# Lecture 1: Introduction

CS343 – Operating Systems

Branden Ghena – Fall 2020

Some slides borrowed from:
Stephen Tarzia (Northwestern), Jaswinder Pal Singh (Princeton), and UC Berkeley CS162

# Today's Goals

- Welcome to Operating Systems!

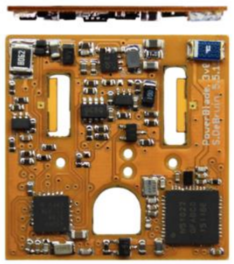
- How will this class operate?
- What is an Operating System?
- What will you learn in this course?

- Course Overview
- What is an OS?
- Operating Systems History
- CS343 Focus

- **Course Overview**
- What is an OS?
- Operating Systems History
- CS343 Focus

# Branden Ghena (he/him)

- Assistant Faculty of Instruction
- Education
  - Undergrad: Michigan Tech
  - Master's: University of Michigan
  - PhD: University of California, Berkeley
- Research
  - Resource-constrained sensing systems
  - Low-energy wireless networks
  - Embedded operating systems
- Teaching
  - Computer Systems
    - Intro to Computer Systems
    - Operating Systems
    - Microprocessor System Design

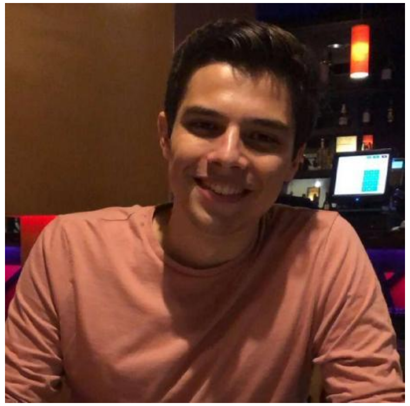
Things I love


# Course Staff

- Teaching Assistant
  - Conor Hetland
    - PhD student working with Peter Dinda
    - TA’d for W20 version of OS

- Peer Mentors
  - Calypso McDonnell
    - Senior, Computer Science
  - Michael Cuevas
    - Senior, Computer Science

  - Both took W20 version of OS



# Class Format

- Lecture
  - Pre-recorded and available on canvas

- Questions and Answer Sessions
  - Zoom call during class time
  - Come having watched lecture already
  - Ask questions and get more in-depth on topics

# Staff Roles

- Office Hours
  - 12 hours per week (3 per person including professor)
  - At a variety of times to work for many timezones
- Lab Discussion
  - 1 hour per week
  - Focused on tools and tips for doing the labs
    - C, Unix tools, Debugging, Specific lab advice
- Piazza
  - All week long, but not necessarily any time of the day

# Course Grade

- 20% Midterm (first half of the course)
- 20% Final (second half of the course)
- 60% Labs

- This class is NOT curved

# Lab Logistics

- Getting Started Lab – 05%
  - Learn how everything works
- Producer-Consumer Lab – 10%
  - Concurrency and locks
- Queuing/Scheduling Lab – 10%
  - OS application scheduling
- Device Driver Lab – 20%
  - Driver for a GPU
- Paging Lab – 15%
  - Memory management

- Getting started lab is special
  - One week deadline (due 09/24)
  - Must do alone
  - All-or-nothing grading
- Normally teams of 2 or 3 students
  - Find partners now!

# Lab Deadlines

- Labs are normally due at 11:59:59 pm Central Time
  - 20% lost points per day late

- Slip days
  - Everyone gets **two slip days**
  - Used to extend a project deadline by a full 24 hours with no penalty
  - Automatically applied as best helps your grade
  - Warning: cannot be used on Getting Started Lab

# These labs can be very challenging

- Dealing with C code
- Handling a large code base
- Dealing with concurrency!!

- Give yourself enough time to get the lab done on time
- You’ll learn a lot through the challenge

