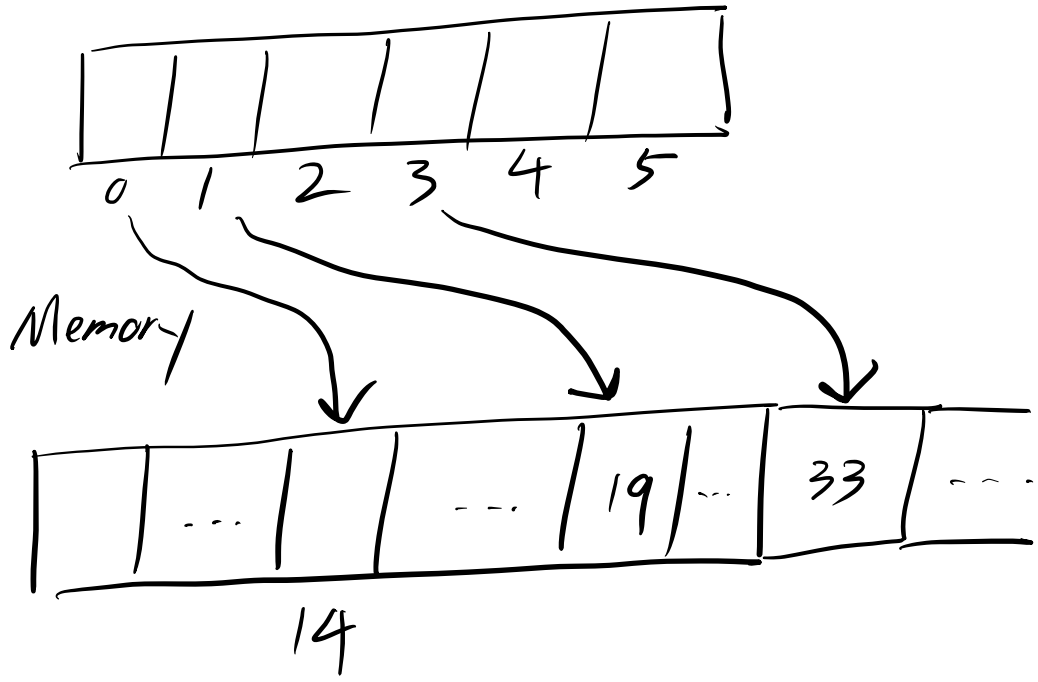
Memory Management

- goal of MM
- fs vs MM
- paging
- process image
- 3 problems of paging

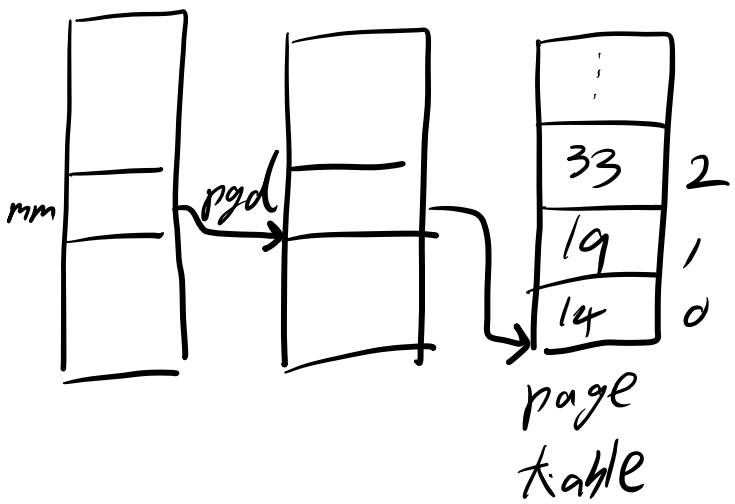
<goal of MM>

- store processes in memory efficently
- solution (paging) : process = $\sum$ page (1page = 1page frame = 4KB)
  - Memory = $\sum$ page frame.
  - store pages in page frames
  - remember page location in task_struct → mm → pg
    ↑ process 마다 가지고 있는 것

