CS 153
Design of Operating Systems

Winter 2016

Lecture 1: Course Introduction

Instructor: Zhiyun Qian

Slides modified from

Harsha Madhyvasta and Nael Abu-Ghazaleh

Class Overview

- Monitor class webpage for information
  - http://www.cs.ucr.edu/~zhiyunq/cs153/
  - Will send out link to webpage

- Lecture slides, homeworks, and projects will be posted on class webpage
- Assignment turn-in through iLearn
  - Digital only, no paper copy (experiment)
- Announcements through iLearn and posted on class webpage
- Piazza for discussion forums; emails to be sent out soon

Textbooks

- Anderson and Dahlin, ***Operating Systems: Principles and Practice (required)***
- Andrew S. Tanenbaum, **Modern Operating Systems (recommended)**
- Silberschatz, Galvin, and Gagne, ***Operating System Concepts***, John Wiley and Sons, 8th Edition **(recommended)**

Class Overview

- Grading breakdown
  - 3 projects (15% each)
  - 3 homeworks (5% each)
  - Mid-term (15%)
  - Final (25%)
  - Extra credit (4%)

- Collaboration policy
  - Feel free to discuss with other students in class
  - But, every student should write solutions to homeworks independently and every project group should write code independently

Projects

- Project framework this time: Pintos
  - Projects are in C
  - Very good debugging support
  - Test cases come with the default code base
  - Used in OS class at several other universities

- You have first two weeks of the quarter to get familiar
  - Make sure to attend the first lab
  - Go over the Pintos documentation (on the course web page)

Projects are HARD!

- Probably the hardest class you will take at UCR in terms of development effort
- Working on the projects will take most of your time in this class
- Biggest reason the projects are hard: legacy code
  - You have to understand existing code before you can add more code
  - Preparation for main challenge you will face at any real job

Project recommendations

- Do not start working on projects at last minute!
  - You are graded for how well your code works, not for how many hours you have put in or how many lines of code you wrote
  - Debugging is integral process of development

- Make good use of help available
  - Post questions on piazza
  - Take advantage of TA office hours
  - Labs

