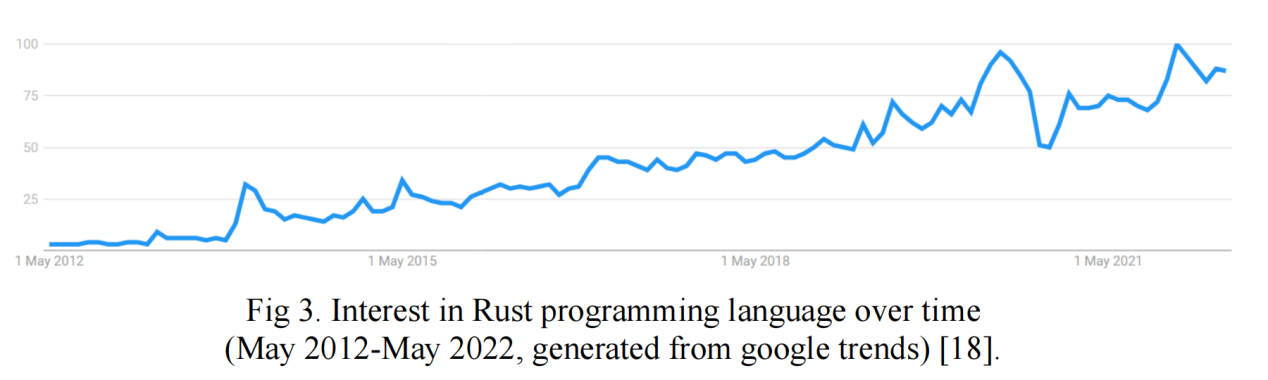
RUST 调研报告

1. 简介

Rust最初是由[Mozilla](https://baike.baidu.com/item/Mozilla/7148071?fromModule=lemma_inlink" \t "/Users/yangyuhang/Documents\\x/_blank)研究院的Graydon Hoare设计创造，然后在Dave Herman, Brendan Eich以及很多其他人的贡献下逐步完善的。 Rust的设计者们通过在研发Servo网站浏览器布局引擎过程中积累的经验优化了Rust语言和Rust[编译器](https://baike.baidu.com/item/%E7%BC%96%E8%AF%91%E5%99%A8/8853067?fromModule=lemma_inlink" \t "/Users/yangyuhang/Documents\\x/_blank)。

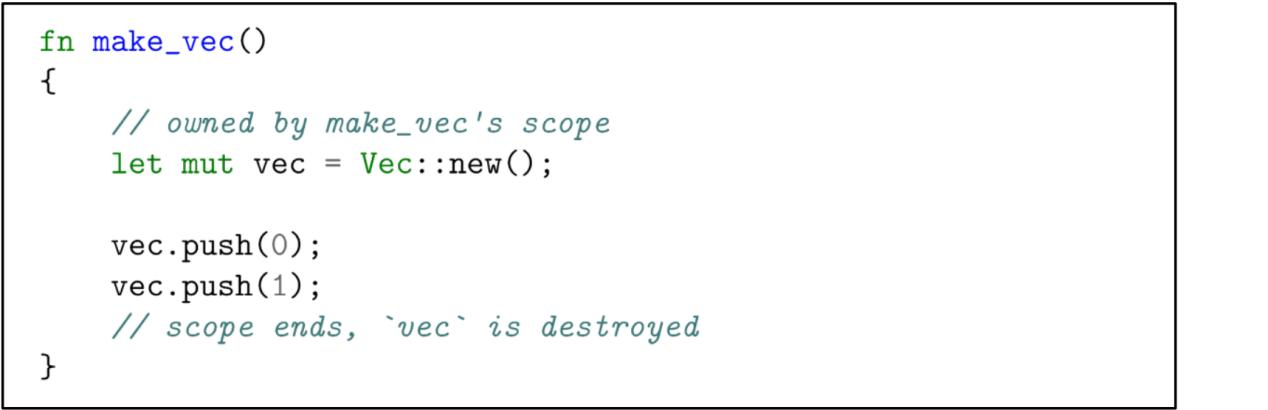
从发布起，Rust受到了广泛的关注，连续七年在 Stack Overflow开发者调查的“最受喜爱编程语言”评选项目中折取桂冠。开发者对于Rust语言的关注度也在逐步提升（如下图）



