从 Rust 说起的

现代程序语言设计与内存安全

BRETHLAND YANG 2021.3.26

What's Rust?

- 一门披着 OCaml 皮的 C++?
  - 图灵完备的类型系统，表达式语言，并选择性吸收了 Haskell 的类型类概念。
- 高效可靠，零成本抽象，与 C / C++ 兼容的 ABI
  - Affine Type System Ownership
- 强大的异步编程支持
- 极为现代化的工具链和活跃的社区
- 以实用为第一目标

Syntax

- let 定义变量
- loop, for, if, match, while…
- fn, mod, use
- struct, enum, trait, impl…
- async, await
- where ::<T> turbo.fish

Types

- 基本类型（数值，二元，字符（UTF-8））
- 组合类型
  - 积集（同质，异质，有名）
  - 和集
- 函数类型
  - 闭包
- 指针，引用
- Trait

Lifetime and Borrow

- 一个变量最多只能使用一次（mutable）
- 可以有多个不可变引用，但只能有一个可变
- 当一个变量没有引用时，将被清除
- 解决了悬空指针，多次释放等问题。
- 一定程度上的线程安全
  - 干掉需求就是最好的解决方法（x）-> MPSC 协程（std 一小部分就随便实现了 Go 语言层面的独特功能）
- 显式地指定 Lifetime（and Lifetime elision）

Modern Language

- 我们究竟需要什么?
- 从问题出发
  - 抽象能力与性能（C 到 C++，C 到 Python）
  - 跨平台（Java 与 Erlang 的虚拟机，LLVM 系）
  - 并发编程（不可变与 Channel，锁与信号量，单线程 -> 协程（异步））
  - 内存安全（完全的脱离底层 Lisp，C++ 的挣扎史）
  - 类型系统（Haskell 系， Java 系，我们需要 Subtyping 吗）
  - 工具链（包管理，构建工具，运行时与调试器）

好的语言和好用的语言

- 心智负担和学习曲线
- 活跃的社区和虚心理智的设计者
- 叫好与叫座（有人愿意开发软件包吗）
- 它能被工程化吗?
- 使用者本身的素质差异
- 在国内……
  - 一门语言的命运啊，不止要……还要……

Play with Memory

- 访问了不该访问的
- 修改了不该修改的
- 泄漏了不该泄漏的
- 是谁改了我，而我又改了谁?
- 分配与释放的模型符合直觉吗?
- 与内存的搏斗就是与记忆的搏斗?

Rust

并不能保证你的安全，但它尽到了职责。

Life is long, so I choose Rust.