Syllabus

Course Code	CAS CS 392
Course Title	Rust, in Practice and in Theory
Semester	Spring 2025
Instructor	Nathan Mull
Meeting Time	Tuesday and Thursday, 5:00PM-6:15PM
Meeting Location	CDS 364
Midterm Date	March 6
Grade Breakdown	30% Assignments
	40% Final Project
	20% Midterm Exam
	10% Participation

CAS CS 392: Rust, in Practice and in Theory is a course about the programming language Rust. Rust is a type-safe, memory-safe programming language that is becoming a popular alternative to C and C++ in settings where performance and memory usage are major concerns. It's self-described as having "high-level ergonomics" and "low-level control." Practically speaking, this means clear, concise code with fewer memory bugs. Theoretically speaking, this means the use of a rich type system based on the notion of linearity to enforce memory-safety before any code has actually been run.

Despite its popularity, Rust is still daunting to learn, even for experienced programmers. There are several concepts in Rust that don't appear in any other popular languages. And even if you become a proficient Rust programmer, it doesn't mean you have a deep understanding how Rust works, or why it is a better alternative to other low-level languages.

In this course, we'll spend the first half of the semester learning Rust. This can include topics like borrowing, lifetimes, traits, smart pointers, and concurrency. We'll spend the second half implementing a subset of Rust. This will help us better understand the details of Rusts type system and borrow checker.

General Information

Disclaimer

Please read this section carefully. This is a new course. It's the first time I'm offering it, it's on a quite modern topic and, to be frank, I'm not sure how well it's going to go. All of this is to say: if you're looking for a smooth learning experience, then I don't recommend taking this course. This course will require a level of academic independence and maturity that I expect of advanced undergraduate students. There's going to be experimentation, some things that work, some things that don't, some things with bad documentation, some things that will require you to look things up outside of the course material (gasp). But if you stick with the course, I hope it can be a valuable experience for all of us.

Course Structure

Please read this section carefully. This course will be treated as a hybrid of a *flipped classroom* and a *workshop*. Each lecture will have corresponding readings which you will be expected to have read *before* lecture, which will be required to answer the pre-lecture quizzes.

The first part of lecture will be dedicated to a short overview of the day's topic. Each lecture will have corresponding discussion questions. *I want to think of this part of lecture as closer to your standard humanities course*. It should be a high-participation discussion, with live-coding and board work. It's during this part that you should make sure you're on the same page as everyone else with respect to the material.

The second (primary) part will be dedicated to working with the material from the reading. This will almost always just mean working on the homework assignments. This means that homework assignments will be long, but you will have lots of opportunities to work on it and get help.

I hesitate to do this, but I'm going to say that **lectures are required**. You are allowed to miss a handful of lectures due to conflicts, but this course only works if you want to participate. If you feel like you cannot do this (be it because of social anxiety or something else) then we can have a conversation and see what other options there are.

Minutiae

This course meets Tuesdays and Thursdays from 5:00PM to 6:15PM in CDS 364.

Prerequisites

The formal requirements of this course are *CAS CS 320: Concepts of Programming Languages* and *CAS CS 210: Computer Systems*. You will have the best time with this course if you took CS320 during the Fall 2024 semester. It is not strictly a requirement, but if you did not, then you will have to catch up a bit on typing judgments. We'll have a crash course on this after the first half of the course, but it will take some time. Ideally, we should be able to create groups in which at least one member has this experience.

The dependence on 210 is quite mild. You need to be comfortable with the terminal, and you need to have some sense of low-level memory management. That said, the more versed you are in systems, more you'll get out of learning Rust.

Learning Outcomes

- ▷ Learn the rudiments of Rust, a notoriously difficult but ultimately rewarding low-level programming language
- ▷ Better understand the use-cases of Rust, when it works well, when it doesn't

- ▷ Identify the pitfalls of Rust programming and determine how to avoid them
- ▷ Learn to read/practice reading academic papers
- ▷ Learn to read/practice reading formal specifications of programming languages
- ▷ Implement an interpreter for a subset of Rust in Rust
- ▷ Prove properties like progress and preservation for a formal specification of Rust (properties which make Rust a ”well-designed” programming language)

Evaluation

The grade breakdown for this course is as follows

30%	Assignments (5% each, 2 dropped)
40%	Final Project (10% each part)
20%	Midterm Exam
10%	Participation

Midterm Exam

The midterm exam will be held on Thursday March 6th during class. This is right before the spring recess so plan accordingly. The exam will be on the fundamentals of Rust.

Final Project

The bulk of the grade in this project will be dedicated to the Final Project. As mentioned above, the final project is an interpreter for a subset of Rust written in Rust. There will be 4 parts to the project

- ▷ The parser
- ▷ The evaluator
- ▷ The type/borrow checker
- ▷ An extension

Course Resources

Piazza

Gradescope

Policies

Collaboration

Academic Integrity

Disability Statement

Diversity Statement

Sexual Misconduct

Generative AI

Student Resources