CS439: Principles of Computer Systems

Dr. Alison N. Norman
Department of Computer Science
The University of Texas at Austin
Fall 2013

Who am I?

- Education
  - Undergrad in CS from Georgia Tech
  - MS and Ph.D. in CS from UT Austin
- Research
  - Supercomputing
- Family
  - Married with two children (boys!) and two dogs

Today's Plan

- Introduce and motivate course themes
- Course organization and logistics
- Quiz

Why are we here?

Two main goals:

- Learn the low-level software abstractions that make the computer work
  - Operating System
  - Network
  - Various aspects of memory management
- Use these topics as a case study to understand large-scale system design

System Design

How do we construct systems that are

- reliable
- portable
- efficient
- secure

?

What is an OS?

- No universally accepted definition
  - Is it everything that comes on a computer?
    - Used to be, then came Microsoft (US v. Microsoft, 1998)
    - Now this varies widely
- Program that is always running
  - Ha.

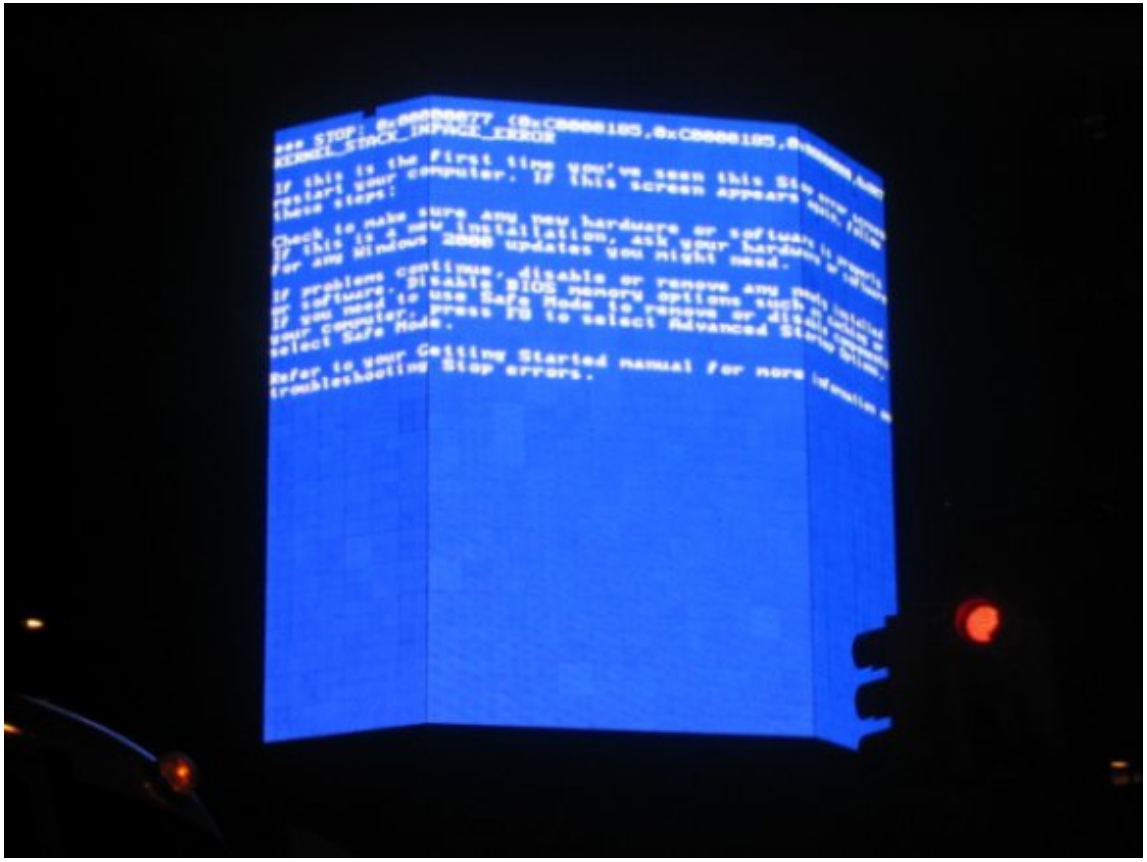
Operating System: A definition

Software that manages a computer's resources

- makes it easier to write the applications you want to write
- makes you want to use the applications you wrote by running them efficiently

Why Study Operating Systems?

