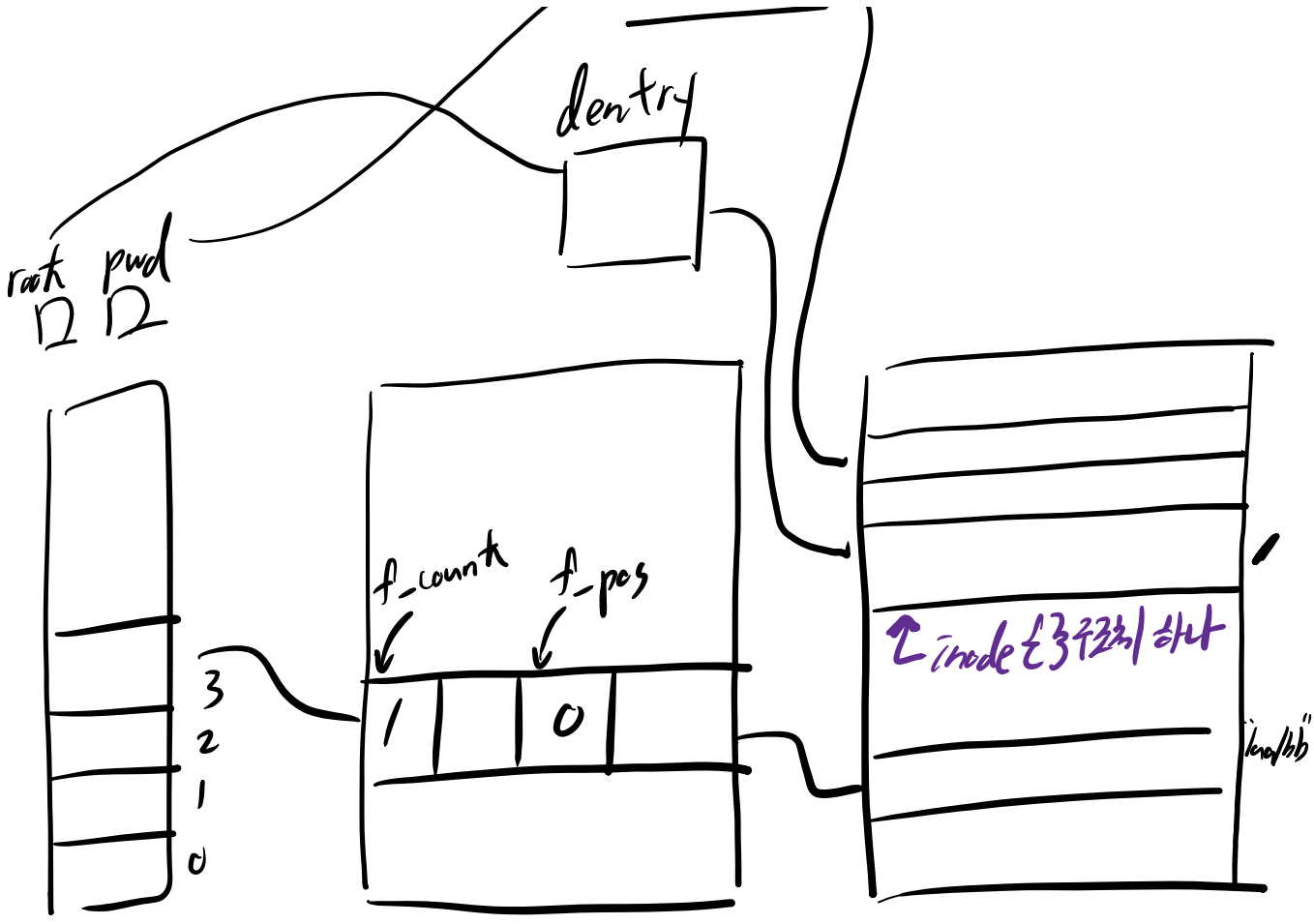


└ VFSmount 구조체'들'이 해당 해시 테이블에 저장된다.

- process & file system
  - fd table ← file descriptor
  - file table
  - inode table
  ← 전부 메모리에 caching 되어 있음
- open, read, write, close, lseek, dup, link
- chroot, chdir
- absolute path
- relative path




fs
files
P2's
Process
descriptor
fd table
file table
(linked list of
'file{}')
struct file {
f_list
f_dentry
f_op
f_pos
f_count
:
}
dentry



i_ino, i_rdev, i_count
멤버 변수를 가지고 있다.

< X: 각 프로세스마다 root, pwd, fd table을 가지고 있다!! >

< 특정 프로세스가 열고있는 file에 대한 'file{}' 구조체의 주소를 담고있는 array임! >

P1:

- x = open("/aa/bb", ...)

  ① find inode of "/aa/bb" ???

  ② cache into the Inode into memory

  ③ allocate file[3] in file table

  ④ find an empty place in fd table and allocate that empty place.

  ⑤ chaining

  ⑥ return fd

- x = read(y, ...)

  ① go to inode y

" '

② read n bytes

③ store in buf

④ increase f-pos