FIXME: Add logo above in footer and header *FIXME*: Add guard page

Embecosm Rust-GCC - Fixed time mission

FIXME: Fix the title

Thanks

I'd like to thanks to Open Source Security and Embecosm for funding us to work on this project. Jeremy Bennett, for his management and the experience he brought to our development, and Brad Spengler, for being kind and benevolent.

I would also like to thank the entire Embecosm team for being so nice and caring towards me.

Particularly, I would like to thank Philip Herron for being my mentor, manager and especially friend when working on this project.

I would also like to thank my girlfriend for being so supportive and helpful during my stay in Germany.

Introduction

Context and Complexity of the task

Currently, not many systems programming native language. But Rust is coming. However, only one implementation! Contribute to a second implementation (no details as that is for the Subject part)

In our current programming ecosystem, not many programming languages are usable to target small embedded architectures as well as large multithreaded applications. These languages, where speed of execution is a major focus, are mostly comprised of C and C++.

These two languages can be categorized as "systems-oriented", and "native", meaning that they are able to target even the lowest level of programming and are compiled directly to native instructions for the CPU. Programs compiled using these languages offer very small overhead and are extremely fast, at the cost of increased mental load for the programmer. What these languages are not, however, is "safe". A recent study conducted by Microsoft (*source*) showed that around 70% of bugs found in their software were "memory issues", where memory is not handled properly by the programmer: memory leaks, use-after-frees, double-frees, out-of-bounds acesses... are all common C/C++ programming mistakes that in turn can lead to vulnerabilities, exploitable by attackers. These numbers are not due to Microsoft's lack of talent: The Google Chrome project reported the exact same number two years ago (*source*).

This issue has lead numerous companies to invest in programming language research, with the hopes of creating a safe, fast, systems-oriented native programming language, the most notable being Rust.

Rust is still quite a young programming language, being only around 15 years old, but offers a competitive alternative to C and C++. It focuses on safety as well as speed of execution, achieving speeds similar (or faster in some cases!) to programs written in C or C++. Furthermore, it also targets the embedded market, providing more expressivity than C.

However, a stark difference with C/C++ and brake to Rust's adoption is the lack of specification or standard. Some companies do not consider the language stable or mature enough to earn a place in their technological stack. Furthermore, only one implementation of the language currently exists, the official `rustc` compiler. This compiler is written in Rust and thus faces bootstrapping issues. It also uses `LLVM` as its compiling framework, making it available on a lot of hardware architectures but not all.

In an effort to improve the reach of the language, be it in terms of the amount of people using it or for more niche architectures to use Rust programs, an alternative compiler is being developed. This compiler, `gccrs`, aims to integrate the Rust language among the GNU Compiler Collection project. The GNU Compiler Collection (GCC) contains multiple compilers for multiple languages, such as C (`gcc`), C++ (`g++`), Fortran (`gfortran`) or Ada (`gnat`). It is a pillar of the free software movement, making it more engaged than "regular" open source projects, and has been developed for more than 30 years, enabling it to target a multitude of architectures.

The GCC project is an old, complex codebase written in C++11, of around 19 million lines of code, making any changes to it extremely complex but also extremely interesting.

Objective and extent of the mission

The project started in 2014, and has been worked on full-time since 2019 by Philip Herron, my mentor and manager for this internship. It's been making fast progress and has been evolving

very quickly. I started working on the project one year and a half ago as a side-project, and participated in Google Summer of Code on this project.

FIXME: Reformat this last sentence *FIXME*: Add more

The goal of this internship was, overall, to contribute to the state of the compiler. Specifically, some complex Rust concepts such as macros, privacy restrictions or const generics were not handled yet, and needed to be worked on to achieve a valid Rust implementation as soon as possible.

FIXME: Add more? Yes!

0. Project since 2014, full time since 2019?
1. Been working on this project for a year before that
2. Google Summer of Code student
3. Improve existing codebase, very big C++ project, hard to maintain and work on but worth it.

Subject

Remind the history of your subject

1. Project started in 2014
2. Philip started working on it full time in 2019
3. Steady funding and progress since then

Motivate your choice of internship

1. Extremely interesting project, that is close to my heart
2. Chance of making big changes, making a mark on a project
3. Lots of things to work on, very varied subject

Is in adequation with your major

1. No? Not embedded systems
2. However, it is low-level programming and systems programming. And that is a big part of GISTRE.

Positioning

Tether your subject and the company's field

1. Embecosm works in compilers
2. However, mostly backend
3. Open Source Security provides GCC plugins for security purposes, notably for the Linux kernel.
4. Lot of interest around a second implementation of the Rust language which could benefit from said plugins.

Present the market and its context

1. There's no market. It's not a market. Stop it.
2. People interested in funding the project maybe? ## Course of the internship

Parts of the internship

1. Development
2. Research
3. Promotion (talks, trade fairs)
4. Hiring (interviews)

Detailed Description

1. Same as above, detailed

Gantt Diagram

1. Add pretty diagram! How? Interlace the PDF?

Engineering approach

One or more structured approache(s)

1. The borrow-checker
2. The macro repetitions implementation
    1. Figuring out an algorithm
    2. Issue reporting
    3. Still more to do and maintain!

Concrete explanation

1. Show and explain what you did concretely (vs the company, the project, the team, the client...)
    1. Versus David Edelhson

Illustrated analysis

1. Detail 2 to 3 competences in relationship with your major

Added value of the internship

Internship's interest for the company

Internship's interest for the project

1. Qualitative and quantitative
2. Show your results

School's implication - On the school's side?

FIXME: Rework the title

1. Usage of concepts and methodologies used in class
2. Acquisition of new skills

Synthesis and Conclusion

Introspection

1. Between the beginning and the end of the internship

Evolution of your career plans/personal project

1. Confirmation or evolution of your professional project

Vision of the business world