- [x] 1. Last time
- [x] 2. Final exam
- [x] 3. Your questions
- [x] 4. Wrap-up

---

## 2. Final exam

- 110 minute exam
  - stay seated at 100 mins
  - closed book
  - TWO two-sided sheets allowed

## Material

- Readings
- Labs
- HWs
- Classes

[see midterm topic list] → see l14.txt

Post-midterm topics (not guaranteed to be necessary or sufficient)

- virtual memory
  - page replacement policies (FIFO, LRU, clock)
  - thrashing
  - mmap()
- I/O
  - architecture
  - how CPUs and devices interact
    - mechanics
    - polling vs. interrupts
    - DMA vs. programmed I/O
  - device drivers
- synchronous vs. async I/O
- context switches
- User-level threading
- Disks

- geometry
- performance
- interface
- scheduling (skipped in class, covered in book)

File systems
- basic objects: files, directories, metadata, links, inodes
- how does naming work?
- types of file layout
  - extents/contiguous, linked, index
  - classic Unix + FFS are variants of indexed
- analogy between inode and top-level page directory (aka L1 page table)
- tradeoffs
- performance

Crash recovery
- ad hoc
- copy-on-write (COW)
- journaling (redo logging, undo logging, undo+redo)
  - WAL

RPC, client/server systems

Case study: NFS

marquee user of RPC

mostly skipped in class, covered in book

protection and security

stack smashing / buffer overflow

Unix security model

access control, privileges, setuid, attacks

trusting trust

bootup, from power-on

static linking + loading is a key tool

bootstrap process

H/W copies firmware into read/write mem

firmware is mini OS

runs bootloader program, which ultimately begins kernel

kernel invokes init(1) (or init(8))

init(1) invokes login(1) → (or init(8))