# Quarantine quarters continue

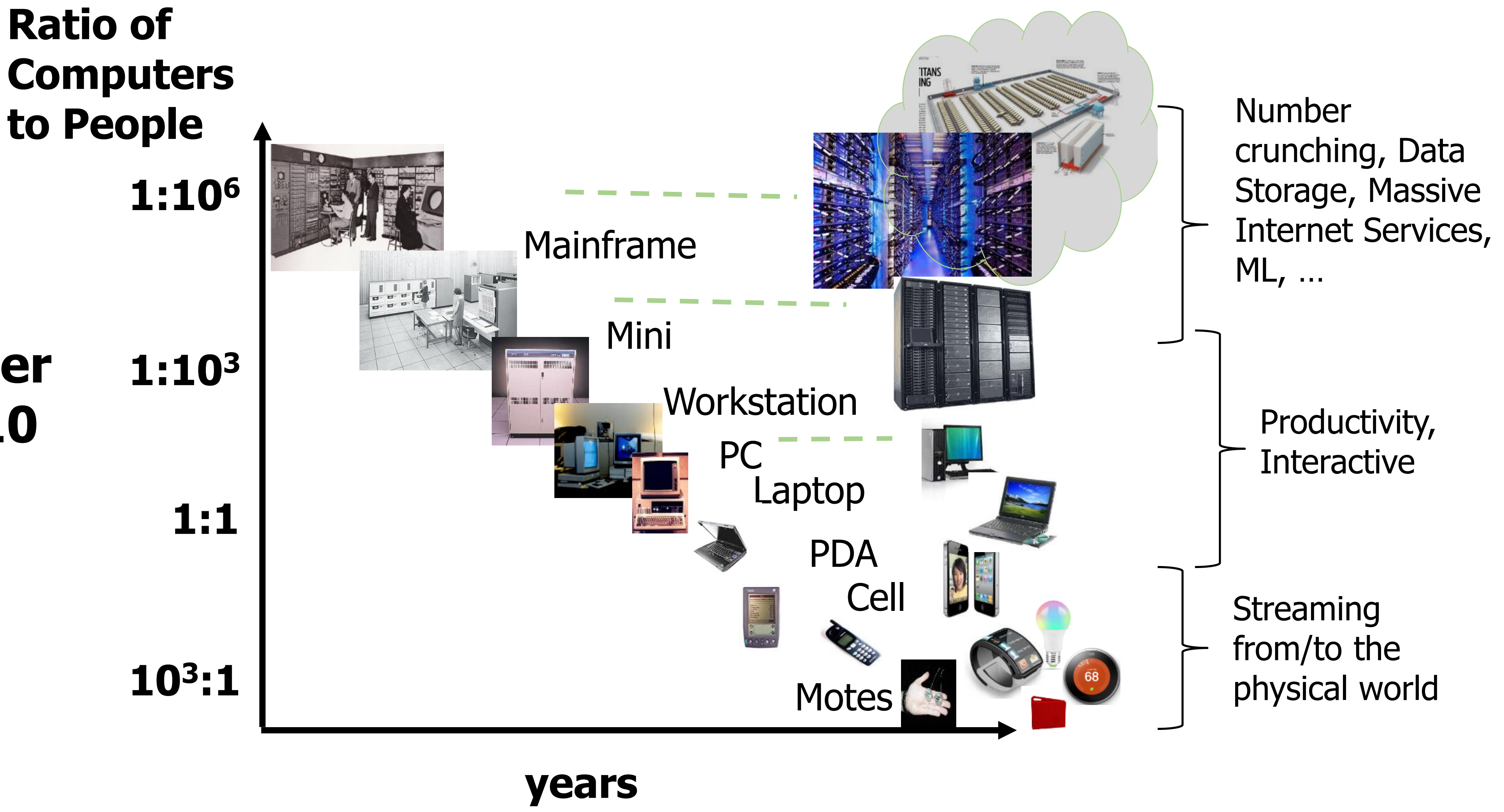
- I am new to remote teaching
  - Let us know what could change to help you learn

- If you are having a hard time keeping up with the class for any reason, *let us know*

- Course Overview
- **What is an OS?**
- Operating Systems History
- CS343 Focus

# Computers come in incredible diversity

**Bell's Law: New computer class every 10 years**



# Computing timescales are increasingly large

**Jeff Dean (Google AI): "Numbers Everyone Should Know"**

| | |
|---|---|
| L1 cache reference | 0.5 ns |
| Branch mispredict | 5 ns |
| L2 cache reference | 7 ns |
| Mutex lock/unlock | 25 ns |
| Main memory reference | 100 ns |
| Compress 1K bytes with Zippy | 3,000 ns |
| Send 2K bytes over 1 Gbps network | 20,000 ns |
| Read 1 MB sequentially from memory | 250,000 ns |
| Round trip within same datacenter | 500,000 ns |
| Disk seek | 10,000,000 ns |
| Read 1 MB sequentially from disk | 20,000,000 ns |
| Send packet CA->Netherlands->CA | 150,000,000 ns |

