CS 318 Principles of Operating Systems

Fall 2018

Lecture 1: Introduction

Ryan Huang

Slides adapted from Geoff Voelker's (UCSD) and David Mazières' (Stanford) lectures

Bad News…

- **This is a TOUGH course**
  - "*Low level (C) programming absolutely necessary.*"
  - "*Need to be fearless about breaking code (and then fixing it later).*"
  - "*You should have a strong grasp of C for this class. Knowing assembly language is also a plus.*"
  - "*need to be confident in touching and modifying large systems of code*"
  - "*IT IS CHALLENGING*"

Bad News…

- **This is a TOUGH course**
- **Requires significant time commitment**
  - "*The projects are insanely time consuming*"
  - "*If you're worried about your course load this semester, maybe consider putting this class off for a later year*"
  - "*The workload is much much heavier than your average CS course…Be prepared to spend entire weeks working on nothing but the material for this course. If you start only one week in advance you WILL NOT finish without at least two all-nighters! I typically started two weeks out, was still stressed, and got an average of 3 hours of sleep every night on the weeks where a project was due.*"

Good News

- **There aren't many such hardcore courses in CS curriculum ☺**
  - You don't have to take it if you are not interested in it at all
- **It's hard, but rewarding in the end**
  - "*The project are very hard. But completing them is very rewarding.*"
  - "*I loved this course, it was very challenging but very satisfying and I learned a lot.*"
  - "*You learn a lot about operating systems and computers in general.*"
- **A highly valued skill after graduation**
- **We will try our best to help you**

Lecture 1 Overview

- **Course overview**
- **Administrative**
- **What is an Operating System?**
- **Walk-through of OS basics**

Quick Survey

- **How many juniors? seniors?**
- **Any non-CS majors?**
- **Graduate students?**
- **Why are you taking this class?**

Course Overview

- **An introductory course to operating systems**
  - Classic OS concepts and principles
  - Prepare you for advanced OS and distributed system course
  - OS concepts often asked in tech interview questions ☺

- **A practice course for hands-on experience with OS**
  - Four large programming assignments on a small but real OS
  - Reinforce your understandings about the theories

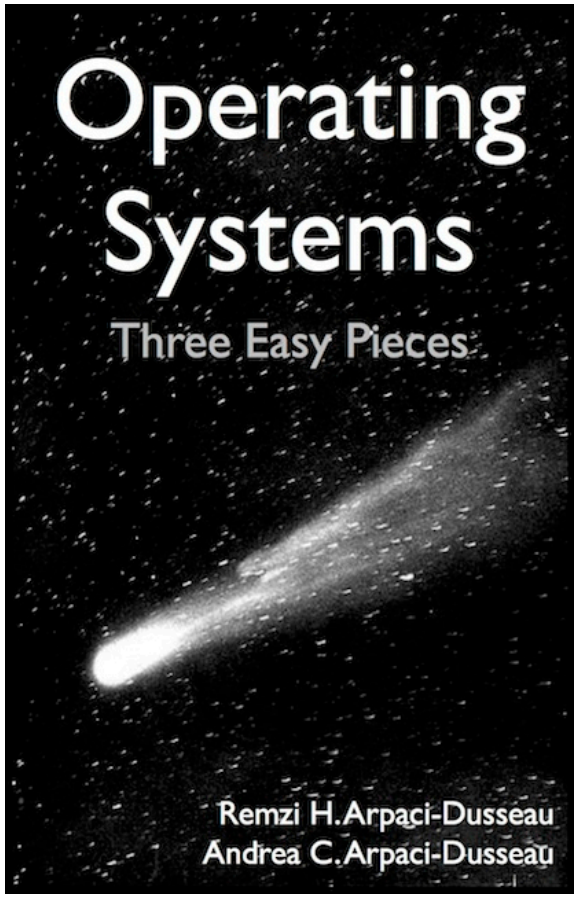
Topics Covered

- **Threads, Processes**
- **Concurrency, Synchronization**
- **Scheduling**
- **Virtual Memory**
- **I/O**
- **Disks, Filesystems**
- **Protection & Security**
- **Virtual Machines**

