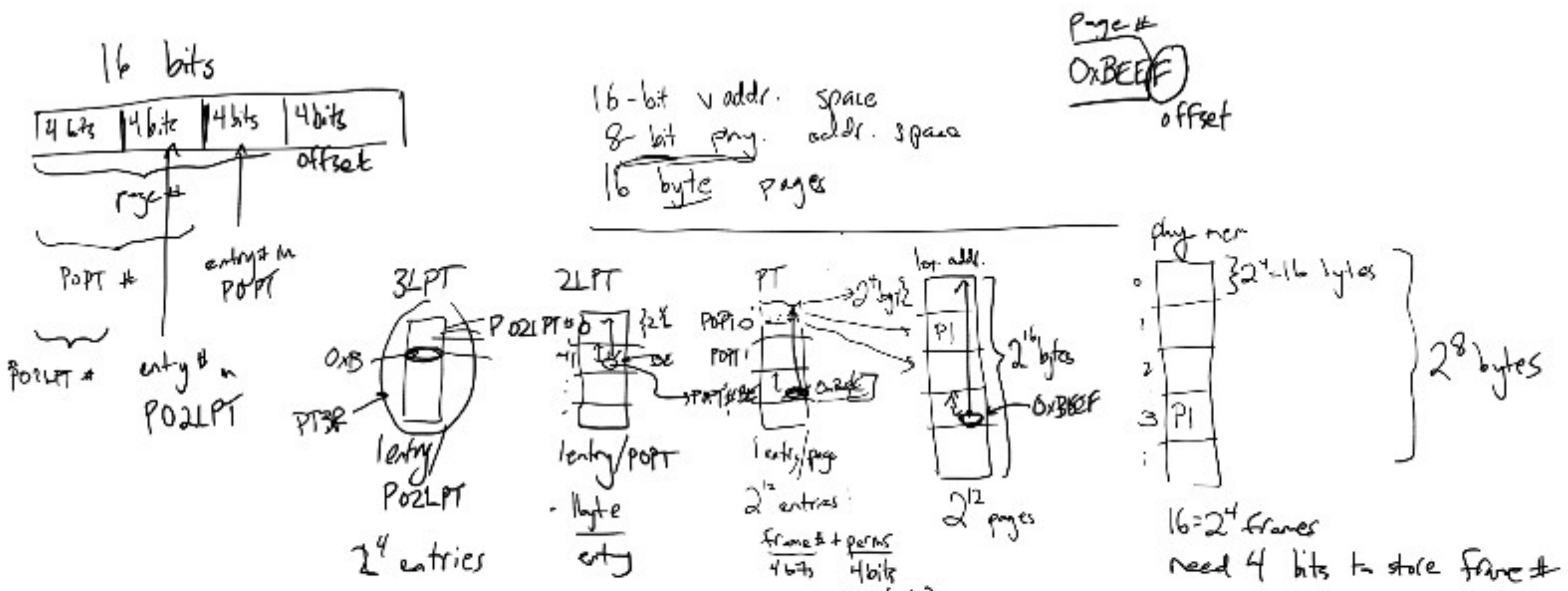
Lecture 18: Free space & recovery

- Managing free space
- FS consistency
- Journaling
- (review, quiz)



16-bit v addr. space
8-bit phy. addr. space
16 byte pages

2^4 entries

$= 2^4$ bytes

$= 1$ page

$= 2^8$ bytes.

$= 2^8 b \cdot \frac{1 \text{ P02LPT}}{2^4 b}$

$= 2^4$ pages in 2LPT

$\frac{8 \text{ bits}}{\text{entry}} \cdot \frac{\text{byte}}{8 \text{ bits}} = \frac{1 \text{ byte}}{\text{entry}}$.

