Lecture 4: threads & Scheduling

- More on PCB
- Threads
- Show & tell
- Scheduling
What goes in the PCB?

- Everything the OS needs to know to manage the process
- Registers: processor state (flags)
- Process state, if waiting, what for?
- Scheduling info (e.g., priority, history)
- Permissions (e.g., userid, groupid)
- State of resources (e.g., open files, network connections)
  - File descriptors
- Identifier (process ID), parent
- Configuration (example: signal handlers:
  - Location of code (in proc) to handle signals from OS (like interrupts)
- Processes have their own address spaces
  - completely isolated (except via inter-process communication)
  - need context switches to communicate expensive (cache invalidation)

- "Processes" that share address space are called threads.
  - have own local vars, shared global variables
Scheduling
- When CPU becomes free, what to do next?

Criteria:
- Fairness
- Efficiency:
  - Scheduler itself
  - Avoid context switches
- Priority:
  - Specified by users
- Simplicity.
- Responsive ress:
  - Respond quickly to user requests
FCFS (First come, first served)
- put incoming jobs in queue (at end)
- to schedule, select a job (from front)

Pros:
- Fair (if all jobs finish)
- Simple
- Scheduler super efficient
- Not many context switches

Cons:
- No notion of priority
  - Average waiting time can be bad.
Preemption

- set a timer (will cause interrupt after a delay)
  - when timer goes off, schedule another process.

Round robin: Preemptive FCFS

- schedule process for fixed amount of time (quantum)
- go to end of line after timeout.