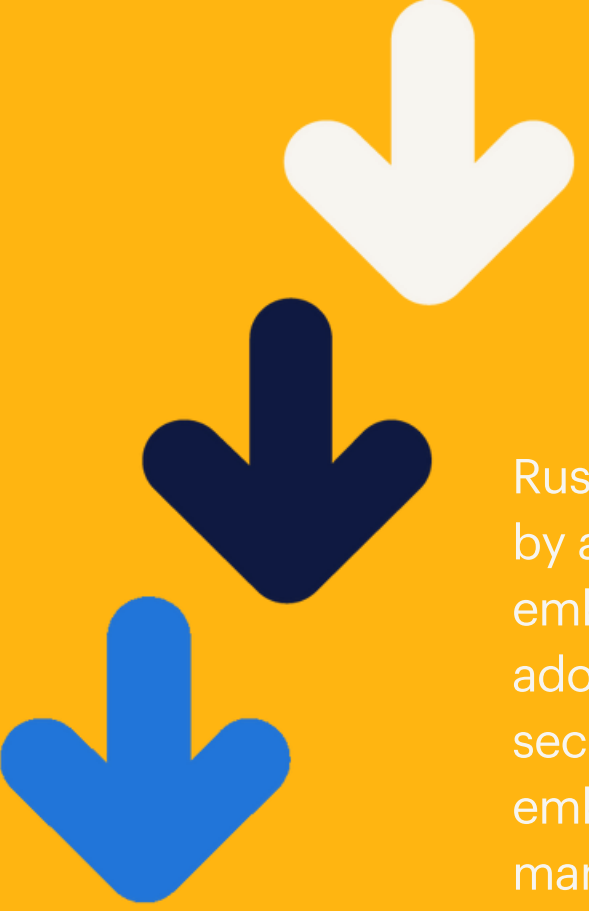
programming in rust

get a head start
with these tips.



Rust as a programming language is loved by a large audience, especially by embedded software engineers. They adore Rust for its performance and its security. Among them is Devon, embedded software engineer and project manager embedded software solutions at AUSY.

Devon: “A few years ago, I followed a presentation on Rust, but I didn’t do very much with it afterwards. Then I read a few Reddit threads on it, and I started studying this programming language.”

“This is not an easy language to learn, but I’m convinced that Rust offers a great added value in embedded software development. There are many tips and tricks that have helped me to get started, but there are also things that I would have preferred to learn earlier on in the process. I would love to share my best practices with you!”

the borrow checker is your friend.

One of the things that make Rust unique, is the Borrow Checker. Everyone that's starting to learn Rust knows the following problem: the Borrow Checker shows an error message during compilation.

Your first thought would probably be to quickly make some adjustments to fix the error message, but it's better to understand why this error message actually occurred. Was your software design developed properly? Unlike other languages, Rust forces you to think about the structure of your software and how the different parts should interact with one another beforehand.

strong types.

Strong types are recommended in Rust. You can use a const such as ERR_INVALID_ARG = -1 in C, but you have to use an enumerator with possible outcomes such as Enum Err { invalid, valid } in Rust.

result and option.

When calling a function in C doesn't give value, a nullpointer is often used. If this happens unexpectedly, your program could be completely ruined. Rust tries to avoid these situations with Result and Option.

With Result, you'll either get an OK or an Err, with Option this will be Some or None. This helps to make it clear in the code which data will be returned from a function and which data won't be.

the ? operator.

A great shortcut in propagating errors in your code logic is the use of the ? operator. When you place a question mark after a Result, this operator will make sure that the program will continue with an Ok result. With an Err result, the function is terminated and the error is returned as a result to the code that called the function. You don’t need an explicit return statement.

use of unsafe.

