Why Rust?

What makes Rust important?

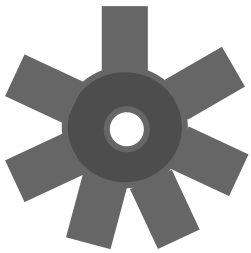
We'll cover the following

Why Is Rust Different?

There are many reasons why Rust is different from other programming languages.

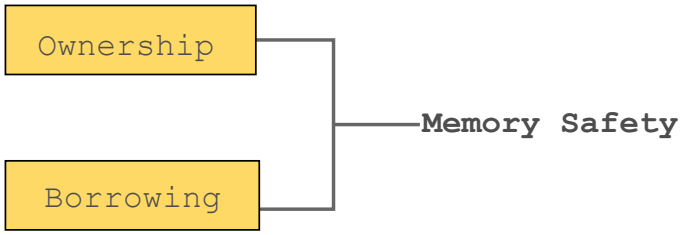
Speed

Rust pays no penalty for the abstraction and only pays for the features that are actually used. This improves the code quality and readability without affecting the run time cost.



Safety

Rust does not have a garbage collector. It ensures memory safety using ownership and its borrowing concept (discussed in the last chapter). The most significant difference between C, C++, and Rust is writing safe code. In Rust,

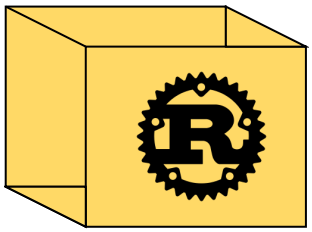


the objects are managed by the

programming language from the beginning to the end. The proper amount of memory required is allocated and automatically deallocated by the system when it is no longer in use.

Cargo Manager

System languages like C and C++ never had a standard package manager. Rust provides a cargo manager that has all the rust libraries and frameworks to not only help developers make fantastic applications, but also distribute code to end-users.



Cargo Manager

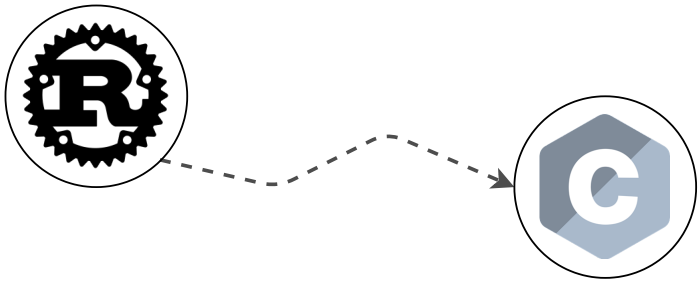
Error Messages

The error messages in Rust are displayed using color. Formatting and misspellings in the program are commonly suggested. It not only displays the line which has the error but, also the type of error.

```
warning: unused variable: `y`
 --> main.rs:4:9
  |
4 |     let y = &x;
  |         ^ help: consider using `_y` instead
  |
  = note: #[warn(unused_variables)] on by default

warning: value assigned to `x` is never read
 --> main.rs:6:5
  |
6 |     x += 1;
  |     ^
  |
  = note: #[warn(unused_assignments)] on by default

error[E0506]: cannot assign to `x` because it is borrowed
 --> main.rs:6:5
  |
4 |     let y = &x;
  |             - borrow of `x` occurs here
5 |
6 |     x += 1;
  |     ^^^^^^ assignment to borrowed `x` occurs here

error: aborting due to previous error
```

Efficient C Binding

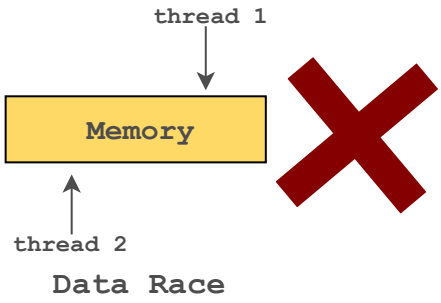
The Rust language can interoperate with the C language. It provides a foreign function interface to communicate with C API's. Due to its ownership rules, it can guarantee



memory safety.

Threads without Data Races

Data race is a condition where two or more threads can access the same memory location. Rust uses the concept of ownership to avoid data races.



Data Race

Let's dive into the basics of Rust in the next chapter.

Or, you can go back to the Learn Rust from Scratch course homepage.