



**MONTANA**  
**STATE UNIVERSITY**

# CSCI 366 - Systems Programming

...

Instructor: Carson Gross

[carson.gross@montana.edu](mailto:carson.gross@montana.edu)

# Course Goals

“Introduce students to fundamental concepts in computer systems, including software environments and development tools, computer architecture and organization, concurrency, information management, network communications, and operating systems based on cloud computing.”

# Course Goals

- Become familiar with common tools used in Systems Development
  - CLion, CMake, git/Github, Google Test
- Become more familiar with C as a programming language
  - Threading, locking, memory management, managing application state
- Understand assembly programming
  - MIPS, Some x86
- Understand computer architecture at a high level
  - Main components of a computer, binary representations, basic logical gates
- Understanding Networking
  - Socket programming, Cloud Architecture, SQL

# Instructor Introduction

- My name Carson Gross
- Just arrived from Northern California
- Academics
  - BS in IE/OR from Berkeley
    - (I dropped out of CS)
  - MS in CS from Stanford
- Career
  - Interned at Google
  - Staff Engineer at Guidewire Software
  - Gosu Programming Language
  - Founder/CTO at LeadDyno
  - Creator of many open source projects
    - <http://intercoolerjs.org>
    - <https://htmx.org>
    - <https://hyperscript.org>

# Classroom Mechanics

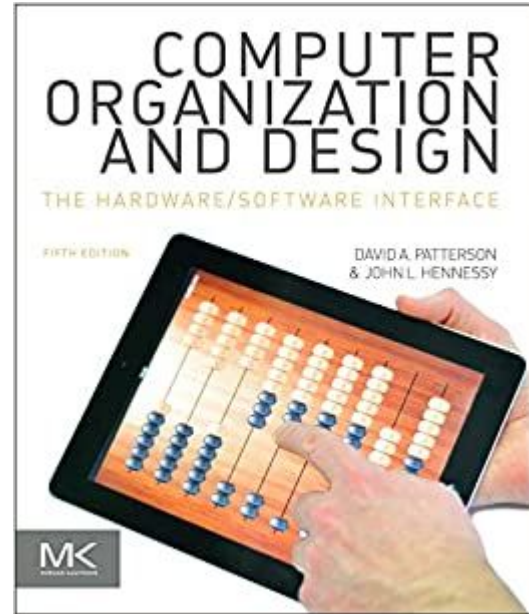
- Lectures are at 9AM, MWF
- All lectures will be live streamed via YouTube from my office
- All assignments, lectures and tests will be available online

# Note On Course Difficulty

- This was a *very* tough course for me at Berkeley, because I found the low level mechanics of computing difficult to follow
- Try to keep in mind that it's all "just code" and that computer systems are layered
- Don't try to understand all the layers at once: try to understand a given layer and then move on, assuming that the previous layer works
- *It's Just Code!*

# Books

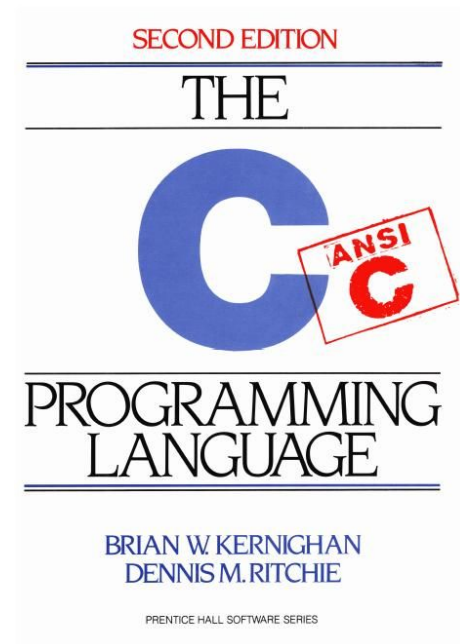
Computer Organization & Design  
*The Hardware/Software Interface*  
MIPS Edition





