Lecture 3: Processes

- Context switching
- PCBs
- Initialization
- Traps (exceptions, interrupts, syscalls)
- Privilege
Need to save data about process:

- memory (base & limit)
- registers (IP, SP, ...) 
- processor Flags

Process Control Block (PCB) (data structure)

Stored in OS's region of memory
OS's memory region

running

ready

waiting

dev1

dev2

P1 PCB

P2 PCB

registers
base/bounds...

P3 PCB

P4 PCB

read from

dev 1
Context Switch: Changing from 1 proc. to another

- update state of PCBs
- load regs for new proc
- jump to IP of new proc.
Need privileged instructions to isolate applications

- Change base/bounds
- Modify int. vector

Privilege bit is a flag in the processor
  - if clear: priv. instructions cause exceptions instead of executing.

Terminology: “kernel mode” (priv.) vs. “user mode” (unpriv.)
  - also called “supervisor mode” or “ring 0
  - not to be confused with “root” or “administrator”
When to set/reset priv. bit?

- Initialization: want priv. bit set.
- Interrupt
- Process termination
- HW exceptions
- Request from processes

set priv. bit

System Calls

traps: anything that causes priv. to get set.
- Interrupts: cause branch to int, set priv. bit
- System calls: requests from applications to OS
  cause a branch to syscall handler routine:
  location stored in dec. register.
  sets priv. bit.

HW exceptions: causes branch to exc. handler routine
  set priv. bit.

priv. bit cleared during
"return from syscall"
instruction
(not necessarily a)
"return"

whenever priv. bit becomes high, must
jump to predetermined address (inside kernel)

to prevent abuse (thus "trap")
Init. routine:
- Set up IV, syscall handler, exc. handler
- Set up devices
- Set up "init" process
  \[ \text{read config, launch other processes} \]
- "return" from syscall \( \Rightarrow \) init process.

Syscall Handler:
- Just like a fo call:
  - Save callee saved registers
  - check permissions (eg. is process allowed to)
  - do its I/O, update PCB state, ...
  - schedule another process (maybe)

Exc./Int. handlers:
- Similar to syscall handler, need to
  - Save all proc. state since
  - Exc./Int. is unexpected.
Each OS has a list of available system calls:

**Monolithic kernel**
- System calls: application level requests
  - `open a file`
  - `read from open file`
  - `open network connection`
  - `launch a new process`
  - `wait for another process to end`
  - `exit (terminate process)`
- Large amount of kernel code
- Kernel bugs are bad

**Virtual machine manager (VMM)**
- System calls forwarded to another process (guest OS)

**Microkernel**
- Only system calls
  - Only `send message` syscall to request the disk process to read from disk
  - Some processes (drivers) need access to addresses corresponding to devices

**Driver for disk**
- P17
- MMIO add of disk