Today

- IPC (inter-process communication)
  - named pipes
  - signals
  - shared memory

- Scheduling
  1) First-Come, First-Serve
  2) Round-Robin
  3) Priority + Decay

- System Calls
  - calling conventions
  - software interrupts
  - exception / privilege levels

- The first process
  - bootstrapping
  - more processes

- Concurrency
  - threads
  - multiprocessor / multicore
IPC

1) named pipes
   - file FIFO

    open(pipe)
    write(pipe, 2K)
    os
    sleep (K)
    wake
    read (pipe)

    open(pipe)

Bounded 1K pipe

(kernel's job)

(un)-bounded?
-blocking?

vs. async
non-blocking
2) Shared Memory
   + larger shared region
   + performance
     + no copying
     + no switching kernel

IPC
Scheduling

Which process ("thing", task) to run next?

Where - - -

- Multicore (symmetric/ asymmetric)
- GPU APU
What's a good scheduler?

- Throughput
- Latency
  - Time to finish
  - "Observed latency"
- Req. $\rightarrow$ res

- CPU is doing something "useful"
- Or not!!
Round Robin

- Keep a FIFO queue of tasks to run
- After some fixed time \( q \) (quantum) (1ms - 10ms)
  - move \( P_2 \) to back of FIFO
  - pop & run \( P_4 \) from FIFO

Fair? Good?
Scheduling Algorithms: Round-Robin

2 jobs; each takes 10 s

RR:

0 1 2 3 4 5 6 7 8 9 10

Avg. completion Time? 9.5 s! (assuming switch is free)

Running each to completion serially?

\[(10 + 20) / 2 = 15 s\]
Priorities

Each process (task) has an associated priority.

Choose the first process from the highest priority queue.

Starvation and decay!