Honors Operating Systems

Robert Grimm
New York University
Course Overview

- **Prerequisite**
  - Undergraduate operating systems

- **Three goals**
  - Gain an appreciation of existing systems research
  - Perform systems design and implementation yourself
  - Develop your communication skills

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

- Read papers
  - What is the problem and why is it important?
  - How is the solution new or different from other work?
  - What are the contributions and limitations?

- Write one paragraph review
  - One sentence summary
  - Key strengths
  - Key weaknesses
  - Anything else important to you
Submit the review by email (by 10am on day of class)
   And by paper if you want my feedback

Read other students’ reviews
   We have a mailing list just for reviews

Participate in class discussion
   I provide slides to review material and guide discussion

Readings and reviews are essential!
Topics

- Historical perspective
  - Early operating systems
- Structure and organization
  - Where to draw the line between kernel and userland?
  - How to isolate applications from each other?
- Managing concurrency
  - Who controls scheduling and how?
Topics (cont.)

- Communication
  - Two paradigms: exchange of data vs. computations
  - An early attempt at security
  - A complete distributed system

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

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

- Internet-scale services
  - Clusters, clusters, clusters
    - Incl. how not to do it
- Mobile and pervasive computing
  - Management of updates and conflicts in the presence of disconnection
  - Structuring and services
- Pulling back
  - How to design systems?
  - Our No. 1 principle
Operating vs. Distributed Systems

- Operating systems manage resources on a *single* machine
- Distributed systems aim to make *several* machines look more like one
  - Ideal: Transparency
  - Reality
    - Failures
    - Concurrency
    - Communication latency
    - Security
  - This is where the action is…
Projects

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

- Topic: operating or distributed systems
  - You may build on your own research, but the class project must have its own contribution
Some (Wild) Ideas

- It’s all about web services
  - How do SOAP, XML-RPC, HTTP POST differ in expressive power?
  - How do the different technologies/systems perform?
- It’s all about P2P, DHTs, CDNs
  - What design choices are there and how do they affect performance?
    - Measure alternatives on PlanetLab
  - Can we reconcile server-driven with client-driven distribution?
- The spam stopper (instead of filter)
Hints on Methodology

- If you don’t quite understand the issues, build a simple test system and refine it.
- Tools are your friend
  - CVS: You will make mistakes
  - make: You don’t have time to do things by hand
- Shoot for a working system quickly instead of aiming for the perfect system
  - Drawback: You may have to refactor/rewrite some
Do not optimize your system without measuring and profiling *first*

Document early and everything
- At the code-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

- 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 outside sources
Administrivia

- One web site
  - http://www.cs.nyu.edu/rgrimm/teaching/sp04-os/

- Two mailing lists
  - g22_3250_001_sp04@cs.nyu.edu
  - g22_3250_001_sp04-readings@cs.nyu.edu
  - Subscribe to both lists
  - Post only plain-text messages with hard line endings
    - No HTML!

- x groups
  - Start forming groups today, notify me by next Tuesday
Administrivia (cont.)

- Official office hours
  - Wednesday 2-3
  - 715 Broadway, room 711
- If you have questions/need to talk, contact me!
Meet and Greet
Let’s Get Started
What is an Operating System?

- Manages hardware resources
  - Hides the gory details and provides a convenient API
    - 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
The Red Line

- To do its job, operating system must be privileged
  - Only the *kernel* can execute privileged 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 should the kernel provide?
What is the key innovation of Unix?

What other important feature offers considerable power?

How does protection work in Unix?
  - In hindsight, what are shortcomings?

What else is noteworthy in the paper?
Three Design Considerations

- Make it interactive
- “Keep it simple, stupid” (KISS)
  - Not just economy (efficiency) but also elegance of design
- “Eat your own dog food”
The Nucleus of a Multiprogramming System

- How does the RC 4000 multiprogramming system differ from Unix?
  - How does RC 4000 process hierarchy differ?
  - How is it the same?
  - What style of communications does the RC 4000 use?
    - Could we make this simpler?
- Does this structure remind you of any other systems?