Introduction to the Course

Performant Software Systems with Rust — Lecture 1

Baochun Li, Professor
Department of Electrical and Computer Engineering
University of Toronto

About me

```
1 function hello(name) {
2     return "hello " + name
3 }
```

Baochun Li

- B.Eng., CS, Tsinghua University (1995)
- M.Sc. & Ph.D., CS, Univ. of Illinois, Urbana-Champaign (1997 & 2000)
- Computer Engineering Group, ECE, Univ. of Toronto (2000 - present)
- IEEE Fellow (2015)
- Fellow, Canadian Academy of Engineering (2023)
- Fellow, Engineering Institute of Canada (2024)

My research

- Research interests
  - Cloud computing, distributed machine learning
  - Computer networking
  - Security & privacy
- Vision
  - Bridge the gap between theory and practice
- Small research team
  - alumni $\longrightarrow$ academia: 22
- Website: iqua.ece.toronto.edu

To develop a practical and in-depth understanding on best practices of building modern software systems with Rust

Why Rust?

- Reliable
  - Memory safe
  - Thread safe — no race conditions
  - Static types
  - Runtime errors $\longrightarrow$ compile-time errors

Why Rust?

- Highly performant
  - Full control over low-level details
  - No garbage collectors
  - Minimized memory footprint
  - Threads and stackless coroutines

Why Rust?

- Excellent ecosystem and tooling
  - Surprisingly useful compile-time error messages
  - The cargo package manager & crates.io
  - Documentation & docs.rs

What do we cover (tentative)

- Basic Programming Concepts and Ownership
- Structs and Enums
- Options and Error Handling
- Strings, Vectors, Hash maps, and Slices
- Generic types, Traits, and Lifetimes
- Functional Rust with closures and iterators
- Smart Pointers to Data on the Heap
- Fearless Concurrency
- Best Practices, Design Patterns, and Idiomatic Rust

Rust is a language for the next 40 years

- High **and** low level
- Code for web assembly, containers, or bare metal chips
- Fast, reliable, productive — pick three
- Fearless concurrency

As a recent example, OpenAI's Codex CLI was previously implemented in TypeScript, and re-implemented in Rust two months ago. After its release on August 8, 2025, its performance is much better.

A note about AI

- AI has been widely known to be effective as a coding assistant
  - At Anthropic, 80% of the code is written by AI
  - At Ampcode.com, 95% of the code is written by AI
- But what about the cream — **5-20%** — at the top?
  - Humans are still essential, at least today
  - But without knowing the language at the finest detail, you cannot write that last 5%!

In this course, we want to talk about making not just any software, but **perfect**, **highly performant** software, that AI cannot make (yet).

Textbook

The Rust Programming Language

Course Website

https://www.eecg.toronto.edu/~bli/ece1724

- At a glance
- Syllabus
- Lecture slides and video recordings

Announcements and grades will be posted to q.utoronto.ca

Office Hours

Mondays, 1-2 p.m., BA 4118

or by appointment

Assignments

1. Reversi board game
2. Search utility
3. Web client
4. Web server

- Do not use AI except code completion

Course Project

- Objective: Building a performant software framework
- Done in teams of 2-3 students
- Third-party frameworks from crates.io can be used
- Some inspiring project ideas will be announced for teams to choose from
- Teams can, of course, start with their own project ideas
- Do not use AI to write new code; you can use AI to locate bugs, as long as you include prompts (e.g. `CLAUDE.md`) and conversation histories in your GitHub repo

Academic Integrity

- Do not copy from other students or provide your code to others
- Trivial to detect