Course Materials

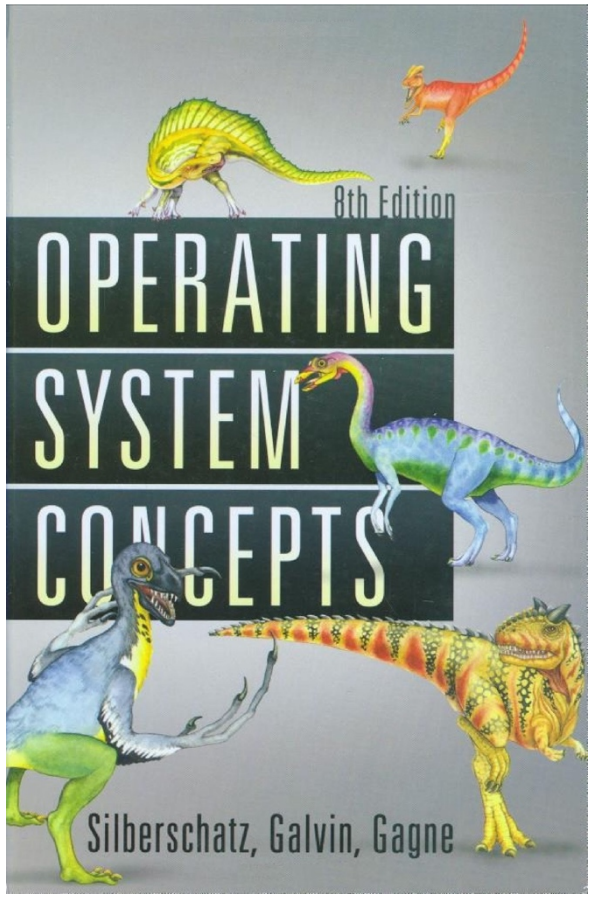
- **Course materials**
  - Lectures are the primary references
  - Textbooks are supplementary readings
  - Occasionally non-required papers

Textbook



Remzi Arpaci-Dusseau and **Andrea Arpaci-Dusseau**,
Operating Systems: Three Easy Pieces**,** Version 0.91