- To learn how computers work
- To learn how to manage complexity through appropriate abstractions
- To learn about system design
  - Performance vs. simplicity, HW vs SW, etc
  - Design trade-offs made in the past do not necessarily apply now
  - Those made now will not necessarily apply in the future
- Operating Systems are everywhere!



Where's the Operating System?

Las Vegas!

Where's the Operating System?

New York!

Operating Systems: More than One Hat

- Referee
  - Manages shared resources
- Illusionist
  - Infinite memory! Your own private processor!
- Glue
  - Provides standard services which the hardware implements

Operating Systems as Referee

- Resource allocation
  - Coordinates multiple applications and users to achieve fairness and efficiency
- Isolation
  - Protects processes from one another
    - One application's bugs should not crash another (or the whole system!)
    - If it does crash, should fail gracefully
- Communication
  - Allow processes to work together

Operating Systems as Illusionist

Illusion of resources that are not really present

- Virtualization: processor, memory, screen space
- Entire computer!

Operating Systems as Glue

Provides standard services to simplify application design and facilitate sharing

- File system, virtual memory, networking
- Decouples hardware and application development
- Start, stop, and clean up after a program

Evaluating an Operating System

- Reliability
  - OS does exactly what is designed to do
- Security
  - OS cannot be compromised by a malicious attacker
- Portability
  - OS does not change as hardware changes
- Performance
  - efficiency, overhead, fairness, latency, throughput, predictability

Reliability

- The ability of a computer-related hardware or software component to consistently perform according to its specifications.
- In theory, a reliable product is totally free of technical errors (yeah, right)
- Availability: percentage of time system is useful
  - Depends on *MTTF* and *MTTR*

