

Overview of Operating Systems



ECE 373

Overview

- What is an OS?
- A brief history of the OS
- Types of Operating Systems
- A view of the OS from above
- Different OS internals
- System calls
- Device drivers

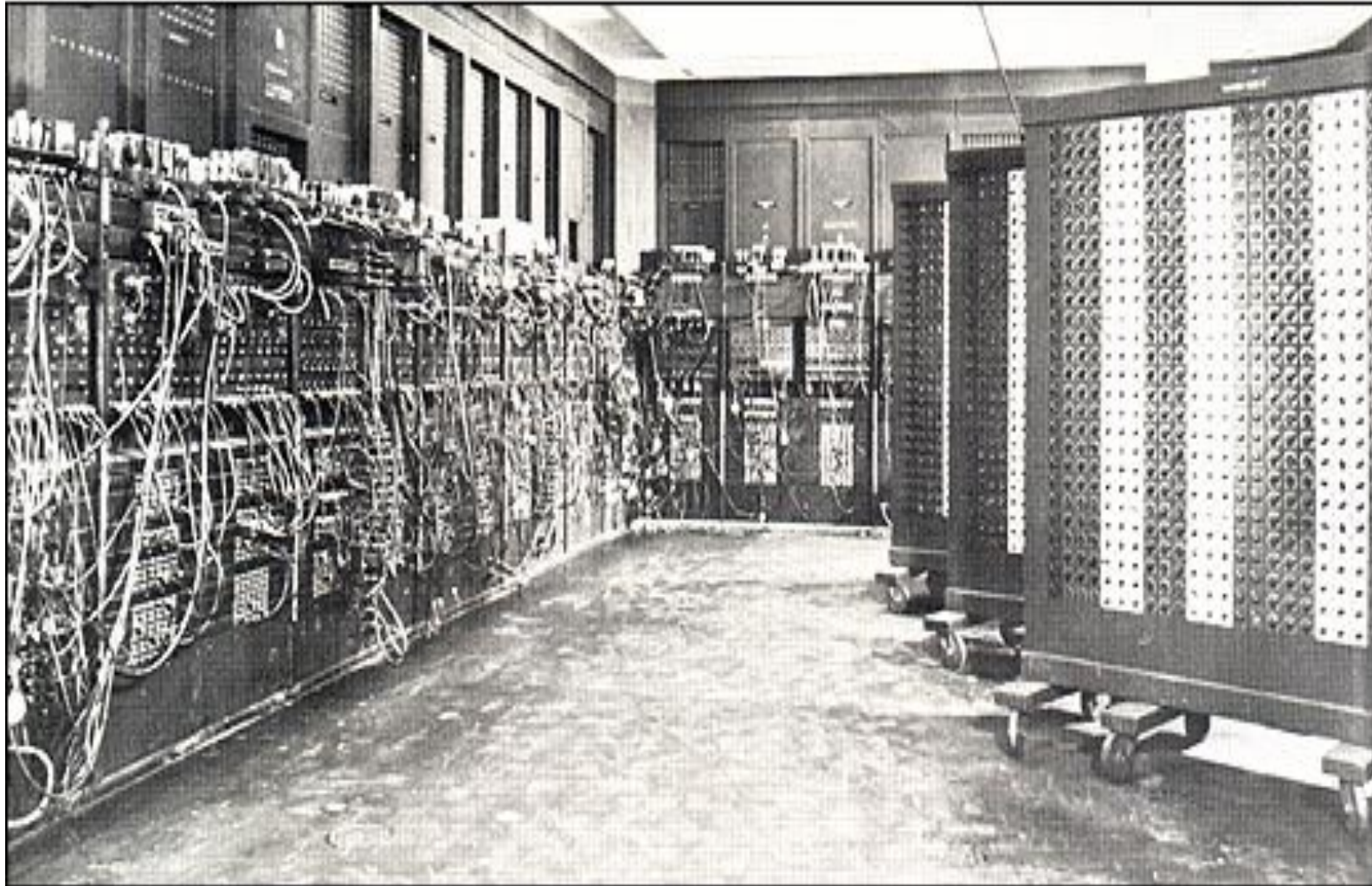


What is an Operating System?

