Introduction

CS 450 – Spring 2019 - Hale
A bit about your instructor

• IIT tells me I have to do this
• I’ve been doing systems research for about 10 years
  • Computer architecture
  • Virtual machines and high-performance hypervisors
  • Systems security
  • High-performance operating systems (see http://nautilus.halek.co)
  • Parallel computing
• Grew up in Texas. PhD at Northwestern, joined IIT in 2016
What is an Operating System?
• Not easy to answer!
• Depends on what the needs of the system are
• We can try to think about it in terms of its place in the HW/SW stack:
Typical HW/SW stack

- Hardware
- Operating System
- Libraries
- Applications
Things are more complicated these days though!
Cloud

VM

Applications
Libraries
Operating System
Hypervisor
Hardware

Applications
Libraries
Operating System
Hypervisor
Hardware

Applications
Libraries
Operating System
Hypervisor
Hardware
Supercomputing

- Applications
- Libraries
- Normal OS (Linux etc.)
- Communication Channel
- Hardware
- Applications
- Libraries
- Super fast OS
How about the *roles* of an OS?
Library (Abstractions)
• What if you had to write a keyboard driver and a video driver for every program you wrote?
• Abstract away the hardware
• What are the best abstractions? Are they universal?
• (no)
Accountant (Resource Management)
• Allow several programs (and/or users) to use the same machine
• Fairly allocate resources in the system
• Schedule things so they happen in a timely manner
Cop (Security, Isolation, Protection)
• Separate privilege
• Sandbox programs
• Protect programs from one another
OS/kernel distinction
What will we cover?
OS Architecture(s)
• Monolithic
• Microkernel
• Unikernel
• Hybrid
• User vs. kernel
Virtualization
• The illusion of having the whole machine
• Particularly CPU and memory
• Crosscutting topic in systems! (Broadly, resource virtualization)
• (demo)
Concurrency
• OSes must handle events that occur *concurrently*

• Allow things to happen in this manner for *independent* processes

• Allow things to happen in this manner for *interacting* processes

• Subtle distinction between concurrency and parallelism
I/O
• Need to interact with devices
• How?
• MMIO/PIO
• Interrupts
• Drivers, devices, PCI, etc.
File Systems and Persistence
• Disk (device characteristics, scheduling)
• We need data to *stick around*. How do we accomplish that?
• Abstractions (file, directory)
• Interfaces
• Resilience
Advanced Topics
• Networked and Distributed Systems
• Programming Models
• Multicore, SMP, Parallelism
• Virtual Machines
• Containers, Unikernels, Emerging Trends
Overarching Themes
• Taming complexity
• Scaling
• End-to-end principle
• Policy vs. Mechanism
• Secure practices
• Good programming patterns
Why should you care?
• Understanding the machinery (man behind the curtain!)
• Understand performance of programs
• Put things you’ve learned together
• Understanding and building complex systems is fun! Especially when they work!
• OS dev very rare (and sought after) skillset
Logistics

• Office hours
• Homeworks
• Reading
• Projects (user vs. kernel)
• Exams
Things to do now

• Take a look at the course webpage
• Make sure you’ve signed up for Piazza
• Do the assigned reading
• Start on project 1a when I post it tonight
Summary
• We will be covering *a lot* of ground
• You will be doing *a lot* of programming (this is the only way to get better at it!)
• This course will be challenging! But you will rise to the challenge