$\text{PT} = 2^{12}$ bytes

$= 2^{12} \text{ bytes} \cdot \frac{1 \text{ P0PT}}{2^4 \text{ bytes}} = 2^8 \text{ P0PT}$

Entry 0xB of TLPT: $0x\overset{\uparrow}{D}\overset{\uparrow}{2}$ perms frame #

$D: \overset{rwxv}{1101}$, (13) 8+4+1

Entry 0xE of 2LPT (in frame #2): $0x\overset{\text{perms}}{D}\overset{\text{frame}}{A}$

∴ page # BE of PT in frame A

Entry 0xE of PT (in frame A): $0x\overset{\text{perm}}{5}\overset{\text{frame \#}}{3}$

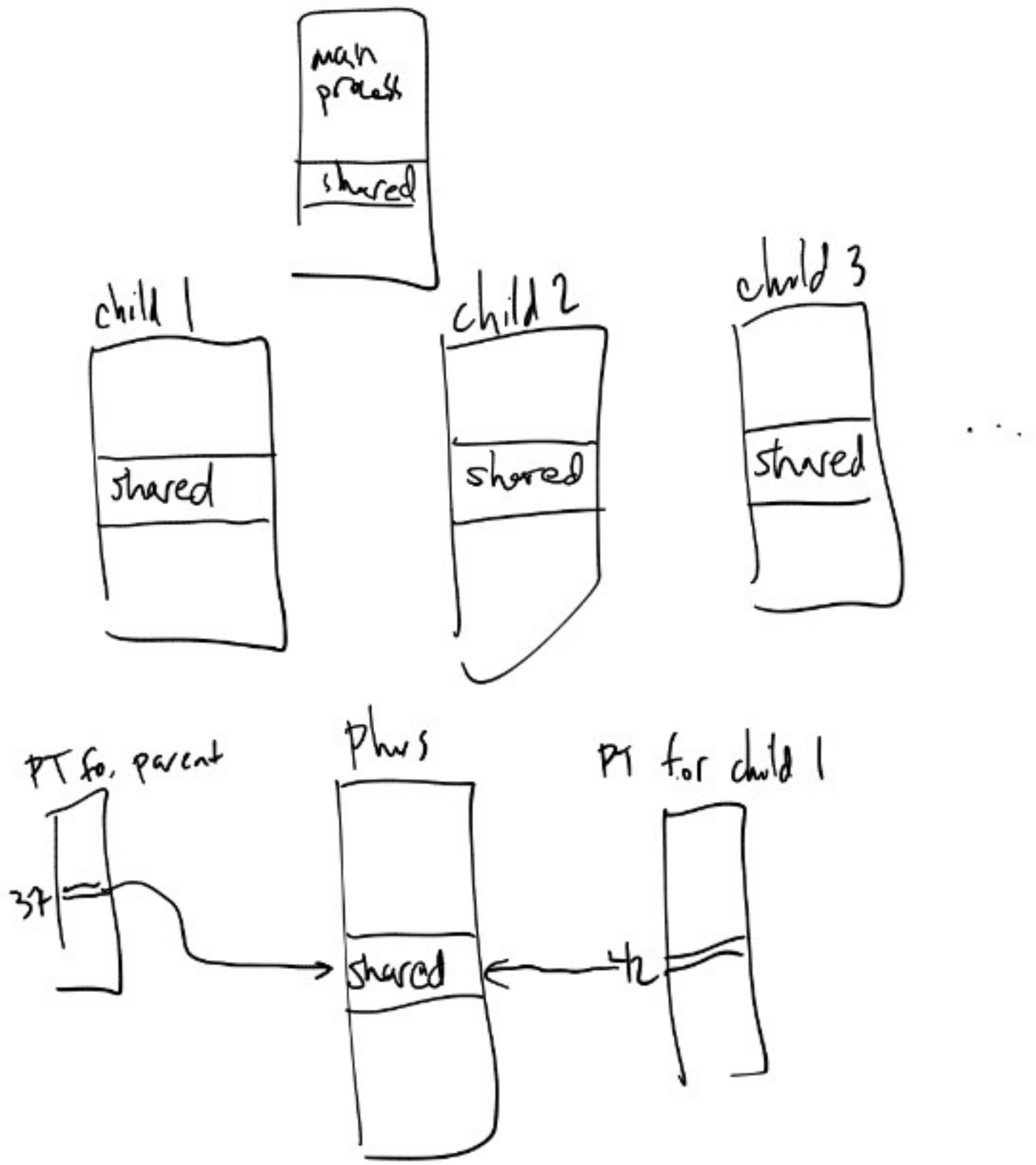
$0x5: \overset{rwxv}{0101}$ writable valid.

