Why study Operating Systems?

Why study Operating Systems?

Why study Operating Systems?

Why study Operating Systems?

To learn how computers work
To learn how to manage complexity through appropriate abstractions
- infinite CPU, infinite memory, files, semaphores, etc.
To learn about system design
- performance vs. simplicity, HW vs. SW, etc.
Because OSs are everywhere!
Why study Operating Systems?
Because you are worth it!

CS 372
A capstone course
- PL meets Data Structures meets HW meets Algorithms meets Software Engineering...and they all meet you!
- Projects
  - build components of an OS
  - enhance software engineering skills

Where's the OS?
Las Vegas

Where's the OS?
New York
Three levels of learning

How to approach problems
- Example: problem definition, design-space exploration, case studies
- Goal: When faced with a similar problem, you should be able to devise a solution
- Time-scale: big, long-term payoff

Three levels of learning

How to apply specific techniques
- Example: time-tested solutions to hard problems such as concurrent programming, two-phase commit, transactions...
- Goal: Become a good engineer
- Time-scale: useful both now and in 20 years

Three levels of learning

How, in detail, current OSs work
- Example: FS, network stack, internal data structures, VM, ... of Linux, XP, Solaris, etc.
- Goal: Well... know, in detail how current OSs work!
- Time-scale: Better be now, because all will have changed tomorrow

What is an OS?

An Operating System implements a virtual machine whose interface is more convenient* that the raw hardware interface

* easier to use, simpler to code, more reliable, more secure...
Operating System Services

- **Manager**
  - of physical resources such as CPU, memory, disks, networks, displays, cameras, etc.
- **Poet**
  - abstracts physical resources to create processes, threads, files, directories, users, etc.
- **Coordinator**

Operating System Services

- **Manager**
  - of physical resources such as CPU, memory, disks, networks, displays, cameras, etc.
- **Poet**
  - abstracts physical resources to create processes, threads, files, directories, users, etc.
- **Coordinator**
  - allows multiple applications/users to work together in efficient and fair ways
**Grading Policy**

- Exams (closed books)
  - Midterm: 25%
  - Final: 35%
- Projects: 40%
- No formal homework
  - Practice problems and solutions posted on web site

**Logical OS Structure?**

- Applications: Quake, Sql Server, System Util, Shells, Windowing & graphics
- OS Interface: Naming, Windowing & Gfx, Networking, Virtual Memory, Access Control
- Physical m/c Intf: Device Drivers, Memory Management
  - Interrupts, Cache, Physical Memory, TLB, Hardware Devices

**Administrivia**

- **Textbook**
  - Further readings, notes, as distributed in class
- **Other useful texts**
  - Operating System Principles, Bic & Shaw, 2nd edition
  - Modern Operating Systems, Tanenbaum, Prentice Hall

**Where to go for help**

- Ask questions in class
- Attend office hours
  - Lorenzo: T/TH 1:30-2:30
  - Amit: M: 12:00-1:00; W: 4:30-5:30; F 9:30-10:30
- Don’t send questions by email
  - I want to know you
  - I don’t want to be subject to DOS attacks
- Your primary avenue for resolving questions is office hours
Academic Integrity

- Submitted work should be your own
- Encouraged collaboration:
  - discuss problem sets, projects
  - point of confusion, question interpretation, solution approaches
- Dishonesty has no place in any community
  - You may NOT copy code from another group, or any course project material similar to the one used in this class
  - It is never OK to look at the written work of another student or show them yours until grading is done—including whiteboard discussions, or help in debugging.
  - Use the "Gilligan Island" rule.
  - When in doubt, ask!

Issues in OS Design

- Structure: how is an operating system organized?
- Sharing: how are resources shared among users?
- Naming: how are resources named by users or programs?
- Protection: how is one user/program protected from another?
- Security: how to authenticate, control access, secure privacy?
- Performance: why is it so slow?
- Reliability and fault tolerance: how do we deal with failures?
- Extensibility: how do we add new features?

Issues in OS Design

- Communication: how can we exchange information?
- Concurrency: how are parallel activities created and controlled?
- Scale, growth: what happens as demands or resources increase?
- Persistence: how can data outlast processes that created them?
- Compatibility: can we ever do anything new?
- Distribution: accessing the world of information
- Accounting: who pays bills, and how to control resource usage

Why is this material critical?

- Concurrency
  - Therac-25, Ariane 5 rocket (June 96)
- Communication
  - Air Traffic Control System
- Persistence
  - Denver Airport
- Virtual Memory
  - Blue Screens of Death
- Security
  - Data Theft at McCombs School of Business