1. 优势
2. Safety(language)：

一种变成语言的安全性主要归结于其防止和发现问题（如缓冲区越界）的能力，这类问题又主要归结于内存管理。

1. Rust使用**所有权系统**来决定内存的分配和释放，提高了运行的表现。简单来说：当一个变量被声明时，内存被分配；当变量不在范围内时，内存则被释放（好像变量的范围是内存的所有者）。通过禁止编程者直接手动管理内存，防止了use after 和 double free 等漏洞。所有权系统将内存和变量的生命周期直接联系起来。



（2）内存在同一时间只能有一个所有者。这会导致函数之间无法共享参数的问题，这个问题通过一个**借阅系统**来实现，其规则如下：只能存在一个可变的参数或者多个不可变的参数。两种不能同时实现，即意味着最多只能有一线程修改内存且没有其他线程读取该内存。

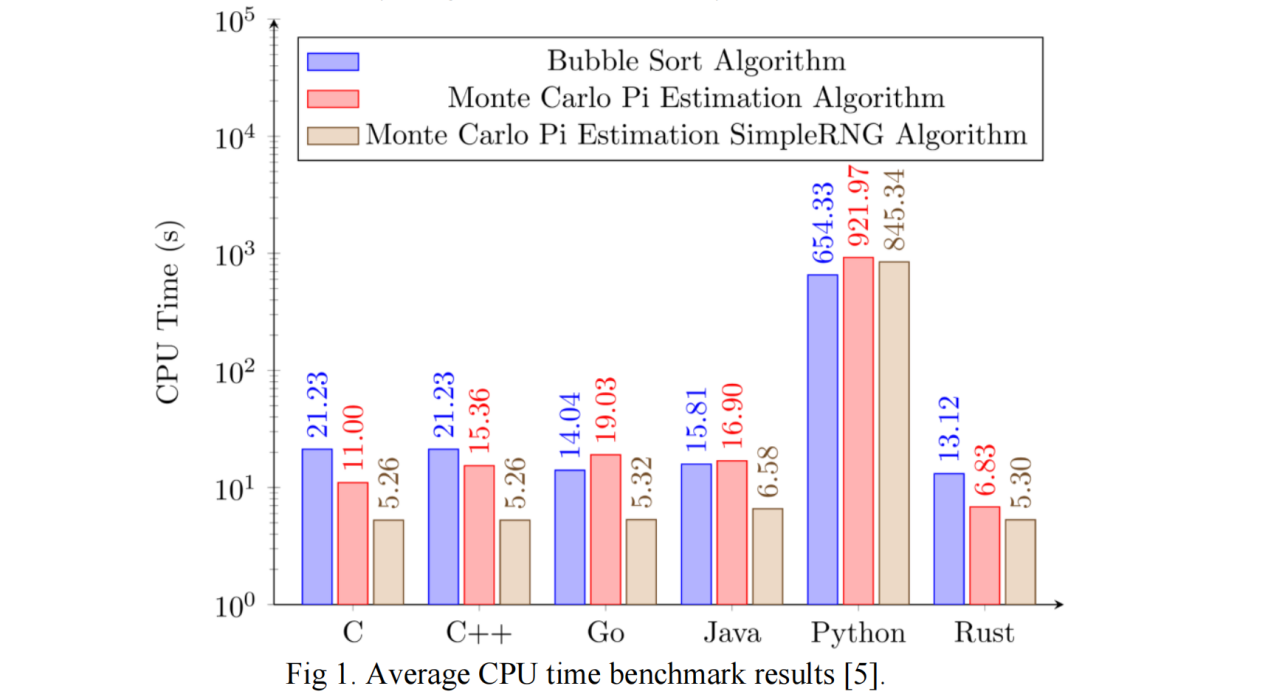
（3）rust还存在许多其他的安全特性，如：缓冲区访问的自动边界检查

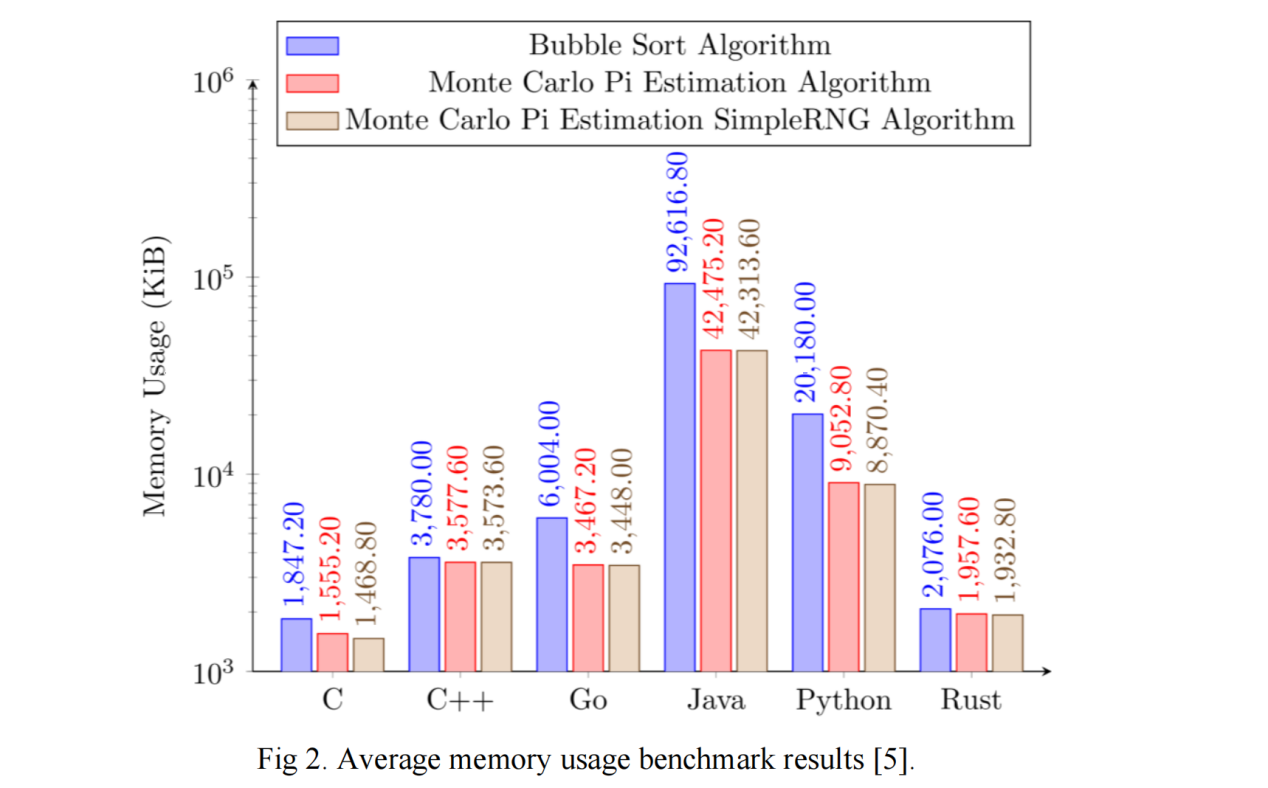
1. Performance：

（1）Rust使用零损耗抽象的概念，在不限制性能的同时简化了语言。例如：

1. 单形态化：允许用户创建通用化的函数，并在编译时转化为具体类型的函数。
2. 标准库函数：不需要重新创建通用类型，提供了性能和互用性更好的库。
3. rust没有内存垃圾收集器。垃圾收集器：
4. 垃圾收集器增加了内存和cpu的使用
5. 垃圾处理器很难控制，可能会导致意外的暂停

下图反映了六种语言的性能：





1. Others
2. 隔离（isolation）零拷贝的软件故障隔离：

Rust支持软件故障隔离（SFI）比任何主流语言都要低的开销。SFI封装

软件中不受信任的扩展，且不依赖于硬件地址空间。

1. 分析（analysis）有效的静态信息流分析

Rust实现精确高效的静态信息流量控制（IFC）。IFC确保不受信任的模块不会通过未经授权的渠道泄露敏感数据,从而为其提供安全保障。

1. 自动化（automation）自动检查点

三、总结

综上。Rust语言相较于传统C语言，有上述的诸多特性和优势，因此通过rust语言对FreeRTOS进行重写，将对其性能和安全性有巨大优化。

四、参考文献

### [1]William Bugden, Ayman Alahmar. Rust: The programming language for safety and performance.*arXiv preprint arXiv:2206.05503, 2022*

[2]Abhiram Balasubramanian, Marek S Baranowski, Anton Burtsev, Aurojit Panda, Zvonimir Rakamarić, Leonid Ryzhyk .*Proceedings of the 16th workshop on hot topics in operating systems, 156-161, 2017*