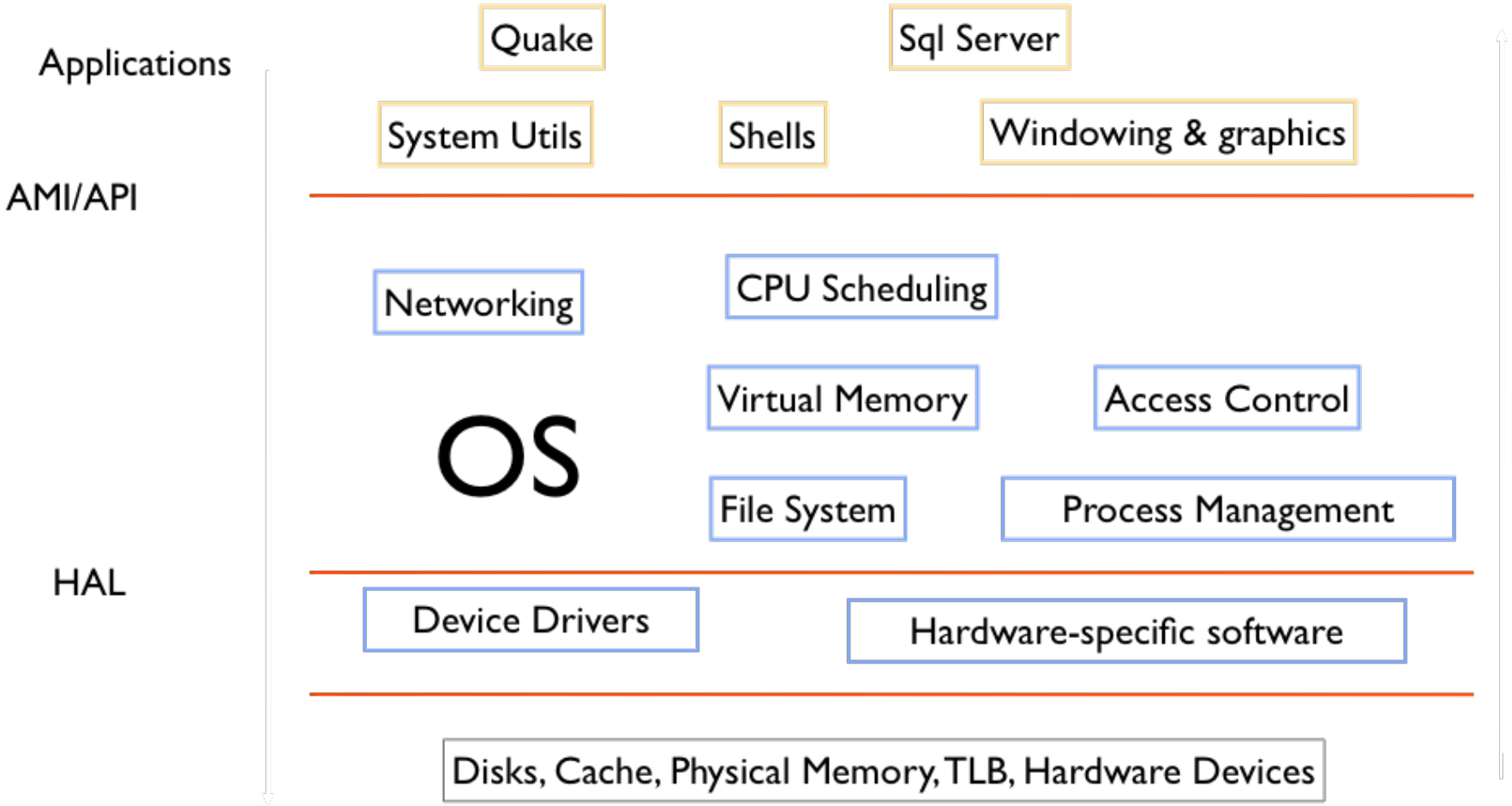
Security

- Includes privacy: data on the computer only accessible to authorized users
- Strong fault isolation helps, but not enough
  - Security mechanisms should not prevent legitimate sharing!
- Enforcement mechanism
  - Ensures only permitted actions are allowed
- Security policy
  - Defines what is permitted

Portability

- OSs can live longer than your cat!
  - must support applications not yet written
  - must run on hardware not yet developed
- Three interfaces
  - Abstract Machine Interface (AMI)
    - between OS and apps: API + memory access model + legally executable instructions
  - Application Programming Interface (API)
    - function calls provided to apps
  - Hardware Abstraction Layer (HAL)
    - abstracts hardware *internally to the OS*

Logical OS Structure



Performance

- Efficiency/Overhead
  - how much is lost by not running on bare hardware?
- Fairness
  - how are resources divided?
- Response time
  - how long does a task take to complete
- Throughput
  - how many tasks complete per unit of time
- Predictability
  - are performance metrics consistent over time?

What You'll Learn in this Course

1. How to approach problems
   - Fundamental issues
   - Design space
   - Manage complexity
   - Case studies

   Goal: You will be able to devise good solutions to similar (and very different) problems.

What You'll Learn in this Course

2. Specific techniques you should be able to apply to other problems
   - Time-tested solutions to hard problems
   - Goal: be a good engineer
3. Details(ish) of modern operating systems
   - Lots of material, changes quickly
   - Not a priority of this class
     - Would rather you know the abstractions so that you can apply your knowledge to the next OS

Things You'll Encounter

- Design Problems
  - Understand the problem and define it
  - Understand the space of possible solutions and previous approaches
  - Formulate your own approach and justify it
- Implementation Issues
  - Real systems are more difficult to build than explain
  - The devil is in the details

Course Organization and Logistics

What knowledge you need to begin this course

- Prerequisites: CS429(H) with a grade of at least a C-
- Solid basic understanding of hardware
- Solid programming skills (especially in C)

You must understand the components to understand the implications of how they interact!

Teaching Staff

- Teaching Assistants:
  - Aming Ni
  - Navid Yaghmazadah
  - Ben Bowley-Bryant
  - Clare Coleman
  - Di Huynh
- And, obviously, me.

Course Materials

- Website: Go-to place for information
  - Syllabus, Schedule, Projects, Homeworks, Slides, Useful links, Feedback form
- Textbooks:
  - *Computer Systems: A Programmer's Perspective* by Bryant & O'Hallaron (from 429)
  - *Pearson Custom Computer Science: Custom Edition for University of Texas – Austin* containing chapters of an OS textbook and a C programming textbook
    - Hard copies available at the UT Co-op beginning September 10th
  - *Operating Systems and Middleware: Supporting Controlled Interaction* by Max Hailperin (online, follow links on syllabus or schedule)
- Piazza: discussion board
  - Course "CS439N"
  - Many of you received an invitation
- Canvas: grade center
  - I don't know much about Canvas, but it has got to be better than Blackboard…
- iClicker: participation counts, get one and get it registered!
  - Participation points begin next week