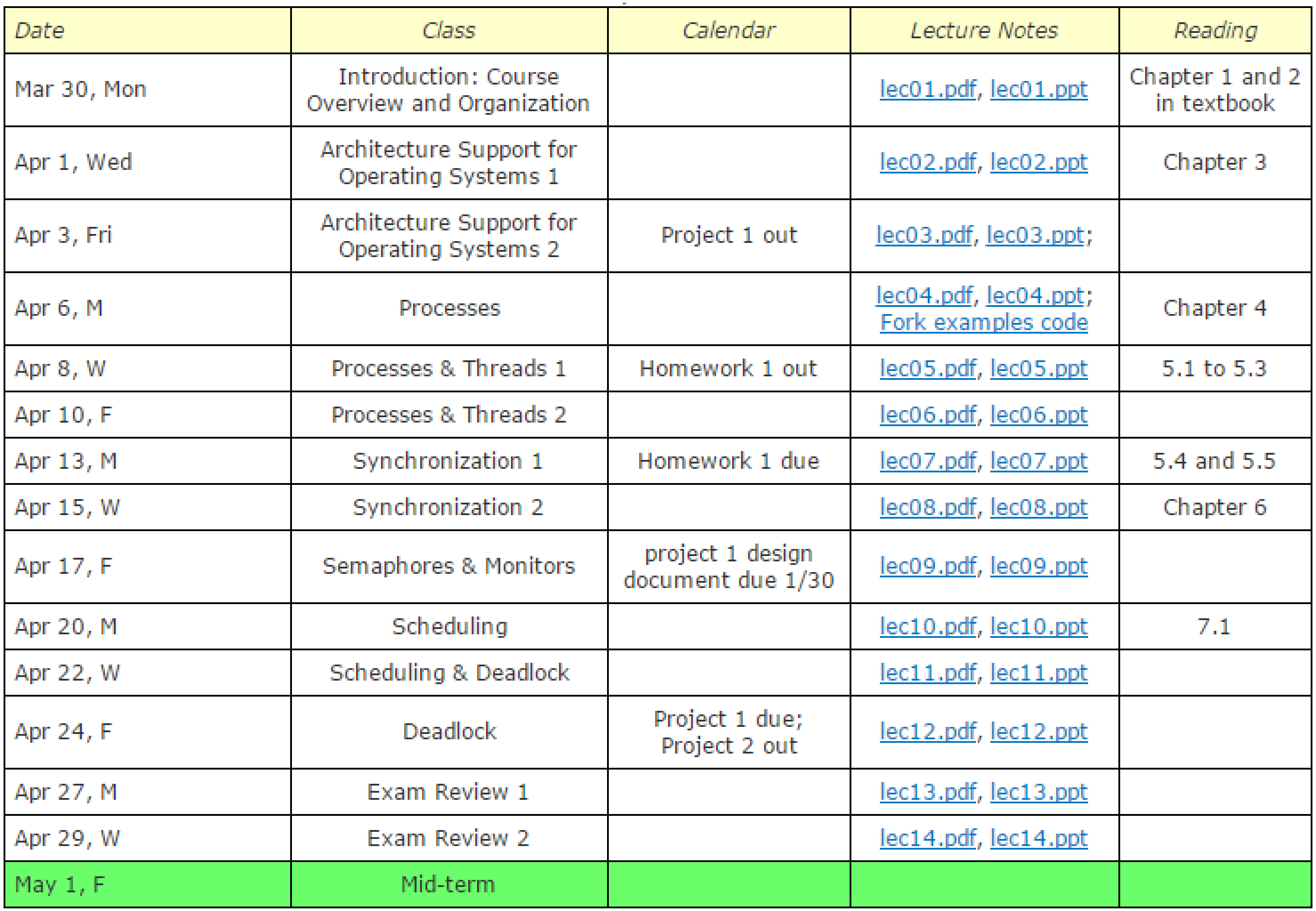
Project logistics

- Three projects to be done in groups of two
    - When you have chosen groups, send your group info to the TA for your lab
        - Joshua Frear
    - Send email if unable to find partner and we'll form groups
    - Option to switch partners after project one

- For every project, design document due a week before the project is due (5 points out of 15 points for project)
    - Walkthrough questions
    - Incentive to think through early what you need to do

Homeworks and Exams

- Three homeworks
  - Can expect similar questions in the exams
- Midterm (early May.)
  - In class
- Final
  - Covers second half of class + selected material from first part
    - I will be explicit about the material covered
- No makeup exams
  - Unless dire circumstances



Date	Class	Calendar	Lecture Notes	Reading
Mar 30, Mon	Introduction: Course Overview and Organization		lec01.pdf, lec01.ppt	Chapter 1 and 2 in textbook
Apr 1, Wed	Architecture Support for Operating Systems 1		lec02.pdf, lec02.ppt	Chapter 3
Apr 3, Fri	Architecture Support for Operating Systems 2	Project 1 out	lec03.pdf, lec03.ppt;	
Apr 6, M	Processes		lec04.pdf, lec04.ppt; Fork examples code	Chapter 4
Apr 8, W	Processes & Threads 1	Homework 1 out	lec05.pdf, lec05.ppt	5.1 to 5.3
Apr 10, F	Processes & Threads 2		lec06.pdf, lec06.ppt	
Apr 13, M	Synchronization 1	Homework 1 due	lec07.pdf, lec07.ppt	5.4 and 5.5
Apr 15, W	Synchronization 2		lec08.pdf, lec08.ppt	Chapter 6
Apr 17, F	Semaphores & Monitors	project 1 design document due 1/30	lec09.pdf, lec09.ppt	
Apr 20, M	Scheduling		lec10.pdf, lec10.ppt	7.1
Apr 22, W	Scheduling & Deadlock		lec11.pdf, lec11.ppt	
Apr 24, F	Deadlock	Project 1 due; Project 2 out	lec12.pdf, lec12.ppt	
Apr 27, M	Exam Review 1		lec13.pdf, lec13.ppt	
Apr 29, W	Exam Review 2		lec14.pdf, lec14.ppt	
May 1, F	Mid-term			

Submission Policies

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

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

- Late policy (also on course webpage):
  - 4 slack days across all three projects
    - Will use the ilearn submission timestamp to determine the days
    - 2% bonus point if you dot not use any of the slack days
  - 10% penalty for every late day beyond that

Recipe for success in CS153

- Start early on projects
- Attend labs and office hours
    - Take advantage of available help
- Make sure to attend lectures
    - Going over slides is not the same
- Read textbook material before class
- Ask questions when something is unclear
    - 4% participation and extra credit – may bump up your grade if on borderline. Face recognition ☺

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?

Questions for today

- Why do we need operating systems course?
- Why do we need operating systems?
- What does an operating system need to do?

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
  - Understanding how an OS works helps you develop apps
  - System functionality, debugging, performance, security, etc.
- Pervasive abstractions
  - Concurrency: Threads and synchronization are common modern programming abstractions (Java, .NET, etc.)
- 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

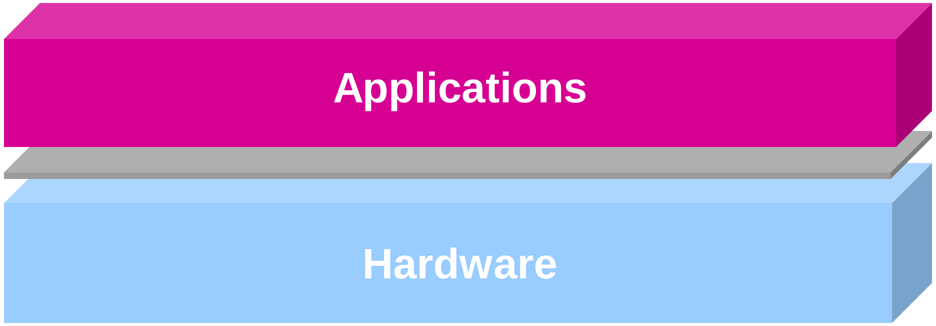
- Why do we need operating systems course?

