☑ 1. Last time
☑ 2. Final exam
☐ 3. Your questions
☐ 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] $\rightarrow$ l14.txt

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

performan

- 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
    - RPC: transparent or not?
- 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) / init(8)

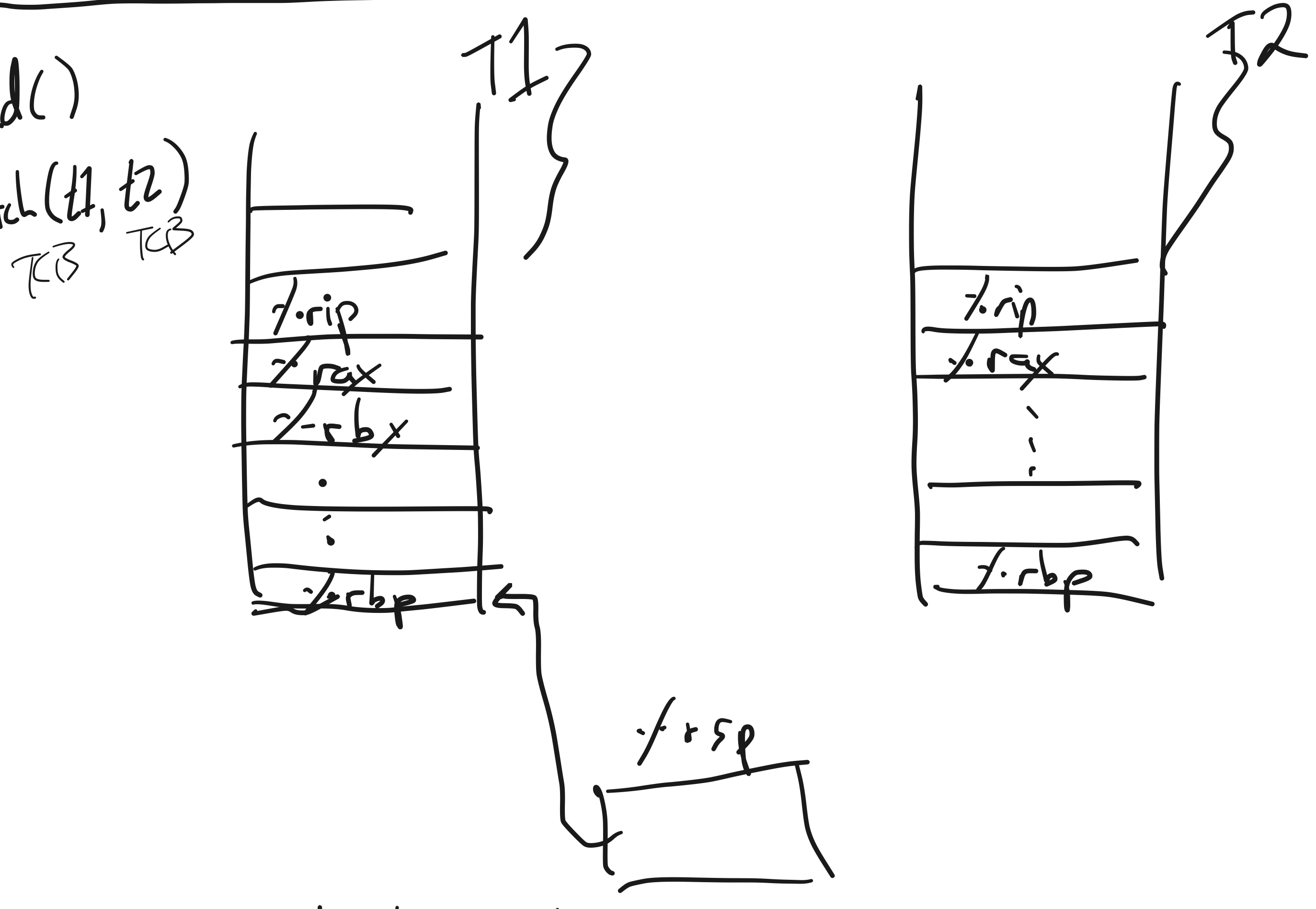
init(1) invokes login(1)

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

---

yield()

switch(t1, t2)



