Taken from: codilme.com

Why is the Rust programming language so popular?

Krzysztof Wróbel, March 27, 2023.

Rust was created to ensure high performance similar to that offered by C and C++, but with emphasis on code safety, the lack of which is the Achilles heel of these two languages. However, **Rust** has more than just memory safety on its side. High performance while processing large amounts of data, support for concurrent programming, and this together with an effective compiler are other reasons why well-known software heavyweights now use this programming language. Firefox, Dropbox, Cloudflare, and many other companies from startups to large corporations use **Rust** in production.

In this article, you can find more about Rust's main characteristics, what it is used for, and why you should consider adopting it for your software projects.

Rust's growing popularity

According to the Stack Overflow Developer Survey 2022 conducted among over 80,000 developers, **Rust is the most beloved programming language.** And it won the title for the seventh year running.

In 2020, Linux kernel developers proposed writing new Linux kernel code in Rust. To be clear, they did not want to rewrite the entire Kernel, which was originally written in C, but to add new code in Rust that would work with the existing infrastructure. No less a figure than Linus Torvalds, the father of the open-source operating system Linux, welcomed the idea and is keen to see the project results. This idea is in continuous development — at this moment, it is still described as experimental, but "good enough that kernel developers can start working on Rust abstractions for subsystems and write drivers and other modules."

Moreover, Google is also planning to use Rust in the Linux kernel after bringing support for the systems programming language Rust to Android. The entire operation is aimed at reducing security flaws. Microsoft, meanwhile, has turned to Rust to reduce memory-related bugs in Windows components.

Facebook has also forged closer ties with Rust, joining the Rust Foundation, an organization created in 2021 to drive the development of Rust and make it "a mainstream language of choice

for systems programming and beyond.” Facebook joins Amazon Web Services, Google, Huawei, Microsoft, and Mozilla in using Rust in some capacity.

All these are clear signs that **Rust has been gaining traction in the past few years**. But is it just another hype technology that promises a lot, but will soon vanish?

What is so special about Rust language?

Rust is a statically-typed programming language designed for performance and safety, especially safe concurrency and memory management. Its syntax is similar to that of C++. It is an open-source project developed originally at Mozilla Research. In 2021, the Rust Foundation has taken the torch and is driving the development of the language.

Rust solves problems that C/C++ developers have been struggling with for a long time: memory errors and concurrent programming. This is seen as its main benefit. However, these are not the only examples of how Rust is different from C++.

Of course, one can argue that modern C++ is putting greater emphasis on memory safety (e.g. by implementing smart pointers), but many problems remain unresolved. One of them is ‘use after free errors,’ which happen when a program continues to use a pointer after it has been freed, e.g. calling the lambda function after freeing its reference captured objects.

In Rust, on the other hand, you have the borrow checker—the part of the compiler that ensures that references do not outlive the data to which they refer. This feature helps you eliminate memory violation bugs. Such problems are detected at the compile time, and garbage collection is unnecessary.

Additionally, in Rust, each reference has a lifetime, where you can set the scope for which that reference is valid. This feature solves the problem with references that are no longer valid, and also distinguishes Rust from C and C++.

The importance of correct memory management becomes immediately apparent when you realize that over the past 12 years around 70% of all security bugs in Microsoft products have been memory safety issues. The same number is also reported for Google Chrome.

In Rust there are two modes of writing code: Safe Rust and Unsafe Rust. Safe Rust imposes additional restrictions on the programmer (e.g. object ownership management), thereby ensuring that the code works properly. Unsafe Rust gives the programmer more autonomy (e.g. it can operate on raw C-like pointers), but the code may break.

The unsafe Rust mode unlocks more options, but you need to take extra care to ensure that your code is truly safe. To do so, you can wrap it in higher-level abstractions which guarantee that all uses of the abstraction are safe. As with other programming languages, using unsafe code should be approached with care in order to avoid undefined behavior and minimize the risk of segfaults and vulnerabilities resulting from memory unsafety.

Rust's dual-mode model is one of its biggest advantages. In C++, on the other hand, you never know you've written unsafe code until somewhere down the line your software crashes or a security breach rears up.

Concurrent programming made easier

Rust also makes it easier to write concurrent programs by preventing data races at compile time. A data race occurs when at least two different instructions from different threads are trying to access the same memory location simultaneously, while at least one of them is trying to write something and there is no synchronization that could set any particular order among the various accesses. Access to the memory without synchronization is undefined.

In Rust, data races are detected. If a given object access does not support many threads (i.e. is not marked with an appropriate trait), it needs to be synchronized by a mutex that will lock access to this particular object for other threads. To ensure that operations performed on an object will not break it, only one thread has access to it.

From the perspective of other threads, operations on this object are atomic, which means that an observed state of the object is always correct and you cannot observe any intermediate state resulting from an operation performed on this object by another thread. **Rust language can check if we are performing any incorrect operations on such objects and inform us at compile time**.

Other languages employ synchronization methods, but they are not related to the objects they refer to. It is the developer who needs to take care to lock the object before using it. In C, for example, a compiler allows a developer to write buggy code. As a result, bugs are detected while a program is already in production - or even worse, while someone is trying to hack it.