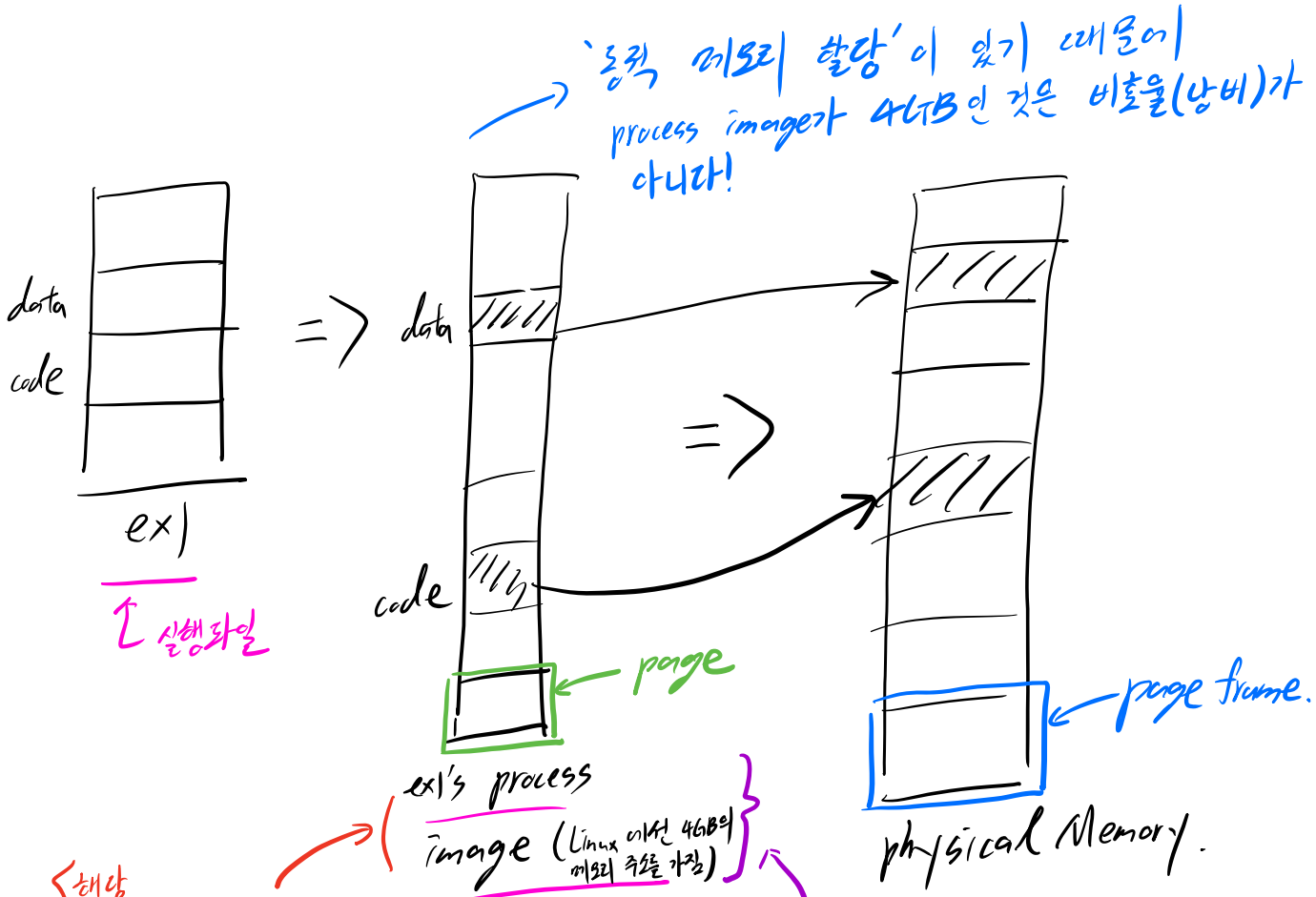
ex) P_1



< P1's task_struct >



ex) gcc -o ex1 ex1.c



<해당 process image가 하드디스크 내 Virtual Memory에 적재된다.>

X pointer로 알 수 있는 변수의 주소값은 가상메모리 내의 주소이다!!!

