Lecture 18: Free space & Recovery

- Managing free space

- FS consistency

- Journaling

-(review, quiz)
Entry $0B$ of TLPT: 0x2A

Entry $0E$ of 2LPT (in frame 42): 0xAD

Entry $0E$ of PT (in frame A): 0x55

Address 0x3ECEF is present (valid): no page fault

but page 3ECE is not readable; cant increment

segmentation fault
- parent: `print`
- parent: `exit`

- Synchronization:
  - acquire: Release mutex on shared variable.
  - wait/exit:
    - parent
    - children
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, impact

 reasoned for small
- search overhead
disks, grows for
- large/full
  disks

con: large disk, large bitmap

not so bad, maybe a pro.

<table>
<thead>
<tr>
<th>1 byte (of map)</th>
<th>1 sector</th>
</tr>
</thead>
<tbody>
<tr>
<td>2^3 bytes of disk</td>
<td>2^3 bits</td>
</tr>
</tbody>
</table>

- linear search expensive
- if bitmap in memory:
  - disk is small, search is
  - cheap.
- may not be able to allocate A/B/C/G contiguously.
Linked list allocation

Pros:
- Fast: $O(1)$ to find new block
- Deallocation easy

Cons:
- Need tons of seeks to find contiguous chunk (prohibitive)
- Slow to allocate more than one block at a time

Put next pointer in each free block

Linked list of block of pointers

To allocate free block:
Look into next free index block
Allocate free blocks from it
If gone: allocate free next block
Update head

Pros: same as LL, can allocate many at once