What is an Operating System?

- SW that makes HW useful
- Provides interface for users to mess with HW
 - Stores financial data, prints it when needed
 - Displays the monsters, creates the pew-pew noise when the trigger button is pressed
 - Reads the data from the network, displays the video, plays the music
- User programs?

Not all computers had an OS



Among the first assignments given to Eniac, first all-electronics digital computer, was a knotty problem in nuclear physics. It produced the answer in two hours. One hundred engineers using conventional methods would have needed a year to solve the problem

A brief history

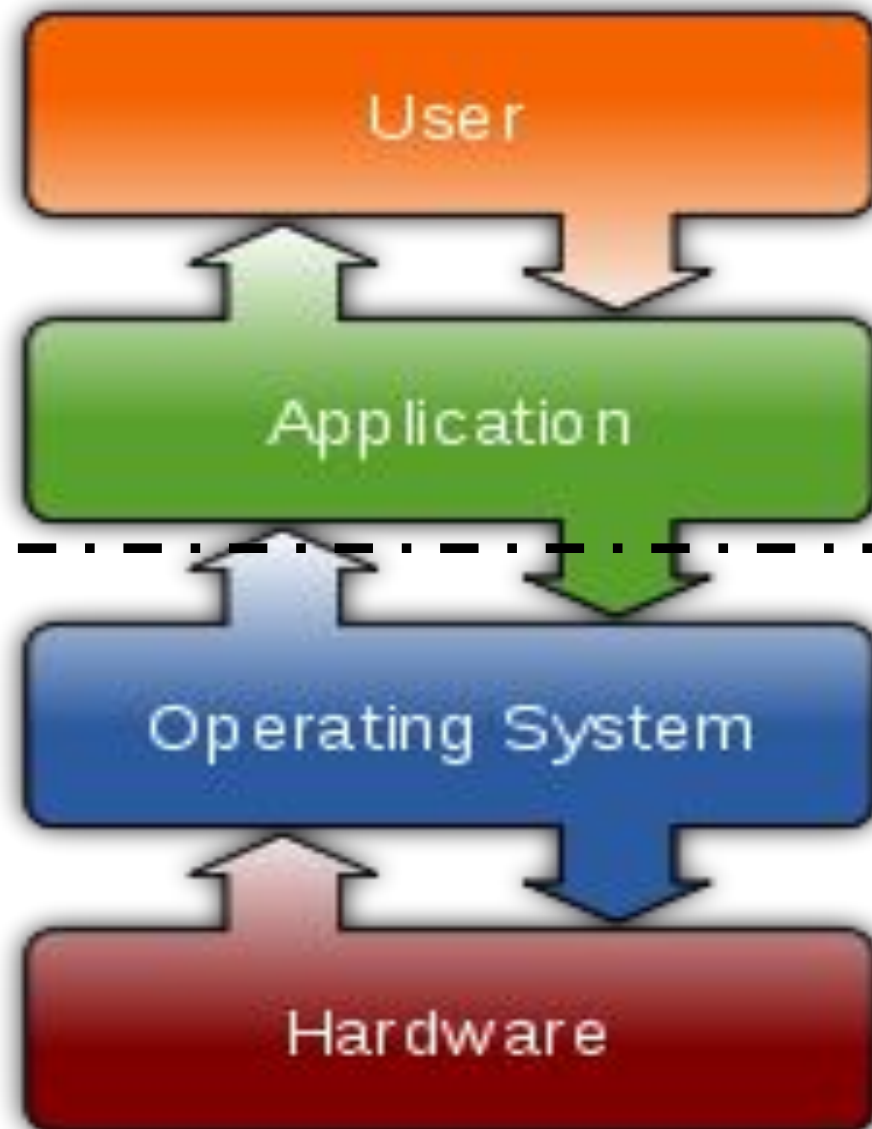
- Mainframes
 - MCP, OS/360, MITS, TOPS-10, Multics, UNIX, VMS, Linux
- Microcomputers
 - CP/M, MS-DOS, AppleDOS, Mac OS, AmigaOS, Windows, Linux
- Embedded
 - Wind River, Symbian, Android, iOS, WinCE, RockBox, ThreadX, uCLinux, brickOS, DD-WRT, Open-WRT, Linux

