  | CSDN（ID：CSDNnews）

我从事专业的 C++ 编程工作已经四年多了，然而，三个月前我找到了一份新工作需要使用 Rust。我想通过本文，分享一下我个人关于在两种语言之间切换的经验和想法。

声明：本文不打算比较 C++ 和 Rust，文中的观点源自个人的经历。

# ****1 C++ 和 Rust 的工作介绍****

我认为，个人从事的工作会极大地影响对语言的体验，下面我来简单介绍一下背景。

我使用 C++ 大多从事的是编写数据库的工作。数据库不是常见的应用程序，通常它们独占服务器，可以使用大量的系统调用，而且在有些情况下甚至可以完全绕过内核。另一方面，这意味着数据库是研究特定语言的一个有趣的样本，因为你需要关注性能，为客户提供良好的用户体验，同时还需要保证正确。

鉴于公司的保密协议，所以我不能分享工作细节。我运行 Rust 的环境是具有高负载和高标准性能要求的数字运算异步服务器。它有点类似于数据库，但用户体验可能并不那么重要。

就本文而言，我认为这两者非常相似。

# ****2反复谈及的方面****

Rust 具有良好的内存安全保证。这个话题已被反复讨论，所以不在此赘述。虽然我已经使用 C++ 四年多了，但即便是已经过审查和合并的代码，偶尔也会遇到内存问题引发的服务器崩溃。很难说有多少这样的代码进入了生产环境，因为人们只是在发生段错误时重新启动服务器。模糊测试确实可以确保覆盖一些奇怪的案例，但终究不是灵丹妙药。总的来说，我认为交付 Rust 代码比 C++ 代码更安心。

# ****3构建系统****

在日常工作中，最令我厌烦的工作莫过于构建 C++ 代码。作为一名开发人员，我希望能够编写一个简短的命令来构建整个项目。“在构建服务器之前，你只需要运行这两个命令……”，这句话简直太可怕了，因为这表明构建过程是多步骤的。每次都需要运行这两个命令？只有当这两个文件发生变化时，需要运行这两个命令，这句话什么意思？如果我使用 sanitizers 构建项目，这些命令会如何变化？构建过程不支持 sanitizer 是什么意思？为什么构建脚本突然开始输出链接错误？

我说了这么多，其实就想表达一个观点：缺乏统一的构建系统真的很令人苦恼。Bazel 是朝着正确方向迈出的一步。CMake 至今仍是我的噩梦。

在经历了这一切后，我感觉 Rust 将我从地狱拉到了天堂。你只需编写一个简短的命令来构建整个 Rust 项目。更重要的是，这个世界上所有其他的 Rust 项目使用的都是同一个构建系统，因此你无需将构建脚本从他们的系统转换到自己的系统。你只需要 Cargo.toml 中的一行代码，就可以将构建过程所有的依赖项都包含进来。它甚至会自动将正确的编译标志传递进来。

# ****4编译器****

来自两个编译器的错误消息真的很让人崩溃，你需要付出大量努力才能正确理解和修复。

在 C++ 中，错误消息的大小都是以千字节为单位衡量的。终端模拟器中的无限滚动是绝对必须的，因为 C++ 编译器太喜欢输出文本了。几年后，你会形成某种直觉，只要看到错误多少就能判断自己是应该阅读错误还是看看代码。通常错误消息越大，随便看看代码的作用就会大于阅读错误消息。我感觉，如果 C++ 不改变模板的定义方式，这个问题永远无解。

在 Rust 中，编译器错误（在修复所有拼写错误之后）通常是非常糟糕的消息。通常，这些错误消息表明你需要通过某种方式重新组织代码，或花一些时间调整生命周期，所以你不可能错误地使用内存。虽然这个过程需要时间，而且很烦人，但正确的方法是认真听取编译器的建议。说起来很惭愧，通常听取建议有助于编写更好的代码。此外，错误消息可以完整地显示在一页内，很方便阅读。

# ****5类型系统****

在 Rust 的类型系统中表达想法是一种享受。

首先，非常感谢没有鸭子类型的泛型。Traits 清楚地表达了类型希望看到的合约结构或函数。这也有助于编译器生成有用的错误消息。我们获得的错误消息不再是：“对第 Y 行的方法 clone() 的无效引用”，而是“类型X没有实现 Clone”，清晰明了。

其次，枚举非常强大。Result 和 Option 是非常强大的概念，每个人都在使用。这两个枚举是所有库（包括标准库）表达容易出错的计算和可选值的通用语言。在 C++ 中，我们有 (1) 返回错误代码；(2) 返回无效值；(3) 引发异常；(4) 进程崩溃。所有有问题的选项和每个库都使用不同的选项。除了 Result 和 Option 之外，我发现定义标记枚举的功能也非常方便。

IMG_256

# ****总结****

总的来说，切换到 Rust 后，我感觉日常工作体验得到了极大地提升。Rust 的工具非常友好，语言富有表现力且功能强大。我非常喜欢 Rust，而且我希望Rust 将来的发展能够避免 C++ 的一些陷阱。