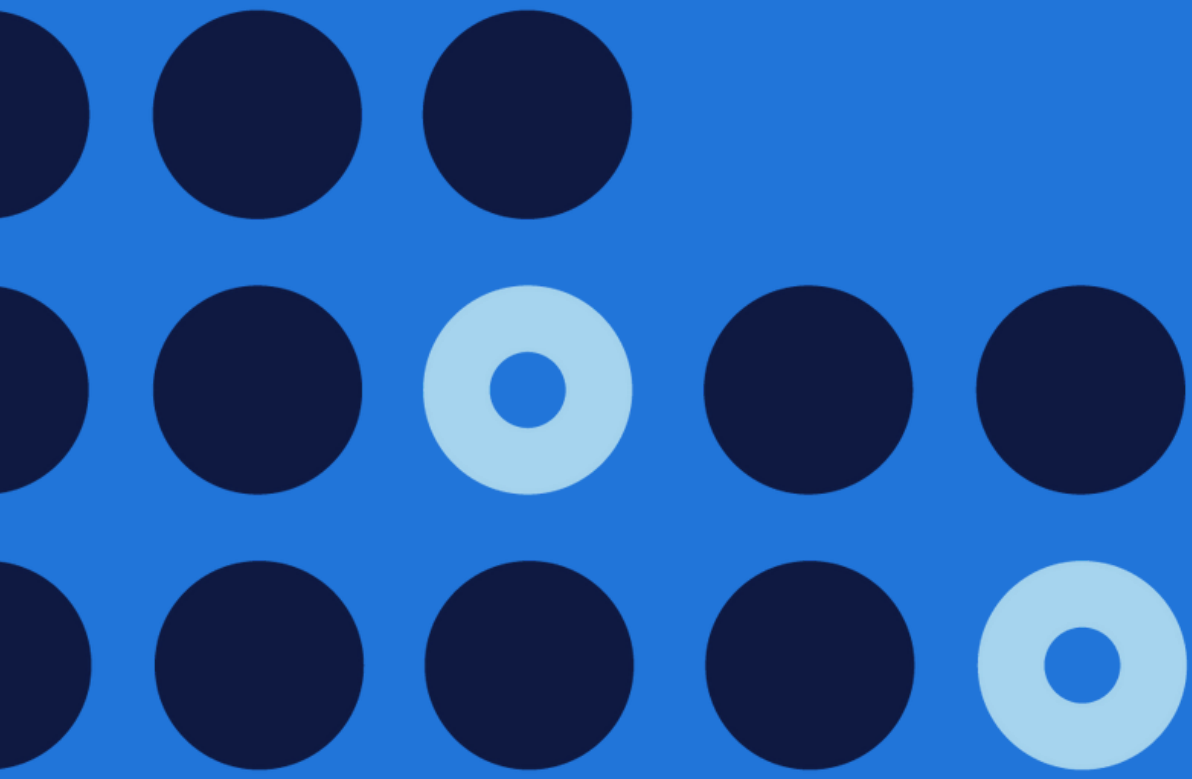
It is possible to indicate in the Rust compiler that you’re using unsafe code in your program by putting code in an unsafe block. The compiler won’t check this block and the memory management will be entirely your responsibility. You should try to use the unsafe block as little as possible - unless you don’t have another option, for example when calling C code from Rust.

tooling.

An extensive tooling landscape was created to help write code in Rust. Clippy and rustfmt help you to make sure your syntax is correct and check for frequently made mistakes. Furthermore, the Rust toolchain also includes support to carry out unit tests and documentation tests. No more example code that doesn't compile!

features.

If you want to develop different versions of your software application (because of licences or if you work with a paid/non-paid version), you can use features in Rust. This helps you to enable or disable parts of your software. The Rust compiler makes sure that the disabled parts will not feature in the final application.



compilation time.

One of the disadvantages of Rust, is that the compilation of the software is a bit slower than C or Go once your project gets more complex and includes multiple crates. One option is to divide your project into different parts and compile them separately.

This forces the developer to think about the architecture of the application. It also gives you the possibility to enable and disable components using features.

If you have multiple Rust projects with shared dependencies, you can use sccache. This will reuse the compiled dependencies instead of compiling them over and over again. You'll save disk space and compilation time!

don't reinvent the wheel.

Even though Rust is a relatively young language, there are many popular and useful crates that will help you with developing parts of your software. Don't create your own logging or serialisation library, but check out the crates.io website and see whether someone has already made a library containing the functionality you're looking for.

If you're using Rust in your embedded software development, make sure to work with the Embedded-HAL crate.

any questions?

Do you have any questions on writing code in Rust? Would you like to share your experience with me? Or are you an experienced Rust developer looking for a new challenge? Let me know, and we'll meet up for some coffee!

- devon kerkhove
- devon.kerkhove@ausy.be
- +32 473 21 64 54

ausy by randstad.

our ambition.

Put simply: we want to inspire you with our passion for technology. We want to bring out the best in our clients as well as our talents. We use the power of technology to reinvent companies. And we offer our experts the opportunity to work on the technological solutions of the future.

www.ausy.be