Types of OS's in the wild

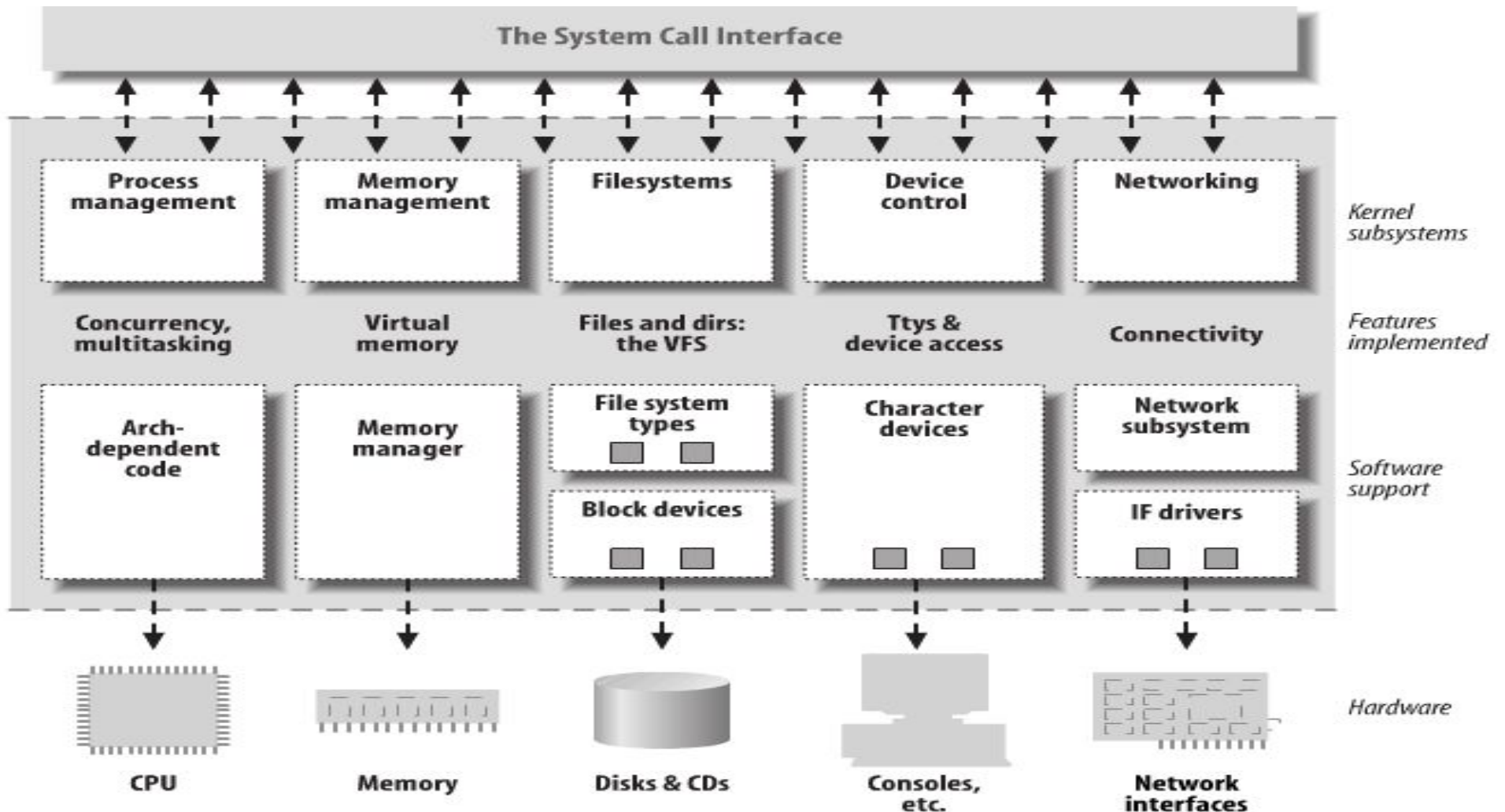
- Single-user (Phones, PC's)
- Multi-user (Servers, mainframes, "cloud")
- Real-time (Stop lights, SpaceX rocket control)
- Embedded (Watch, routers, car engine, mp3)