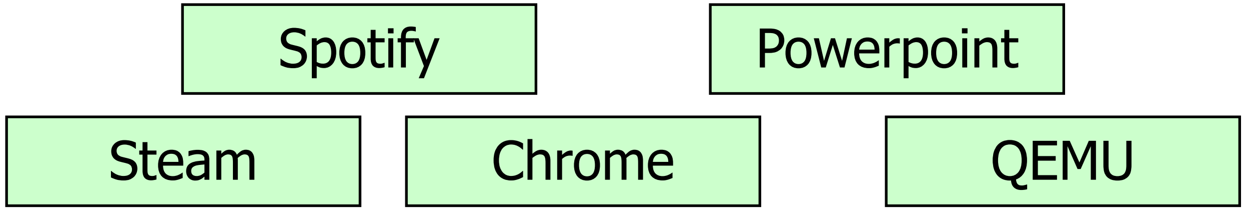
# Operating systems are at the heart of these challenges

- OSes make advancing technology available to rapidly evolving applications.
  - Provide **abstractions** to applications to enable hardware compatibility
  - Manage **sharing of resources** across many applications

- Good operating systems do these quickly, efficiently, and securely

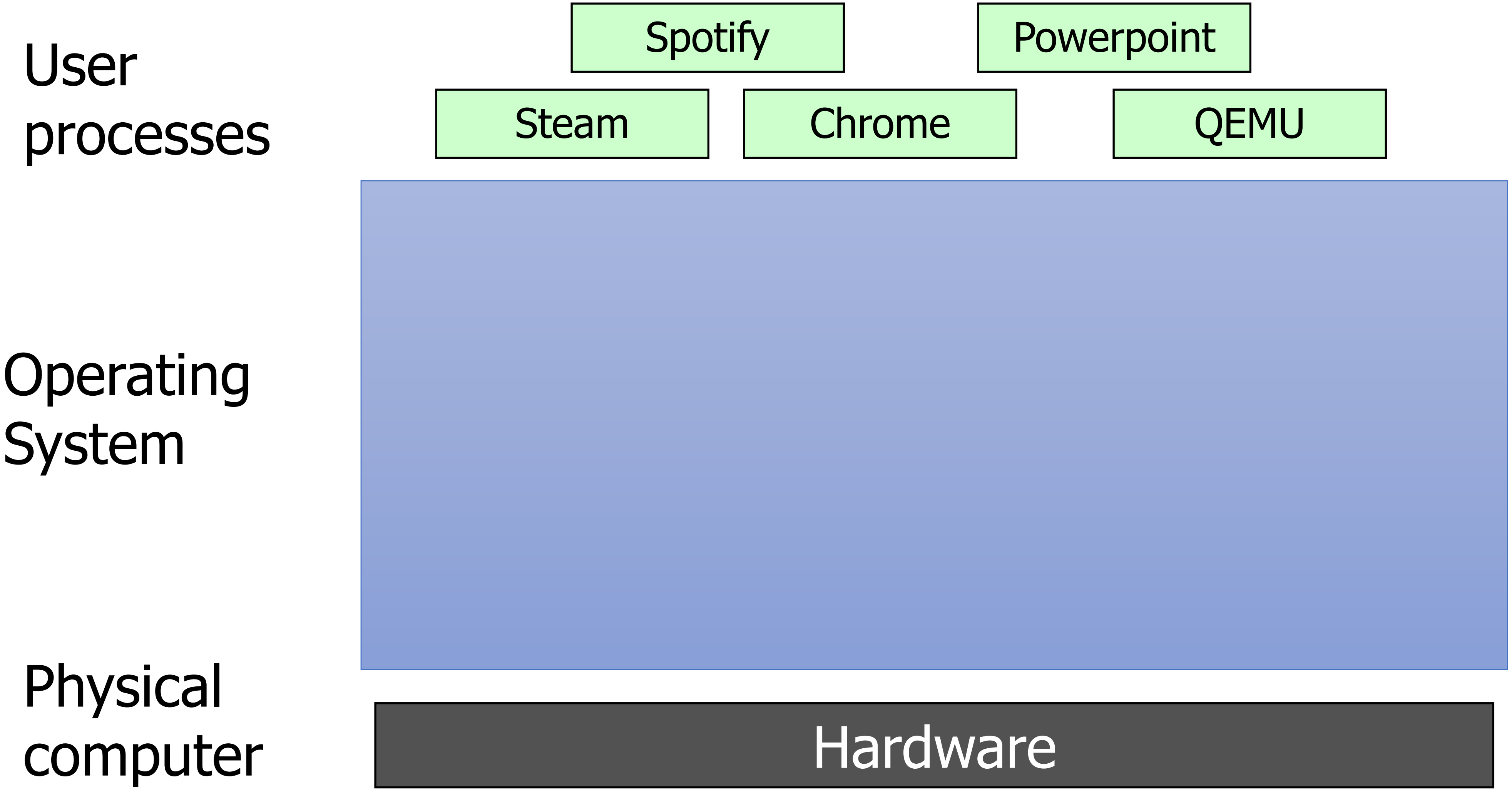
# What is an operating system?

