Rust 从入门到精通01-简介

程序员可乐、

于 2022-07-05 15:25:48 发布

1504

收藏 2

分类专栏： Rust 从入门到精通 文章标签： rust 开发语言 后端

版权

Rust 从入门到精通

专栏收录该内容

12 篇文章3 订阅

订阅专栏

1、rust 从哪里来？

Rust语言在2006年作为 Mozilla 员工 Graydon Hoare 的私人项目出现，而 Mozilla 于 2009 年开始赞助这个项目。第一个有版本号的 Rust 编译器于2012 年 1 月发布。Rust 1.0 是第一个稳定版本，于 2015年5月15日发布。

Graydon Hoare 是一个职业编程语言工程师，其日常工作就是给其他语言开发编译器和工具集，但是不会参与语言本身的设计，由于这种工作性质，他接触过很多编程语言，了解各种语言的优缺点。

比如 c 和 c++，性能比较好，但是类型系统和内存都不太安全；一些拥有 GC 的语言，比如Java、golang、kotlin等，内存是安全的，但是性能却比较低。于是 Graydon Hoare 萌生了自己开发一门语言的想法，这门语言就是 rust，他对 rust 的期望如下：

①、必须安全，不易崩溃；

②、不需要引入 GC，注重性能；

③、应该拥有广泛的特性，让程序员写出易于维护、调试，且更安全更高效的代码。

2、rust logo

Rust 的 logo 承载了创造者对该语言的期望：

①、Rust 这个单词是由 Trust 和 Robust 组合而成，暗示了信任（Trust）和鲁棒性（或健壮性、Robust）。

②、Rust Logo 的形状与一种叫做锈菌的真菌相似，这种真菌生命力非常顽强，在其生命周期内可产生多达 5 种孢子类型，这 5 种生命形态还可以相互转化。也就是鲁棒性（健壮性）非常强。其 Logo 上面的 5 个小圆孔，与锈菌的 5 种生命形态相对应，也在暗示 Rust 语言超强的健壮性。

3、rust 是什么？

rust 是新时代的 C 语言。

①、Rust 是一门通用型语言。

②、Rust 语言的内存方案针对的是 C 语言的不足

一、禁止对空指针和悬垂指针进行解引用

二、读取未初始化的内存

三、缓冲区溢出

四、非法释放已经释放或未分配的指针

注意：内存泄露在 rust 中依然存在，也就是说内存泄露不在rust承诺的内存安全范围内。

③、安全且无缝沟通 C 语言。

④、Rust 是具有混合范式的 ”面向过程“式的编程语言。

⑤、和 C 语言类似，担负了时代的使命。

4、rust 语言版本说明

rust 语言版本从三个不同的维度来讲，包括三个概念。

4.1 语义化版本

其格式为：主版本号.次版本号.修订号，依次用 句点 隔开。

简单说下语义版本号递增规则：

①、主版本号：当做了不兼容的 API 修改。

②、次版本号：当做了向下兼容的功能性新增。

③、修订号：当做了向下兼容的问题修正。

4.2 发行版本

https://github.com/rust-lang/rust/branches/all

①、master 分支：对于一个发行版本

②、beta 分支：对应 Beta 版本，也叫测试版本。

③、stable 分支：对应 Stable 发行版，也叫稳定版。

通常稳定版和测试版每 6 周 发布一次，一般来说，会先发布测试版，经过一段时间测试，在进行发行稳定版。

4.3 Edition 版次

可以看到每个通过 cargo工具（Cargo 是 Rust 的构建系统和包管理器，后面会详细介绍） 创建的 rust项目的 Cargo.toml文件。

版次的意义和语义化是正交的概念，方便rust自身的进化。目前rust有两个版次，分别是 rust2015 和 rust2018。

因为rust每6周发布一次新的语义化版本，版本更新迭代速度快，但是版次通常每 3 年发布一次，之所以存在版次和语义化两种概念，我们可以看这样一个例子：

例如添加可能与代码中的标识符冲突的新关键字，比如 ”async"和“await”，刚开始rust是没有异步的概念，后面我们要添加这两个关键词，但是旧代码可能会把这两个关键词当做变量名来使用，这时候如果只有语义化的概念，我们升级rust版本号，就会造成旧代码编译报错。但是增加了版次的概念就不同了，我们对于之前的代码编译方式选择之前的版次，新的代码编译方式选择新的版次，那么问题就解决了。

因此，如果你正在使用 Rust 2015，并且您的某个依赖项使用 Rust 2018，那么一切正常

开发者可以在 Rust 包管理配置文件 Cargo.toml 来配置 edition = “2018”，表示依赖 2018 版次。

编译的时候可以通过：

Rustc --edition 2015|2018|2021

5、rust 编译过程

rust的编译器是 rustc，它是一种多道编译器，其编译过程要经过多道工序，其编译过程如下：

①、Rust 是跨平台语言，一次编译，到处执行。

②、Rust 编译器是一个 LLVM 编译前端，它将代码编译成 LLVM IR ，然后经过 LLVM 编译为相应的平台代码。

③、Rust 源码经过分词和解析，生成 AST （抽象语法树）。然后把 AST 进一步简化为 HIR(High-level IR)，目的是让编译器更方便的做类型检查。HIR 会被进一步编译为 MIR(Middle IR),这是一种中间表示，它在 Rust 1.12 版本中被引入，主要用于以下目的：

一、缩短编译时间：MIR 可以帮助实现增量编译，当你修改完代码重新编译的时候，编译器只计算更改过后的部分，从而缩短了编译时间。

二、缩短执行时间：MIR 可以在 LLVM 编译之前实现更细粒度的优化，因为单纯依赖 LLVM 的优化粒度太粗，而且 Rust 无法控制，引入了 MIR 就增加了更多的优化空间。

三、更精确的类型检查：MIR 将帮助实现更灵活的借用检查，从而可以提升Rust 的使用体验。

最终，MIR 会被翻译成 LLVM IR ，然后被 LLVM 的处理编译为能在各个平台上运行的机器代码。

————————————————

版权声明：本文为CSDN博主「程序员可乐、」的原创文章，遵循CC 4.0 BY-SA版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/ysvae/article/details/125620688