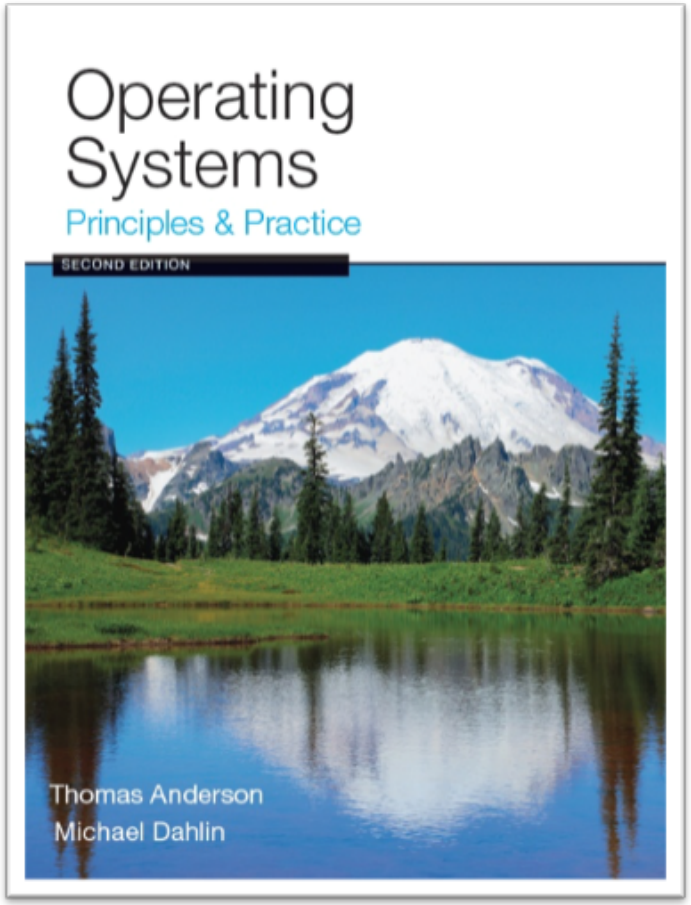
Textbook

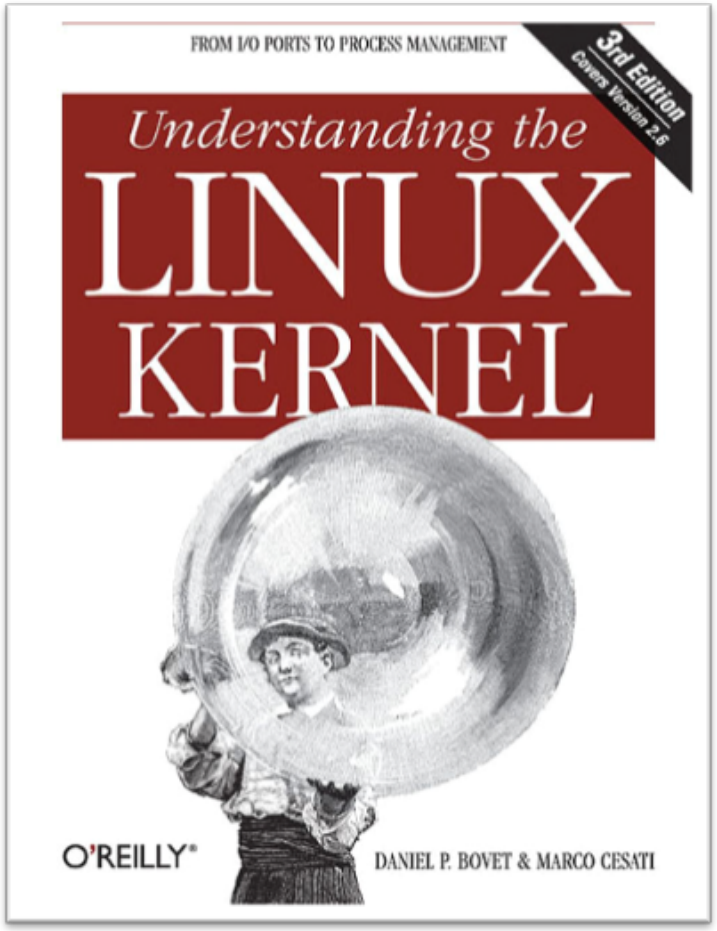


Silberschatz, Galvin and Gagne
Operating Systems Concepts

Textbook

Other Recommended Textbooks





Course Staff: Instructor

- **Prof. Ryan Huang**
  - Web: https://cs.jhu.edu/~huang
  - Office Hours: Tue 3-4pm, Thu 10:30-11:30am, Malone 231 (or by appointment)
- **Research Areas**
  - Operating Systems
  - Cloud and Mobile Computing
  - Specializes in Reliability and Availability
- **Research Lab**
  - https://orderlab.io
- **Close Interactions with Industry**

Course Staff: Teaching Assistants

- **Chang Lou**
  - Office Hours: Mon 4-5:30pm, Wed 3:30-5pm, Malone 122
- **Will Pryor**
  - Office Hours: Thu, Fri 5-6:30pm, Malone 122
- **Zach Silver**
  - Office Hours: Tue 5-6:30pm, Thu 3:30-5pm, Malone 122

Important Links

- **Course Website:**
  - https://www.cs.jhu.edu/~huang/cs318/fall18
  - Course syllabus and schedule
  - Lecture slides
  - Homework handouts
  - Project descriptions and references
- **Discussion Forum:**
  - https://piazza.com/class/jkysofgrqho5fh
- **Staff mail list:**
  - cs318-staff@cs.jhu.edu

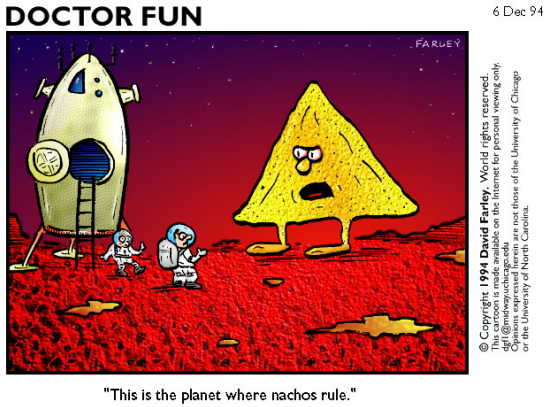
Homework

- **Five homework assignments throughout the semester**
  - help you check understanding about the lectures
  - prepare you for the exams
- **The homework assignments will *not* be graded**
  - solutions released ~a week later
  - amount learned from doing homework is proportional to effort
  - your choice on how much effort

Project Assignments

- **Implement parts of Pintos operating system**
  - Developed in 2005 for Stanford's CS 140 OS class
  - Written in C, built for x86 hardware
    - can run on a real machine!
  - Use hardware emulator (QEMU/Bochs) during development

pinto beans



nachos

Project Assignments (2)

- **One setup lab (lab 0)**
  - due next Thursday (done individually)
- **Four substantial labs:**
  - Threads, User processes, Virtual memory, File system
- **Implement projects in groups of up to 3 people**
  - Start picking your partners today
- **Warning: each project requires significant time to complete**
  - Don’t wait until the last minute to start!!

Project Assignments (3)

- **Automated tests**
  - All tests are given so you immediately know how well your solution performs
  - You either pass a test case or fail, there is *no* partial credit
- **Design document**
  - Answer important questions related to your design for a lab
- **Coding style**
  - Can your group member and TAs understand your code easily?

Project Design and Style

- **Must turn in a design document along with code**
  - Large software systems not just about producing working code
  - We supply you with templates for each project's design doc
- **TAs will manually inspect code**
  - e.g., must actually implement the design
  - must handle corner cases (e.g., handle `malloc` failure)
  - will deduct points for error-prone code
- **Code must be easy to read**
  - Indent code, keep lines and functions short
  - Use a consistent coding style
  - Comment important structure members, globals, functions

Project Lab Environment

- **The CS department ugrad and grad lab machines**
  - Running Linux on x86
  - The toolchain already setup
- **You may also use your own machine**
  - We have written instructions[1] for setting up the environment
  - Unix and Mac OS preferred. Windows needs additional setup
  - Final grading will be done on department lab machines
    - make sure to test your submission there

[1] https://www.cs.jhu.edu/~huang/cs318/fall18/project/setup.html

Exams

- **Midterm**
  - Covers first half of class + questions related to projects
  - Tentatively Tuesday, October 23rd
- **Final**
  - Covers second half of class + selected materials from first part
    - I will be explicit about the material covered
  - Also include project questions

Grading

- **Midterm: 15%**
- **Final: 35%**
- **Project: 50%**
  - Breakdown for five labs:
    - 601.418/618: 2%, 8%, 10%, 14%, 16%
    - 601.318: 2%, 12%, 15%, 21%, 6% (bonus points)
  - For each project
    - 60% based on passing test cases
    - 40% based on design document and style

