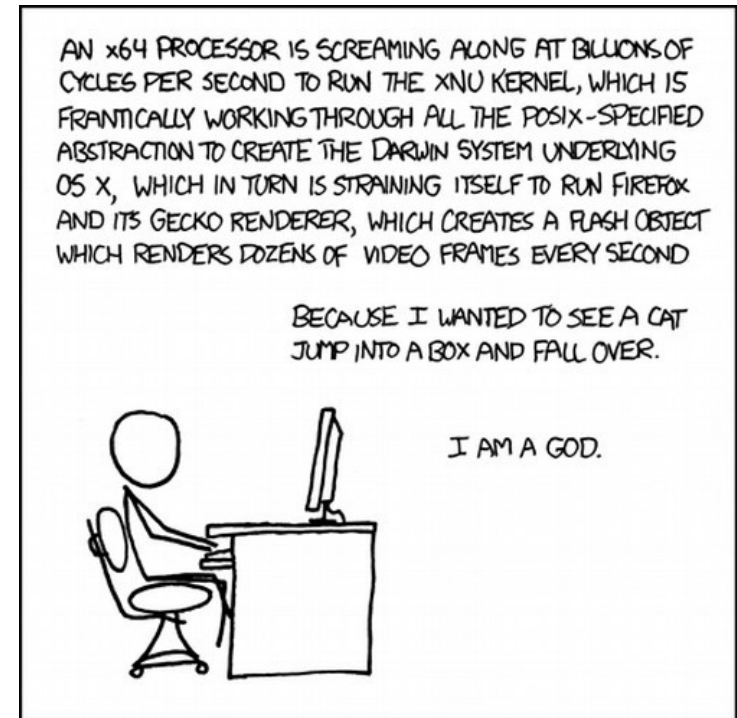


CS 261

Spring 2024

Mike Lam, Professor



<https://xkcd.com/676/>

Computer Systems I: Introduction

Question

- What will be the output of this C program?

```
#include <stdio.h>
int main() {
    double a = 1e20;
    double b = -a;
    double c = 3.14;
    if (((a+b) + c) == (a + (b+c))) {
        printf("Equal!\n");
    } else {
        printf("Not equal!\n");
    }
    return 0;
}
```

- A) “Equal!”
- B) “Not equal!”
- C) Neither of the above

Question

- Which of the following versions of a “matrix copy” routine will run the fastest?
 - A)

```
for (int i = 0; i < 2048; i++) {  
    for (int j = 0; j < 2048; j++) {  
        dst[i][j] = src[i][j];  
    }  
}
```
 - B)

```
for (int j = 0; j < 2048; j++) {  
    for (int i = 0; i < 2048; i++) {  
        dst[i][j] = src[i][j];  
    }  
}
```
 - C) Neither; they will always run at approximately the same speed.

What's happening?

- Something about our **mental model** of these programs does not match the **system** on which we're running them.

Systems

- What is a “system?”

Systems

- What is a “system?”
 - Set of interacting components
 - More than the sum of its parts



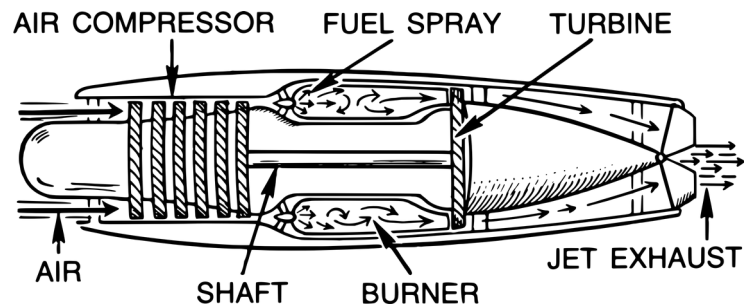
Jet engine



Computer

Systems

- What is a “system?”
 - Set of interacting components
 - More than the sum of its parts