Layers on top of layers



The OS overview



 features implemented as modules

Process Scheduler



Process Scheduler

- OS thread that handles process scheduling
- Complicated piece of software
- Multithreaded operating systems need this
- Batch operating systems much simpler
- Timeslices...



Process Scheduler cont.

- Processes scheduled on CPU
 - Algorithms vary how this happens
 - Processes can be "pinned" or "affinitized"
- Code is architecture-dependent
 - x86 scheduler is different than ARM scheduler
 - e.g. Big/little - getting blurrier today
- Pre-emption

Memory Manager



Memory Manager



- RAM
- Swap
- Virtual vs. Physical
- Kernel vs. Userspace
- Allocate and free

I/O Subsystem



I/O Subsystem

- I/O – input/output
- Framework for data movement in system
- Many subsystems
 - PCI
 - USB
 - Storage
 - Network
 - Many more...



Interrupts

Interrupts

- Signal triggering a CPU there's something to do
- Drives the drivers
- IPI's
- Legacy interrupts
- MSI and MSI-X interrupts
- Shared interrupts
- <http://en.wikipedia.org/wiki/Interrupt>



System Calls

System Calls

- Main conduit from userspace to kernel space
- Linkage between C library and kernel through some glue in the kernel core
- System call glue written in assembly for the architecture
- Examples of common system calls
 - `open()`, `write()`, `read()`, `close()`
- The "man" pages chapter 2 are where the system calls live

System call man pages

```
[ppwaskie@ppwaskie-hc1 ~]$ man open
```

```
OPEN(2)                                Linux Programmer's Manual
```

```
OPEN(2)
```

NAME

`open`, `creat` - open and possibly create a file or device

SYNOPSIS

```
#include <sys/types.h>
```

```
#include <sys/stat.h>
```

```
#include <fcntl.h>
```

```
int open(const char *pathname, int flags);
```

```
int open(const char *pathname, int flags, mode_t mode);
```

```
int creat(const char *pathname, mode_t mode);
```

DESCRIPTION

Given a pathname for a file, `open()` returns a file descriptor, a small, non-negative integer for use in subsequent system calls (`read(2)`, `write(2)`, `lseek(2)`, `fcntl(2)`, etc.). The file descriptor returned by a successful call will be the lowest-numbered file descriptor not currently open for the process.

System calls for all

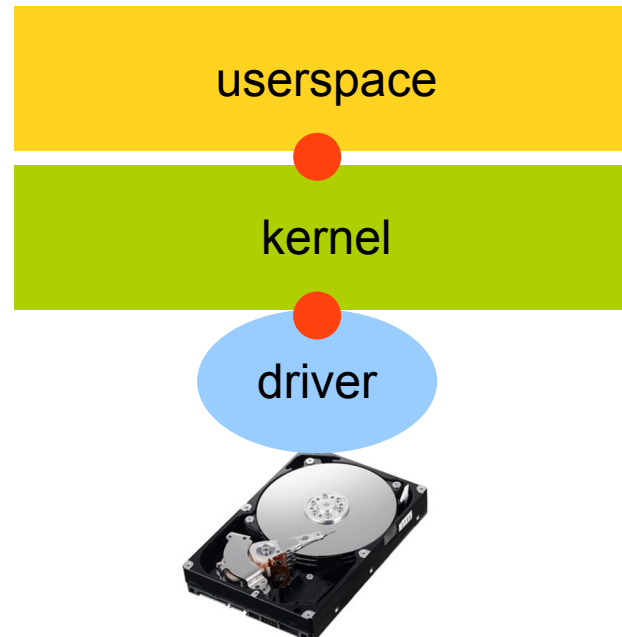
- Not all driver interfaces use the same system calls
- **Block and char drivers use** `open()`, `write()`, `close()` **for example**
- **Network drivers use** `socket()`, `select()`, `send()`, `shutdown()` **for example**