Late Policies

- **Late submissions receive penalties as follows**
  - 1 day late, 10% deduction
  - 2 days late, 30% deduction
  - 3 days late, 60% deduction
  - after 4 days, no credit
- **Each team will have 72-hour grace period**
  - can spread into 4 projects
  - for interview, attending conference, errands, etc., no questions asked
  - use it wisely

Collaboration and Cheating Policies

- **Collaboration**
  - Explaining a concept to someone in another group
  - Discussing algorithms/testing strategies with other groups
  - Helping debug someone else's code (in another group)
- **Do not look at other people's solutions**
  - Including solutions online (e.g., GitHub)
  - We will run comprehensive tools to check for potential cheating.
- **Do not publish your own solutions**
  - online (e.g., on GitHub) or share with other teams
- **Cite any code that inspired your code**
  - If you cite what you used, it won't be treated as cheating
    - in worst case, we deduct points if it undermines the assignment

Do Not Cheat

- **It *will* be caught**
- **The consequence is very high**
- **The truth is: you always get better outcome by not cheating**

How *Not* to Pass CS 318?

- **Do not come to lecture**
  - The slides are online and the material is in the book anyway
  - Lecture is the basis for exams and directly relates to the projects

- **Do not do the homework**
  - It's not part of the grade
  - Concepts seem straightforward...until you apply them
  - Excellent practice for the exams, and project

How *Not* to Pass CS 318?

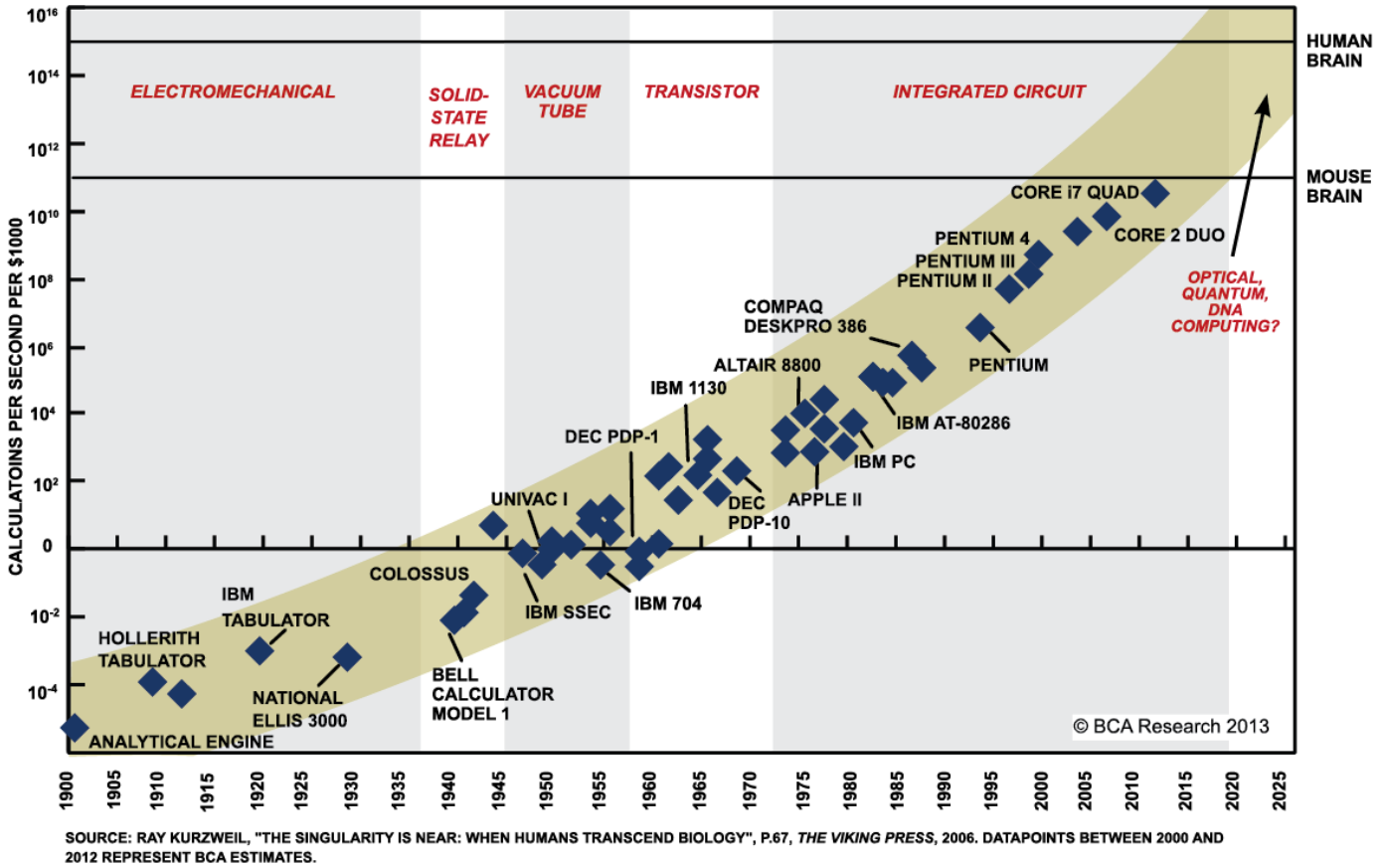
- **Do not ask questions in lecture, office hours or online**
  - It's scary, I don't want to embarrass myself
  - Asking questions is the best way to clarify lecture material
  - Office hours and email will help with homework, projects
- **Wait until the last couple of days to start a project**
  - We'll have to do the crunch anyways, why do it early?
  - The projects cannot be done in the last few days
  - Repeat: **The projects cannot be done in the last few days**
  - (p.s. The projects cannot be done in the last few days)

Questions

- **Before we start, any questions?**

Why Study Operating Systems?

