

CSE 120

Principles of Operating Systems

Fall 2014

Lecture 1: Course Introduction

Geoffrey M. Voelker



8:00 AM!

The image shows a large, empty lecture hall or auditorium. Rows of black seats with orange cushions are arranged in a tiered fashion, facing towards the front. At the front of the room, there is a long black chalkboard mounted on a wall with a wood-grain texture. The ceiling is white with several long, recessed fluorescent light fixtures. To the right, a white wall features horizontal grey panels. In the background, a wooden door is visible, and a small black speaker is mounted on the wall. The overall atmosphere is quiet and ready for a lecture.

Lecture 1 Overview

- Class overview, administrative info
- What is an operating system?

Personnel

- Instructor
 - ◆ Geoff Voelker
 - » Office hours: Mon 3-4pm in CSE 3108
- TAs
 - ◆ He “Lonnie” Liu
 - ◆ Peng “Ryan” Huang
 - ◆ Mohit Kothari
- Discussion
 - ◆ Fri @ 2pm, Center 119
 - ◆ No discussion first week

CSE 120 Class Overview

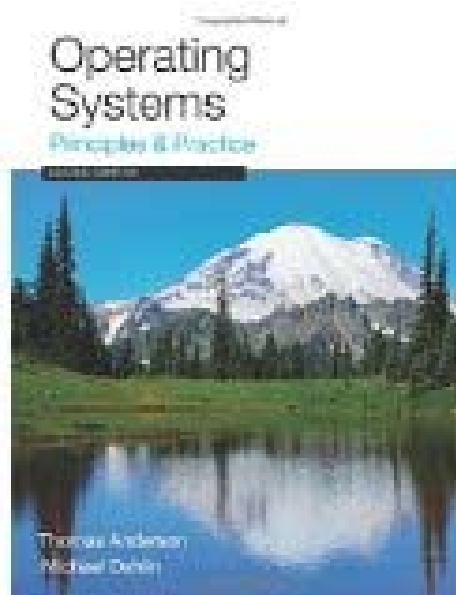
- Course material taught through class lectures, textbook readings, and handouts
- Course assignments are
 - ◆ Homework questions (primarily from the book)
 - ◆ Three large programming projects in groups
- Discussion sections are a forum for asking questions
 - ◆ Lecture material and homework
- Other forums
 - ◆ Mailing list (cse120@cs.ucsd.edu)
 - ◆ Discussion board (<http://piazza.com>)

Homeworks

- There will be 4-5 homeworks throughout the quarter
 - ◆ Reinforce lecture material...no better practice
- Collaboration vs. cheating
 - ◆ I encourage you to 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

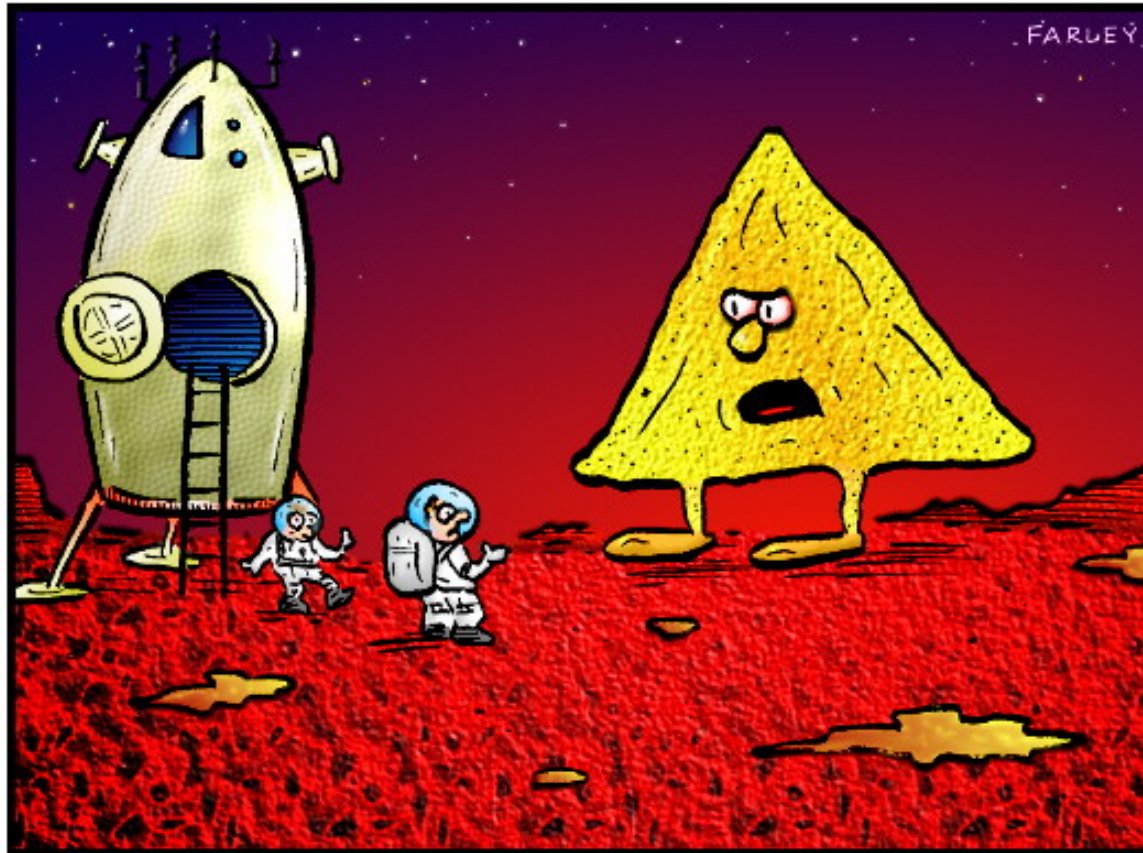
- Anderson and Dahlin, *Operating Systems: Principles and Practice*, 2nd Edition, Recursive Books, ISBN 978-0985673529 (2014)