address 0xBEEF is present (valid) ∴ no page fault

but page BEE is not readable ∴ can't increment

$\boxed{\text{segmentation fault.}}$

4(b)

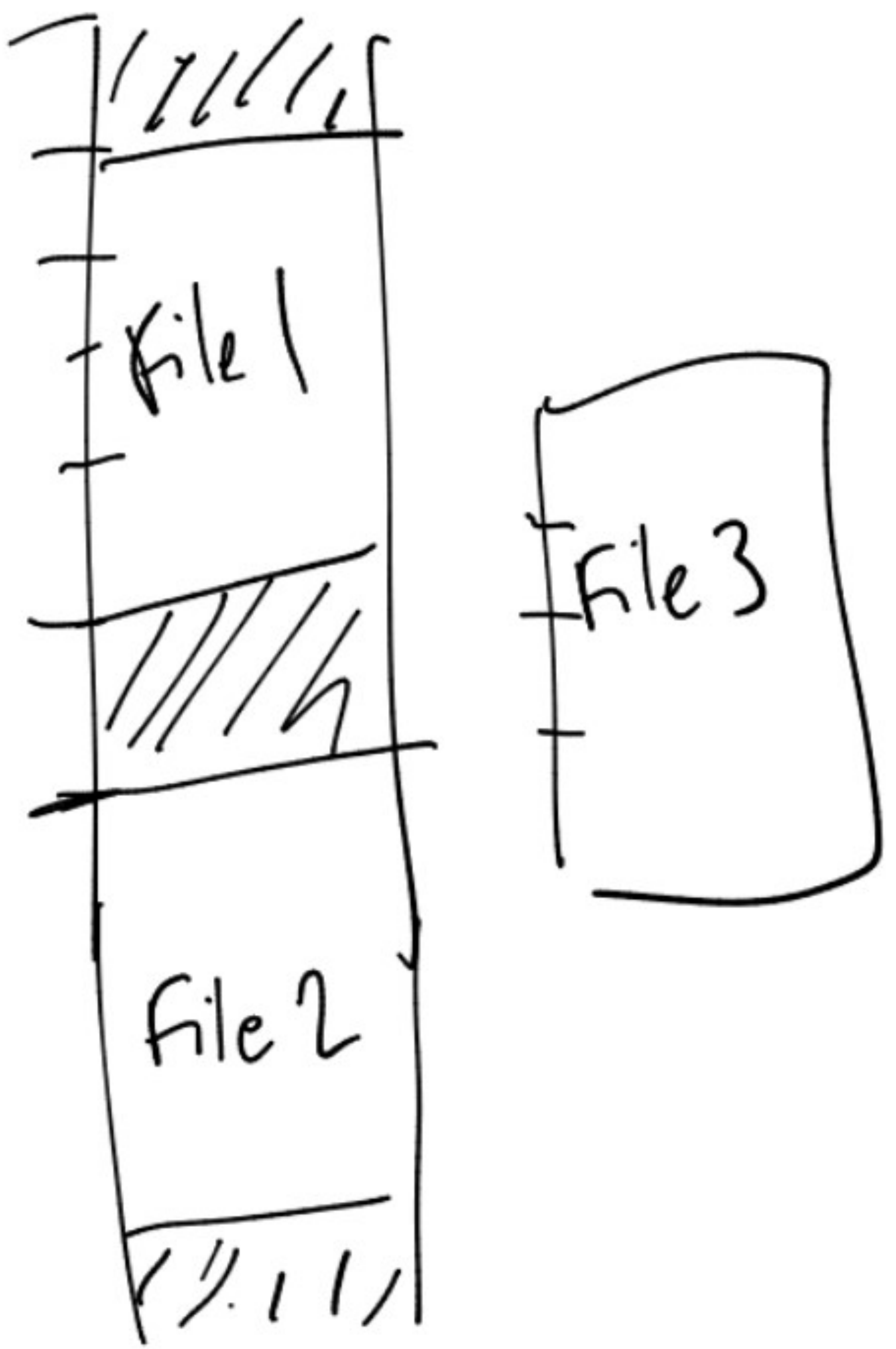


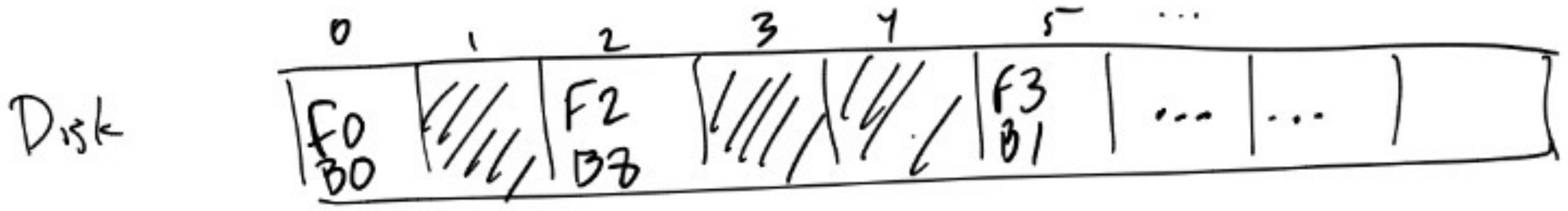
Syscalls

- fork child prosesses: create new PCBs, new PTs
- memory map system call:
  - update PTs of parent & child procs to point to same frame
- Synchronization:
  acquire & release mutex on shared variable.
- wait / exit:
  parent — children

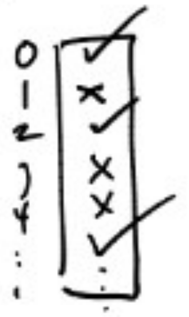
- parent : print
- parent : exit.

Contiguous allocation





array: 1 entry / sector, "free" or "used" (1 bit)



Bitmap allocation