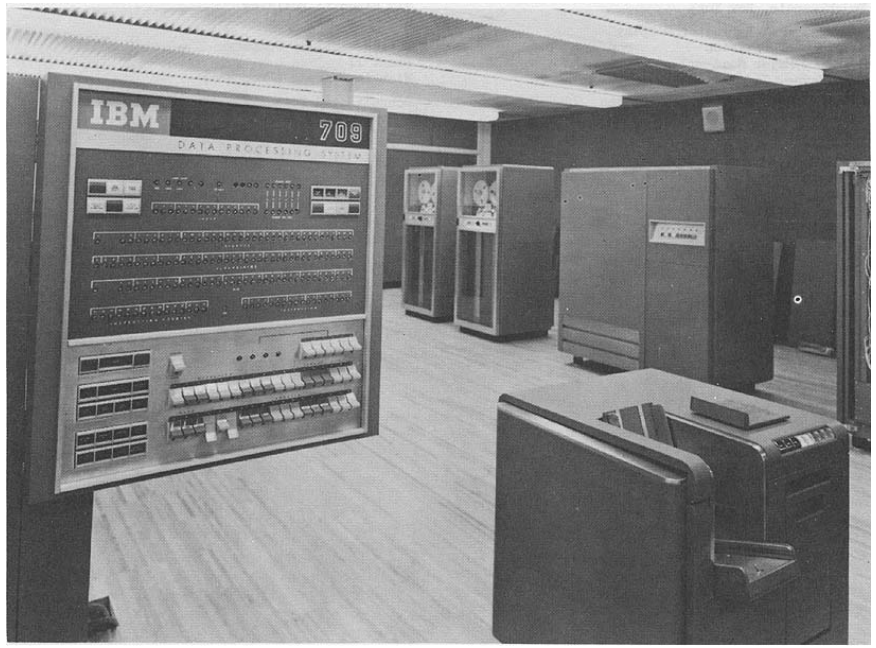
- **Technology trends**



SOURCE: RAY KURZWEIL, "THE SINGULARITY IS NEAR: WHEN HUMANS TRANSCEND BIOLOGY", P.67, *THE VIKING PRESS*, 2006. DATAPOINTS BETWEEN 2000 AND 2012 REPRESENT BCA ESTIMATES.

Why Study Operating Systems?

- **Technology trends**



IBM 709

CPU: ~4000 mult/div per sec.

memory: 32K 36-bit words

price: $2,630,000+

size: half room

CPU: 1.85 GHz dual-core

memory: 2 GB

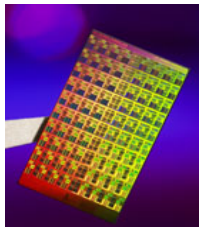
price: $329

size: 9.4 in × 6.6 in

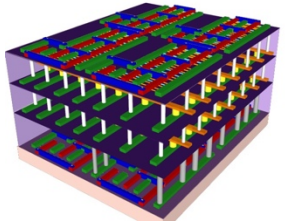
iPad

Why Study Operating Systems?

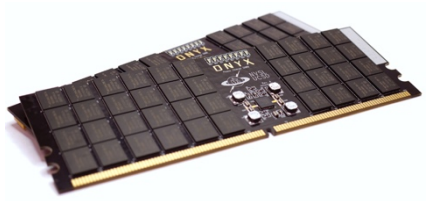
- **Technology trends**



manycore

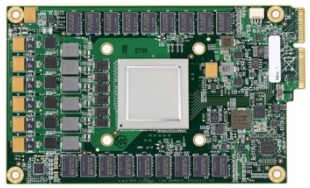


3D stacked chip



persistent memory

accelerators



Tensor Processing Unit

smartphones

IoT device

self-driving cars

robots

data centers

...

Why Study Operating Systems?