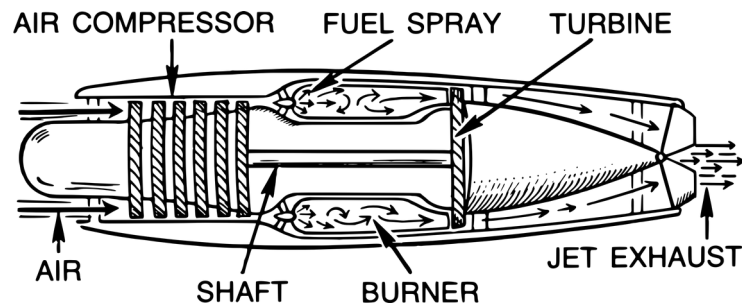
Jet engine



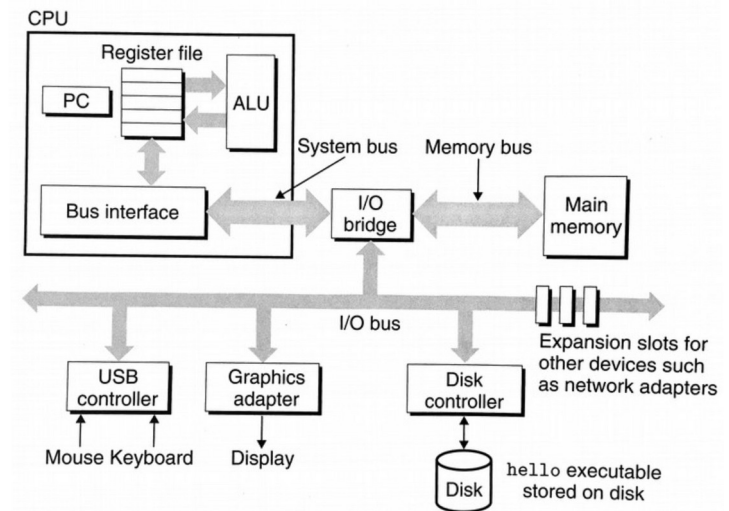
Computer

Systems

- What is a “system?”
 - Set of interacting components
 - More than the sum of its parts



Jet engine



Computer

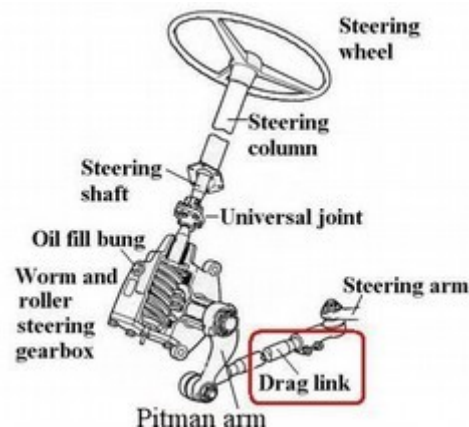
Systems

- A **computer system** consists of multiple hardware and software components that work together to run user applications.
 - We use complex computer systems every day
 - Our goal: peel back (some of) the complexity
 - See (some of) what's “under the hood”



Systems

- What is a *process*? What is a *file*?
 - These are examples of **abstraction**; "fake" views of reality that reduce complexity for users
 - Key ideas: **ignore details** and **focus on interfaces**
 - Especially important in large, complicated systems
 - Understanding abstractions can improve your ability to use them effectively



