COS 318: Operating Systems

Introduction

Margaret Martonosi and Vivek Pai
Computer Science Department
Princeton University

http://www.cs.princeton.edu/courses/archive/fall11/cos318/
Today

- Course staff and logistics
- What is an operating system?
- Evolution of computing and operating systems
- Why study operating systems?
Instructor

- Prof. Margaret Martonosi, 204 CS Building, mrm@cs.princeton.edu
  Office hours: Tue 3-5pm
- Prof. Vivek Pai, CS322 vivek@cs.princeton.edu
  Office hours: Thu 3-5pm

Teaching Assistants

- Mark Browning, mrbrowni@princeton.edu
  - Office Hours: Mon 12:30-2:30pm
- Xianmin (Sam) Chen, xianmincc@princeton.edu
  - Office Hours: Fri 10am-12pm
- Srinivas Narayana, narayana@princeton.edu
  - Office Hours: Fri 2-4pm.

- All TA Office hours are in the “Fishbowl”: Friend 010
What you will learn

- What an OS does. What services are provided, what functions are performed, what resources are managed, and what interfaces and abstractions are supported.
- How the OS is implemented. How the code is structured. What algorithms are used.
- Techniques, skills, and "systems intuition" (e.g., concurrent programming).
- Peeks at current research topics.
COS318 in Systems Course Sequence

- Prerequisites
  - COS 217: Introduction to Programming Systems
  - COS 226: Algorithms and Data Structures

- 300-400 courses in systems
  - COS318: Operating Systems
  - COS320: Compiler Techniques
  - COS333: Advanced Programming Techniques
  - COS425: Database Systems
  - COS471: Computer Architecture

- Courses needing COS318
  - COS 461: Computer Networks
  - COS 518: Advanced Operating Systems
  - COS 561: Advanced Computer Networks
Information & where to get it!

- Website
  - Materials will go here: projects, schedule, lecture/precept slides...
  - ~0 paper handouts!

- Textbook:
  - Keep up with readings!

- Questions about coursework, logistics, projects, etc: Enroll in Piazza
Besides Lecture

- Regular precept
  - Time: Tuesday 7:30pm – 8:30pm
  - Location: default is this room, CS 105

- First precept: Tues Sep 20
  - Will cover a bit of x86 assembler review in addition to project-specific topics.

- Project 1 Design review
  - Monday Sep. 26, 6pm -- 9pm
  - Sign up online (1 slot per team)
  - Project 1 deadline: Oct 5
Exams, Participation and Grading

◆ Grading
  ● First 5 projects: 45% with extra points
  ● Midterm: 15%
  ● Final Exam: 15%
  ● Final project: 15%
  ● Reading & participation: 10%

◆ Midterm and Final Exam
  ● Test lecture materials and projects
  ● Midterm: Thursday of midterm week, Oct 27

◆ Reading and participation
  ● Do your reading BEFORE each lecture
  ● Occasional quizzes just to check on this.
The Projects

Projects
1. Bootup
2. Non-preemptive kernel
3. Preemptive kernel
4. Interprocess communication and driver
5. Virtual Memory
6. File Systems

How
- Pair up with a partner for projects 1, 2, 3
- Different partner for 4, 5
- On your own for #6
- Each project takes 2-3 weeks
- Design review at the end of week one
- All projects due Wednesdays at NOON!

The Lab aka “The Fishbowl”
- Linux cluster in 010 Friend Center, a good place to be
- You can setup your own Linux PC to do projects
Project Grading

- **Design Review**
  - Signup online for appointments
  - 10 minutes with the TA in charge
  - 0-5 points for each design review
  - 10% deduction for missing the appointment

- **Project completion**
  - 10 points for each project

- **Late policy of grading projects**
  - 1 hour: 98.6%, 6 hours: 92%, 1 day: 71.7%
  - 3 days: 36.8%, 7 days: 9.7%
Why Piazza?

- Instructors’ Goal: Want to view this course as a learning community, where we all contribute to asking and answering questions.
  - Piazza helps provide a forum for this.
- Easier for students to answer each other’s questions.
- Easier for one of us (2 profs + 3 TAs) to see and answer questions (or endorse your answers) in a timely manner.
- Please use it instead of email, unless the question is of a personal/private nature.
Ethics and other issues

- **Do not put your code or designs or thoughts on the Web**
  - Other schools are using similar projects
  - Not even on Facebook or the like
- **Follow Honor System**: ask when unsure, cooperation OK but work is your own (or in pairs for projects)
Today

- Course staff and logistics
- What is an operating system?
- Evolution of computing and operating systems
- Why study operating systems?
Let’s begin at the beginning...