In Rust lang, many concurrent programming-related problems (though not all) are solved, as they are found immediately at compile time.

Some obstacles to overcome while programming in Rust

Of course, it is not all roses. **Rust is a relatively new technology**, so some desired libraries may not yet be available. Still, the Rust package library crates.io has been growing fast since 2016, and the vibrant community of Rust developers is a good omen for the further development.

Also, for developers not used to working with a language where errors in the code are detected at compile time, it may be annoying to get many error messages. As a result, developing code is not as fast as in more popular languages, like Python. However, **Rust's developers are doing their best to make these error messages as informative and actionable as possible**.

Even if it may be somewhat irritating to get so many error messages when coding, stay focused on the bigger picture. Memory safety enforced at compile time prevents bugs and security vulnerabilities from happening, when your software is already in production. Correcting them in this phase will definitely cost you in nerves and money alike.

Last but not least, writing Rust code requires more effort, as the entry threshold is pretty high. You need to spend some time mastering the language. A good knowledge of C++ or any other object-oriented programming language is also required.

But if you overcome all these obstacles, **the benefits of using Rust will be the best reward for your efforts.**

You can also read a ccomparison between Rust vs Go.

What is Rust used for

Rust is already a mature technology that is used in production. As a systems programming language, it allows you to maintain control over low-level details. You can choose whether to store data on the stack (used for static memory allocation) or on the heap (used for dynamic memory allocation). Here it is important to mention RAII (Resource Acquisition Is Initialization), a code idiom that is mainly associated with C++, but is also present in Rust: every time an object goes out of scope, its destructor is called and its owned resources are freed. You do not have to do this manually and you are protected from resource leakage bugs.

This ultimately allows memory to be used more efficiently. Tilde used Rust in their Skylight product, particularly to rewrite certain Java HTTP endpoints. This enabled them to reduce their memory usage from 5GiB to 50MiB.

Since Rust does not have a garbage collector continuously running, its projects can be used as libraries by other programming languages via foreign-function interfaces. This is an ideal scenario for the existing projects where it is critical to ensure high performance while maintaining memory safety. In such projects, **Rust code can replace some parts of software**, where performance plays a crucial role, without the need to rewrite the entire product.

Rust is a low-level programming language with direct access to hardware and memory, which makes it a great solution for embedded and bare-metal development. You can use Rust to write operating systems or microcontroller applications. In fact, there are a number of operating systems written in Rust like: Redox, intermezzOS, QuiltOS, Rux, and Tock. Mozilla, where the language was originally designed, uses it in its browser engines.

High performance and safety are the features that made Rust so appealing to scientists that started using it to perform heavy data analysis. Rust is blazingly fast, making it an ideal choice for computational biology and machine learning, where you need to process large amounts of data very quickly.

At CodiLime, we are also experimenting to determine if Rust can replace C in performance in network scenarios calling for high data throughput. In our PoC, we built an application in Rust using DPDK libraries (written in C) to ensure performance while also maintaining memory safety.

Types of applications that can be written with Rust

Rust can be used for various types of applications. However, in some areas this programming language is especially popular. Check out for which project types and industry areas Rust might be the go-to option.

System-level programming

- **Operating systems (OS)**
  The Rust programming language has been used in the development of operating systems due to its low-level control, efficiency, and memory safety. It ensures safe memory access and prevents undefined behavior, making it easier to build secure operating systems.
- **Device drivers**
  Rust is well-suited for device driver development, particularly in embedded systems, due to its low-level control, performance, and memory safety. Rust's type system and borrow checker make it easier to write safe and efficient device drivers that work seamlessly with the underlying hardware.
- **Embedded systems**
  This language is also increasingly being used for embedded systems thanks to features like great low-level control and performance. Rust's borrow checker ensures that memory is safely managed and eliminates the risk of null pointer dereferencing, making it a safe and efficient choice for embedded systems development.

Web development

- **Backend services**
  Here, the most important features are Rust’s performance, scalability, and safety – its async/await support provides easy concurrency for building high-performance backend services.
- **Web applications**
  Rust is often chosen for web applications as it offers efficient memory management. It allows developers to write high-performance and memory-safe applications.

Data science

- **High-performance computing**
  The Rust support for parallelism and its async/await pattern facilitates building efficient and safe parallel programs.

- **Machine learning (ML)**
  Rust is not the most common choice for ML-related projects for now, but it provides some libraries such as rusty-machine and tch-rs that provide useful tools for building machine learning models.

Why use Rust

To sum up, the **main reasons for embracing Rust in your next software project include**:

- High performance while ensuring memory safety.
- Support for concurrent programming.
- The growing number of Rust packages at crates.io repository.
- A vibrant community driving the development of the language.
- Backwards compatibility and stability ensured (Rust has been designed for the next 40 years).

Conclusion

Rust has great performance, tooling, and an active community on its side that is continuously working on language improvement. Moreover, if you need a solution with a greater focus on safety than C and C++ and you don't want to compromise on speed, Rust is a good choice for you. If you worry that Rust may not be "mature" enough for your software development project, be reassured that this is no longer the case. Dozens of companies, like Figma, 1Password or Amazon, have already used Rust in their development work.