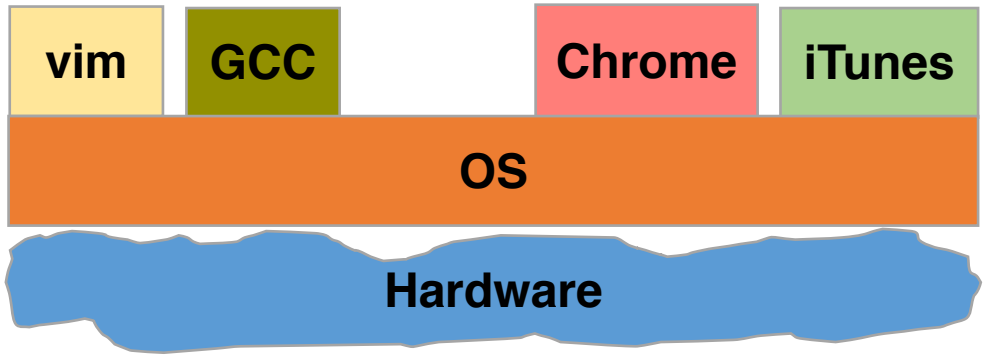
- **An exciting time for building operating systems** — some of you
  - New hardware, smart devices, self-driving cars, data centers, etc.
  - Facing OS issues in performance, battery life, security, isolation
- **Pervasive principles for systems in general** — many of you
  - Caching, concurrency, memory management, I/O, protection
- **Understand what you use** — all of you
  - System software tends to be mysterious
  - Understanding OS makes you a more effective programmer
- **Complex software systems** — all of you
  - Many of you will go on to work on large software projects
  - OSes serve as examples of an evolution of complex systems

What Is An Operating System?

- **Anyone?**
  - (Yes, I know that's why you're taking the course)
  - (Note: There are many answers)

What Is An Operating System?

- **Layer between applications and hardware**



- **All the code that you didn't have to write to implement your app**

OS and Hardware

- **Manage hardware resources**
  - Computation (CPUs)
  - Volatile storage (memory) and persistent storage (disk, etc.)
  - Communication (network, modem, etc.)
  - Input/output devices (keyboard, display, printer, camera, etc.)
- **Provides abstractions to hide details of hardware from applications**
  - Processes, threads
  - Virtual memory
  - File systems
  - …

OS and Hardware (2)

- **Mediate accesses from different applications**
  - Who has access at what point for how much/long
- **Benefits to applications**
  - Simpler (no tweaking device registers)
  - Device independent (all network cards look the same)
  - Portable (across Win95/98/ME/NT/2000/XP/Vista/7/8/10)

OS and Applications

- **Virtual machine interface**
  - The OS defines a logical, well-defined environment
  - Each program thinks it owns the computer
- **Provides protection**
  - Prevents one process/user from clobbering another
- **Provides sharing**
  - Concurrent execution of multiple programs (time slicing)
  - Communication among multiple programs (pipes, cut & paste)
  - Shared implementations of common facilities, e.g., file system

Questions to Ponder

- **What is part of an OS? What is not?**
  - Is the windowing system part of an OS?
  - Is the Web browser part of an OS?
  - This very question leads to different OS designs
- **How different are popular OSes today?**



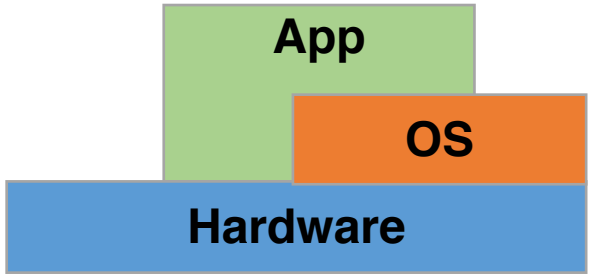

Questions to Ponder cont'd

- **OSes change all of the time**
  - Consider the series of releases of Windows, Linux, OS X
  - What drives the changes in OS?
  - What are the most compelling issues facing OSes today?
- **How many lines of code in an OS?**
  - Win7 (2009): 40M
  - OS X (2006): 86M
  - Linux (2011): 15M
  - What is largest kernel component?

Walk-through of OS basics

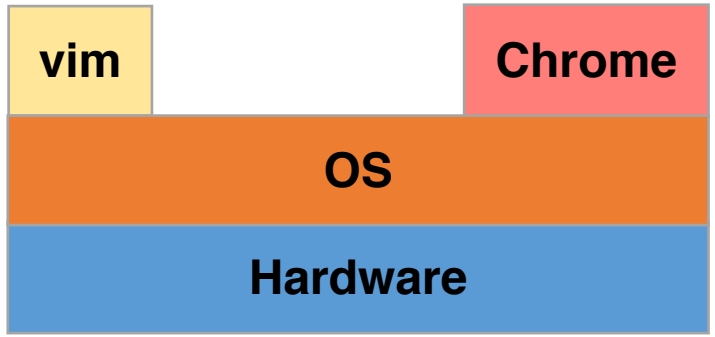
A Primitive Operating System

- **Just a library of standard services**



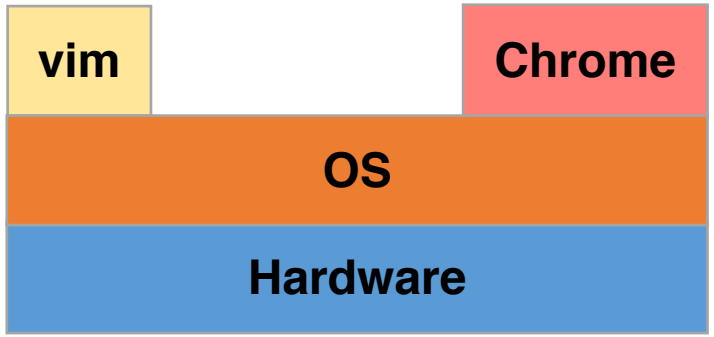
- **Simplifying assumptions**
  - System runs one program at a time
  - No bad users or programs
- **Problems: poor utilization**
  - ...of hardware (e.g., CPU idle while waiting for disk)
  - ...of human user (must wait for each program to finish)