리눅스에선 하나의 process image가 4GB여기 때문에, 4GB 미만의 physical Memory에는 해당 process image가 못들어온다!

※ file << process

Disk >> Memory

< 3 problems of paging >

- process too big
- page table too big
- address computing time getting slow.

< solution for process too big >

- demand paging
  - Virtual Memory
  - page fault (INT 14 → do_page_fault())
  - locality of reference (참조의 지역성)
    - 시간 지역성
    - 공간 지역성
    - 순차 지역성

< • 현재 필요한 페이지가 physical Memory에 부재한 경우'를 `page fault'라 한다.
• `page fault'가 발생하면, 해당 페이지를 가상 메모리에서 찾아야 한다.
• 운영체제가 page fault를 해결하는 과정을 `demand Paging'이라 한다. >

〈 중간 정리 〉

- 실행파일(.exe)는 '프로그램 파일'이다.
- 프로그램은 명령어(코드)와 정적 데이터의 묶음이다.
  (동적 데이터가 직접적으로 들어있지 않다. '명령어'형태로 존재하고 있을 것이다.)
- 실행파일(프로그램)은 크기가 작다. (몇 KB 밖이 안된다. 동적 데이터를 가지고 있지 않기 때문인 것 같다.)
- 어떤 실행파일이 실행되면, 'process image (4GB)'가 디스크 내에 생성된다.
  ↳ 디스크 내 가상메모리 공간에 process image가 저장됨.
- process image는 Code Section, heap Section, stack Section, Data Section으로 나뉘어져 있다.
- 실행파일에 있었던 요소(명령어, 정적 변수, 전역변수)들이 Process Image의 적합한 섹션에 저장된다. (동적 변수는 해당 변수를 생성하는 명령어가 실행된 후에, 그제서야 Process Image의 heap 영역에 저장된다.)
- Process Image 내 저장되어 있는 요소들 중, 실행함에 있어서 현재 필요한 요소들만 physical Memory로 이동된다.
  ↳ demanding Page.

<goal of fs>

- store files in Disk efficently
- solution :
  - file = Σ Block
  - Disk = Σ Block
  - store file blocks in disk blocks
  - remember block location in inode
    → i_block[]

file 마다 가지고 있는것

- 3 problems of paging

1. process too big
2. page table too big
3. address mapping getting to slow.

<Solution for "process too big">

- process = Σ page
- store only active pages in memory frame.
- remember frame location in map table

< Solution for page table too big >

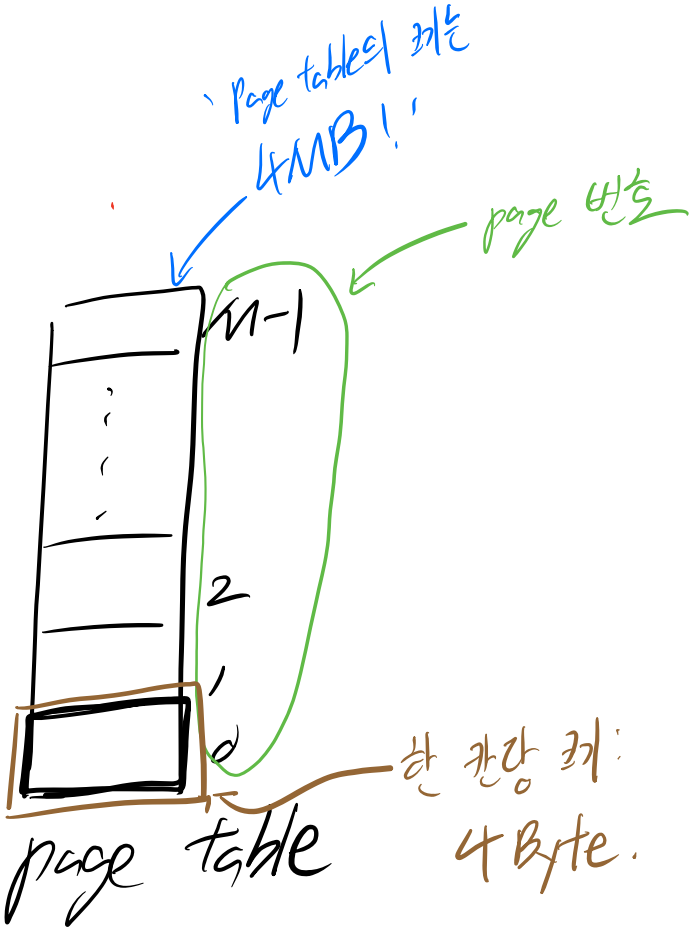
- page table = Σ dir
- store only active dirs. In mem frame
- remember fr location In dir table

