Marco Perronet

Software Engineer

perronet.github.io

github.com/perronet

linkedin.com/in/marco-perronet

perronet.marco@gmail.com

About me

I have four years of experience in computer science, comprising both of software engineering and research.

I moved to Germany after my Bachelor's degree to pursue a Master+PhD program at **Max Planck Institute**, where I worked on research projects in the field of real-time operating systems.

Most recently, I moved to London for an internship at **Meta**, and I am now working as a software engineer at **Bloomberg**.

Languages

- **Italian** *(mother tongue)*
- **English** *(fluent)*
- **French** *(beginner)*
- **German** *(beginner)*

Extras

I grew up in the alps and I love anything that is related to mountains. I can climb, both outdoors with ropes and bouldering indoors, and I love skiing. In my free time I like to play the acoustic guitar (maybe one day I'll get an electric one) and I'm an avid Super Smash Bros player.

Skills

My main programming languages are **C++**, **Rust**, **Python**, and **OCaml**. I have experience with designing and implementing distributed systems using middleware such as **Kafka** and **RabbitMQ**. Being interested in low-level programming and operating systems (both user and kernel space), I am experienced with **Unix**, **C** programming, and **Bash** scripting.

Experience

2022 Dec-now — **Backend Software Engineer** — **Bloomberg**, London

My team owns the core system which processes and routes orders on the market. I built several mircoservices from scratch, from the high-level design to the implementation, which was fully in **C++**, with **Python** and **Docker** used for integration testing. These services are event-driven and communicate through pub-sub middleware such as **Kafka**. I learned how to design a highly available and scalable back-end, which is challenging due to the need for high data throughput.

2019-2022 — **PhD Student** — **Max Planck Institute For Software Systems**, Germany

I worked in the field of real-time operating systems, and my project focused on trace-based response-time analysis on **Linux**. I designed and developed *DMXtrace*: a tool written in **Rust** that traces the processes running on the system, extracts a formal model, and uses it to analyze the timing correctness of the system. [Paper] [Code]

2022 Jun-Aug — **Software Engineer Intern** — **Meta**, London

Infer is an open-source static analysis tool developed at Meta. During my internship, I extended Infer to support an analysis based on declarative logic programming with Datalog, which enables the detection of potential null pointer exceptions. I studied the approach and implemented it in **OCaml**. [Website] [Code]

Education

2019-2022 — **Master's degree in CS** — Technische Universität Kaiserslautern, Germany

Thesis: "Dynamic Extraction of Real-Time Models from Arbitrary Workloads on Unmodified Linux Kernels"

2016-2019 — **Bachelor's degree in CS** — Università degli studi di Torino, Italy

Thesis: "Monitoring the Linux scheduler with `trace_sched*` events"

Projects

Linux kernel exploration

I fiddle with the kernel codebase both for my research work and personal interest, and made a contribution by proving the existence of a minor bug in the real-time scheduler. I collaborated with a kernel developer to create a patch to fix it. [Patch]

Simple chess AI

I implemented chess in **C++** and then trained a neural network to play using a dataset of chess positions scored by a popular chess engine. I used common machine learning libraries such as **Tensorflow** and **Scikit-learn**, and **Pandas** to clean and process the dataset. [Code]

Treecodes

Treecodes are an alternative to QR codes which encode information inside the topology of a tree. I designed, implemented, and evaluated different types of efficient encoding/decoding strategies using **C++** for the final implementation and **Python** for prototyping. [Demo on the website] [Code]

NP to SAT transform

I implemented an efficient transpiler in **C** that turns a (Turing machine, input problem) tuple into a formula for SAT solvers. [Code]