CSci 4061

Introduction to Operating Systems

Administrivia, Intro

Welcome to 4061!

- Me:
  - Jon Weissman
  - UMn CS&E Professor since 1999
  - Call me Jon

Class Structure

- Lectures are in person and identical
  - Need remote: register for UNITE
  - If need to pivot, we will do it FAST
- Labs are in person and identical
  - Go to any TA for help
- COVID-19
  - See syllabus; masking optional

Class Structure (cont’d)

- Lecture
  - Cover concepts and abstractions
  - Provide examples and use cases
- Labs
  - Brief week review, examples, Q&A, exercise to submit
- Class engagement
  - Short Canvas quiz over the weekend
  - Submit exercise solution from lab
  - Both Informally graded and based on prior week

Introduction

- CSCi 4061 is a rigorous course
  - **Systems programming focus**
- Expected background
  - Some machine organization and architecture
  - Some C or C++
  - Know how to login, navigate, edit, program, debug on Linux system

Course Outcomes

You will learn how to:

- Write code that exploits OS features to write "systems programs"
- Write code that is efficient, reliable, and possibly secure

You will also learn a little bit about OS internals but mostly the EXTERNAL interface

Course Outcomes (cont’d)

You will learn about the UNIX/Linux interface but not every parameter setting

(*you are expected to search out some details*)

You will learn about general systems programming concepts beyond just OS

Maybe something on ethics

Course Guideposts

- Adding top-down perspective
  - I want to build the Web (browser, server) what do I need?
- Numerous “landmarks”
  - Core systems concepts

Why C?

- High-level languages are too far away from the machine
- Examples of applications that must be fast and use low-level OS facilities:
  - JVM
  - Web browser/server
  - DB engine
  - Text editors
  - Compiler
  - Shell
  - Any app that needs direct hardware access (screen, camera, audio, …)
  - …

Android (anecdote)

- Sensor application
  - gcc (C) compiled code takes X time
  - Java compilation: 8-18X time

- C is tough but you will be up for it!

Our Perspective on OS

Two views

- internal view: what is inside the OS?
- **user view**: what can the OS do for me?

User view focus

- Abstractions
- APIs/Libraries

To Be Successful in 4061 …

- Be able to hunt down materials on your own (beyond the book)
- Be willing to learn by doing
- Be able to work effectively with others
- Ask questions

Books

- Required: **Unix Systems Programming: Communication, Concurrency, and Threads, Robbins and Robbins** (R&R)

- Optional (inside-view):
  - Operating Systems Concepts, Silberschatz, Galvin, and Gagne (S&G)
  - Modern Operating Systems, Tanenbaum (MOS)

- Optional (Systems Programming):
  - **Linux Systems Programming by Love** (LSP)
  - Unix Systems Programming, Haviland et al
  - Advanced Programming in the UNIX Environment, Stevens

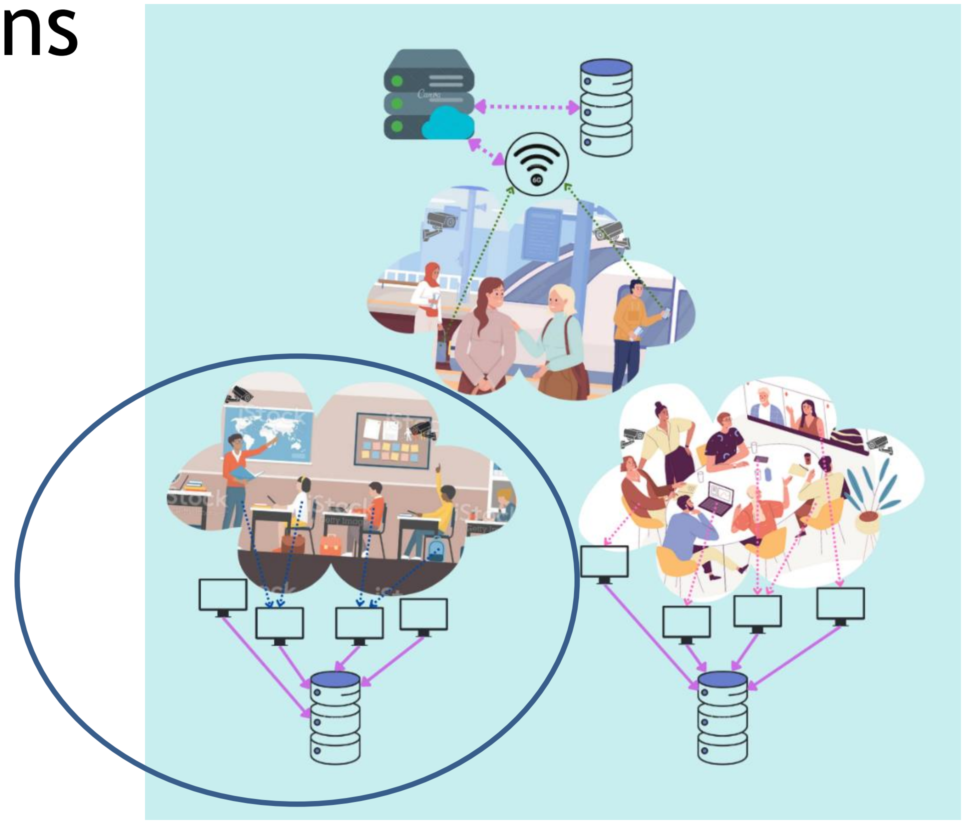
- Optional (C programming): see course resource module

