CS 153
Design of Operating Systems

Winter 23

Lecture 1: Introduction/Historical development

Instructor: Nael Abu-Ghazaleh
Slide contributions from
Chengyu Song, Harsha Madhyvasta and Zhiyun Qian

Class and Teaching Staff

- Class will be in person only
  - I expect labs as well, although I will confirm with TAs first

- Instructor: Nael Abu-Ghazaleh
  - I am a Professor in CSE and ECE
  - Office hours will be available online
    - Time will be announced
    - Hope to meet many of you during office hours

- TAs: Lian Gao, Xuezixiang Li and Zhenxiao Qi
  - Office hours TBA
  - They are leads for Labs

Class Resources

- Check class webpage for information
  - http://www.cs.ucr.edu/~nael/cs153/
- Lecture slides, homework, and projects will be posted on class webpage
- Assignment turn-in through Gradescope
  - Digital preferred, but if not please make sure legible
- Piazza for discussion forums; link on website
  - Stay on top of things – falling behind can snowball

Textbook

- Apraci-Dessau and Apraci-Dessau**, OS, 3 easy pieces** (required + free!)
  - Really well written book, rare in academic textbooks
  - Read! (especially if you can before class)

- Other pretty good books:
  - Anderson and Dahlin, ***Operating Systems: Principles and Practice***
  - Silberschatz, Galvin, and Gagne, ***Operating System Concepts***, John Wiley and Sons, 8th Edition

Class Mechanics Overview

- Grading breakdown
  - OS Fundamentals:
    - 4 homeworks (20% total)
    - Two exams: Midterm and Final (20% each)
  - OS projects (40% total)
    - Xv6 Operating system
    - Book uses examples from it
    - 4 projects (used to be 2, splitting into halves)
      - To keep the TA load under control, they will grade each two together
  - To pass class you must pass Fundamentals and Projects
  - Engagement/extra credit (2%+)
    - Includes attendance in lab. and lecture, participation on piazza, etc.
    - You learn much better if you are interested and engaged

Submission Policies

- Homeworks due on ilearn by the end of the day (will be specified on ilearn)

- Code and design documents for projects (if applicable) due by the end of the day (similarly will be specified on ilearn)

- Late policy (also on course webpage):
  - 4 slack days across all deliverables
    - Will use the ilearn submission timestamp to determine the days
    - 2% bonus to HW and Labs if you dont not use any of the slack days
  - 10% penalty for every late day beyond slack days

Projects

- Project framework: xv6
  - Projects are in C
  - Good debugging support
  - Used in OS class at several other universities

- Start to get familiar immediately
  - We will start labs. next week
  - Go over the xv6 documentation (on the course web page)
  - Optional Lab 0 to help get familiar with what xv6 is

Projects can be difficult!

- Reputation as a hard class in the CS curriculum because of projects (IMO)
  - You *must learn gdb* if you want to preserve your sanity! ☺
  - Hopefully you wont think its that hard by the time we are done

- Working on the projects will take a lot of time

- Biggest reason the projects are hard: legacy code
  - You have to understand existing code before you add more code
  - Preparation for main challenge you will face at any real job

Project logistics

- Projects *can* be done in groups of two or individually
  - When you have chosen groups, send your group info to your TA
  - Use the find a partner feature in piazza
    - email if unable to find partner and we'll try to connect
  - Option to switch partners after project two

- First step is to conceptually understand the project
  - Then come up with implementation plan
    - Fail and fail again
    - Debug, debug, debug (systems are unforgiving)
      - gdb is your friend
    - →success!!

Recipe for success in CS153

- Start early on projects
- Attend labs and office hours
  - Take advantage of available help
- Be engaged, interested, curious
- Make sure to attend lectures
  - Going over slides is not the same
- Try to read textbook material before class
- Ask questions when something is unclear
  - 2%+ participation and extra credit – may bump up your grade if on borderline. Face recognition ☺

Questions for today

- Why do we need operating systems course?
- Why do we need operating systems?
- What does an operating system need to do?
- Looking back, looking forward

Objectives of this class

- In this course, we will study problems and solutions that go into design of an OS to address these issues
  - Focus on concepts rather than particular OS
  - Specific OS for examples

- Develop an understanding of how OS and hardware impacts application performance and reliability

- Examples:
  - What causes your code to crash when you access NULL?
  - What happens behind a printf()?
  - Why can multi-threaded code be slower than single-threaded code?

Soap box – why you should care?