User processes

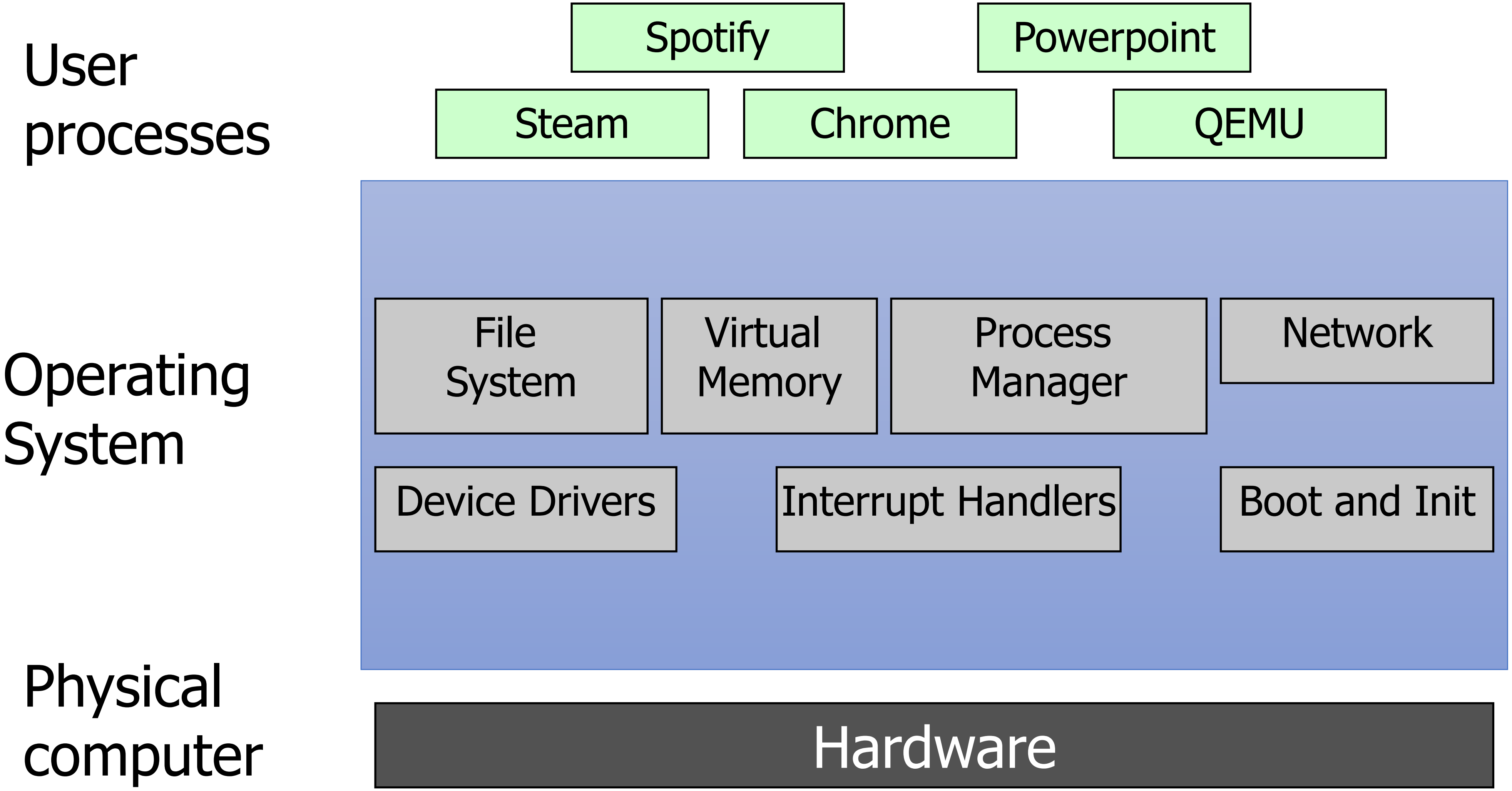



Physical computer