abstraction

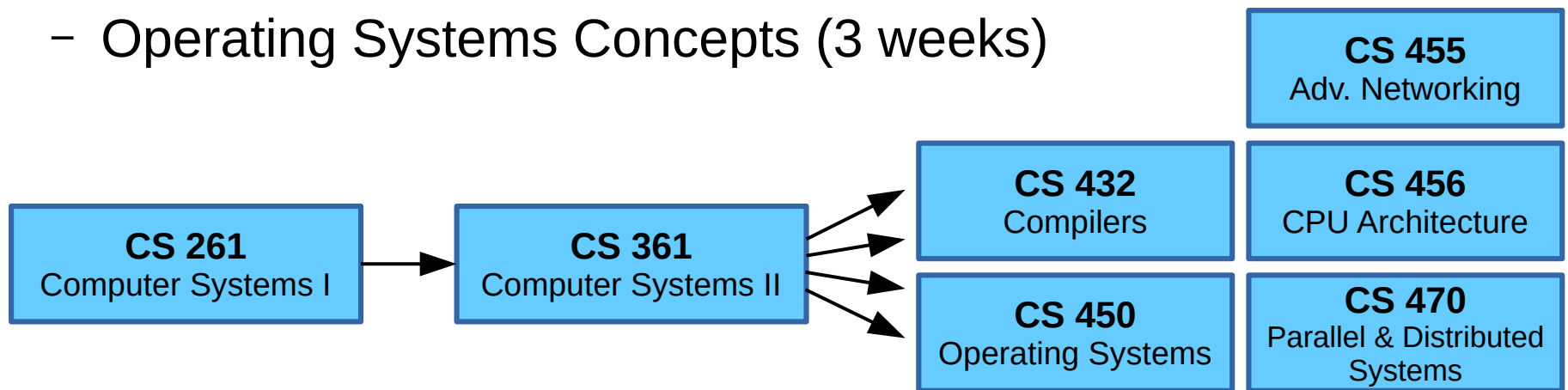


Course Objectives

- Explain machine-level representation of data and code
- Summarize the architecture of a computer
- Explain how complex systems are built from simple components
- Translate high-level code into assembly and machine language
- Write code to emulate the functionality of a computer
- Cultivate a sense of control over computer systems
- Gain an appreciation for software development tools
- Develop a sense of play when writing code
- Appreciate the complexity of systems-level software

Systems courses

- CS 261 units:
 - C and Linux (3 weeks)
 - Binary Representations (2-3 weeks)
 - Assembly and Machine Code (2-3 weeks)
 - Computer Architecture (3 weeks)
 - Operating Systems Concepts (3 weeks)



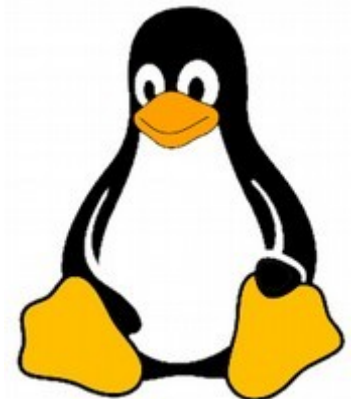
Fundamentals of digital,
single-process systems

Multi-process systems
and networking

In-depth study of a particular
kind of complex system

CS 261

- What this course is NOT:
 - Programming 101 – I will assume you can program
 - However, we will spend a few weeks learning C
 - Electronics 101 – we won't be going THAT deep
 - If you're interested, see PHYS 140/150 or 240/250 then 371/372
 - Linux 101 – but you have the Unix Users Group
 - Weekly meetings: Wed, 6:30pm, in King Hall 236
 - <https://www.jmunixusers.org>



CS 261

- This is not an “**easy**” course
 - *But you **can** succeed!*
 - I will set you up for success
- Commit to prioritize this course
 - Be prepared to **read** and **work** a lot
 - Don't be afraid to experiment
 - Practice a *growth mindset*: “I can't do it **YET**”
 - Take advantage of office hours and Piazza
 - Start projects **early** and ask questions



