I/O Devices
You Need to Get Out More!

How does a computer connect with the outside world?
I/O Architecture

CPU

MEM

Memory Bus

General I/O Bus
(PCI)

Graph

Peripheral I/O Bus
(SCSI, SATA USB)

Graph

CPU

MEM

PCIe

I/O CHIP

eSATA

USB
Interacting with a Device

Registers
(what the OS sees)

Abstraction
(what the user sees)

Internals
(what is needed to implement the abstraction)
Interacting with a Device

Registers

Interface

Status

Command

Data

(Internals

(what the OS sees)

(what is needed to implement the abstraction)
Interacting with a Device

Microcontroller
Memory
Other device specific chips

Internals
(what is needed to implement the abstraction)

Registers

| Status | Command | Data |
OS controls device by reading/writing registers

while (STATUS == BUSY)
    ; // wait until device is not busy
write data to DATA register
write command to COMMAND register
    ; // starts device and executes command
while (STATUS == BUSY)
    ; // wait until device is done with request
Tuning It Up

- CPU is polling
  - use interrupts
  - run another process while device is busy
  - what if device returns very quickly?

- CPU is copying all the data to and from DATA
  - use Direct Memory Access (DMA)

while (STATUS == BUSY)
    ; // wait until device is not busy
write data to DATA register
write command to COMMAND register
    // starts device and executes command
while (STATUS == BUSY)
    ; // wait until device is done with request
From interrupt-driven I/O to DMA

**Interrupt driven I/O**
- Device ➔ CPU ➔ RAM

for
- CPU issues read request
- device interrupts CPU with data
- CPU writes data to memory
From interrupt-driven I/O to DMA

Interrupt driven I/O
- Device \(\leftarrow\) CPU \(\rightarrow\) RAM
  - for
    - CPU issues read request
    - device interrupts CPU with data
    - CPU writes data to memory

+ Direct Memory Access
- Device \(\rightarrow\) RAM
  - CPU sets up DMA request
  - Device puts data on bus & RAM accepts it
  - Device interrupts CPU when done
Communicating with devices

- **Explicit I/O instructions (privileged)**
  - _in_ and _out_ instructions in x86

- **Memory-mapped I/O**
  - map device registers to memory location
  - use memory load and store instructions to read/write to registers
How can the OS handle a multitude of devices?

- Abstraction!
  - Encapsulate device specific interactions in a device driver
  - Implement device neutral interfaces above device drivers

- Humans are about 70% water...
  - ...OSs are about 70% device drivers!