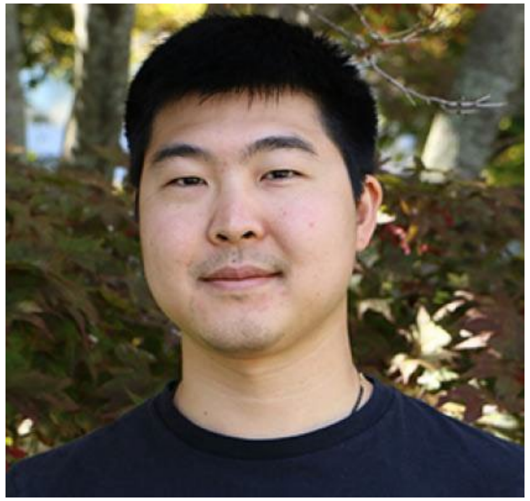
Software Systems

Day 1 – Course Overview

Instructor



- Steve Matsumoto (he/him) – just call me Steve.
- Contact: smatsumoto@olin.edu
- Office hours: Wed/Thu 9-10am, MH 329
- 4th year at Olin, 3rd time teaching SoftSys.
- Research in computing education, security/privacy, cryptocurrency.
- Hobbies: trivia, board games, cooking, mixology
- My first programming experience was writing scripts for my TI-83 Plus to help me with my math homework. Nothing fancy, just pretty BASIC stuff.

Course Assistants

- Course Assistants (CAs) work a lot of magic behind the scenes to make sure this course runs smoothly.
- You might catch them in class or in CA hours.
- Our CAs:
  - Caitlin Coffey (she/her)
  - Jonas Kazlauskas (he/him)
  - Krishna Suresh (he/him)

What’s this course about?

- Systems programming in C
- Operating systems design and implementation
- Software synchronization

What is a computer actually doing when it runs your code?

What effect does that have on a program's behavior or performance?

Intro Activity

- We're going to dive deeper into what this course is about through an introductory activity.
- The goal of this activity is to experience some of the problems that you might encounter in this course, but in a non-programming context.
- Feel free to be creative, ambitious, etc. – but don't overthink it.
- We'll do a debrief discussion afterwards.

Intro Activity: Setup

- Form teams of 4-6 at tables.
- Each of you will be given a set of materials.
- For this activity, you can **only** use the materials given.
  - You cannot use your own materials or get additional ones.

Intro Activity: Instructions

- For the next 10 minutes, do the following:
    - Read how to make a paper plane: https://www.foldnfly.com/1.html
    - Make as many paper planes as possible.
    - For a plane to count:
        - It must be able to fly a "reasonable" distance (depending on the room layout).
        - You must legibly write "Yay SoftSys Spring 2023 FTW!" along the bottom of **both** wings.
        - There should be no writing on the plane other than on the bottom of the wings.
    - You can split the workload amongst yourselves however you like.

Intro Activity: Discussion

- We'll have a 10-minute table discussion, then share with the class.
- Discuss the following with your table:
  - How did you split up the workload/tasks?
  - What went well with your arrangement? What didn't go so well?
  - Given the benefit of hindsight, what might you do differently?
  - If you could do this again, how many paper planes do you think you would make? Why?
  - What do you think this activity has to do with SoftSys or what a computer is doing when it runs code?

Intro Activity: Debrief

- Some lessons to take away:
  - Performance can depend on a lot more than how much resources you have.
  - There are many ways to optimize around limited resources.
  - Understanding dependencies and bottlenecks within a process is important.

What’s this course about?

- Systems programming in C
- Operating systems design and implementation
- Software synchronization

What is a computer actually doing when it runs your code?

What effect does that have on a program's behavior or performance?

What’s C?

- General purpose programming language dating from the early 1970s
- Replaced by Java and Python in intro classes, because C is a steeper learning curve
- Now used for:
    - Implementing operating systems or run time systems
    - Network programming (clients and servers)
    - High-performance and scientific computing
    - Graphics/GPU programming
    - Embedded systems
    - Digital signal processing
    - Hardware synthesis

What about operating systems?

- How code is turned into an executable program.
- How programs run.
- How files and memory are stored and managed.
- How data is represented.
- How computers do multiple things at once.

How does software synchronization work?

- Software synchronization isn’t as easy as it might seem at first.
- Multiple things can read/write the same memory at once.
- Need to also make sure that multiple threads or processes don’t get stuck waiting for the same resource.
- Implementation patterns for common synchronization problems.

