Who am I?

• Education
  – Undergrad in CS from Georgia Tech
  – MS and Ph.D. in CS from UT Austin
• Research
  – Supercomputing
• Family
  – Married
  – Three children: two boys (8 and 4) and one girl (1)
  – Two dogs
Today’s Plan

• Introduce and motivate course themes
• Course organization and logistics
• Homework assignment
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
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?
Where’s the Operating System?
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 is 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

Applications
- Quake
- Sql Server
  - System Utilities
  - Shells
  - Windowing & graphics

AMI/API
- Networking
- CPU Scheduling
- Virtual Memory
- Access Control
- File System
- Process Management

OS
- Device Drivers
- Hardware-specific software

HAL
- Disks, Cache, Physical Memory, TLB, Hardware Devices
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
Summary

- Operating Systems are infinite loops that manage resources
- Key ideas: coordination and abstraction
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:
  – Mihir Mehta
  – Yi-Hsuan Hsieh
  – Briana Gray
  – Cody Hill
  – Kaelin Hooper
  – Jon Lee
  – Robert Lynch
  – Sage Moore
  – Plus one
  – Jeremy Hintz

• 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)
  - *Operating Systems: Three Easy Pieces* by Remzi H. Arpaci-Dusseau and Andrea C. Arpaci-Dusseau (online, follow links on syllabus or schedule)
  - *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
  - Please ignore Canvas’s calculation of your grades

• **iClicker:** participation counts, so 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 55-60 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 lecture and homework
  – an additional homework question
  – then discussion of that week’s homework
• Second half will be discussion of current project
Homeworks

• Weekly homeworks (ten total)
• Designed to help you prepare for the exams
• All but one question will be posted online (linked from the schedule) and due at 9:45am on Thursdays
• One question will be solved and turned in during discussion section
• Must turn in both sets of problems to receive credit for that homework
• Graded on an okay/not okay basis (Binary!)
• Lowest two dropped
  – These are your excused absences
Projects

- There will be 5 projects in this course
- Pair or group programming
- Each project is accompanied by a design document
- They will not be equally weighted
  - You must get Project 2 working
- They will be difficult
  - Systems programming is difficult!
- Your life will be easier if you learn 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 hours 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
    • None on last project
  – Due 11:59pm on select Fridays
  – More information soon

• Homeworks (8%)
  – Written
  – Due 9:45a 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 9/30 and 11/4 (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
    • The other 20% is for “excused absences”.
  – Using laptops or other digital devices forfeits your participation credit

• Final grades will be curved (a little)
  – 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

• Extra credit: 10 points on final exam for reading 75% of Piazza posts (as measured by Piazza)
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.
• Work hard, play hard
How to Get Help

• Ask questions!
  – In class
  – Office Hours
    • In online syllabus and on Office Hours webpage
    • All begin Monday, 8/31
  – Piazza
    • Use Anonymous feature if necessary

• Many resources/tutorials on course main page
  – Online Lectures, C tutorials, link to B&O student site,...
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 week) will provide introduction and review (among other topics)
Learning C and Linux

• Resources on the website
  – I’ll add more
• Next week’s discussion sections will review C and Linux
• ASK Questions
  – Early and often
That said...

• I want to know where you are now so that we can plan that review

• Assignment:
  – On Canvas, take the practice quiz “C and Linux Quiz” by Friday 11:59p.
    • Questions should cover familiar knowledge
  – Quiz includes an OS Concept Inventory
    • These questions likely are completely unfamiliar, as is the terminology
Announcements

• No discussion section this week
• Canvas quiz (ungraded) due Friday night
• Homework 1 will be posted on Friday
• Slides for Monday will be posted prior to class on Monday (maybe Friday?)
• Office hours begin next week
• Read the syllabus
• It’s going to be great!