Lecture 4: Threads & Scheduling

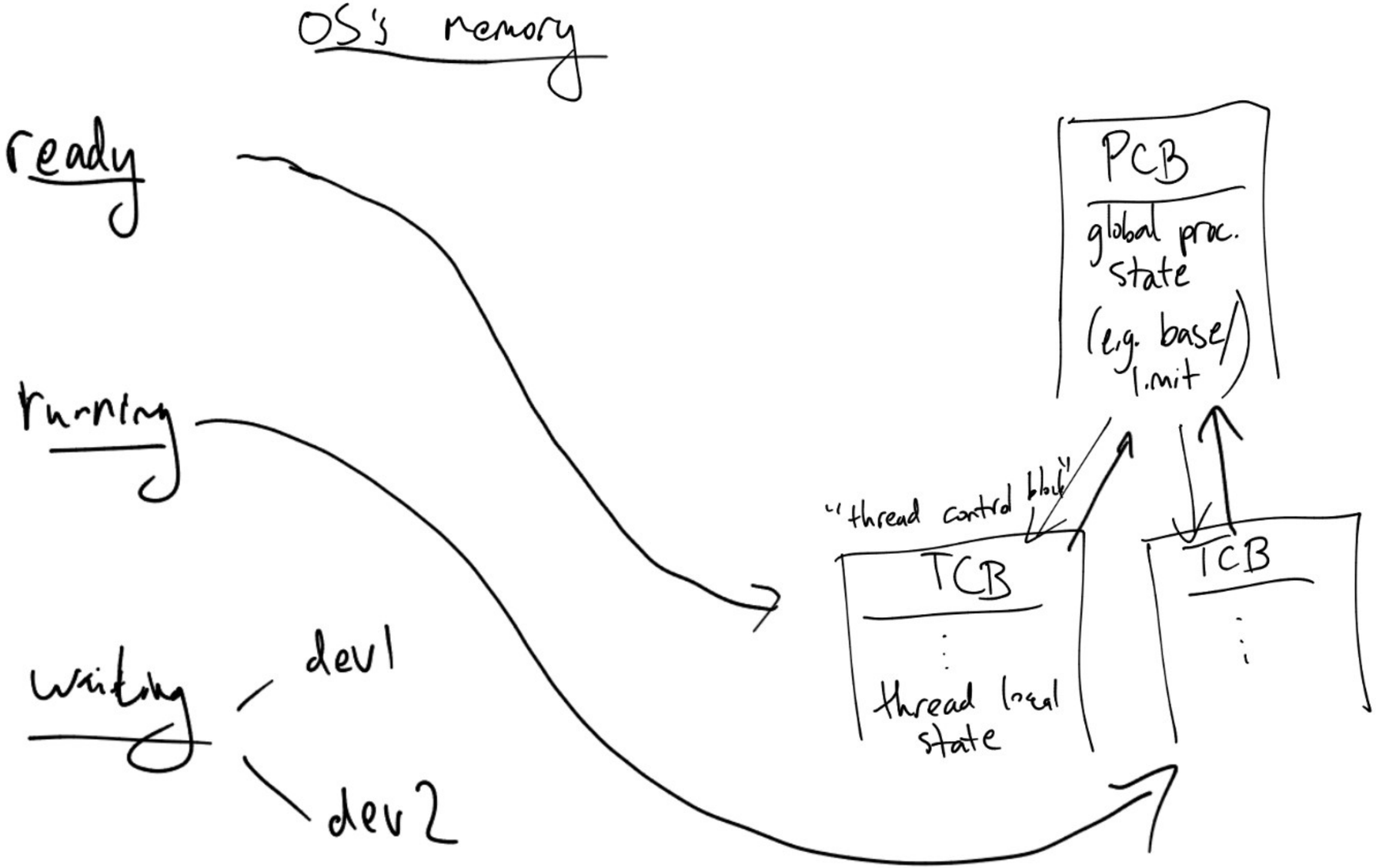
- More on PCB
- Threads
- Show & tell
- Scheduling

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 (eg. 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-comm syscalls)
  - need context switches to communicate expensive (cache invalidation)
- "Processes" that share address space are called threads.
  - have own local vars, shared global variables


OS's memory
ready
running
waiting
dev1
dev2
PCB
global proc. state
(e.g. base/limit
"thread control block"
TCB
thread local state
TCB

Scheduling

- when CPU becomes free, what to do next?

Criteria:

- Fairness
- Efficiency:
  - scheduler itself
  - avoid context switches
- Responsiveness:
  - respond quickly to user requests
- Priority
  - specified by user
- Simplicity.

FCFS (First come, first served)

- put incomming jobs in queue (at end)
- to schedule, select a job (from front)

Pros:

- Fair (if all jobs finish)
- simple.
- scheduler super efficient
- not many ctx 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 procs for fixed amount of time (quantum)
- go to end of line after timeout.