Note about course plans

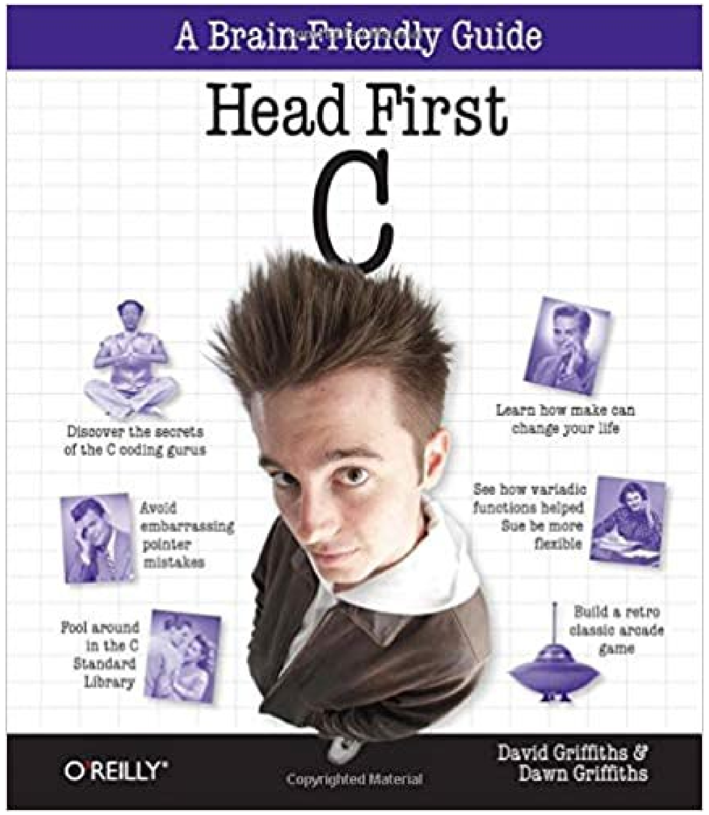
- We're doing a bit of a course redesign this semester.
- Please bear with us if things are a bit rough.
- You can **always** come talk to me about potential changes to the course.
  - If you're willing to put in some work to develop readings/questions/exercises or curate resources, I'll also give you extra credit.

What do we do in this course?

- Readings
- Quizzes
- In-Class Exercises
- Homework Exercises
- Projects

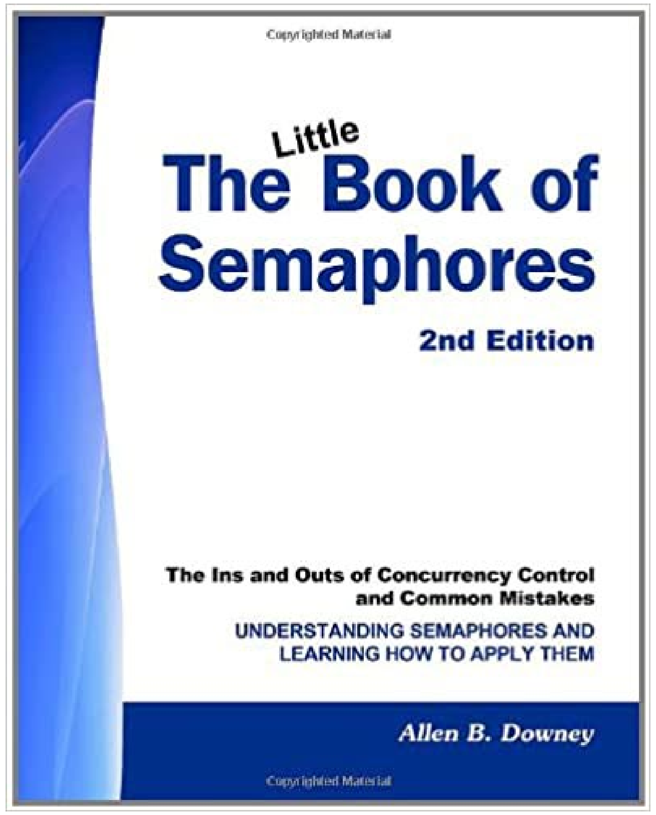
Course Books

- Books used in this course:


A Brain-Friendly Guide
Head First
C
Discover the secrets of the C coding gurus
Learn how make can change your life
Avoid embarrassing pointer mistakes
See how variadic functions helped Sue be more flexible
Fool around in the C Standard Library
Build a retro classic arcade game
O'REILLY
Copyrighted Material
David Griffiths & Dawn Griffiths

Think OS:
A Brief Introduction to
Operating Systems

Allen B. Downey



The Little Book of Semaphores

2nd Edition

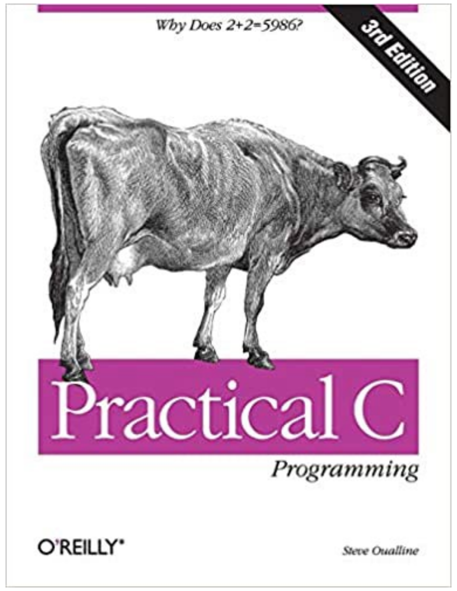
The Ins and Outs of Concurrency Control and Common Mistakes