Device drivers



Device drivers

- The software that controls specific pieces of hardware
- Driver takes common commands from the OS and translates into hardware-specific stuff
- APIs in the OS provide common interfaces for services



Types of drivers

- Three main classes of drivers
 - Block drivers
 - Character drivers
 - Network drivers
- Driver class can apply to many types of devices
 - USB device can be char, block, or network
 - Interface cards (PCIe) can be network or block
 - Graphics typically char devices

Block drivers



Block drivers

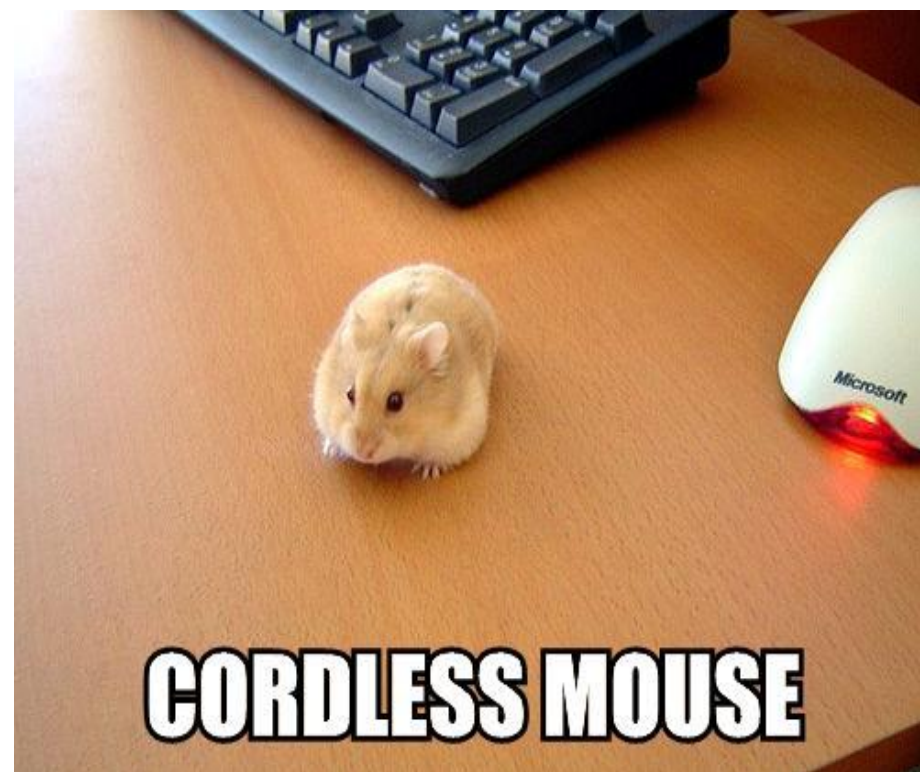
- Drivers that take and give data in rigid chunks of data (blocks...duh)
- Blocks are typically powers of 2, such as 512 bytes, 2048 bytes, etc.
- SCSI layer and SATA layer send blocks to disk drivers
- Blocks pushed down onto sectors on the hard drive
- SSD's, NVMe...