Nachos Project

DOCTOR FUN

6 Dec 94



© Copyright 1994 David Farley. World rights reserved.
This cartoon is made available on the Internet for personal viewing only.
dgfl@midway.uchicago.edu
Opinions expressed herein are not those of the University of Chicago
or the University of North Carolina.

"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 abundantly clear (or not so clear) very soon
 - ◆ Programming environment will be C++ on Unix (Linux)
 - ◆ 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 looking for partners now

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 Linux (but not Windows)
 - ◆ Note: We will test and grade on uAPE machines
 - ◆ Be sure to test your projects there as well

Exams

- Midterm
 - ◆ Tuesday November 4th (put in calendar)
 - ◆ Covers first half of class
- Final
 - ◆ Tuesday December 16th (put in calendar)
 - ◆ Covers second half of class + selected material from first part
 - » I will be explicit about the material covered
- No makeup exams
 - ◆ Unless dire circumstances
- 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: 15%
 - ◆ Think of these collectively as a take-home midterm
- Midterm: 25%
- Final: 30%
- Projects: 30%
 - ◆ Each project is 10% of your 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 15% of the grade
 - ◆ Excellent practice for the exams, and some homework problems are exercises for helping with the project
 - ◆ 15% is actually a significant fraction of your grade (could be difference between at least one letter grade)

How *Not* To Pass (2)

- 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 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 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
 - ◆ (p.s. The projects cannot be done in the last few days)

Class Web Page

<http://www.cse.ucsd.edu/classes/fa14/cse120-b/>

- Serves many roles...
 - ◆ Course syllabus and schedule (updated over quarter)
 - » Lecture slides
 - ◆ Homework handouts
 - ◆ Project handouts
- Supplemental readings on Unix, monitors, and threads
 - ◆ e.g., seminal research paper describing the early Unix system
 - ◆ FYI only, but you might find it interesting
 - ◆ 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.?

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

CSE 120 Course Material

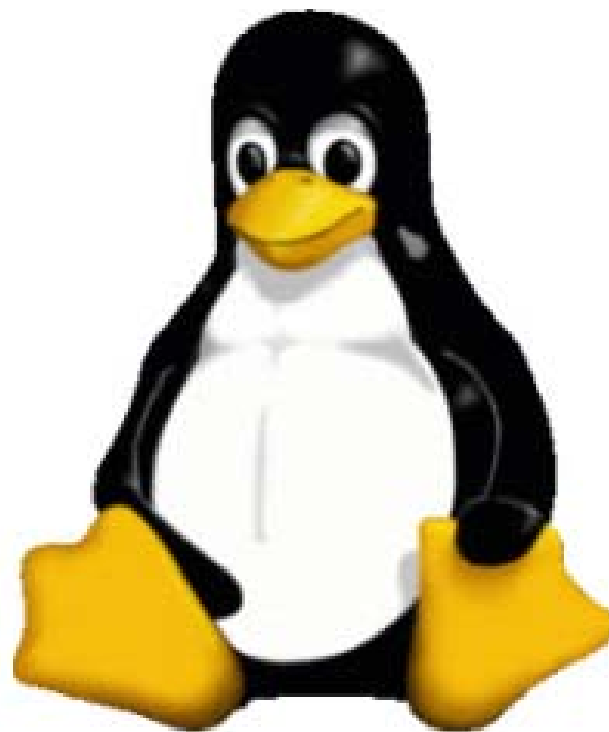
- This course addresses classic OS concepts
 - ◆ Services provided by the OS
 - ◆ OS implementation on modern hardware
 - ◆ Co-evolution of hardware and software
 - ◆ Techniques for implementing software systems that are
 - » Large and complex
 - » Long-lived and evolving
 - » Concurrent
 - » Performance-critical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