- When you write a program, what happens?
Managing and Abstracting Hardware Resources

- **Hardware to manage:** CPU, Primary memory, Secondary memory devices (disk, tapes), Networks, Input devices (keyboard, mouse, camera), Output devices (printers, display, speakers)

- **Resources to manage:**
  - CPU Cycles
  - Network and memory bandwidth
  - Energy / battery-life (mobile)
  - …
What is an OS?

- Resource Manager of physical (HW) devices ...
- Abstract machine environment. The OS defines a set of logical resources (objects) and operations on those objects (an interface on the use of those objects).
- Allows sharing of resources. Controls interactions among different users.
- Privileged, protected software - the kernel. Different kind of relationship between OS and user code (entry via system calls, interrupts).

- Birthplace of system design principles!
  e.g., Separation of Policy and Mechanism.
What Does an Operating System Do?

- Provides a *layer of abstraction* for hardware resources
  - Allows user programs to deal with higher-level, simpler, and more portable concepts than the raw hardware
  - E.g., files rather than disk blocks
  - Makes finite resources seem “infinite”

- Manages the resources
  - Manage complex resources and their interactions for an application
  - Allow multiple applications to share resources without hurting one another
  - Allow multiple users to share resources without hurting one another
How to Mitigate Complexity? Abstraction!

- Hide underlying details, and provide cleaner, easier-to-use, more elegant concepts and interfaces.
  - Also provides standardized interfaces despite diversity of implementation underneath.
- Key CS principle
- Key to understanding Operating Systems

**Examples**
- Threads or Processes (Fork)
- Address spaces (Allocate)
- Files (Open, Close, Read, Write)
- Network Messages (Send, Receive)
One Abstraction Example: Disk

Disk hw and operations are very complex

- Multiple heads, cylinders, sectors, segments
- Wait for physical movement before read or write
- Data stored discontiguously
- Sizes, speeds vary on different computers
- IT WOULD BE HORRIBLE TO WRITE CODE SPECIALIZED FOR EACH DISK!

- OS provides simple read () and write() calls as the API
  - Manages the complexity transparently, in conjunction with the disk controller hardware
  - Such I/O abstractions have outlived several storage technologies!
Resource Management

4 sub-issues:

- Resource Allocation
- Resource Virtualization
- Resource Reclamation
- Resource Protection
Resource Allocation

- Computer has finite resources
- Different applications and users compete for them
- OS dynamically manages which applications get how many resources
- **Multiplex** resources in space and time
  - Time multiplexing: CPU, network
  - Space multiplexing: disk, memory

- E.g., what if an application runs an infinite loop?

```c
while (1);
```
Resource Virtualization

- OS gives each program the illusion of effectively infinite, private resources
  - “infinite” memory (by backing up to disk)
  - CPU (by time-sharing)
Resource Reclamation

- The OS giveth, and the OS taketh away
  - Voluntary or involuntary at runtime
  - Implied at program termination
  - Cooperative
Protection

- You can’t hurt me, I can’t hurt you
- OS provides safety and security
- Protects programs and their data from one another, as well as users from one another
- E.g., what if I could modify your data, either on disk or while your program was running?
Mechanism vs. policy

- Mechanisms are tools or vehicles to implement policies
- Examples of policies:
  - All users should be treated equally
  - Preferred users should be treated better
Today

- Course staff and logistics
- What is an operating system?
- Evolution of computing and operating systems
- Why study operating systems?
A Typical Academic Computer (1988 vs. 2008)

<table>
<thead>
<tr>
<th>Feature</th>
<th>1988</th>
<th>2008</th>
<th>Ratio</th>
</tr>
</thead>
<tbody>
<tr>
<td>Intel CPU transistors</td>
<td>0.5M</td>
<td>1.9B</td>
<td>~4000x</td>
</tr>
<tr>
<td>Intel CPU core x clock</td>
<td>10Mhz</td>
<td>4×2.66Ghz</td>
<td>~1000x</td>
</tr>
<tr>
<td>DRAM</td>
<td>2MB</td>
<td>16GB</td>
<td>8000x</td>
</tr>
<tr>
<td>Disk</td>
<td>40MB</td>
<td>1TB</td>
<td>25,000x</td>
</tr>
<tr>
<td>Network BW</td>
<td>10Mbits/sec</td>
<td>10GBits/sec</td>
<td>1000x</td>
</tr>
<tr>
<td>Address bits</td>
<td>32</td>
<td>64</td>
<td>2x</td>
</tr>
<tr>
<td>Users/machine</td>
<td>10s</td>
<td>&lt; 1</td>
<td>&gt;10x</td>
</tr>
<tr>
<td>$/machine</td>
<td>$30K</td>
<td>$3K</td>
<td>1/10x</td>
</tr>
<tr>
<td>$/Mhz</td>
<td>$30,000/10</td>
<td>$3,000/10,000</td>
<td>1/10,000x</td>
</tr>
</tbody>
</table>

