Jackson Goldberg

Programming Languages

Fall 2022

A Poor Man's Introduction to Rust

Rust was created to be a c and c++ alternative to provide the same high performance while keeping memory safety which the two other languages lack. In 2021 Rust won the most beloved programming language for the 6th year in a row in the stack overflow developer survey. Rust has also piqued the interest of Linux kernel developers, saying that new features and updates to the linux kernel will incorporate rust. Over the last two years we have seen a lot of interest in rust from FANG corporations specifically as the language gets more mature. More and more companies are wanting to implement rust into their production and it begs the question, what makes Rust so special?

What puts Rust apart?:

Both Microsoft and Google report that 70% of all security bugs in products have been memory safety issues. Rust was created in order to tackle these issues. One common issue is when a program calls a pointer's lambda function after freeing its reference captured objects. This is essentially a garbage collection issue. The way that Rust tackles this is something called the borrow checker. The borrow checker ensures that references do not outlive the data to which they refer to. Another unique feature to Rust is that each reference has a life span that you can set, this helps solve the same problem. To top all of that off Rust has a two mode system where you can program in Safe-Rust which provides more restrictions on the programmer but ensures safe

memory usage, and Unsafe-Rust which gives the programmer more autonomy over memory but less security.

Another great feature which makes Rust really unique is the detection of Data Races on a particular section of memory/object. When Rust compiles it is able to detect whether or not a specific part of memory will be an issue with read/write given multiple threads attempting to access it. This is something that c and c++ do not have and it is important because it removes a lot of human error caused by forgetting to put safety locking on an object.

Uses and Limits of Rust:

Since the Rust compiler detects a lot of memory errors at compile time the programmer ends up having to sort through a lot of errors before they get running code. The advantage to this is when it does compile you can be pretty confident that you won't be having any loose pointers or references. The downside to this is that coding is not a very fast task in this language, especially compared to a language like python. The use cases for this language are very interesting however. Since you don't need to worry about memory issues and since Rust does not have a garbage collector, you are able to port rust code through an interpreter into existing production projects without needing to rewrite the entire project. This feature allows rust to be very adaptable and can be used in specific situations, like with writing new features for linux.

Recap:

Rust is a mature language used by many leading companies in the tech space. Its usefulness comes primarily from its memory security and dynamic use cases while still maintaining high performance. Its drawbacks come in the form of it being difficult to

learn and it being more difficult to get to a point where there are no errors. Rust is mainly seen as an alternative to c and c++.