Semester-specific info

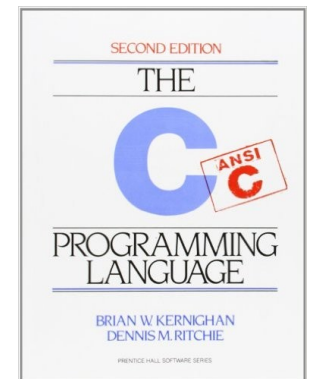
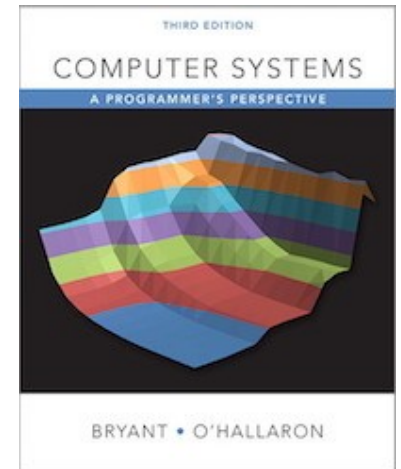
- The remaining slides are specific to Spring 2024
 - All slides are posted on the website (calendar page)
- Health and safety concerns
 - If you test positive for COVID or the flu, or are coughing and/or sneezing frequently, **please stay home**
 - Contact me ASAP regarding missed class
 - If you feel ill but well enough to attend class (and are NOT coughing/sneezing frequently), please consider wearing a surgical or N95/KN95 mask to protect others
 - These policies may change
 - Changes will be announced via Canvas message

Textbooks

- Required textbook: “Computer Systems”
 - “CS:APP” textbook from Carnegie-Mellon
 - A practical, example-filled introduction
 - Electronic rental available via RedShelf
 - Reserve copy at the Rose library

Important: Readings are listed on their associated quiz

- Recommended book: "The C Programming Language"
 - Brian Kernighan and Dennis Ritchie (creator of C)
 - This is “the book” about C (we’ll refer to it as “CPL”)
 - Scanned excerpts on Canvas (do not redistribute!)
 - Reserve copy at the Rose library



Course Grades

Quizzes and Labs 25%

Programming Projects 25%

Module Tests 15%

Exams 35%

- Quizzes and labs are **formative**
 - Designed to help you learn
- Tests/exams are **summative**
 - Designed to assess what you have learned
- Caution: don't trust your overall grade in Canvas until after the midterm!
- Projects are **both**
 - Designed to help you learn C and reinforce other course concepts
 - Also designed to assess whether you are ready for CS 361

Course Components

- **Public website** (w3.cs.jmu.edu/lam2mo/cs261)
 - Syllabus, **calendar**, project descriptions, and resources
 - Links to lecture videos (YouTube, already posted)
 - Most recorded for Fall 2020; all still relevant this year
 - Links to slides (Fall 2022 versions posted)
 - Some Spring 2024 revisions may be posted as well if needed
- **Canvas course**
 - Quizzes, lab submissions, and module tests
 - Grades and private files (e.g., lab solutions)
 - Access to Piazza Q&A
- **Student server** (stu.cs.jmu.edu)
 - Project development and submission
- **Piazza**
 - Q&A (especially re: projects)

**Make sure you can
access ALL of these!**

Course Design

- This is a **flipped class**

- Research shows active learning is more effective than passive learning
- Ahead of time: watch lecture, do reading, take quiz
- During class: review last lab and work on new lab in small groups
- Outside class: work on projects, take module tests, and study for exams

	Monday	Tuesday	Wednesday	Thursday	Friday
In-class		Lab		Lab	
Out-of-class	Lecture videos, reading, and quiz		Lecture videos, reading, and quiz		
	Project work	Project work	Project work	Project work	Project work (deadlines every 2-3 weeks)

Video playlists, quizzes, and labs all have a common tag (the first day is “01”)

Class Policies

- Class attendance is necessary and expected
 - We will be completing labs most class periods
 - Find a group (2-4 people) to work with consistently
 - Use a name card for the first half of the semester
- Every person should fill out a separate copy of the lab
 - Work together and check each other
 - Ask for help when you are stuck or want to confirm something (your goal is to get to the right answers by the end of class)
 - Getting “stuck” or confused is intended; it’s how you learn!
 - Resist the urge to “speedrun” the labs or to work solo

Class Policies

- Submit as PDF on Canvas/Gradescope when done
 - **Scan as a black-and-white PDF**
 - Instructions: <https://wiki.cs.jmu.edu/student/canvas/start>
 - DO NOT submit raw photos
 - Double-check to make sure it “went through” and that it is legible
 - Submit before leaving class even if you’re not done yet
 - Guarantees at least partial credit if you don’t finish or forget to submit later
- Labs are “lightly graded” (w/o individual mistakes marked)
 - Solutions will be posted on Canvas (under Files → Lab Solutions)
 - Bring your solution to the next class for review
 - Come to office hours if you have further questions