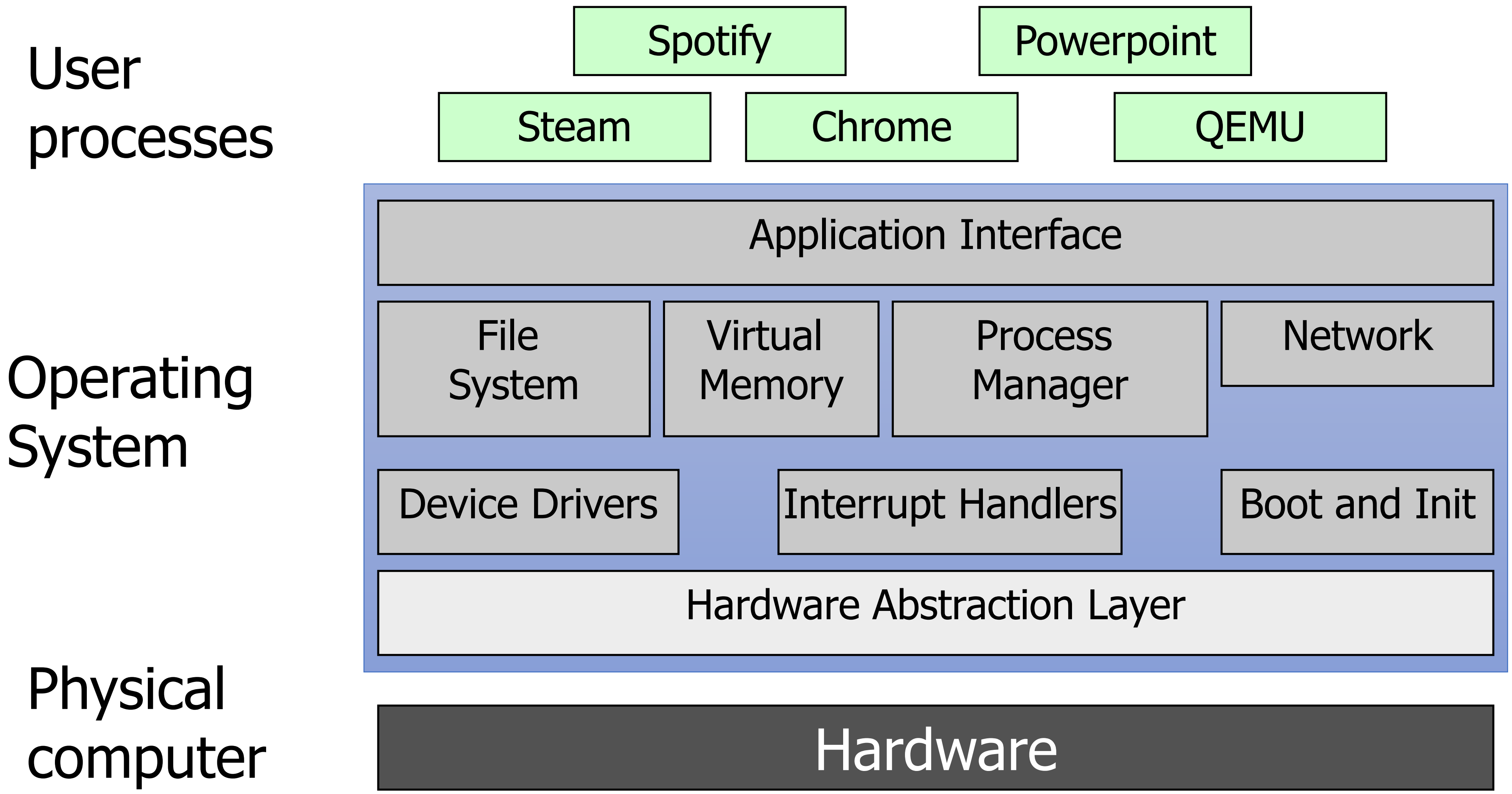
# What is an operating system?