↳ 해당 'dir table'도 physical memory에 넣어 야함.

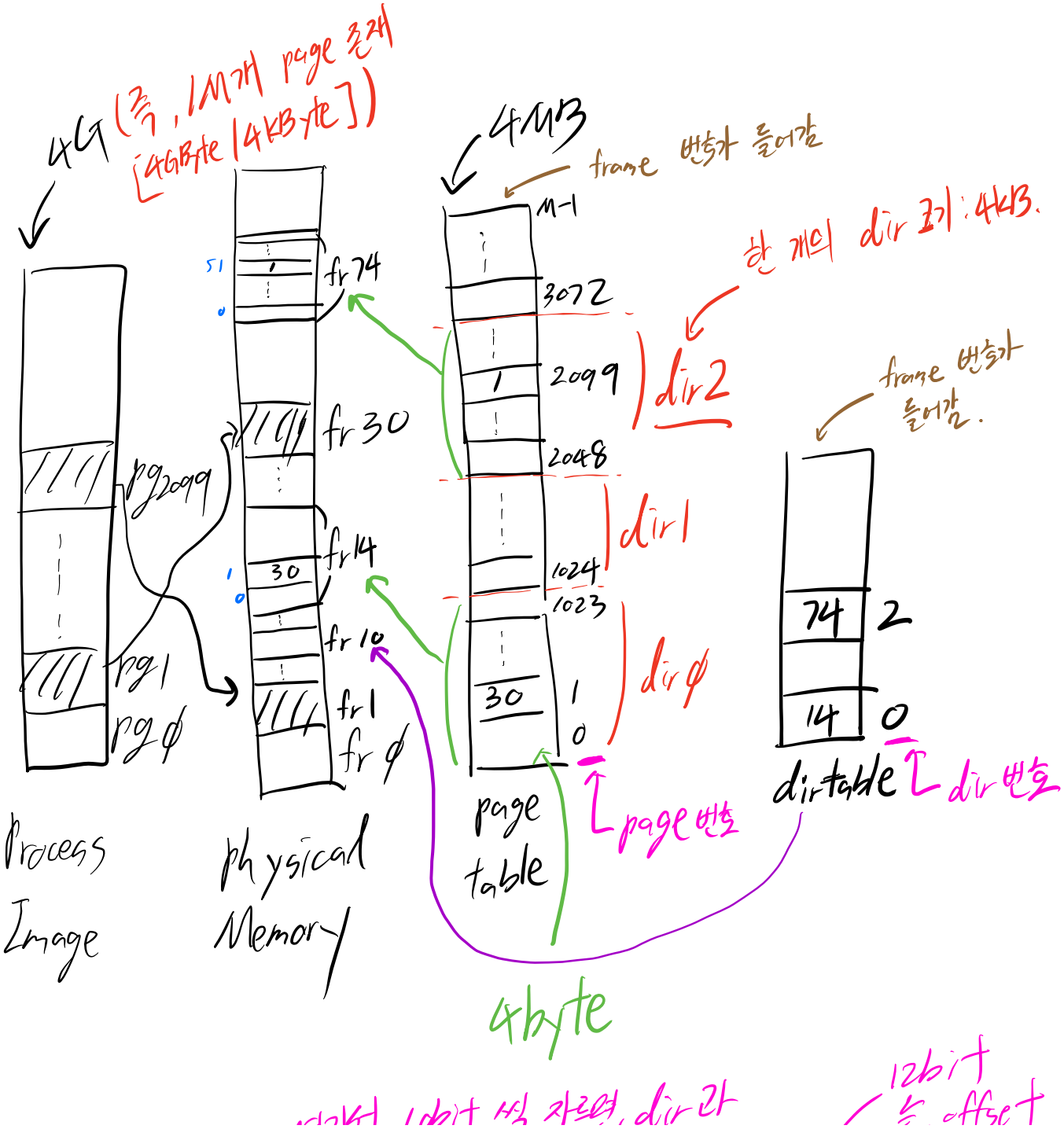
~~page~~의 크기는 4K! => 4K를 페이지 한 개의 크기(4KByte)로 나눠면, Process Image가 대표하는 총 페이지 개수는 1M (1백만) 개이다.

