Honors Operating Systems

Robert Grimm
New York University
Meet and Greet
What Is an Operating System?
Two Takes

Traditionally: Operating system
- Manages resources on a single machine

Increasingly: Distributed system
- Tries to make several machines look more like one
- Ideal: Transparency
- Reality: Communication overhead, concurrency, failures, malicious users
Manages hardware resources

- Hides the gory details and provides a convenient API
  - CPU, memory, storage, networking, display, keyboard, mouse, printer,...

- Multiplexes shared resources
  - Time and space multiplexing

Provides isolation and protection

- Applications cannot clobber each other or their resources
To do its job, the OS must be privileged

Only the *kernel* can execute special instructions

Applications request operations from kernel

Kernel provides *system call* interface

- open, read, write, fork, pipe, execute, wait,...

Applications set up arguments and then *trap* to kernel

Kernel performs service and returns to application

Where to draw the line? What abstractions to provide?
This Course
Overview

* Prerequisite
  * Undergraduate operating systems

* Three goals
  * Gain an appreciation for existing systems research
  * Perform systems design, implementation, and evaluation
  * Develop your (technical) communication skills

* Two components
  * Reading, reviewing, and discussing papers
  * Performing a term-long project
Process

- Read papers
  - What is the problem and why does it matter?
  - What is the solution and how is it new/different?
  - What are the contributions and limitations?
- Write one paragraph review (per paper)
  - One sentence summary
  - Key strengths and weaknesses
  - Anything else important to you
Process (cont.)

- Submit review by email (by 8am on day of class)
  - Also by paper if you want my individualized feedback
- Read other students' reviews
  - Subscribe to mailing list today
- Participate in class discussion
  - I provide slides to review material and guide discussion
- Readings and reviews are essential!
Topics

- Historical perspective
  - Early operating systems: RC 4000, Unix, Multics
- Structure and organization
  - Where to draw the line between kernel and userland?
  - How to isolate applications from each other?
- Managing concurrency
  - Who controls what runs on a computer and how?
Topics (cont.)

- **Communication**
  - Two paradigms: exchange data vs. exchange computations
  - A complete distributed system
  - How to deal with failure?

- **Virtual memory**
  - Implementation, interface, measurement
  - Value-added service: Recoverable virtual memory

- **File systems**
  - Local, client/server, peer-to-peer
Topics (cont.)

- Security
  - Capabilities (revisited), labels
  - Hardware support: trusted computing
- Mobile and pervasive computing
  - Disconnected operation
  - Coordinating storage
  - Application structure and supporting services
Projects
Projects

* In groups of 2-3, you perform your own systems work
  * Group charter
  * Project proposal
  * Literature search
  * Mid-term report
  * Final report and talk

* Topic: operating and distributed systems
  * You may build on your own research, but the class project must have a distinct component
Some (Biased) Ideas

- How can systems benefit from language technologies?
  - Identify something that is hard to express/enforce
  - Design an extension to C or Java and implement with xtc
- How build stream processing systems?
  - Lots of languages would benefit from a common platform
  - Start by talking to my student Robert Soulé
- Do you believe the authors?
  - Pick one or more related systems and repeat the evaluation
Hints on Methodology

- If you don't quite understand the issues, build a simple test system and refine it

- Shoot for a working system quickly instead of aiming for the perfect system
  - Drawback: you may have to refactor/rewrite several times

- Tools are your friends
  - CVS: you will make mistakes
  - JUnit, DejaGNU: you will make mistakes
  - make/ant: you don't have time to do things by hand
More Hints on Methodology

- Do not optimize your system without measuring first
- Make sure you understand your measurement results
  - Expect to do more measurements
- Document early and everything
  - At the function level: if you can't describe it, don't code it
  - At the system level: check for (in)consistency
A Few More Things
Collaboration Policy

- Do discuss readings and topics with each other
- But write reading summaries individually
- Help each other with project questions
- But clearly identify any ideas, code, etc. from others
One web site
  * http://cs.nyu.edu/rgrimm/teaching/sp11-os/

One mailing list
  * g22_3250_001_sp11@cs.nyu.edu
  * Subscribe today
  * Post only plain text (no HTML)

n groups
  * Start forming groups today, group charter due Tuesday
Getting in Touch

- Office hours
  - By appointment
  - 715 Broadway, room 711
- Don't hesitate to stop by, send me an email
  - rgrimm@cs.nyu.edu