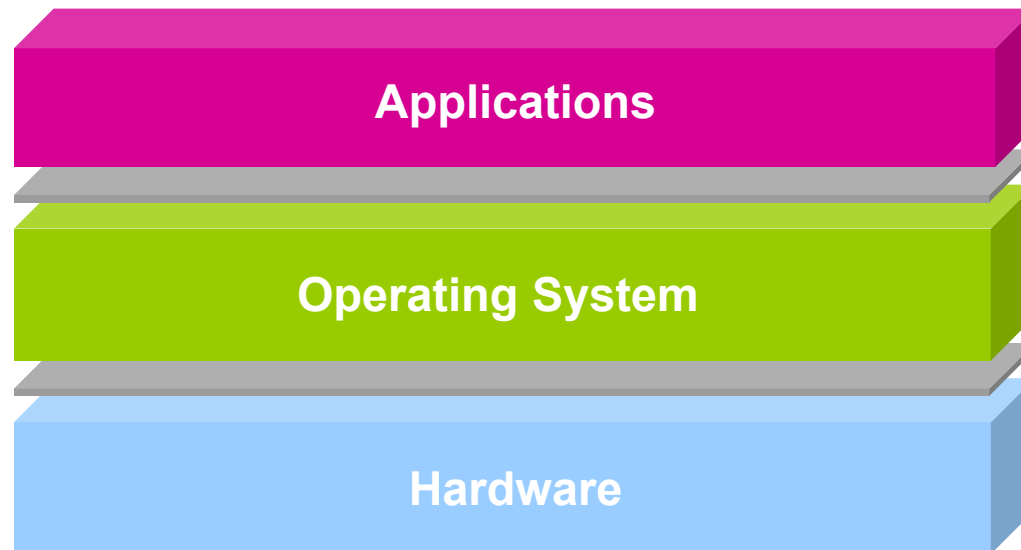


What is an Operating System?

- How would you answer?
 - ◆ (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)

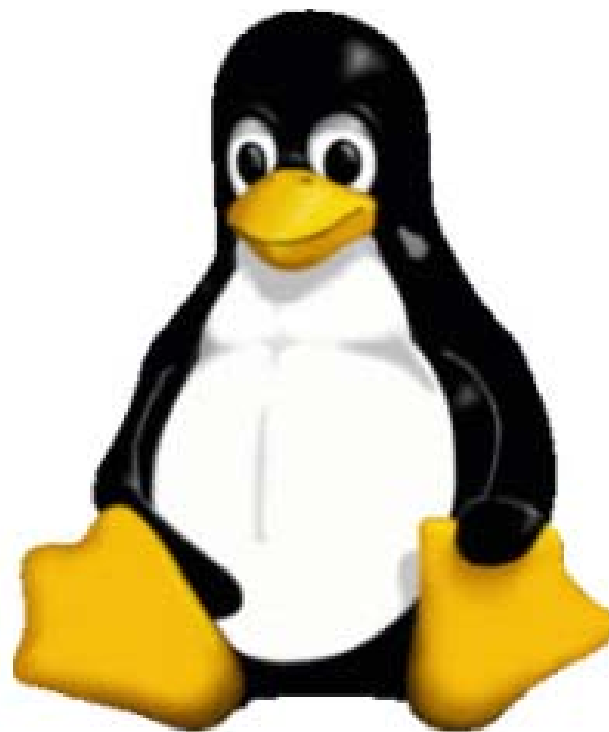
- Benefits to applications
 - ◆ Simpler (no tweaking device registers)
 - ◆ Device independent (all network cards look the same)
 - ◆ Portable (across Windows95/98/ME/NT/2000/XP/Vista/7/8/...)
 - ◆ Transportable (same program across different OSes (Java))

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?



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?
- 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: 5M
 - » Apache: 1.7M
 - » JDK: 6M
 - ◆ If you become a developer, you will face complexity

For next class...

- Browse the course web
<http://www.cse.ucsd.edu/classes/fa14/cse120-b/>
- Read Chapters 1 and 2
 - ◆ Start reading Nachos paper (online)
- Start thinking about partners for project groups
- No discussion this Friday (nothing to discuss)
- Let the fun begin!