# What is an operating system?



# What is an operating system?



# What's part of the OS?

- **OS kernel** – the only code without security restrictions

- Process scheduling (who uses CPU)
- Memory allocation (who uses RAM)
- Accesses hardware devices
  - Outputs graphics
  - Reads/writes to network
  - Read/write to disks
  - Handles boot-up and power-down

- **OS distribution** – the kernel + lots of other useful stuff

- GUI / Window manager
- Command shell
- Software package manager
  - "app store", yum, apt, brew
- Common software libraries
- Useful apps:
  - Text editor, compilers, web browser, web server, SSH, anti-virus, file-sharing, media libraries,

# Before operating systems

- User could only run one program at a time.
- Had to insert the program disk before booting the machine.
- Program had to control the hardware directly
  - This is a nuisance because hardware is complicated
  - Program will only be compatible with one set of hardware
- For example (at right) 1983 “King’s Quest” game for IBM PC Jr.


# Embedded systems often run without operating systems

- "Bare-metal" embedded systems
- Application must handle:
  - Boot and initialization
  - All hardware it wants to interact with

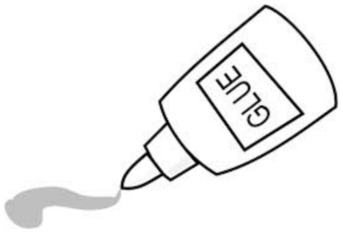
- Applications are not portable
  - Rewrite, mostly from scratch, for new microcontroller

- No malloc, no segfaults

# What is an Operating System?

- Referee
  - Manage protection, isolation, and sharing of resources
  - Resource allocation and communication

- Illusionist
  - Provide clean, easy-to-use abstractions of physical resources
    - Infinite memory, dedicated machine
    - Higher level objects: files, users, messages
    - Masking limitations, virtualization




- Glue
  - Common services
    - Storage, Window system, Networking
  - Sharing, Authorization
  - Look and feel

# Example: File Systems

- Referee
  - Prevent users from accessing other’s files without permission
- Illusionist
  - Files can grow infinitely large
  - Where a file exists in memory or disk isn’t important!
- Glue
  - Default file system types, named directories

- Course Overview
- What is an OS?
- **Operating Systems History**
- CS343 Focus

# Computer History

- Actually check out the textbook!
  - In-depth history
  - Entertaining writing with just the right amount of sarcasm

- This isn't a computer history course
  - But there is a good reason to understand the lineage of the techniques we explore in this course

# Early evolution of computing systems – Batch

- 1955: Batch systems
  - Collect a bunch of program punch cards and write them all one magnetic tape.
  - Run the tape through the mainframe to execute all the jobs in sequence.
- OS responsibility
  - Libraries for I/O
- Problems
  - I/O is VERY slow. 80-90% of total time just waiting.

# Early evolution of computing systems – Multiprogramming

- 1960s: Multiprogramming (IBM OS/360)
  - Keep multiple runnable jobs in memory at once.
  - Allows overlap I/O of one job with computing of another.
    - Uses asynchronous I/O and interrupts or polling to detect I/O completion

- OS responsibility
  - Schedule jobs
  - Monitor I/O

- Problems
  - Still need to submit all jobs in advance

# Early evolution of computing systems – Timesharing

- 1960s-70s: Timesharing (MULTICS, Unix)
  - Multiple user terminals connected to one machine
  - Allows *interactive* use of machine to be efficient (because another user's job can run while you're thinking).

- OS responsibility
  - Multiple users (with permissions!)
  - Scheduling processes
  - Application interface
  - Shell tools

# Later evolution of computer systems – PC

- 1980s-90s: Personal Computers (IBM PC, Macintosh)
  - Graphical user interfaces were developed
  - Mainframe OS concepts (like networking) were applied to PCs
  - Magnetic disks (hard drives) become huge, but still slow

