CS439: Principles of Computer Systems

Dr. Alison N. Norman
Department of Computer Science
The University of Texas at Austin
Spring 2019
Who am I?

• Education
  – Undergrad in CS from Georgia Tech
  – MS and Ph.D. in CS from UT Austin
• Teaching: CS439, CS398T, CS109
• Service
  – WiCS Faculty Advisor
  – Coding in the Classroom, Hour of Code, and Code Orange advisor
  – Lab director for First Bytes and Code Longhorn
• Research: Supercomputing
• Family
  – Married
  – Three children: two boys (11 and 7) and one girl (4)
  – Two dogs (14.5 and 12.9)
• Addressing me:
  – Alison or Dr. Norman
  – she/her/hers
Today’s Plan

• Introduce and motivate course themes
• Course organization and logistics
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?
Where’s the Operating System?
Where’s the Operating System?
Operating Systems: More than One Hat
Operating Systems as Referee

• Manages shared resources
  – 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 interfaces to the hardware

- Simplifies application design and facilitates sharing
- Decouples hardware and application development
- Start, stop, and clean up after a program
- Examples: File system, virtual memory, networking
Evaluating an Operating System
Reliability

• OS does exactly what is designed to do
  – 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
Security

• OS cannot be compromised by a malicious attacker
  – Includes privacy: data on the computer only accessible to authorized users

• Security policy
  – Defines what is permitted

• Enforcement mechanism
  – Ensures only permitted actions are allowed

• Strong fault isolation helps but is not enough
  – Security mechanisms should not prevent legitimate sharing!
Portability

• OS does not change as hardware changes
• OSs can live a really long time
  – Must support applications not yet written
• 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*
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 deliver a response to the user?
• 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.
some learning skills

- Know what you don't understand
  - ooo, how does IPv6 work?

- Knowing what you do and don't understand is a skill! It takes a long time to learn!
  - ooo, it's a skill! It takes a long time to learn!

- When you're confused, identify why
  - ooo, what's going on?

- Build a lot of connections
  - ooh, these two different things I know are actually related!

- Fill in the gaps you want to fill in
  - Today I will learn IPv6!

- Remember learning is something you can get better at!
  - When I'm confused, now I can figure out why!

8:41 PM - 26 Jul 2018
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
THE EMOTIONAL JOURNEY OF CREATING ANYTHING GREAT

99% of what outsiders see

This is the best idea ever!!
This will be fun
This is harder than I thought
This is going to be a lot of work
This sucks I have no idea what I'm doing
Dark swamp of despair
Ok but it still sucks
Quick, let's call it a day and say we learned something
Hmm...
Hey!
Wow

THE EMOTIONAL JOURNEY IS INEVITABLE AND PERHAPS NECESSARY

https://john.do/emotional-journey-creating/
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!
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, group formation, and assignment submission
  – 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
• I’ll talk, you’ll talk
  – Your talking is essential to your learning in the class
  – Try to engage and venture answers---don’t let your classmates do all the work
  – If you find that you always answer, try to take a step back and wait for others to respond
• Talk-break-talk... divided up some way over the 110 minutes we have
• Class ends at ten minutes to the hour
Weekly Discussion Sections

• Review lecture content
• Work an additional problem set problem
• Review the problem set
• Discuss projects
• In groups!
• Lots more talking. By you. :)
More about the Problem Sets

• Assigned weekly (twelve total)
• Designed to help you prepare for the exams
• Posted online (linked from the schedule)
• Work and bring your solutions to discussion section
  – Except this week. This week, you submit through Canvas.
Projects

• There will be 5 projects in this course
  – Each includes implementation and a design document
  – Pair or group programming is required
• They will not be equally weighted
• They will be difficult
  – Systems programming is difficult!
• Your life will be easier if you learn the Linux environment
Pair or Group Programming

• Program together 80% of the time
  – That 20% allows for cases when the bugs won’t let you sleep
• Switch drivers every 20-30 minutes
  – Everyone drives an equal amount
• Keep a pair programming log
  – Help to ensure fairness
• Be respectful
  – Arrive prepared and on time
  – Pay attention when someone is talking
  – Pay attention when someone is coding
  – Maintain personal space
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
  – Take breaks
Evaluation

• Exams and a final (16%, 16%, 22%)
  – Exams are 2/27 and 4/10 (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

• Projects (34%)
  – Build operating system components
  – Due 5:59p (code) and 11:59pm (design doc) on select Fridays
  – More information soon
Evaluation

• Discussion Section Participation (7%)

  To get credit:
  – arrive on time
  – arrive with the assigned problem set completed with an honest attempt at the answers
  – bring a hard copy of your problem set answers and anything else we request
  – display your name tent
  – participate in the group discussion of problem set answers and summary of the lectures
  – be ready to present an answer whenever selected by the TA
  – participate in other parts of discussion
  – stay until discussion section is complete
Evaluation

• iClicker participation (5%)
  – Instant feedback for me
  – Gives you a reason to come to class
  – To get full credit:
    • Register your iClicker
    • Bring your working iClicker
    • Sign in on the roster
    • Answer at least one question with your iclicker
  – Using laptops or other digital devices forfeits your participation credit
  – iClicker credit begins next class!
Evaluation

• 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 85% of Piazza posts every single time I check during the semester (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
  – Uploading material to a third-party website for future students to use
  – **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.
• Sleep, eat
• 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, 1/28*
  – Piazza
    • Use Anonymous feature if necessary

• Many resources/tutorials on course main page
Teaching Staff

• Teaching Assistants:
  – Ashay Lokhande
  – Kiana Alcala
  – Sunaina Krishnamoorthy
  – Will Lin
  – Kishan Patel
  – Liliana Terry
  – Noah Thornton
  – Jesus Palos
  – Joel Swiatek

• And, obviously, me.
“It’s not that I’m so smart, it’s just that I stay with problems longer.”

-Albert Einstein
Problem Set 0

• Problem Set 0 will be posted this afternoon and is due Friday at 11:59pm
• You will find a link to it on the schedule
• Includes:
  – A quick meeting with Alison
  – A Code of Conduct
  – An Academic Integrity quiz on Canvas
  – A “C and Linux Quiz” on Canvas
    • Many questions should cover familiar knowledge
    • Also includes an OS Concept Inventory
      – These questions likely are completely unfamiliar, as is the terminology
  – A quick C-programming assignment (if you understand pointers!)
• This problem set should be submitted electronically through Canvas (unlike the others)
Announcements

• No discussion section this week
• Problem Set 0 due Friday night
• Slides for Monday will be posted prior to class on Monday (maybe Friday?)
• Office hours begin Monday
• Read the syllabus
• It’s going to be great!
• Next time:
  – History of Operating Systems
  – Dual Mode Execution