Course Policies

- The projects in this course are **VERY** important!
 - One purpose of this course is to ensure you are ready to tackle harder projects in CS 361 and the system electives
- Projects are **individual** and **mandatory**
 - A “good faith” submission shows evidence of significant work and investment in writing a solution
 - A good faith submission gets you an “F” (25 points) instead of a zero (!!), in terms of a numeric grade
 - Doing *at least* this on every project is **required** to pass the class

Course Policies

- The JMU Honor Code applies on ALL assignments
 - Violations may be sent to the honor council
 - See relevant section in the syllabus
 - All online quizzes and module tests must be completed **on your own** with no assistance aside from what is allowed in the assignment description in Canvas
- All submitted labs must represent YOUR work
 - You will work in groups to discuss the answers
 - By submitting a PDF on Canvas, you are asserting that these answers are YOUR answers and that you understand WHY you have answered the way you have

Course Policies

- All submitted project code must be **YOUR** work entirely
 - You may talk with others to discuss general approaches (in fact, I encourage this; use *pseudocode* if necessary)
 - However, one goal of the projects in this course is to develop individual competency in C programming, so **you may NOT share code** with anyone who is not a TA or CS 261 instructor
 - This includes letting someone examine or take a screenshot of your code, or “talking it through” with them line-by-line
 - This also includes using an AI-assist tool (e.g., Github Copilot)
 - If you co-work, sit such that you can’t see each other’s screens
 - Do not store your solution in a public online code repository
 - If you have questions about this, please ask!

Course Policies

- There are a total of four sections of CS 261
 - Two Lam sections and two Aboutabl sections
 - Some course materials are shared
 - You are welcome to study with students from other sections, but you must attend and submit assignments to the section you are registered for
 - DO NOT assume assignments are identical or that due dates align

Office hours

- My drop-in office hours are posted on Canvas
 - In person: King Hall 227
 - If I'm unavailable when you arrive, scan the QR code by my door
 - You'll join the same queue as virtual attendees and I will call you when I'm available (you'll leave a cell number as part of sign-up)
 - Virtual via Zoom: bit.ly/lam-office-hours-sp24
 - This is sometimes preferred for coding questions
 - Be prepared to share your screen, and turn your webcam on!
- CS TAs: in-person and virtual office hours: bit.ly/CS-TAs
 - 261-specific TAs TBD

TODOs in the next few days

- If you haven't already:
 - **Take welcome survey on Canvas**
 - **Take syllabus quiz on Canvas** (due Friday)
 - **Read CS:APP Ch. 1 and take Quiz 01** (due Friday)
- Before class on Thursday:
 - Review these slides and the syllabus and come with questions
 - **Watch “Command line and C compilation” lecture videos**
 - **Read 02-CPL excerpts** (on Canvas under Files → Readings)
 - **Take Quiz 02** (due Wednesday)
 - **Make sure you can log into stu**
 - Instructions at the top of Thursday's lab: w3.cs.jmu.edu/lam2mo/cs261/02-cmd_line.html
 - Make sure you can access Piazza
 - Skim the project guide and Project 0 description (on website)

Intro lab (Tuesday)

- Material from Chapter 1
 - Front page: **Computer Organization**
 - Back page: **C Compilation**
- Submit as PDF on Canvas when done
 - **Scan as a black-and-white PDF**
 - Instructions: <https://wiki.cs.jmu.edu/student/canvas/start>
 - DO NOT submit raw photos, and double-check for legibility!
 - Let the instructor know after you submit for verification
 - Once you have verified a satisfactory submission, please feel free to leave – I'll see you on Thursday!

Closing exhortations

- Take care of yourself
 - And if you can, someone else
 - Build (or reconnect with) a support network
 - Protect your boundaries
 - Carve out time to disconnect and rest
 - Talk to someone if things start getting overwhelming
- Have a great semester!