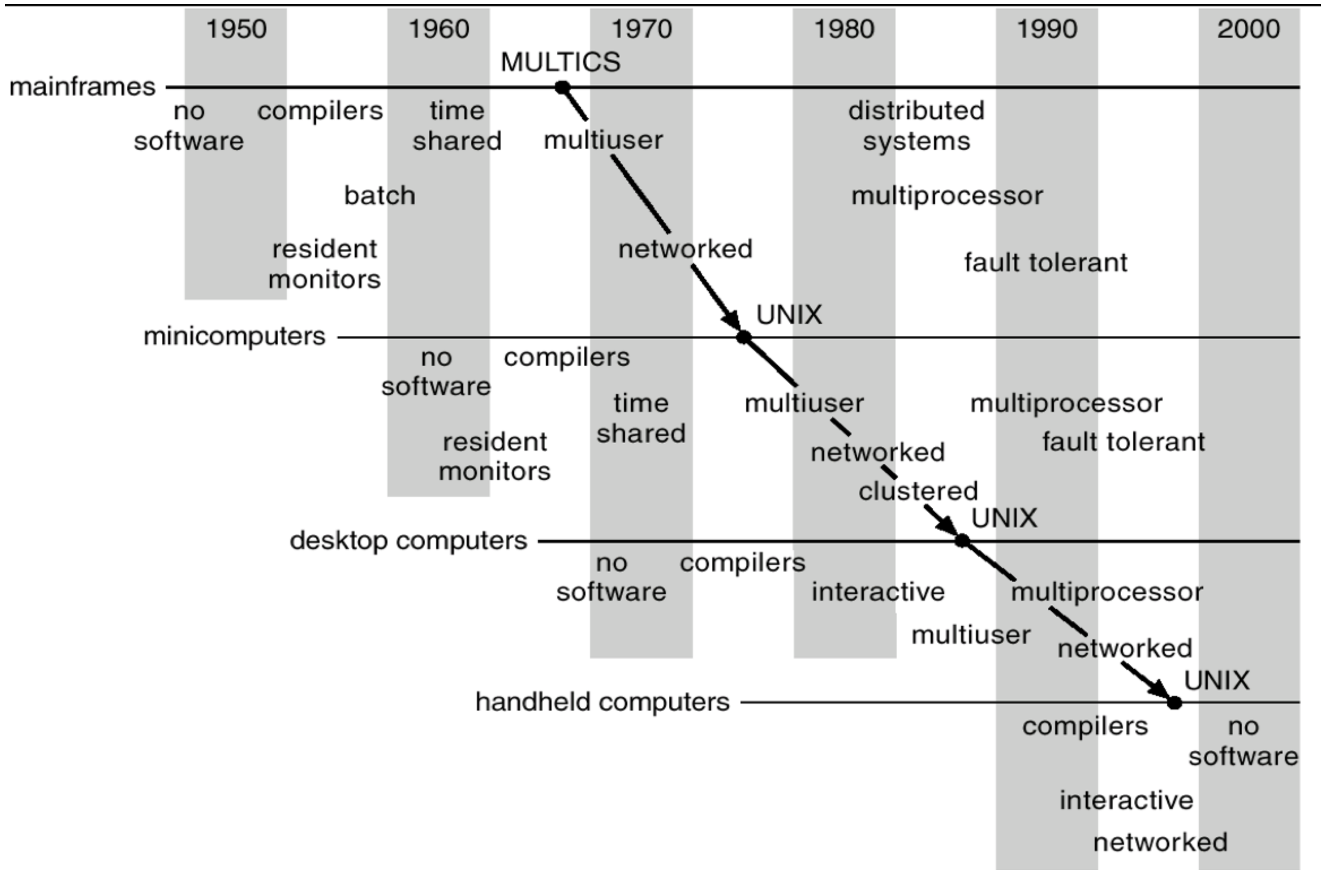
- OS responsibility
  - Look and feel of a system, particularly for non-experts
  - Tools that were distributed with the OS had significant business results

# Later evolution of computer systems – Mobile and Cloud

- 2000s-10s: Mobile and pervasive computing, Cloud Computing
  - Slow hardware is once again common (phones & wearables)
  - OS manages sensitive information like location and internet behavior
  - Fast flash storage is common.
  - Server hardware is shared by many different cloud computing customers

