CS 318 Principles of Operating Systems

Fall 2017

Lecture 1: Introduction

Ryan Huang

Slides courtesy of Geoff Voelker, Yuanyuan Zhou, David Mazières
Lecture 1 Overview

• Course overview

• Administrative

• What is an Operating System?

• Walk-through of OS basics
Quick Survey

• How many graduate students?
• Any non-CS majors?
• 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 class

• **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

• **Course materials**
  - lectures are the primary references
  - textbooks, papers, and handout as supplementary readings
Topics Covered

• Threads, Processes
• Concurrency, Synchronization
• Scheduling
• Virtual Memory
• I/O
• Disks, Filesystems
• Protection & Security
• Virtual Machines
Who Am I?

• Prof. Ryan Huang
  - Web: https://cs.jhu.edu/~huang
  - Office Hours:Tue 4-5pm, Thu 11am-12pm, Malone 231

• Research Areas
  - Operating Systems
  - Cloud and Mobile Computing
  - Software Reliability

• Bio
  - PhD @UCSD, Postdoc @Microsoft Research
  - B.S. (Computer Science) and B.A. (Economics) @Peking University
Course Assistant Team

• **Head CA: (primarily project)**
  - Guoye Zhang
  - Office Hours: Mon, Wed 4:30-6pm, Malone 122 (ugrad lab)

• **CA: (homework + lecture)**
  - Ying Liu
  - Office Hour: TBD

• **CA: (discussion + lecture)**
  - Dewank Pant
  - Office Hour: TBD
• **Portal:**
  - [https://cs.jhu.edu/~huang/cs318/fall17/](https://cs.jhu.edu/~huang/cs318/fall17/)
  - Course syllabus and schedule
  - Lecture slides
  - Homework handouts
  - Project descriptions and references

• **Discussion forum:**
  - [https://piazza.com/jhu/fall2017/cs318418618](https://piazza.com/jhu/fall2017/cs318418618)

• **Staff mail list:**
  - [cs318-staff@cs.jhu.edu](mailto:cs318-staff@cs.jhu.edu)
Remzi Arpaci-Dusseau and Andrea Arpaci-Dusseau, *Operating Systems: Three Easy Pieces*, Version 0.91
Other Recommended Textbook

Operating Systems
Principles & Practice

Understanding the Linux Kernel

Thomas Anderson
Michael Dahlin

Daniel P. Bovet & Marco Cesati
Homework

• ~5 homework assignments throughout the semester
  - help you check understanding about the lectures
  - prepare you for the exams

• The 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 boot on your machine!
  - Use hardware emulator (QEMU/Bochs) during development

pinto beans

nachos
Project Assignments (2)

• One setup project (lab 0)
  - due next Thursday (done individually)

• Four implementation projects:
  - Threads, User processes, Virtual memory, File system

• Implement projects in groups of up to 3 people
  - Start picking your partners today
  - Git version control

• Automated tests

• Design document and style

• Warning: the projects require serious time commitments
  - Don’t wait until the last minute to start
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 provide instructions for setting up the environment
  - Unix and Mac OS preferred. Windows needs additional setup
  - testing will be done on lab machines
    - make sure to test your submission there
Exams

• **Midterm**
  - Covers first half of class + *something related to projects*
  - Tuesday, October 17th

• **Final**
  - Covers second half of class + selected materials from first part
    - I will be explicit about the material covered
  - Also include some project questions

• **No makeup exams**
  - Unless dire circumstances
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, 70% of score based on passing test cases
  - Remaining 30% based on design and style
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

• CAs 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
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**
  - as long as you cite what you used, it’s not cheating
    - in worst case, we deduct points if it undermines the assignment
How Not to Pass CS 318?

• Do not come to lecture
  - Lecture is early, 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 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)
Before we start, any questions about the class structure, contents, etc.?
Why Study Operating Systems?

- Technology trends
Why Study Operating Systems?

- Technology trends

IBM 709

<table>
<thead>
<tr>
<th>CPU</th>
<th>1.85 GHz dual-core</th>
</tr>
</thead>
<tbody>
<tr>
<td>memory</td>
<td>2 GB</td>
</tr>
<tr>
<td>price</td>
<td>$329</td>
</tr>
<tr>
<td>size</td>
<td>9.4 in × 6.6 in</td>
</tr>
</tbody>
</table>

iPad

<table>
<thead>
<tr>
<th>CPU</th>
<th>~4000 mult/div per sec.</th>
</tr>
</thead>
<tbody>
<tr>
<td>memory</td>
<td>32K 36-bit words</td>
</tr>
<tr>
<td>price</td>
<td>$2,630,000+</td>
</tr>
<tr>
<td>size</td>
<td>half room</td>
</tr>
</tbody>
</table>
Why Study Operating Systems?

- **Technology trends**

  - Manycore
  - 3D stacked chip
  - Persistent memory
  - Tensor Processing Unit
  - Smartphones
  - IoT device
  - Self-driving cars
  - Robots
  - Data centers
Why Study Operating Systems?

- An exciting time for building operating systems
  - New hardware, smart devices, self-driving cars, data centers, etc.
  - Facing OS issues in performance, battery life, security, isolation

- Pervasive abstractions and principles for systems in general
  - Caching, concurrency, memory management, I/O, protection

- Understand what you use
  - System software tends to be mysterious
  - Understanding OS makes you a more effective programmer (highly competitive in career)

- 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
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 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)
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

```
#include <stdio.h>
int main()
{
    printf("hello, world\n");
}
```

write() system call implementation
For Next Class...

• Browse the course web
  - https://cs.jhu.edu/~huang/cs318/fall17/

• Read Chapters 1 and 2

• Start exploring Pintos and its documentation
  - Work on Lab 0

• Thinking about partners for project groups
For Next Class...

• Browse the course web:
  - [https://cs.jhu.edu/~huang/cs318/fall17](https://cs.jhu.edu/~huang/cs318/fall17)
• Read Chapters 1 and 2
• Start exploring Pintos and its documentation
  - Work on Lab
• Thinking about partners for project groups

LET THE FUN IN OS BEGIN!