login(1) lets you get a shell and begin executing programs

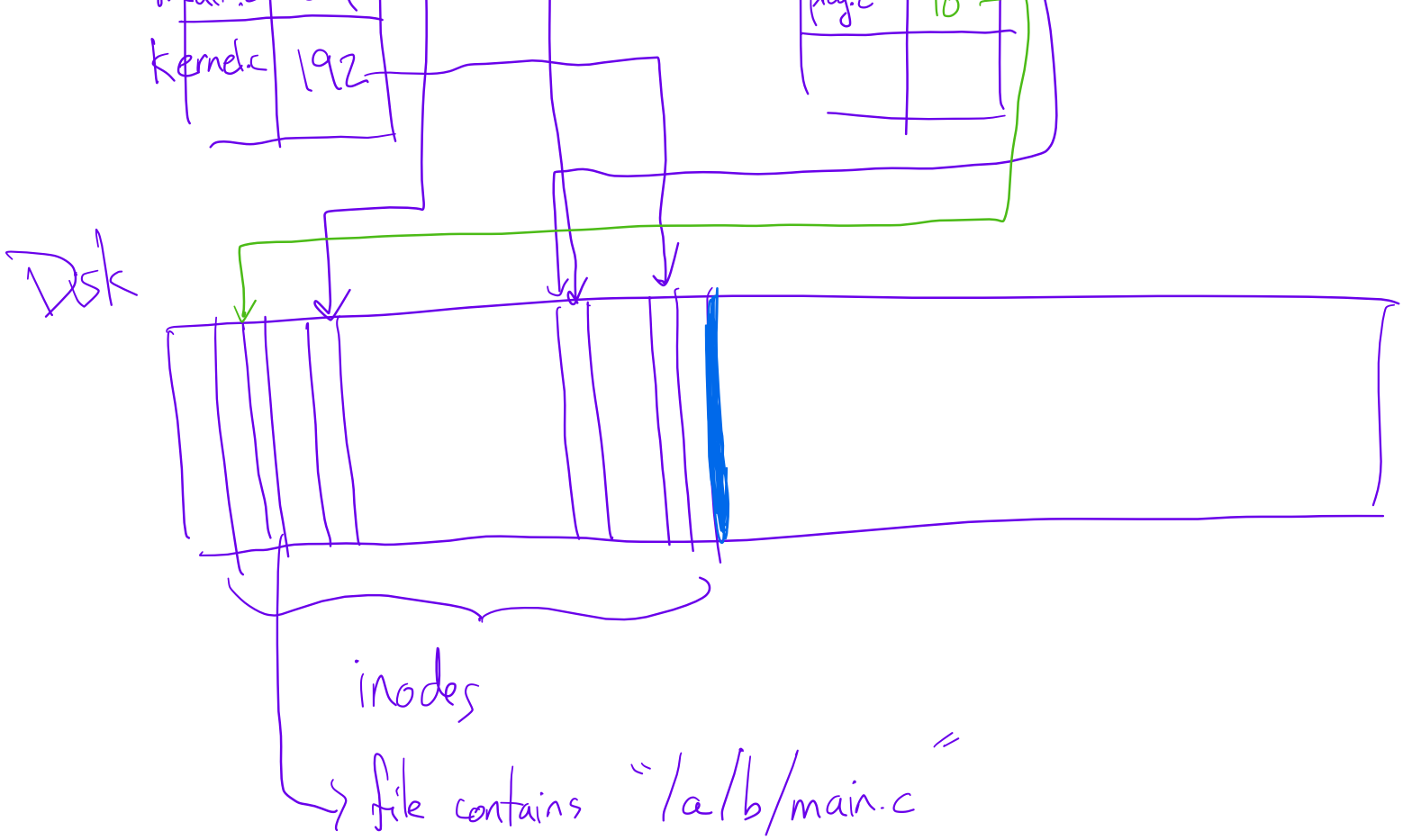
---

/a/b

| | |
|---|---|
| essay | 173 |
| main.c | 64 |

/c/d

| | |
|---|---|
| hello.txt | 34 |
| eng201-final.docx | 173 |



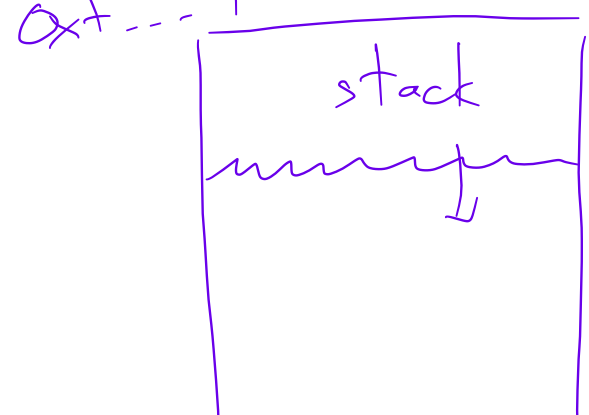

---

```
vm_map(pg_table, 0x200000, pa1, PGSIZE,
                                PTE_P,
                                albc);
vm_map( "  ", 0x300000, pa2, 5*PGSIZE, ..);
vm_map( "  ", 0x301000, pa3, PGSIZE, ..);
```

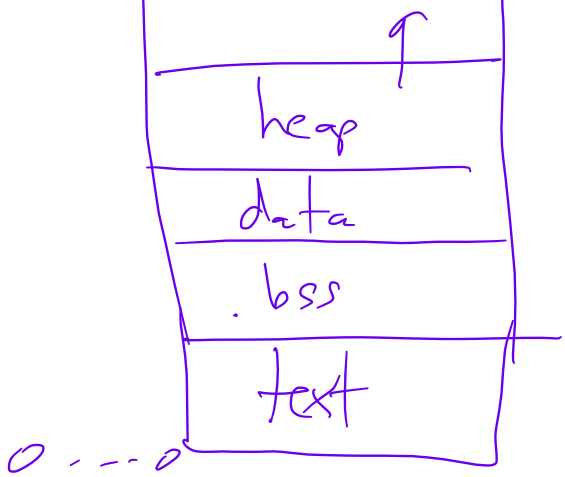
---

```
foo.c:
char *s = "abc---z";
int ex = 5;
int foo;

void f(){
  ---
}
```



```
}
int main () {
    ..
    =
    ..
}
```

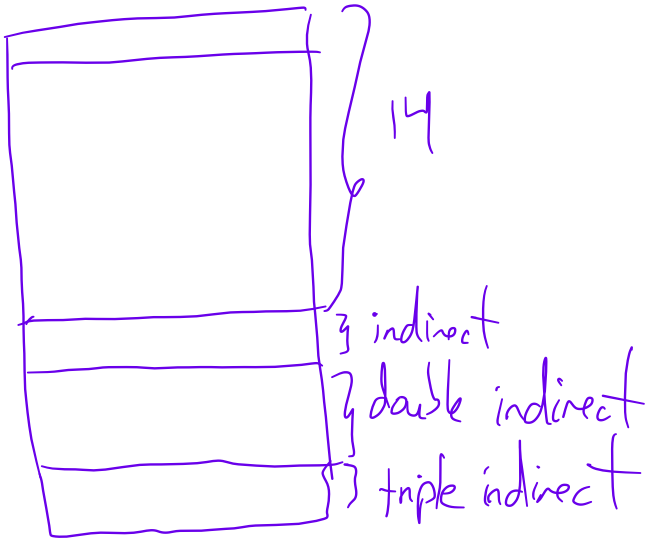


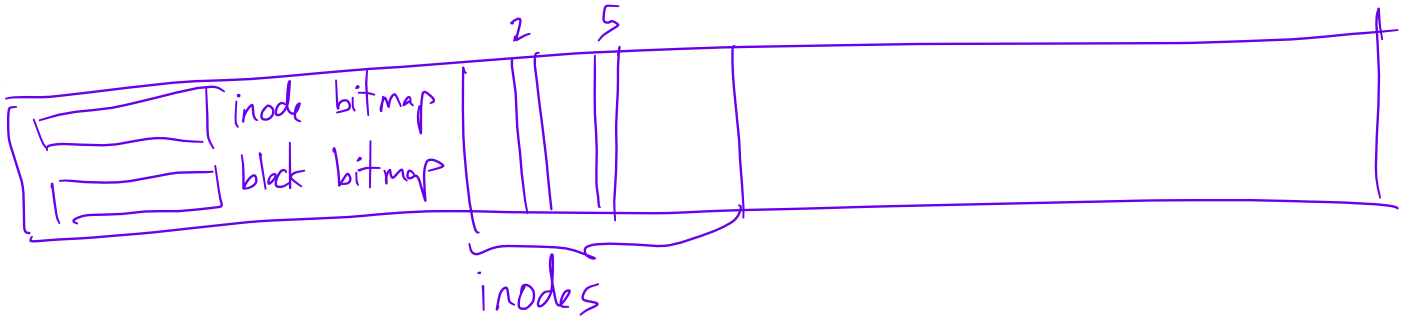

---

redo - undo logging

why do we need undo pass?

---




/foo/bar

fetch inode 2

foo ? 5

fetch inode 5

(d) modify 5 in-memory:
<bar, 7>

(c) set inode 7 as in-use

(b) write to inode 7, the first direct slot

(a) write the actual data

QR code for course feedback

# Operating Systems - Spring 2024, Lecture (SP24:CSCI-UA:202:1:001)

## Students

https://go.blueja.io/HcMkjD89BkqadltyZ9ezrw

To access the evaluation, scan this QR code with your mobile phone.