- Why do we need operating systems?

- What does an operating system need to do?

Why have an OS?

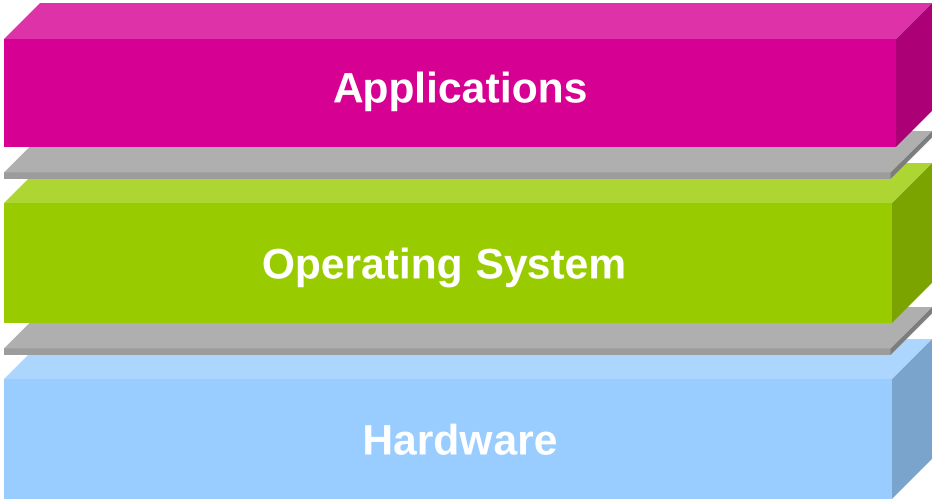
- What if applications ran directly on hardware?



- 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?

Roles an OS plays

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

More technically ☺: OS and Hardware

- The OS virtualizes/controls/mediates access to hardware resources
    - Computation (CPUs)
    - Volatile storage (memory) and persistent storage (disk, etc.)
    - Communication (network, modem, etc.)
    - Input/output devices (keyboard, display, printer, camera, etc.)
- The OS defines a set of logical resources (objects) and a set of well-defined operations on those objects (interfaces)
    - Physical resources (CPU and memory)
    - Logical resources (files, programs, names)
    - Sounds like OO…

The OS and Applications

- The OS defines a logical, well-defined environment…
  - Virtual machine (each program thinks it owns the computer)
- …for users and programs to safely coexist, cooperate, share resources

- Benefits to applications
  - Simpler (no tweaking device registers)
  - Device independent (all network cards look the same)
  - Portable (across Windows95/98/ME/NT/2000/XP/Vista/…)

Fundamental OS Issues

- The fundamental issues/questions in this course are:
  - Structure: how is an operating system organized?
  - Sharing: how are resources shared among users?
  - Naming: how are resources named (by users and programs)?
  - Protection: how are users/programs protected from each other?
  - Security: how can information access/flow be restricted?
  - Communication: how to exchange data?
  - Reliability and fault tolerance: how to mask failures?
  - Extensibility: how to add new features?

Other Questions to Ponder

- What is part of an OS? What is not?
  - Is the windowing system part of an OS? Java?
- Popular OSes today are Windows, Linux, and OS X
  - How different/similar do you think these OSes are?
- Somewhat surprisingly, OSes change all of the time
  - Consider the series of releases of Windows, Linux, OS X…
  - What are the drivers of OS change?
  - What are the most compelling issues facing OSes today?

Pondering Cont’d

- How many lines of code in an OS?
  - Vista (2006): 50M (XP + 10M)
    - What is largest kernel component?
  - OS X (2006): 86M
  - Debian 3.1 (2006): 213M
- What does this mean (for you)?
  - OSes are useful for learning about software complexity
  - OS kernel is only one component, however
    - Linux 3.6: 15M
    - KDE (X11): 4M
    - Browser : 2M+
  - OS is just one example of many complex software systems
    - If you become a developer, you will face complexity

How *Not* To Pass CS 153

- Do not come to lecture
  - It’s nice out, the slides are online, and the material is in the book anyway
  - Lecture material is the basis for exams and directly relates to the projects
- Do not ask questions in lecture, office hours, or email
  - It’s scary, I don’t want to embarrass myself
  - Asking questions is the best way to clarify lecture material at the time it is being presented
  - Office hours and email will help with projects

How *Not* To Pass (2)

- 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**
  - Each quarter groups learn that starting early meant finishing all of the projects on time…and some do not

Wrap-up Preliminaries

- Surefire steps to do poorly in CS 153
  - DON' T come to lecture
  - DON' T ask questions in class when unclear
  - DON' T start projects well in advance
  - DON' T come to office hours
- Any questions about the class structure, contents, etc.?

For next class...

- Browse the course web (especially Pintos docs)
  http://www.cs.ucr.edu/~zhiyunq/cs153
- Read chapters 1 and 2 in textbook
- Start …
  - … tinkering with Pintos
  - … finding a partner for project group

Blank