- Student surveys show low interest coming in
- Computers are an amazing feat of engineering
  - Perhaps the greatest human achievement
- You get to understand how they work
  - OS, Architecture, Compilers, PL, … are the magic that makes computers possible
- Ours is a young field
  - Our Euclids, Newtons, Darwins, … lived in the last half century
  - Many of our giants are still alive
  - So much innovation at an unbelievable pace
  - You can help write the next chapter

Why an OS class?

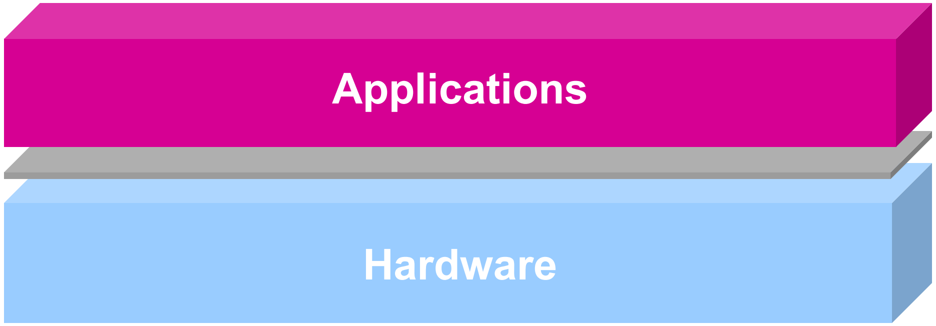
- Why are we making you sit here today, having to suffer through a course in operating systems?
  - After all, most of you will not become OS developers
- Understand what you use (and build!)
  - Understanding how an OS works helps you develop apps
  - System functionality, debugging, performance, security, etc.
- Learn some pervasive abstractions
  - Concurrency: Threads and synchronization are common modern programming abstractions (Java, .NET, etc.)
- Learn about complex software systems
  - Many of you will go on to work on large software projects
  - OSes serve as examples of an evolution of complex systems

Questions for today

- Why do we need operating systems course?
- Why do we need operating systems?
- What does an operating system need to do?
- Looking back, looking forward

Why have an OS?

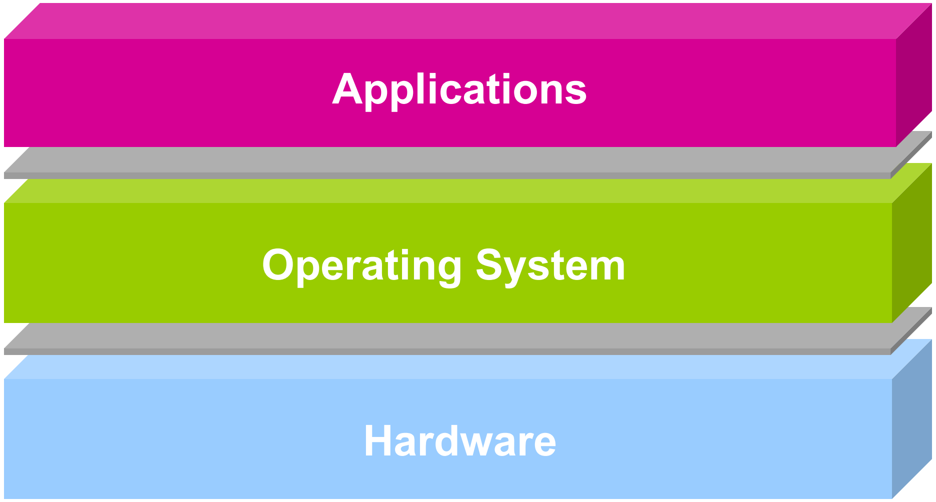
- What if applications ran directly on hardware?



- Problems:
  - Portability
  - Resource sharing

What is an OS?

- The operating system is the software layer between user applications and the hardware



- The OS is "*all the code that you didn't have to write*" to implement your application

Questions for today

- Why do we need operating systems course?
- Why do we need operating systems?
- What does an operating system need to do?
- Looking back, looking forward.

Roles an OS plays

- Beautician that hides all the ugly low level details so that anyone can use a machine (e.g., smartphone!)
- Wizard that makes it appear to each program that it owns the machine and shares resources while making them seem better than they are
- Referee that arbitrates the available resources between the running programs efficiently, safely, fairly, and securely
    - Managing a million crazy things happening at the same time is part of that – concurrency
- Elephant that remembers all your data and makes it accessible to you -- persistence

More technically...

- **Abstraction**: defines a set of logical resources (objects) and well-defined operations on them (interfaces)

- **Virtualization**: Isolates and multiplexes physical resources via spatial and temporal sharing

- **Access Control**: who, when, how
  - Scheduling (when): efficiency and fairness
  - Permissions (how): security and privacy

- **Persistence**: how to keep and share data