Multitasking



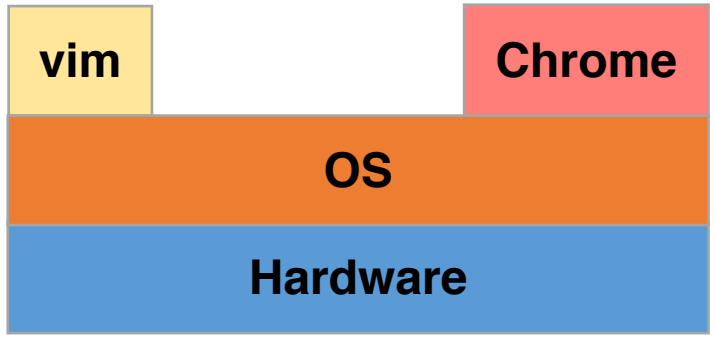
- **Idea: more than one process can be running at once**
  - When one process blocks (waiting for disk, network, user input, etc.) run another process
- **Mechanism: context-switch**
  - When one process resumes, it can continue from last execution point

Multitasking



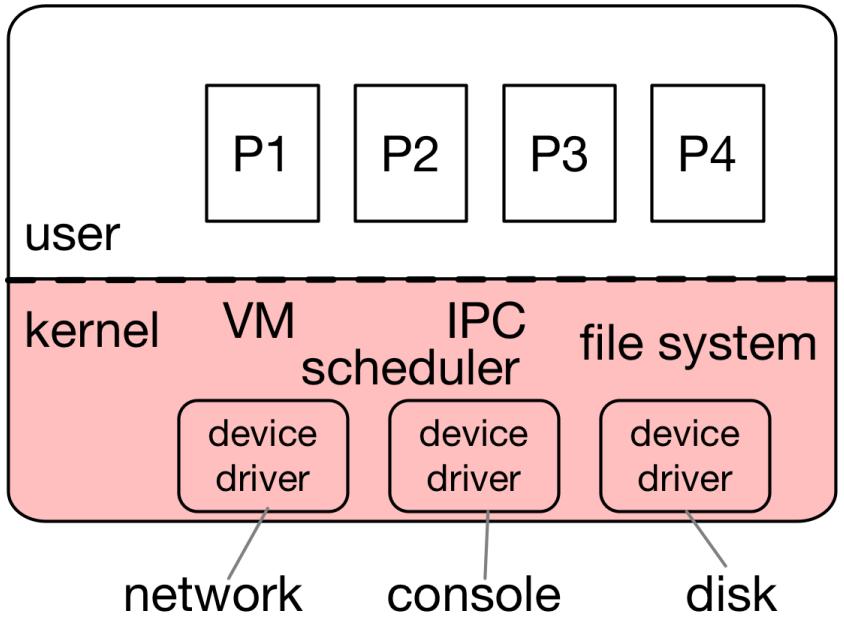
- **Idea: more than one process can be running at once**
- **Mechanism: context-switch**
- **Problems: ill-behaved process**
  - go into infinite loop and never relinquish CPU
  - scribble over other processes' memory to make them fail

Multitasking



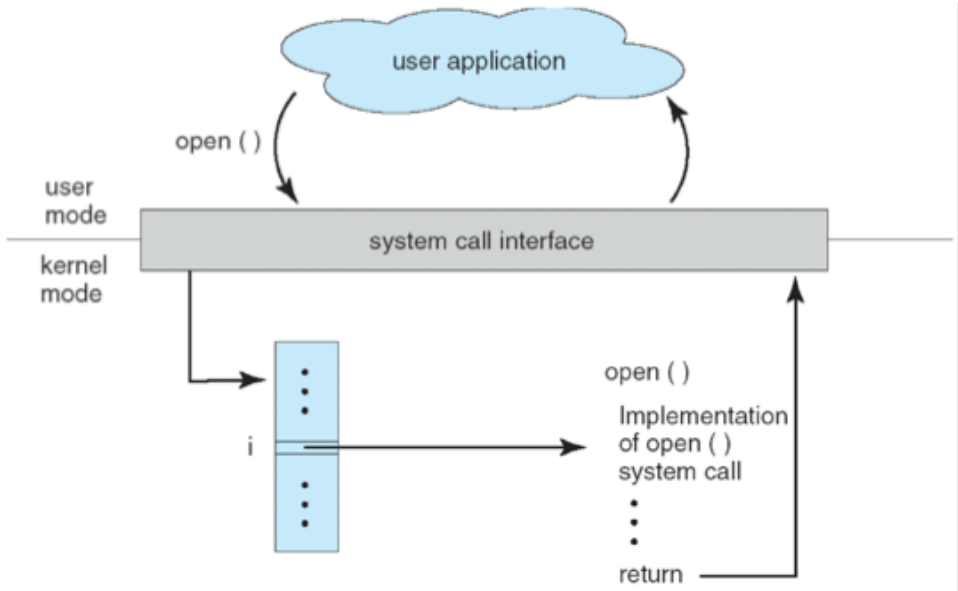
- **Problems: ill-behaved process**
  - go into infinite loop and never relinquish CPU
  - scribble over other processes' memory to make them fail
- **Solutions:**
  - **scheduling**: fair sharing, take CPU away from looping process
  - **virtual memory**: protect process's memory from one another