Char drivers

Char drivers

- Drivers that take a stream of characters
- Character streams often contain some type of opcode with data
- Streams can be variable length (unlike block devices)
- Common char devices are:
 - Keyboard
 - Mouse
 - Printer



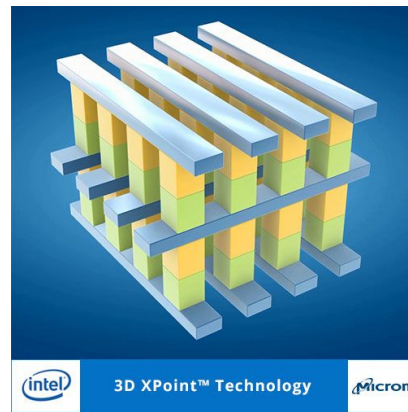
Network drivers

- Network drivers are different animals than block and char drivers
- Unpredictable patterns of traffic
- Network drivers typically not tied directly to system call interface
- Network drivers can be
 - Wireless Ethernet devices
 - Wired Ethernet devices
 - Cell phone 5G transceiver



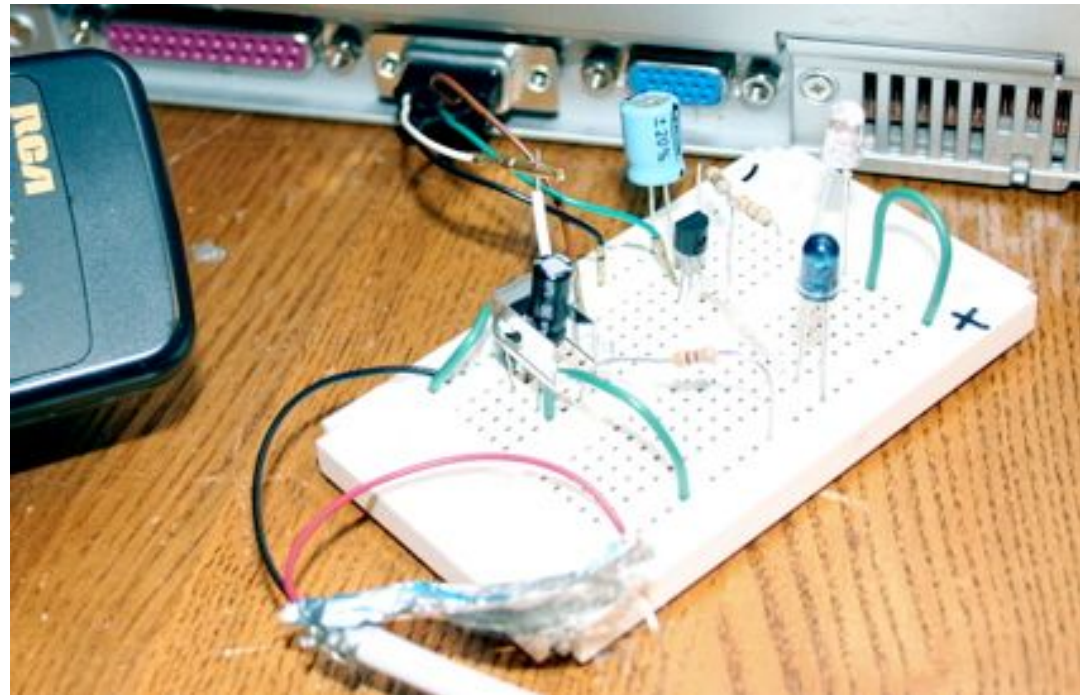
Hybrid devices

- Not all devices fit into one "category"
- Good example, 3D Xpoint NVDIMM (e.g. Optane)
 - ~~New~~ tech from Micron and Intel (controversy)
 - Phase-changing memory lattice
 - Can be block device or char device depending on firmware configuration



Hardware

- Voltages and signals
- Bits and bytes
- Registers and busses
- GPIOs and LEDs
- Sensors and motors



Wrap-up

- Operating Systems: big (complex) programs
- Operating Systems abstract specific hardware interfaces into common APIs for software
- Each block in the OS has a purpose
- Drivers are the abstraction of the hardware, keep "the devil in the details"