to find free block: linear search

pro:

- have enough data/time to search for larger cont. chunks. (with a bit more effort)
- Simple, compact

con: - large disk: large bitmap.

$$\frac{1 \text{ bit} \left(\substack{\text{of}\\\text{map}}\right)}{\text{sector}} \cdot \frac{1 \text{ sector}}{2^{10} \text{ bytes} \left(\substack{\text{of}\\\text{data}}\right)} \cdot \frac{\text{byte}}{2^3 \text{ bits}} = \frac{1 \text{ byte (of map)}}{2^{13} \text{ bytes of data}} \approx \frac{1}{8000} \text{ overhead}$$

not so bad, maybe a pro.

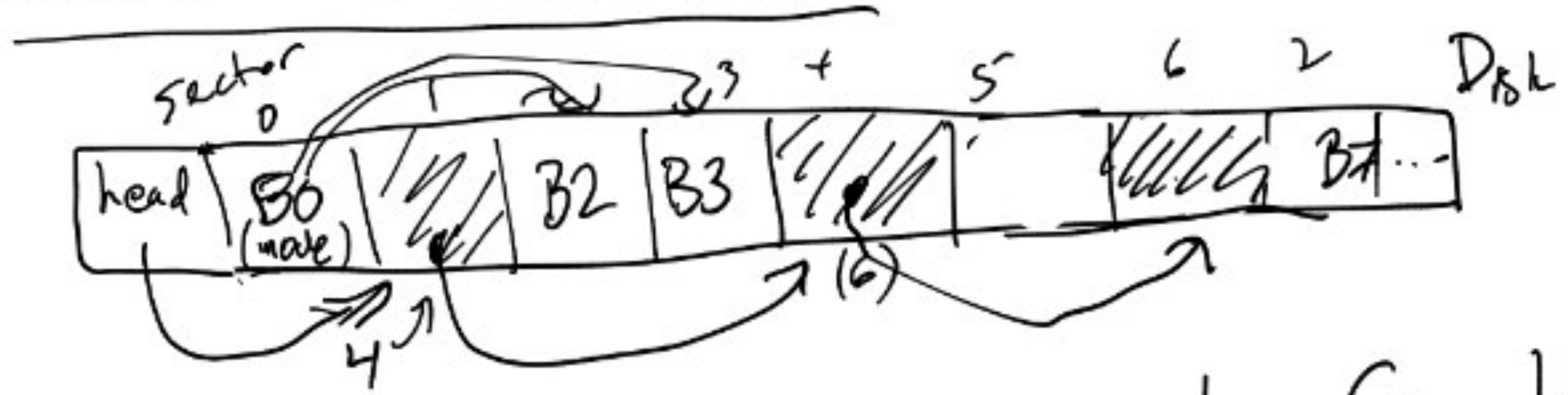
- linear search expensive

  If bitmap in memory: disk is small, search is cheap.