Typical OS Structure



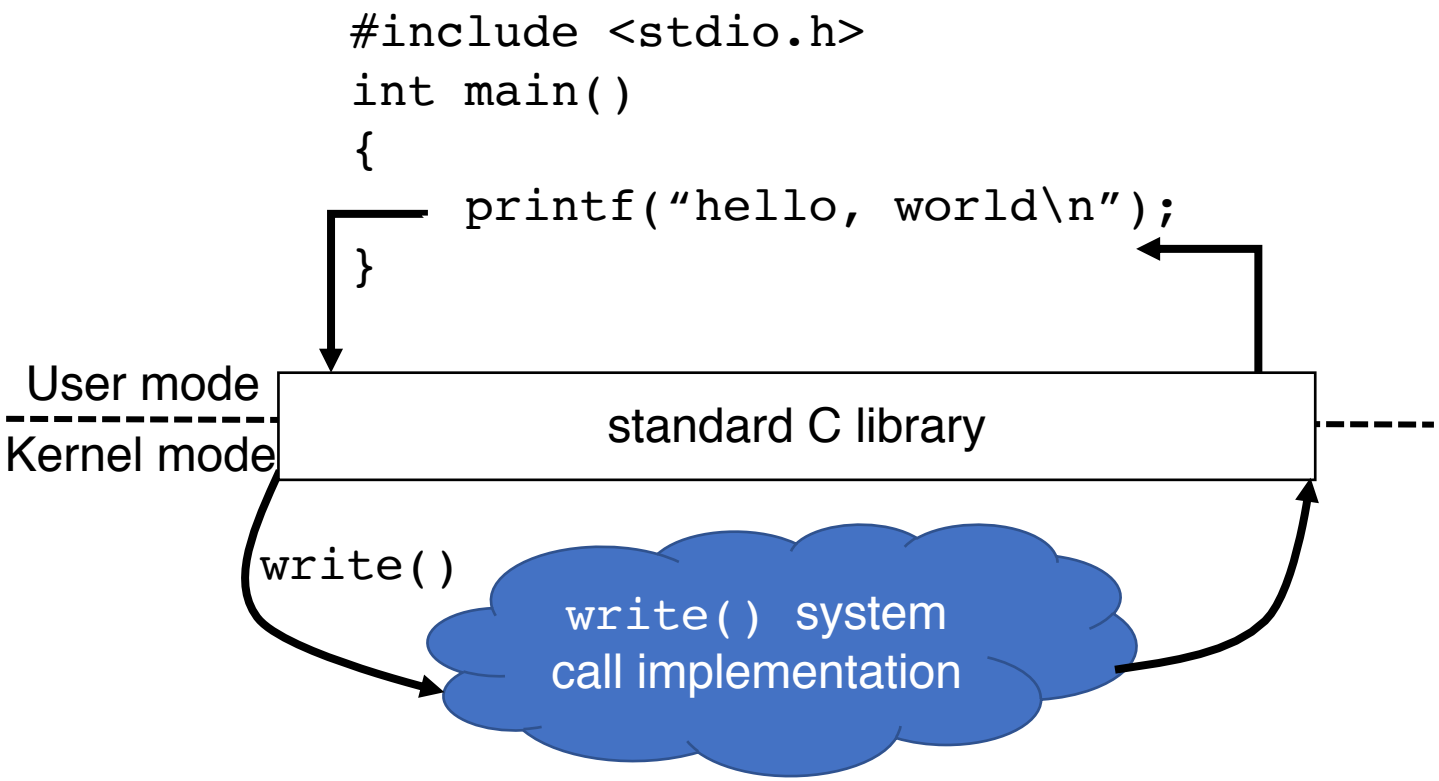
- **Most software runs as user-level processes (P[1-4])**
- **OS kernel runs in privileged mode (shaded)**

System Calls



- **Applications can invoke kernel through system calls**
  - Special instruction transfers control to kernel
  - ...which dispatches to one of few hundred syscall handlers

System Calls



- **Standard library implemented in terms of syscalls**

For Next Class...

- **Browse the course web**
  - https://www.cs.jhu.edu/~huang/cs318/fall18/
- **Read Chapters 1 and 2**
- **Setup Pintos and read its documentation**
  - Work on Lab 0
- **Looking for project partners**

For Next Class...

- **Browse the c**
  - https://www.c
- **Read Chapte**
- **Setup Pintos**
  - Work on Lab
- **Looking for p**