Welcome to my resume! It consists of two parts. The first part is the free-form narrative of what I do work-wise. This is something *I* would be excited to read from a person I am going to work with. The second part is a more traditional bullet-list of companies, positions, and projects. The resume is available as .html and .pdf.

Narrative

I used to do math. Although I no longer do mathematics daily, it is the basis I use to think about programming. I enjoy solving an occasional puzzle. See "Generate All the Things" and "Notes on Paxos" articles as examples of math I like.

I am a programmer. I like writing code just for the sake of it. I like deleting code even more. I like short, simple, robust and beautiful code, which not only gets the job done, but does it in an obviously correct way. See, eg, `ungrammar` for an example of relatively short and self-contained piece of programming.

I am a pragmatist. The above two points sound outright scary, but don't worry :) While I do enjoy encoding lambda calculus in types, that's not what I spend most of my time on. I see most code as something to be replaced and re-written later, and optimize for making changes over time, not for perfection right now. This section from rust-analyzer style guide is a good example of this.

I loathe accidental complexity. I think I spend most of my time trying to make things simpler, trying to remove parts, trying to make foundational APIs more crisp. I have a visceral reaction to the gaps between how the thing *should* be, and how they are. `cargo xtask` pattern shows to what lengths I am willing to go just to get rid of the mess the unix shell is.

I build systems. Software engineering is programming integrated over time, and it's that time dimension that really matters. The shape of the software today is determined by accidental, runaway, viral successes of yesterday. There's a reason why VT100 interface is still programmed against today, and it is not its technical adequacy. This is not my article, but I like it so much that I'll advertise it even in my resume. System's thinking is why I am fascinated with Rust and not, eg, with Kotlin. Since Java with its reasonably fast managed runtime, Rust is the first PL revolution which meaningfully changes how we write software, and not just repacks known-good idioms with a better syntax (which is also important!, just not as exciting!).

I build open source communities. My biggest successes so far I think are `IntelliJ Rust` and `rust-analyzer`. I didn't write the hardest, smartest bits of those. But I tried very hard to make sure that others can do that, by removing accidental complexity, by making contribution enjoyable, by trying to program the architecture which would be robust to time and systems effects.

More generally, I help build moderately large projects, which are combinations of all of the above: people, systematic forces, beautiful mathematical abstractions at the core, and hundreds of thousands of lines of code as a physical manifestation. See "One Hundred Thousand Lines of Rust" series for a bunch of concrete, pragmatic lessons I've learned so far.

I love teaching! See, for example, my Russian Rust Course, TigerBeetle series, consensus lecture, or the article about Pratt parsers.

Oh, and I love writing :-)

Contacts

Name

Alex Kladov

GitHub

https://github.com/matklad

Email

aleksey.kladov@gmail.com

Core Competencies

- Building complex systems software from nothing
- Growing open source communities
- Implementing compilers, IDEs, build tools, databases
- Zig and Rust mastery

Education

- 2014-2016, Masters of Software Engineering (unfinished)
  St. Petersburg State University of RAS, Russia

- 2009-2014, Specialist of Software Engineering
  St. Petersburg State University, Russia

Professional Experience

TigerBeetle

`From Dec 2022`
https://tigerbeetle.com
https://github.com/tigerbeetle/tigerbeetle

At TigerBeetle I help to build correct and fast distributed database for accounting in Zig. I work throughout the stack, but my primary focus areas are consensus, testing, overall build&development process, and knowledge sharing.

Rust Programming Language

`Sep 2015 to Dec 2022`
https://www.rust-lang.org/governance/teams/dev-tools

I was a member of the dev-tools team of the Rust programming language. I am the original author of both IntelliJ Rust and rust-analyzer — the two tools which today power IDE support for Rust in virtually every editor. My work included both the technical task of writing an advanced, incremental, resilient compilers *and* organizing a vibrant community of contributors and maintainers to ensure that my direct involvement is not a requirement.

I made many smaller contributions across the Rust ecosystem. I was a co-maintainer of Cargo in 2016-2018, maintain prominent libraries, and document emerging ecosystem patterns.

NEAR

`Feb 2021 to Dec 2022`
https://near.org

At NEAR, I was a TLM/TL for the contract runtime team which is responsible for secure, reliable, and fast execution of WebAssembly smart contracts.

Ferrous Systems

`Sep 2018 to Feb 2021`
https://ferrous-systems.com

With Ferrous Systems, we brought rust-analyzer project from an MVP to a de-facto standard for the ecosystem. I also helped with teaching people to use Rust efficiently.

Computer Science Center

`Sep 2014 to Sep 2019`
https://compscicenter.ru/teachers/934/

At CSC I taught two major courses:

Programming In Rust, `Winter-Spring 2019`**, video**
: A semester long introduction course, focused on contrasting unique Rust features with more mainstream languages like C++ or Java.

Programming In Python, `Autumn-Winter 2018`**, video**
: A semester long advanced course focusing on the language inner workings and programming idioms.

I have also worked as a teaching assistant for “Algorithms and Data structures” and “Python” courses.

JetBrains

`Sep 2015 to Jan 2018`
https://intellij-rust.github.io

At JetBrains, I have led the development of IntelliJ-Rust plugin for the Rust programming language. The plugin is a Rust “compiler” written in Kotlin, with full-blown parser, name resolution and type inference algorithms, and integrations with build tools and debuggers. Besides solving the technical problems, I’ve created an open source community around the plugin by mentoring issues, writing developer documentation and supporting contributors.

The plugin later became the basis for the stand-alone Rust Rover IDE.

Stepik.org

2012 to 2014

http://stepik.org/

Stepik is a e-learning platform, written in Python, focused on rich variety of practical exercises and ease of creating content. I was on the backend team of three from the start of the project. Among other things, I've worked on exercises subsystem and student's code sandboxing, progress tracking and designed and implemented JSON API interface for the single-page frontend.