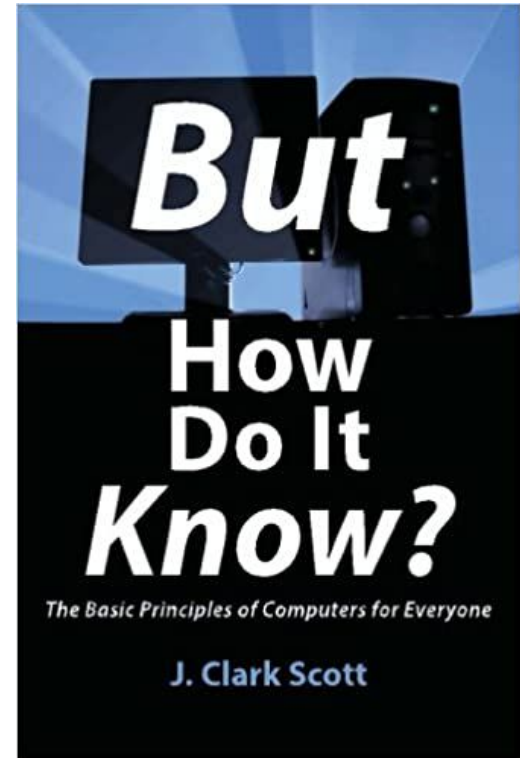
# Books

## The C Programming Language



# Books

But How Do It Know?  
*The Basic Principles of  
Computers for Everyone*



# Course Outline

- Class Introduction & Tools
- System Software Languages
- Assembly & Machine Code
- Computer Architecture
- Networking

# Class Introduction & Tools

Introduction to the tools necessary to do systems-level development. Most will carry over into higher-level development!

- Typing (a crucial skill!)
- Editors (Vim, CLion, etc.)
- Git/Github
- Markdown
- Command Line Tools

# System Software Languages

- System level development is still done primarily in C (!?!?)
- C is a little bit insane:

```
void (*bsd_signal(int, void (*)(int)))(int);
```

<https://www.geeksforgeeks.org/complicated-declarations-in-c/>

- We will also look at C++, Rust and a few others...

# Assembly & Machine Code

- Assembly is a very low level programming language
  - Pretty much just above binary
  - C has been called "glorified assembler" and "portable assembler"
  - We'll see how you feel about that after programming in some assembler
- We will start with Little Man Computer, an excellent tool that demonstrate (roughly) how assembly on early computers worked
- We will move up to both RISC and then CISC (x86-64) assembly from there
- Some Assembly Required in this class...
  - that's a *terrible* joke

# Modern Computer Organization

- I will give a high level introduction to the low level mechanics of computation
- We will briefly discuss logical gates
- Binary representation
  - Integers
  - Floating Points
- Modern CPU
- Cache & Memory
- Etc.

# The Network

- Finally, we will finish up by looking at networking
- Socket Programming
- Overview of Web Programming & Cloud Computing
- SQL Introduction & Overview



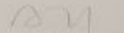
# Grading

- Programming assignments - 70%
  - Designed to give you practical experience with system implementation
  - Done individually, graded with unit tests, checked for plagiarism
- Quizzes - 20%
  - Designed to test your theoretical knowledge of the concepts covered in class
- Homework - 10%
  - Designed to give you practice on the concepts covered in class
- Extra credit - ?%
  - I can offer extra credit for various parts of the course (e.g. implementing part of the project in assembly)

# Cheating

- **Don't**
- Seriously, you are only cheating yourself. You will need to interview to get a job & you don't want to embarrass yourself
- What's OK:
  - Discuss programming assignments (not answers) with other people
  - Helping other people debug (not write) their program
- What's Not:
  - Share code with other people
  - Submit code that you did not write (small snippets from StackOverflow OK)
  - Modify someone else's solution and claim it as your own

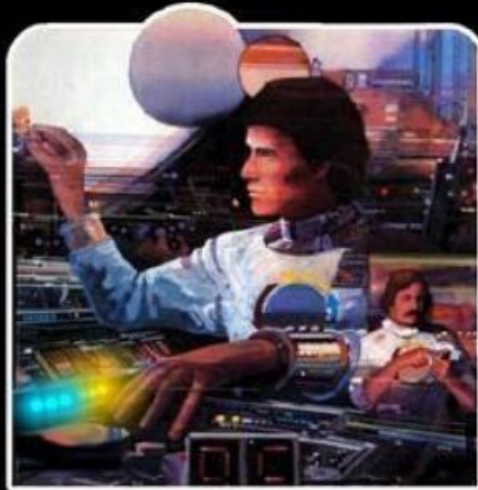
**WORK  
HARD  
&  
BE NICE  
TO PEOPLE**

  
Anthony Barill

# Homework 0

- Install VirtualBox
  - Download the class VirtualBox image (posted tomorrow)
  - Be sure to set CPUs to at least 2 (defaults to 1)!
- If you are going to develop on your own, install the following tools:
  - CLion
  - Nasm
  - CLion NASM plugin
  - CMake
- Play around with CLion, try out debugging

# THE TWO STATES OF EVERY PROGRAMMER



**I AM A GOD.**



**I HAVE NO IDEA  
WHAT I'M DOING.**



**MONTANA**  
**STATE UNIVERSITY**