CSE 120
Principles of Operating Systems

Fall 2002

Lecture 1: Course Introduction

Geoffrey M. Voelker

Lecture 1 Overview

- Class overview
- What is an operating system?
- Operating system modules, interfaces
Personnel

- Instructor: Geoff Voelker
- TAs:
  - Yuchung Cheng: Discussions, homeworks
  - John Rapp: Projects

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
  - Also “guest” discussions from the project TAs
  - Additional OS topics (e.g., how does an OS boot?)
  - Mailing list (cse120@cs.ucsd.edu)
  - Discussion board (http://discus4.ucsd.edu)
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 by me

Textbook

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/Solaris)
  - The projects will require serious time commitments
    » This is not an understatement
- You will do three projects using Nachos (more later)
  - Concurrency and synchronization
  - Multiprogramming
  - Virtual memory
- You will work in groups of four on the projects
  - Start identifying partners now

Labs

- We will use the uAPE lab in the AP&M basement
  - Solaris running on Sun sparc machines
- You can also use your home machine
  - The same project source will work on Linux (but not Windows)
  - We will test on uAPE machines
  - Be sure to test your projects there as well
Exams

- Midterm
  - Thursday, October 24
  - Covers first half of class

- Final
  - Friday, December 10
  - Covers second half of class plus selected material from first part
    - I will be explicit about the material covered

- 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%
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
  - 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 couple of days
  - Some groups last time learned that starting early meant finishing all of the projects on time…and some didn’t
Handouts

- Course syllabus
  - Also online (URL at bottom, linked into my home page, departmental web pages)
  - Will update schedule as quarter progresses
- Nachos paper
  - Original paper describing the Nachos project
- Unix paper
  - 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 there today, having to suffer through a core course in operating systems?
  - It’s not like everyone will become OS developers
- 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
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?

And the **principles** in this course are the design methods/approaches/solutions to these issues.
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)