Graded Coursework

- Engagement: informal
- Exams to test conceptual material and programming skill
- Projects
  - group of 3; same grade IF shared load
  - test programming skill
  - no late work, can drop lowest

New Twist: Edge Computing

- Integrate some research ideas into class operations



ADMIN Questions?

Topics

- OS Overview
- Programs and Processes
- I/O and devices
- File systems
- Communication
- Exceptions
- Threads
- Synchronization
- Memory Management
- Network programming
- System Design

Cross-cutting theme 1

- Concurrency
  - activities (resource sharing) *appearing* to occur at the "**same time**":

    processes, threads, synchronization

    Examples from daily life?

    concurrent: "taking CS, Math, English courses in Fall 2022" (brain)

    parallel: "washing dishes and listening to ipod" (hands vs. ears)

Cross-Cutting Theme 2

- Asynchrony
  - dealing with **unpredictable** events (**in time**): exceptions, devices, I/O

  Examples from daily life?

"when mom wants to talk, my phone rings"

Cross-Cutting Theme 3

- Communication
  - information **transfer**:
    communication, network programming

“Jon -> Project Assignment -> Submit Solution”

What is an OS?

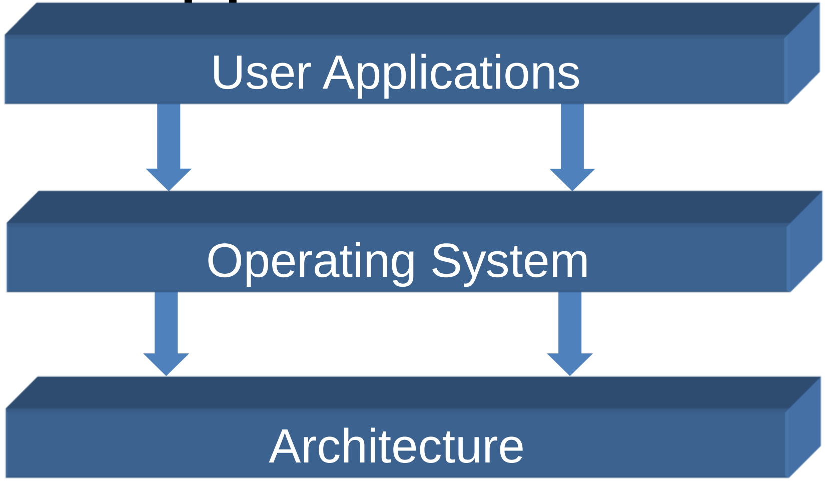
- Simple definition:
- A program that acts as an intermediary between a “user” of a computer and the computer hardware.

Stakeholders?

- User
- “System”
- Owner
- Admin
- Which may be at odds in terms of goals?

Operating Systems: Two Interfaces

- The operating system (OS) is the interface between user applications and the hardware.



- An OS implements a *virtual machine* that is easier to program than the raw hardware

Operating System Roles

- Referee
  - Resource allocation among users, applications
  - Isolation of different users, applications from each other
  - Communication between users, applications

- Illusionist
  - Each application appears to have the entire machine to itself
  - Infinite number of processors, (near) infinite amount of memory, reliable storage, reliable network transport

- Handy-person
  - Libraries, user interface widgets, drivers, …

Example: File Systems

- Referee
  - Prevent users from accessing each other's files without permission
  - Sharing disk space across the file system
- Illusionist
  - Files can grow (nearly) arbitrarily large
  - Files persist even when the machine crashes in the middle of a save
- Handy-person
  - Named directories, printf, …

Summary

- Great to get started with you!
- Always a fun class
- Next time: gentle Introduction to OS concepts (Read Chapter 1)
- Have any questions?