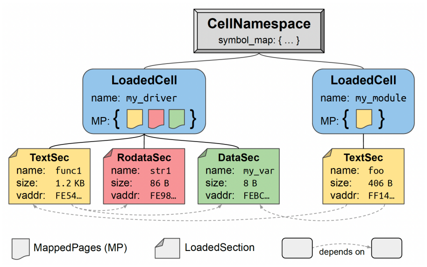
CS340R: Rusty Systems

Spring 2024
Tuesdays and Thursdays, 10:30-11:50
380-380W
Email: **cs340r-spr2324-staff@lists.stanford.edu**



Instructor: **Philip Levis**

- Office hours: **TBD**

CA: **Deepti Raghavan**

- Office hours: **TBD**

CA: **Rae Wong**

- Office hours: **TBD**

This course seeks to ask, and start to answer, a simple question: "What are the most important open research challenges for software systems written in Rust?"

CS340R is a graduate research course intended to explore how Rust, as a language, introduces new research questions. C (and C++) have dominated software systems for so long that their challenges (e.g., asynchrony, memory management, array indexing) are assumed as basic systems problems. Rust, however, has very different challenges. How does this change research?

CS340R is not intended to be a primer on the Rust language. While we will spend the first three weeks learning Rust, this is so that students can begin reimplementing a recent research system in Rust. Through this effort, we will learn how a Rust implementation differs and why. Are there parts of the original system which are hard but are easy in Rust (e.g., memory safety)? Are there parts of the system which are much harder in Rust (e.g., asynchrony, memory layouts, buffer management)? These latter discoveries may indicate places where research could make a contribution.

We will read a wide range of (advanced) research papers describing systems written in Rust. If you have not taken a first paper reading course in systems before (e.g., CS240, CS244, CS244B, or equivalent at another institution), you are not prepared to take CS340R. There is a tremendous amount of institutional knowledge in systems research, and if it is new to you, many of the papers will require a lot of background reading. For example, you should know the answers to some (but probably not all) of these questions:

- What are some of the tradeoffs between thread-based and event-based concurrency?
- What is livelock?
- What is idempotency?
- What are the advantages of a log-structured file system? What is hard?
- What are some differences between a macrokernel and a microkernel?
- What is scatter-gather I/O and when is it useful?
- What is optimistic concurrency control?
- What are some tradeoffs between garbage collection and reference counting?
- What is a replicated state machine? What are the major algorithms used for them?
- What is information flow and how is it used in secure systems?

The course is split into two parts. In the first 3 weeks, students begin to learn the basics of Rust. At the end of these three weeks, students form into project groups, picking a recent system implemented in a language other than Rust. Over the next seven weeks of the quarter, the groups re-implement the system in Rust while we read research papers on systems built in Rust. Students write a short summary (300 words) for papers in weeks 4-10, which we use to guide class discussion.

Syllabus (in progress)

For ACM papers that aren't public access, you can log into the ACM Digital Library using Institutional Login and choosing Stanford. You should be able to log in with your SUNet ID to get access.

Date	Topic	Due
4/2	Course Goals and Rust	
4/4	**System Programming in Rust: Beyond Safety**	**Rust Book**, Chapters 1-5
4/9	**Kernel extension verification is untenable**	**Rust Book**, Chapters 6-10
4/11	**Ownership is Theft: Experiences Building an Embedded OS in Rust**	**Rust Book**, Chapters 11-15
4/16	**The Case for Writing a Kernel in Rust**	**Rust Book**, Chapters 15-19
4/18	**Evolving OS kernels towards secure kernel-driver interfaces**	**Rust Book**, Chapter 20

4/23	**How do programmers use unsafe rust?**	Project Proposal
4/25	**Understanding memory and thread safety practices and issues in real-world Rust programs**	
4/30	**Is rust used safely by software developers?**	
5/2	**Multiprogramming a 64kB Computer Safely and Efficiently**	
5/7	**Tighten rust's belt: shrinking embedded Rust binaries**	
5/9	**RedLeaf: Isolation and Communication in a Safe Operating System**	Project Review
5/14	**Theseus: an Experiment in Operating System Structure and State Management**	
5/16	**Engineering the servo web browser engine using Rust**	
5/21	**Splinter: bare-metal extensions for multi-tenant low-latency storage**	
5/23	**Modular information flow through ownership**	
5/28	**Cornflakes: Zero-Copy Serialization for Microsecond-Scale Networking**	
5/30	**NetBricks: taking the V out of NFV**	
6/4	Wrap-up and Reflections	
6/11	Project Reports Due	