- may not be able to allocate files contiguously.

Reasonable for small disks, search overhead grows for large/full disks

Linked list allocation



put next pointer in each free block

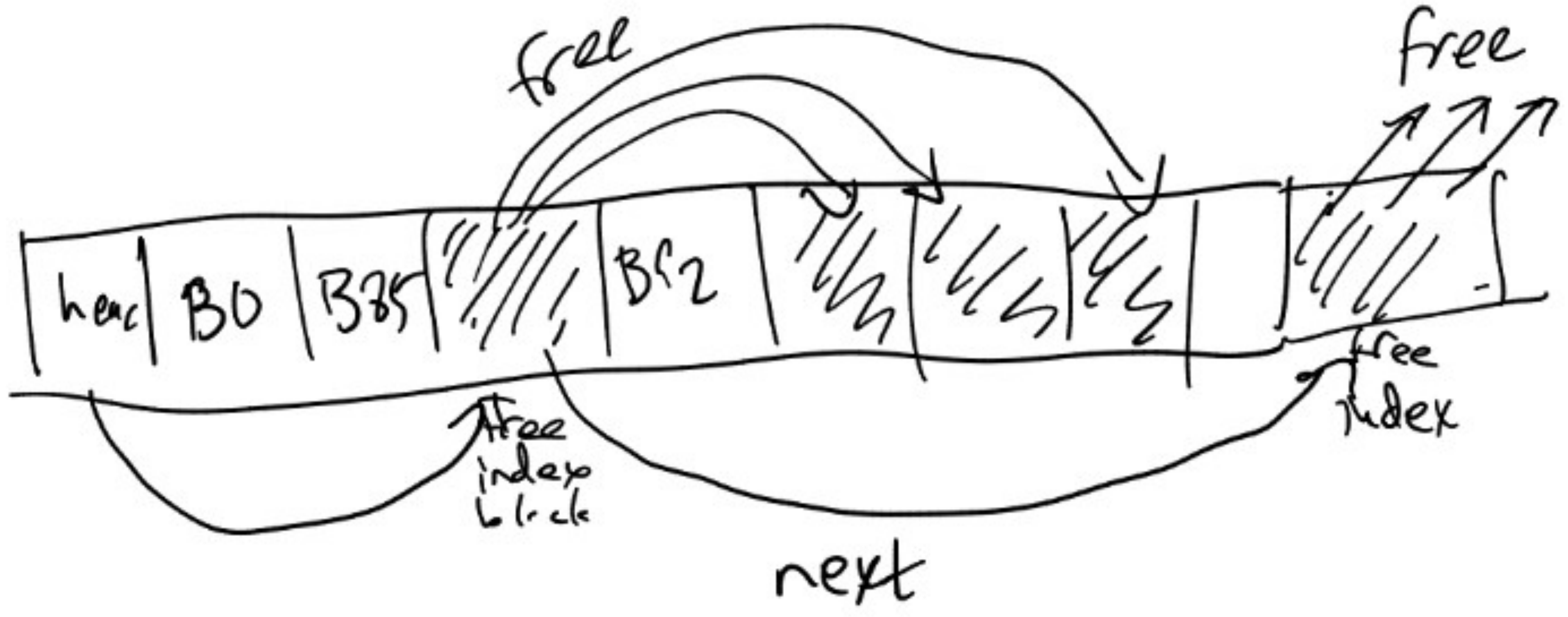
pro:

- fast: O(1) to find new (alloc) block
- deallocation easy.

Con:

- need tons of seeks to find contiguous chunk. (prohibitive)
- slow to allocate more than one block at a time

Linked list of blocks of pointers



to allocate free block:

look into next free index block,
allocate free blocks from it,
if gone: allocate free ind. block,
update head.

pros: same as LL,
can allocate many.. at once