Schedule Overview

- Introduction
- Concurrency and Synchronization
- Memory Management
- File Systems: Use and Implementation
- Networked Systems
- Parallel and Distributed Computing (briefly)
- Security (briefly)

Each Class

- Introduces concepts, covers high-level ideas
- Timing:
  - Approximately 50 minutes of lecture
  - Approximately 5 minutes of break
  - Approximately 45 minutes of lecture

Discussion Sections

- Required!
- You MUST attend your own
- First half will be about homework
  - an additional homework question
  - then discussion of that week's homework
- Second half will be discussion of current project

Homeworks

- Weekly homeworks (eleven total)
- Designed to help you prepare for the exams
- All but one question will be posted online and due at 8:45am on Fridays
- One question will be solved and turned in during discussion section
- Must turn in problem in discussion section to receive credit for that homework
- Graded on an okay/not okay basis (Binary!)
- “Lowest” two dropped

Projects

- There will be 5 projects in this course
- Most will be pair or group programming
- They will not be equally weighted
  - Weights will be announced as they are assigned
- They will be difficult
  - Systems programming is difficult!
- Your life will be easier if you learn and program in the Linux environment

Expected Effort

- This is a hard course that requires a LOT of effort
- Topics are new and detailed
- There are many design tradeoffs to understand
- Systems programming is hard
  - Debugging systems code is worse
- Projects can take 10-15 hours in the beginning, and 30-40 later in the semester
  - If it goes well
  - Start early, stay late

Evaluation

- Projects (32%)
  - Build operating system components
  - 4 slip days total, 2 maximum on each project
  - Due 11:59pm on select Fridays
  - More information soon
- Homeworks (8%)
  - Written
  - Due 7:45a Thursdays *and* at the beginning of discussion section
  - Graded on an *ok*/*not okay* basis

Evaluation

- Exams and a final (16%, 16%, 22%)
  - Exams are 10/1 and 11/6 (mark your calendars!)
    - In the evening. Locations are on the schedule.
  - Final is as scheduled by the registrar
    - Will NOT be at time currently listed
- iClicker participation (6%)
  - Instant feedback for me
  - Gives you a reason to come to class
  - Attend 80% of the classes for full credit
  - Using laptops or other digital devices forfeits your participation credit
- Final grades will be curved
  - If you are on the edge, you need to have shown effort
    - Attended class
    - Turned in all assignments
  - Or I will NOT bump you up

Collaboration and Cheating

- Collaboration
  - Discuss problem sets and programming assignments
  - Discuss possible interpretation of questions, technical details
- Cheating
  - Copying solutions code or programs from someone else, previous semesters' solutions, or public domain
  - Providing material for someone else to imitate
  - Participating in discussion group where one person writes solution and everyone else copies it
  - Penalty for cheating is an F in the course and a referral to the Dean of Students office

How to Succeed in This Course

- Keep up
- Attend class
- Do the reading
- Do the projects (and start them early!)
- Ask questions
- Get to know the people in the class
  - How many people you know is the number one indicator of success
    - study groups, problem discussion, etc.
  - Let me help you…

How to Get Help

- Ask questions!
  - In class
  - Office Hours (in online syllabus)
    - Mine: M 11a-12:30p and W 11:30a-1p GDC 6.816
      - If I am not in 6.816, please check my office, GDC 6.310
    - Others coming
    - If you can't make it to office hours, set up an appointment
    - *All are beginning Tuesday, 9/3*
  - Piazza
    - Use Anonymous feature if necessary
- Online Lectures
  - UC Berkeley's OS course

Other Thoughts

- Enrollment is high
- Workload is heavy
- Grading will be slow
- Use of discussion board essential

C and Linux

C and Linux

- This course relies heavily on C and Linux
- You should have prior knowledge of these from 429
- Your first discussion section (next Friday!) will provide introduction and review (among other topics)
- I want to know where you are now so we can plan that review

Assessment

Learning C and Linux

- Resources on the website
  - I'll add more
- Friday's discussion section will review/teach C and Linux
  - It is the ONLY one that is optional
- ASK Questions
  - Early and often

Summary

- Operating Systems are infinite loops that manage resources
- Key ideas: coordination and abstraction
- It's Going to Be Great!