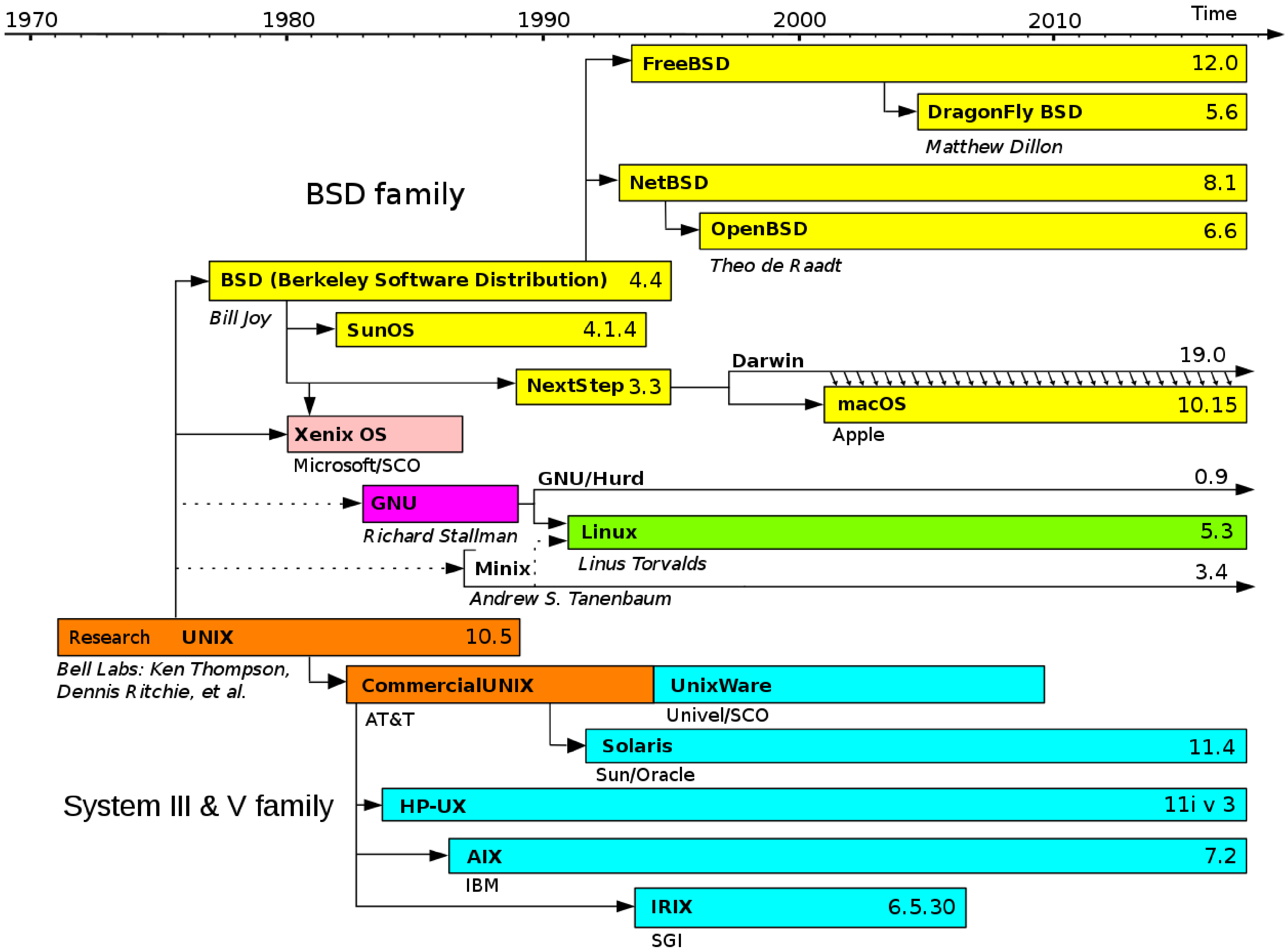
- OS responsibility
  - Diverse hardware drivers
  - Security
  - Massive parallelism

# Operating systems have evolved with hardware in a cycle




- Sophisticated operating systems first arose on mainframes.
- OS ideas migrated to smaller machines as those machines became more powerful.
- In 2019, a **smart watch** has 1gb RAM, 16gb SSD storage, two CPU cores, and a real OS.

# Simplified History of Unix-like Operating Systems




Operating systems are very interconnected

- Course Overview
- What is an OS?
- Operating Systems History
- **CS343 Focus**

# Schedule for first half of the course

1. Concurrency
   - Dealing with the realities of modern-day computing
   - Sources, Control, Challenges

2. Scheduling
   - Managing CPU utilization
   - Workload, Queuing, Real-time

# Schedule for second half of the course

3. Device Drivers
   - Management and abstraction of devices
   - Interrupts, DMA, Abstractions

4. Virtual Memory
   - Management and abstraction of memory
   - Paging, Allocation, Security

5. File Systems
   - Management and abstraction of data
   - Principles, Examples

# Why do we care about OS?

- Performance
  - Speed is influenced by
    - Parallelism, resource contention, memory management
    - Generally OS overhead
- Security
  - Process and data isolation when actually all running together
  - The biggest security vulnerabilities break abstractions
    - Meltdown and Spectre

- Course Overview
- What is an OS?
- Operating Systems History
- CS343 Focus

# Your first tasks

1. Getting Started Lab (due Thursday, September 24)

2. Fill out survey on Piazza

3. Find project partners for future labs