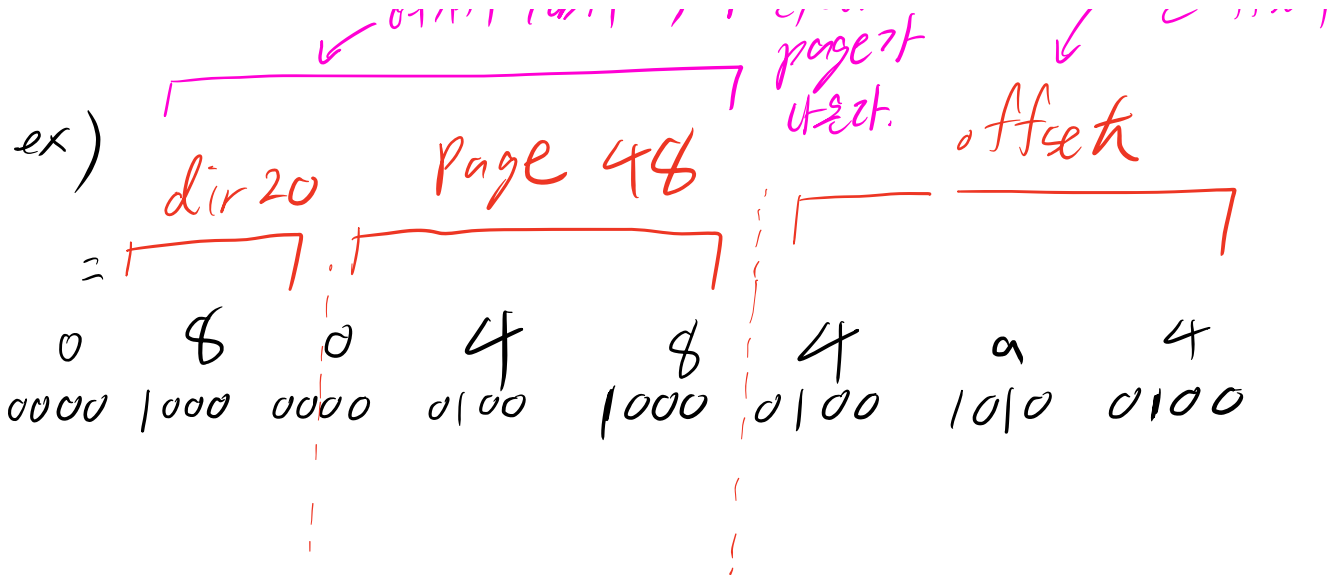
process Image

process Image



page table


4G (즉, 1M개 page 존재 [4GByte/4KByte])
4MB
frame 번호가 들어감
M-1
3072
2099
2048
1024
1023
30
0
1
한 개의 dir 크기: 4KB.
dir2
dir1
dir0
frame 번호가 들어감.
74
14
2
0
dirtable
dir 번호
page 번호
pg2099
pg1
pg0
fr74
fr30
fr14
fr10
fr1
fr0
51
30
1
0
Process Image
physical Memory
page table
4byte
12bit 는 offset


page가 나온다.
ex)
dir 20
Page 48
offset
=
0 8 0 4 8 4 a 4
0000 1000 0000 0100 1000 0100 1010 0100

offset: distance