Moore’s Law! ++
Phase 1: Batch Systems

- Hardware very expensive, only one user at a time
- Batch processing: load, run, print
  - OS linked in as a subroutine library
- Problem: better system utilization
  - System idle when job waiting for I/O
- Development: multiprogramming
  - Multiple jobs resident in computer’s memory
  - Hardware switches between them (interrupts)
  - Memory protection: keep bugs to individual programs
Phase 2: Time Sharing

- Problem: batch jobs hard to debug
- Use cheap terminals to share a computer interactively
- MULTICS: designed in 1963, run in 1969
- Shortly after, Unix enters the mainstream
- Issue: thrashing as the number of users increases
Phase 3: Personal Computer

- Personal computer
  - Altos OS, Ethernet, Bitmap display, laser printer
  - Pop-menu window interface, email, publishing SW, spreadsheet, FTP, Telnet
  - Eventually >100M units per year

- PC operating system
  - Memory protection
  - Multiprogramming
  - Networking
Now: > 1 Machines per User

- Pervasive computers
  - Wearable computers
  - Communication devices
  - Entertainment equipment
  - Computerized vehicle

- OS are specialized
  - Embedded OS
  - Specially configured general-purpose OS
Now: Multiple Processors per Machine

- Clusters
  - A network of PCs
  - Commodity OS

- Multicomputers
  - Supercomputer with many CPUs and high-speed communication
  - Specialized OS with special message-passing support

- Multiprocessors
  - SMP: Symmetric MultiProcessor
  - ccNUMA: Cache-Coherent Non-Uniform Memory Access
  - General-purpose, single-image OS with multiprocessor support

- Chip Multiprocessors
  - 2+ cores per chip COMMON
Today

- Course staff and logistics
- What is an operating system?
- Evolution of computing and operating systems
- Why study operating systems?
Why Study OS?

- OS is a key part of a computer system
  - It makes our life better (or worse)
  - It is “magic” to realize what we want
  - It gives us “power”

- Learn about concurrency
  - Parallel programs run on OS
  - OS runs on parallel hardware
  - Best way to learn concurrent programming

- Understand how a system works
  - How many procedures does a key stroke invoke?
  - What happens when your application references 0 as a pointer?
  - Building a small OS will go a long way…
Why Study OS?

- Important for studying other areas
  - Networking, distributed systems, …
- Full employment
  - New hardware capabilities and organizations
  - New features
  - New approaches
  - Engineering tradeoffs keep shifting as the hardware changes below and the apps change above
Influences in OS Design

- Workload
  - Services & API
  - Internal Structure
  - Policies / Mechanisms
- Metrics
- Hardware Resources
Influences in OS Design

- Metrics: Performance as Bandwidth and Latency.
- Workload: Scientific computations, Database operations, Multi-user.
- Services & API.
- Internal Structure.
- Policies / Mechanisms.
- Hardware Resources: Processor, Memory, Disks, Network, Keyboard, Display, Multiprocessors.
Influences in OS Design

Changing

Metrics
Accessibility, Reliability, No-futz-ness, Energy efficiency, Security

Workload
Productivity applications Games, Multimedia, Web
Process control Personal (PDAs), Embedded, E-Commerce

Services & API

Internal Structure

Policies / Mechanisms

Hardware Resources
Processor, Memory, Disks (?), Wireless & IR, Keyboard(?), Display(?), Mic & Speaker, Motors & Sensors, GPS, Camera, Batteries
Things To Do

- For today’s material: Read MOS 1.1-1.3
- For next time: Read MOS 1.4-1.5
- Make “tent”, leave with me, pick up and use every class.
- Choose a partner for first 3 projects and email vivek/me with your choice.
  - Use Piazza to help find available partners!