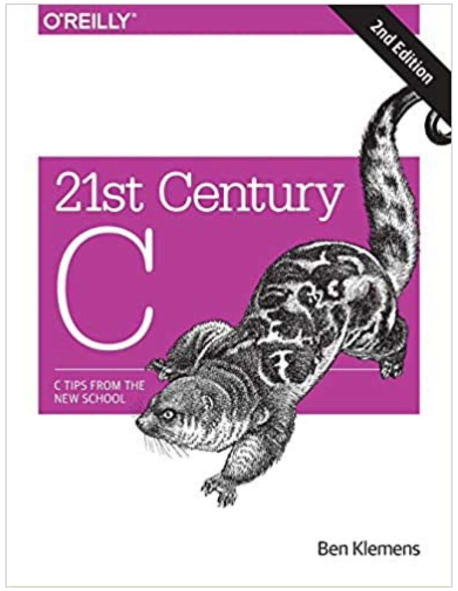
UNDERSTANDING SEMAPHORES AND LEARNING HOW TO APPLY THEM

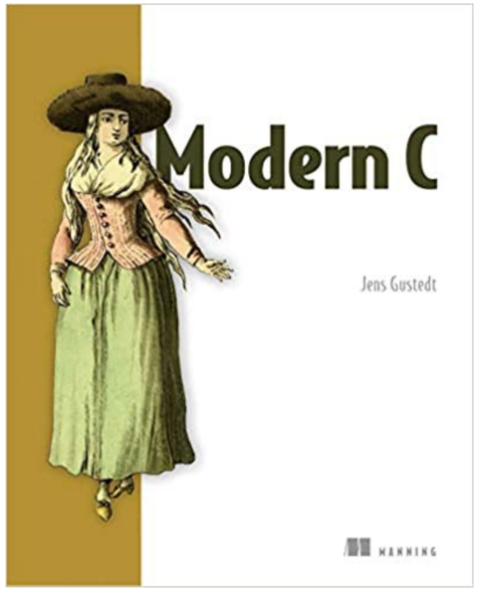
Allen B. Downey

Course Books

- Some people like Head First C, some don't.
- In my opinion, the content level is good, but the presentation is meh.
- If you don't like it, there are other options.







Readings/Quizzes

- Try to at least skim the readings before class.
- We’ll review some material in class, so you have an idea of what you should be getting out of the readings.
- Readings usually have short quizzes to check your understanding.
- There may be other quizzes (on Canvas or in person) to measure your understanding of longer-term concepts.

In-Class Exercises

- We’ll do in-class exercises to help you learn.
- Not all of these are programming – some are more conceptual or exploratory.
- Mostly done at tables or in breakout rooms.
- Feel free to collaborate on Discord as well.

Homework Exercises

- Done out of class and submitted through GitHub.
- Assignment 0 will help you with the process.
- Deadlines throughout the semester (mostly weekly).
- A few optional exercises.

Projects

- Expect to have 2 projects, around 5 weeks each.
- General workflow is the same for both projects:
  - Pick a topic you’re interested in.
  - Find resources to help you learn about that topic.
  - Develop a software product that develops and demonstrates your learning.
- Examples of previous projects will be linked from course webpage.

Assessment

- Overall grading has four categories:
  - Quizzes: 20%
  - Homework: 20%
  - Project 1: 20%
  - Project 2: 20%
- Whatever you score highest in counts for 40% instead of 20%.
- Lowest n quizzes will be dropped (likely 2-4).
- 5 late days in total.

Computational Setup

- This course is taught in UNIX, so you’ll need macOS or Linux.
- Three main options for installing UNIX:
    - ~~Use a Linux virtual machine (VM).~~ Don't use this until we get a new VM image set up for you.
    - Use WSL. This is the preferred method.
    - Set up dual-boot: install Linux alongside your existing OS.
- Each method has tradeoffs, described on the website.
- Other methods are possible, but you’re on your own.

Git Setup

- Set up a GitHub account if you don’t have one already.
- Submit your username on Canvas.
- You’ll get an invitation to join the olincollege GitHub organization, as well as the course repository, softsys-20XX-YY.
- Next time, we’ll go over how to set up your GitHub repos.

Computational and Git Setup!

- Let's take some time to get everyone set up.
- Ask questions of CAs/instructors if you need it.

TODOs

- In general, Canvas always has an up-to-date schedule for what is due when.
- The "day assignments" show what you need to prepare for each class meeting.