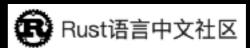
第一课: Rust入门基本原理

Rust内存管理、所有权机制、借用规则

苏林







今天公开课内容

- 1、理解Rust内存管理.
- 2、理解Rust所有权机制.
- 3、理解Rust借用规则.

Rust是内存安全、没有GC(垃圾回收)的高效语言。使用Rust,需要正确理解Rust管理内存的方式。

栈 -> 属于操作系统的概念, 操作系统负责管理栈空间, 负责创建、释放栈帧

Rust是内存安全、没有GC(垃圾回收)的高效语言。使用Rust,需要正确理解Rust管理内存的方式。

堆 GC -> go、python、java。Rust只要能够编译通过,多数情况下没有内存问题.

作用域:

Rust如何使用堆和栈

- A、栈适合存放存活时间短的数据.
- B、数据要存放于栈中,要求数据所属数据类型的大小是已知的.
- C、使用栈的效率要高于使用堆.
- D、Rust将哪些数据存放于栈中? => 布尔值、各种整数、浮点、元组、引用
- E、Rust除了使用堆栈,还使用全局内存区(静态变量区和字面量区).
- F、Rust中允许使用const定义常量。常量将在编译期间直接以硬编码的方式内联(inline)插入到使用常量的地方

Rust位置表达式和值

- A、通过示例来理解变量、位置和值的关系
- B、理解变量的引用
- C、位置和值与Move语义、Copy语义的关联

理解Rust所有权机制

Rust的所有权系统是保证Rust内存安全最关键的手段之一,例如它使得Rust无需GC也无需手动释放内存。所有权系统影响整个Rust,它也使得Rust的很多编码方式和其他语言不太一样。因此,需要掌握好Rust的所有权规则,才能写出可运行的、正确的Rust代码,并且越熟悉所有权规则,在编码过程中就越少犯错。

理解Rust所有权机制

理解Rust的变量作用域

=>

悬垂引用

理解Rust所有权机制

- 1、Rust所有权规则概述
- 2、谁是谁的所有者
- 3、Rust中数据的移动
- 4、所有权移动后修改数据
- 5、移动真的只是移动吗
- 6、Copy语义
- 7、克隆数据
- 8、函数参数和返回值的所有权移动

理解Rust借用规则

- 1、引用和所有权借用
- 2、可变引用和不可变引用的所有权规则
- 3、容器集合类型的所有权规则
- 4、所有权移动后修改数据
- 5、移动真的只是移动吗
- 6、Copy语义
- 7、克隆数据
- 8、函数参数和返回值的所有权移动

理解Rust借用规则

理解可变引用

深入理解Move

QA环节

加群一起交流Rust & Databend







