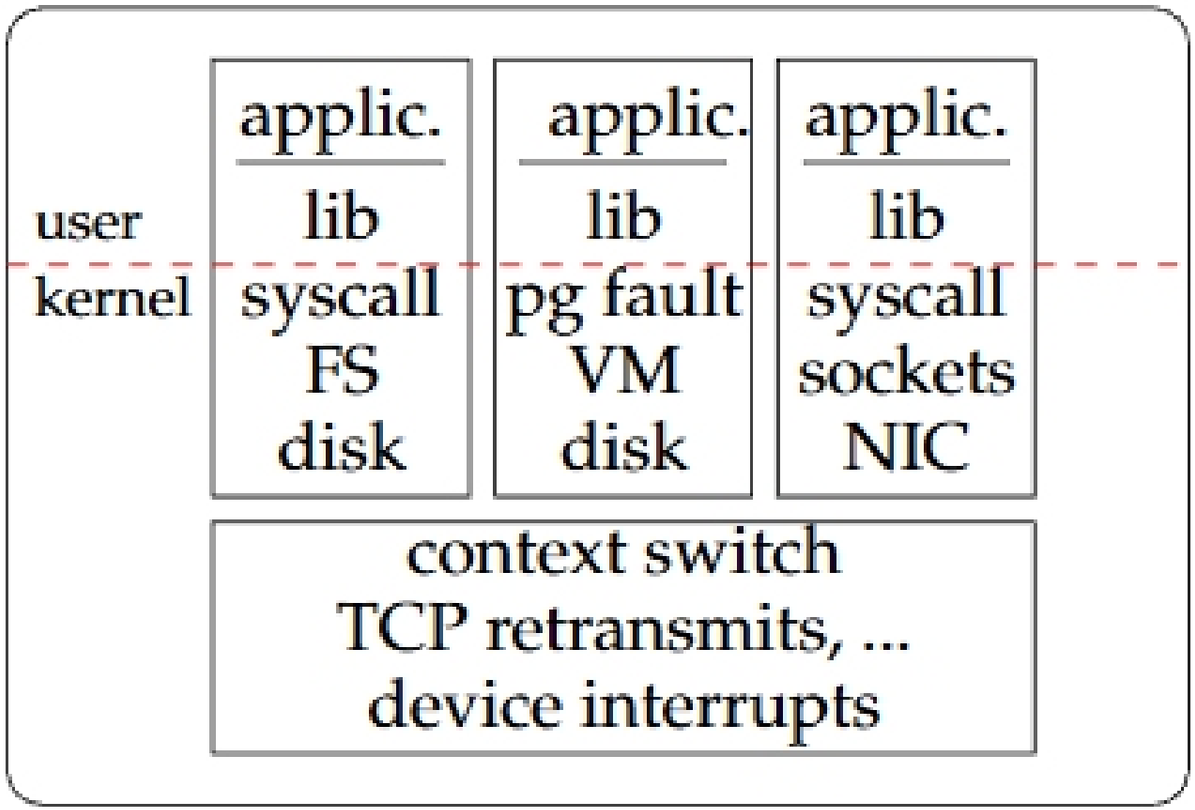
Review: Hardware user/kernel boundary



- **Processor can be in one of two modes**
  - user mode – application software & libraries
  - kernel (supervisor/privileged) mode – for the OS kernel
- ***Privileged* instructions only available in kernel mode**

Execution contexts

- **User – executing unprivileged application code**
- **Top half of kernel (a.k.a. "bottom half" in Linux)**
  - Kernel code acting on behalf of current user-level process
  - System call, page fault handler, kernel-only process, etc.
  - This is the only kernel context where you can sleep (e.g., if you need to wait for more memory, a buffer, etc.)
- **Software interrupt**
  - Code not acting on behalf of current process
  - TCP/IP protocol processing
- **Device interrupt**
  - External hardware causes CPU to jump to OS entry point
  - Network interface cards generate these, disks too, . . .
- **Timer interrupt (hardclock)**
- **Context switch code – change top half thread**

Transitions between contexts

- **User → top half: syscall, page fault**
  - E.g., Read or write to a socket
- **User/top half → device/timer interrupt: hardware**
  - E.g., network transmission complete or a packet has arrived
- **Top half → user: syscall return**
  - Note syscall return can also go to context switch (e.g., if packet has arrived and made sleeping process runnable)
- **Top half → context switch: sleep**
  - E.g., User called read on a TCP socket, but no data yet
- **Context switch → user/top half: return (to new ctx)**