CSE 120
Principles of Operating Systems
Spring 2019
Lecture 1: Course Introduction
Geoffrey M. Voelker
Lecture 1 Overview

• Class overview
• Administrative info
• Introduction to operating systems
Personnel

• Instructor
  ♦ Geoff Voelker
    » Office hours: Wed 4-5pm & Fri 2-3pm

• TAs and Tutors
  ♦ Ujwal Bachiraju (TA)
  ♦ Keerthana Ganesan (TA)
  ♦ Ruohan Hu (Tutor)
  ♦ Naomi McCracken (Tutor)
  ♦ Erin McGinnis (TA)
  ♦ Christopher Weaver (Tutor)

• Discussion
  ♦ Mon @ 3pm in Warren 2005 (but not this week)
CSE 120 Class Overview

- Course material taught through class lectures, textbook readings, and handouts
- Course assignments are
  - Homework questions
  - Three large programming projects in groups
  - Midterm and final exams
- Discussion sections are a forum for asking questions
  - Lecture material and homework
- Other forums
  - Piazza, basement labs
Textbook


FREE

*Operating Systems*

Remzi Arpaci-Dusseau
Andrea Arpaci-Dusseau
Homeworks

- There will be 4 homeworks throughout the quarter
  - Reinforce lecture material
- Homeworks provide practice learning the material
  - Unfortunately, wasted a lot of time and energy dealing with homework cheating in the past
  - So: You get full credit for a technical answer related to the homework question
  - Amount learned from doing homework is proportional to effort
  - Your choice on how much effort
Nachos Project

"This is the planet where nachos rule."
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 more clear very soon
  - Programming environment will be Java on Unix (Linux)
  - The projects will require serious time commitments
    - Waiting until the last minute is not a viable option

- You will do three+ projects using Nachos
  - Concurrency and synchronization
  - System calls, processes, multiprogramming
  - Virtual memory

- You will work in groups of 1-3 on the projects
  - Start thinking about partners
Labs

• We will use the labs in the CSE basement
  ♦ Linux running on x86 machines

• You may also use your home machine
  ♦ The same project source will work on Windows (mostly)
  ♦ Note: We will test and grade on ACMS machines
  ♦ Be sure to test your projects there as well
    » You will be able to test before the deadline

• Why work in the labs?
  ♦ Classmates there to help (and have fun)
  ♦ TAs there to help (will have posted hours in the lab)
  ♦ I will visit the labs to help
Exams

• Midterm
  ♦ Tuesday April 30th (put in your calendar)
  ♦ Covers first half of class

• Final
  ♦ Thursday June 13th (put in your calendar)
  ♦ Covers second half of class + selected material from first part
    » I will be explicit about the material covered

• No makeup exams
  ♦ Everyone must be able to attend these exam dates
    » Unless absolute dire circumstances

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

- Homeworks: 6%
- Midterm: 28%
- Final: 33%
- Projects: 33%
How *Not* To Pass CSE 120

- Do not come to lecture
  - Lecture is far too 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 only 6% of the grade, get full credit for turning anything in
  - Concepts seem straightforward…until you apply them
  - Excellent practice for the exams, and some homework problems are exercises for helping with the project
- Violate academic integrity
  - It is much better to get a 0 for an assignment than to fail the course for academic integrity violations
How *Not* To Pass Even More

- 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 at the time it is being presented
  - 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
Project 1 Scores

START DATE

SCORE

mean  median

DAY.0-2  DAY.2-4  DAY.4-6  DAY.6-8  DAY.8-10  DAY.10-12  DAY.12-14  DAY.14-16
Class Web Page

http://cseweb.ucsd.edu/classes/sp19/cse120-a/

- Serves many roles…
  - Course syllabus and schedule (updated over quarter)
  - Lecture slides
  - Homework handouts
  - Project handouts

- Optional material
  - Entirely for your interest only

- Supplemental readings on Unix, monitors, and threads
  - e.g., seminal research paper describing the early Unix system
  - Concepts in paper might seem obvious and familiar, but they were new at one time
Questions

• Before we start the material, any questions about the class structure, contents, etc.?
Podcasts

• We will have podcasts to supplement the live lectures
  ♦ They are not there as a substitute
• If 8am lectures are the hardest thing about the course, then CSE 120 is going to be easy
Why?

You have a question, Calvin?

Yes! What assurance do I have that this education is adequately preparing me for the 21st century?

Am I getting the skills I'll need to effectively compete in a tough, global economy? I want a high-paying job when I get out of here! I want opportunity!
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, C#, C++, Rust, etc.)
- Complex software systems
  - Many of you will go on to work on large software projects
  - OSes serve as examples of complex systems
This course addresses classic OS concepts

- Services provided by the OS
- OS implementation on modern hardware
- Interaction 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 those mysteries
```
top - 20:48:08 up 275 days,  1 user,  load average: 0.06, 0.07, 0.05
Tasks: 171 total, 1 running, 0 sleeping, 170 stopped, 0 zombie
Cpu(s): 0.1%us, 0.1%sy, 0.0%iow, 0.0%hi, 0.0%si, 0.0%st
Mem: 16467276k total, 1415968k used, 2307248k free, 884340k cached
Swap: 0k total, 0k used, 884340k cached

PID USER   PR NI VIRT  RES  SHR %CPU %MEM    TIME+ COMMAND
14677 voelker  20   0  55548 3232  2364  R   0:00.07  top
24637 voelker  20   0  86300 6364 10244  S   32:06.70 mosh-server
  1 root  20   0 57812 1636  584  S   1:26.73 init
  2 root  20   0    0    0    0  R   0:03.13 kthreadd
  3 root  20   0    0    0    0  R   0:04.38 migration/0
  4 root  20   0    0    0    0  R   0:54.94 ksoftirqd/0
  5 root  20   0    0    0    0  R   0:00.01 watchdog/0
  6 root  20   0    0    0    0  R   0:04.39 migration/1
  7 root  20   0    0    0    0  S   11:22.89 ksoftirqd/1
  8 root  20   0    0    0    0  S   0:00.01 watchdog/1
  9 root  20   0    0    0    0  S   0:18.05 migration/2
 10 root  20   0    0    0    0  S   0:44.37 ksoftirqd/2
 11 root  20   0    0    0    0  R   0:00.01 watchdog/2
 12 root  20   0    0    0    0  R   0:18.06 migration/3
 13 root  20   0    0    0    0  S   0:01.67 ksoftirqd/3
 14 root  20   0    0    0    0  S   0:00.01 watchdog/3
 15 root  20   0    0    0    0  S   0:30.93 events/0
```
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?
  ♦ (Yes, I know that’s why you’re taking the course…)
  ♦ (Note: There are many answers…)
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)
  - Sounds like OO…
The OS and Hardware (2)

- 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/…)
  - Transportable (same program across different OSes (Javascript))
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)
More 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?
More 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?

• 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?
More 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?

- 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?

- 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?
  - Win7 (2009): 40M
  - OS X (2006): 86M
  - Linux (2011): 15M
  - What is largest kernel component?

- What does this mean (for you)?
  - OSes are useful for learning about software complexity
  - OS is just one example of many complex software systems
    » Chrome (2015): 17M
    » Hadoop (2018): 3.9M
    » JDK (2015): 6M
    » Unreal Engine 3: 2M
  - If you become a developer, you will face complexity
For next class...

- Browse the course web
  
  http://cseweb.ucsd.edu/classes/sp19/cse120-a/

- Sign up on Piazza!
- Read Chapters 1 and 2
- No discussion this week
- Start thinking about partners for project groups

- Let the fun begin!