# Rust的特点

Rust是一种系统编程语言，Rust提供以下功能：

* 零成本抽象
* 错误消息
* 移动语义
* 没有数据竞争的线程
* 模式匹配
* 保证内存安全
* 高效的C绑定
* 安全的内存空间分配
* 最少的时间

#### 1.零成本抽象

在Rust中，可以添加抽象而不会影响代码的运行时性能。 它提高了代码质量和代码的可读性，而无需任何运行时性能成本。

#### 2.错误消息

在C++编程中，与GCC相比，错误消息有了很好的改进。 在清晰的情况下，Rust更进一步提高。错误消息显示为(格式，颜色)，并在程序中建议拼写错误。

#### 3.类型推断

Rust提供了类型推断的功能，这意味着它可以自动确定表达式的类型。

#### 4.移动语义

Rust提供此功能，允许在源对象是临时对象时，通过移动操作替换复制操作。

#### 5.没有数据争用的线程

数据争用是两个或多个线程访问同一内存位置的条件。 由于所有权系统，Rust提供了没有数据争用的线程功能。 所有权系统仅将不同对象的所有者传输到不同的线程，并且两个线程永远不能拥有具有写访问权限的相同变量。

#### 6.模式匹配

Rust提供了模式匹配的功能。 在模式匹配中，Rust中的模式与’match’表达式一起使用，以更好地控制程序的控制流。 以下是一些模式的组合：

* 字面量
* 数组，枚举，结构或元组
* 变量
* 通配符
* 占位符

#### 7.保证内存安全

Rust通过使用所有权的概念保证了内存安全。 所有权是C语言的内存控制和java的垃圾收集之间的中间地带。 在Rust程序中，内存空间由变量拥有，并由其他变量暂时借用。 这允许Rust在编译时提供内存安全性，而不依赖于垃圾收集器。

#### 8.高效的C绑定

Rust提供了“高效C绑定”的功能，这意味着Rust语言能够在与自身对话时与C语言进行互操作。 Rust提供了一个“外部功能接口”来与C API进行通信，并利用其所有权系统同时保证内存安全。

#### 9.安全的内存空间分配

在Rust中，内存管理是手动的，即程序员可以明确控制分配和释放内存的位置和时间。 在C语言中，使用malloc函数分配内存然后初始化它，但Rust通过单个~运算符拒绝这两个操作。 此运算符将智能指针返回为int。 智能指针是一种特殊的值，用于控制何时释放对象。 智能指针是“智能的”，因为它们不仅跟踪对象的位置，而且还知道如何清理它。

来源：BootWiki  
链接：https://www.bootwiki/rust/rust-features.html  
著作权归作者所有  
商业转载请联系作者获得授权，非商业转载请注明出处