Lecture 1: Course Introduction

CSE 120: Principles of Operating Systems
Alex C. Snoeren

Lab 0 Due Thursday 4/10
Lecture 1 Overview

- Class overview
- What is an operating system?
- Operating systems and hardware
- Operating systems and applications
Personnel

- Instructor: Alex C. Snoeren
  - Office hours Thursdays 2-3pm or by appointment
  - EBU3B 3114

- Discussion TA: Chengmo Yang
  - Discussion Wed 1-1:50pm in WLH 2204
  - Office hours Tue/Wed 5:30-6:30pm EBU3B B240A
  - Homework grader

- Project TA: Kaisen Lin
  - Lab hours Mon/Wed 2-4pm in EBU3B B230
  - Office hours by appointment in EBU3B 3242
  - Project grader (assisted by Alvin AuYoung)
CSE 120 Class Overview

- Course material taught through class lectures, textbook readings, and handouts

- Course assignments are
  - Homework questions (mostly from the book)
  - Three large programming projects in groups

- Discussion sections are a forum for asking questions
  - Lecture material and homework
  - Additional OS topics (e.g., how does an OS boot?)

- Discussion board [http://webboard.ucsd.edu](http://webboard.ucsd.edu)]
  - The place to ask questions about lecture, hw, projects, etc.
Homeworks

- There will be 4 homeworks throughout the quarter
  - Reinforce lecture material…no better practice

- Collaboration vs. cheating
  - You *should* discuss homework problems with others
    » You can learn a lot from each other
  - But there is a distinction between collaboration and cheating
  - Rule of thumb: Discuss together in library, walk home, and write up answers independently
  - Cheating is copying from other student’s homeworks or solution sets, searching for answers on the Web, etc.
  - Suspicious homeworks will be flagged for review
Textbook

Nachos Project

DOCTOR FUN

"This is the planet where nachos rule."

6 Dec 94
Nachos

- Nachos is an instructional operating system
  - It is a user-level operating system and a machine simulator
    » Not unlike the Java runtime environment
    » Will become abundantly clear (or not so clear) very soon
  - Programming environment will be Java on Unix (Linux/Solaris)
  - The projects will require serious time commitments
    » Waiting until the last minute is not an option

- You will do three projects using Nachos
  - Concurrency and synchronization
  - Multiprogramming
  - Virtual memory

- You will work in groups of 1-4 on the projects
  - Start identifying partners now
Labs

- We will use the uAPE (B230) lab in the basement of the CSE/EBU3B building
  - Linux running on Intel machines
- You can also use your home machine
  - The project source will work on Windows/OS X (with caveats)
  - Graders will test on uAPE machines
  - Be sure to test your projects there as well
Exams

- Midterm
  - Tuesday, May 6th
  - Covers first half of class

- Final
  - Monday, June 9th
  - Covers second half of class + selected material from first part
    » I will be explicit about the material covered

- No makeup exams
  - Unless dire circumstances (we all want to start vacation early)

- Crib sheet
  - You can bring one double-sided 8.5x11” page of notes to each exam to assist you in answering the questions
  - Not a substitute for thinking
Grading

- Homeworks: 20%
  - Think of these collectively as a take-home midterm
- Midterm: 20%
- Final: 30%
- Projects: 30%
  - Each Nachos project is 10% of your final grade
How Not To Pass CSE 120

- 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 do the homework
  - It’s only 20% of the grade
  - Excellent practice for the exams, and some homework problems are exercises for helping with the project
  - 20% is actually a significant fraction of your grade (difference between an A and a C)
How Not To Pass (2)

• Do not ask questions in lecture, office hours, or email
  ◆ Professor is 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 homeworks, 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 couple of days
  ◆ Repeat: The projects cannot be done in the last couple of days
  ◆ Some groups last time learned that starting early meant finishing all of the projects on time…and some didn’t
Class Web Page

http://www.cs.ucsd.edu/classes/sp08/cse120/

- Serves many roles…
  - Course syllabus and schedule (updated as quarter progresses)
    » Lecture slides
  - Announcements
  - Homework handouts
  - Project handouts (tons of info on Nachos, start now)
Questions

- Before we start the material, any questions about the class structure, contents, etc.?
Why Operating Systems?

- Why are we making you sit here today, having to suffer through a core course in operating systems?
  - It’s not like everyone will become OS developers, after all
- Understand what you use
  - Understanding how an OS works helps you develop apps
  - System functionality, performance, efficiency, 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
This course addresses classic OS concepts

- Services provided by the OS
- OS implementation on modern hardware
- Co-evolution of hardware and software
- Techniques for implementing software systems that are
  » Large and complex
  » Long-lived and evolving
  » Concurrent
  » Performance-critical

System software tends to be mysterious

- Virtual memory? Wazzat?

Our goal is to reveal all mysteries
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?
Fundamental OS Issues (2)

- **Concurrency**: how to control parallel activities?
- **Performance**: how to make efficient use of resources, reduce OS overhead?
- **Scale and growth**: how to handle increased demand?
- **Compatibility**: can we ever do anything new?
- **Distribution**: how to coordinate remote operations?
- **Accountability**: how to charge for/restrict use of resources?

And the **principles** in this course are the design methods, approaches, and solutions to these issues.
What is an operating system?

- How would you answer?
- (Note: There is no one good answer)
What is an operating system?

- 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.
The OS and Hardware

- The OS abstracts/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)
The OS and Hardware (2)

- Benefits to applications
  - Simpler (no tweaking device registers)
  - Device independent (all network cards look the same)
  - Portable (same program on Windows95/98/ME/NT/2000/…)
  - Transportable (same program across different OSes (Java))
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
  - Concurrent execution of multiple programs (timeslicing)
  - Communication among multiple programs (pipes, cut & paste)
  - Shared implementations of common facilities
    » No need to implement the file system more than once
  - Mechanisms and policies to manage/share/protect resources
    » File permissions (mechanism) and groups (policies)
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?
  - How would you go about answering that question?
- Somewhat surprisingly, OSes change all of the time
  - Consider the series of releases of WinNT, Linux, OS X…
  - What are the drivers of OS change?
  - What are the most compelling issues facing OSes today?
- How many lines of code in an OS?
  - WinXP (2002): 40M (what is largest component?)
  - Linux (2001): 30M – kernel 2.4M; X11 1.8M; browser 2M
For Next Class...

- Browse the course web
  - http://www.cs.ucsd.edu/classes/sp08/cse120/
- Read Chapters 1 and 2
- Start thinking about partners for project groups
- Log into your account and check out Lab 0
- Chengmo will not be holding a discussion section on Wednesday—starts next week