t1 → stack = %rsp

%rsp = t2 → stack

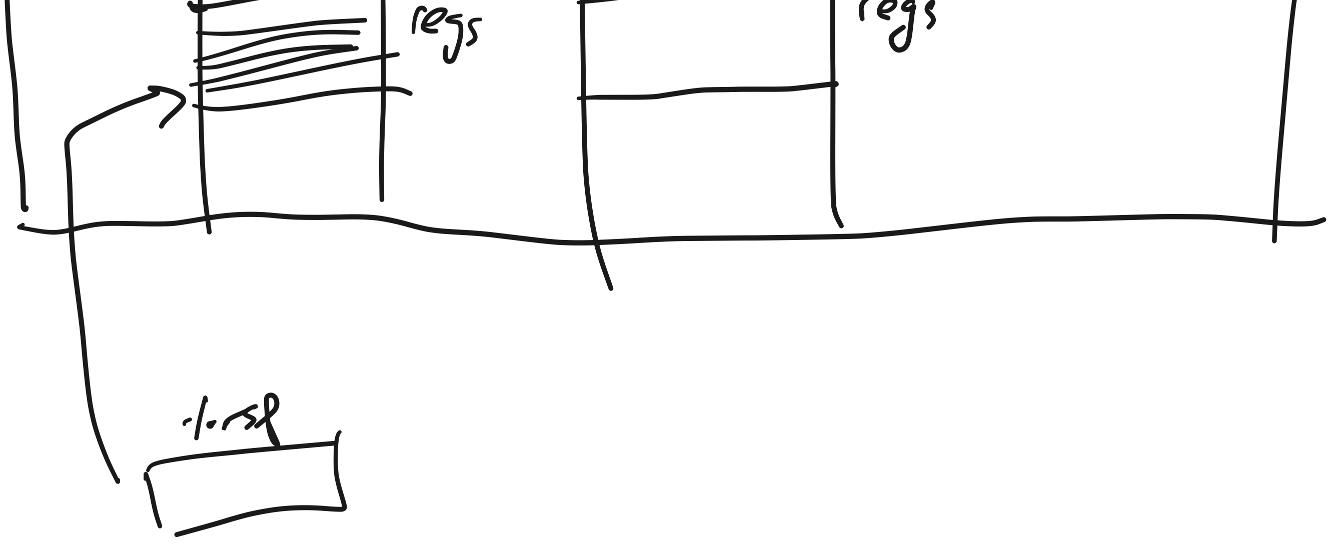
pop

pop

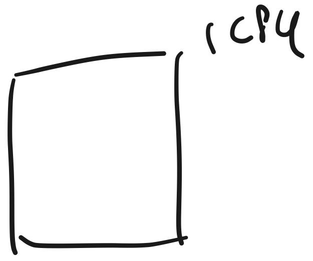
---

processes

PCB



Sp 2020 final, Q4



False

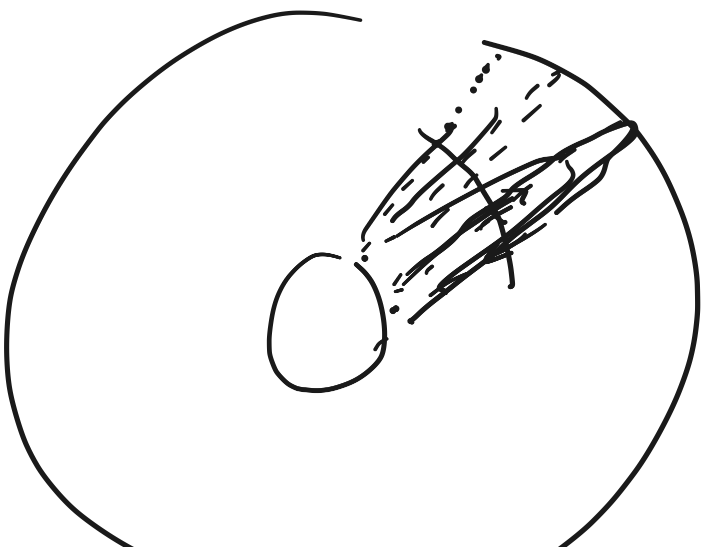
FS goals:
minimal

Sp 2020 Final, Q8

```
{
        proc* p = &procs[pid];
    if(check_perm _ return -1; ) ~~return~~
        memcpy(regs, p->p_registers, sizeof(p->p_registers)).
    return 0;   %eax 1 or 0.
}
```

mmap()

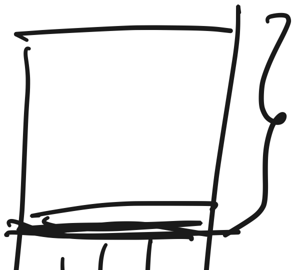
~~kernel calls read()~~

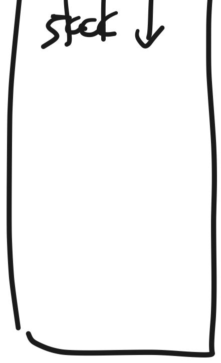


```
struct foo {
    int a;
    int b;
};

foo f;
foo* p = &f;
*p = 2; /* syntax */          int* q = (int*) &f;
                              *q = 2;
p->a = 2;
```

```
return (int*) _ __ ;

int f()
```

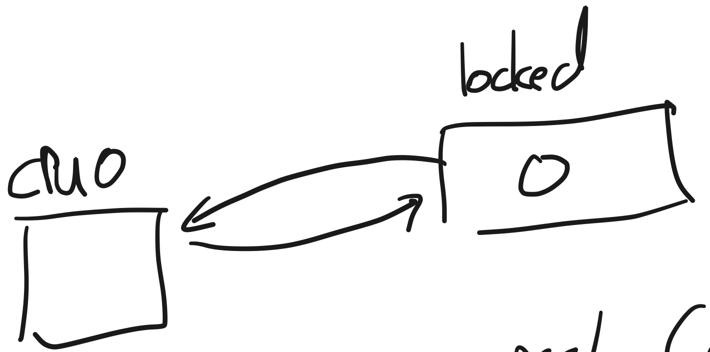
```
virtual_memory_map( - - - )
```





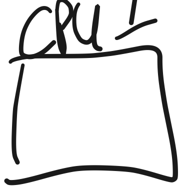
---

spinlock vs. mutexes



xchg (reg, addr)

CPU1



```
acquire:
    while ( xchg (%eax, &locked) == 1 ) ;
```

```
release:
    %eax ← 0
    xchg (0, &locked);
```

mutex:

spinlock

queue of waiters
id of holder