Rust课程3.5
Rustlings答疑 / 拓展内容

徐启航 西安交通大学

unsafe 范式

The paradigm of unsafe

当我们在用unsafe的时候，我们在干什么

- 本质：一个契约的 定义 或者 满足
- 在unsafe中，编译器将契约的检查交给了开发者自身。
- 开发者从享受保护，变成了保护别人的人。因此，开发者需要完备掌握满足该契约所需的背景知识，包括但不限于Rust的借用规则和所有权
- 示例：rustlings中的unsafe

裸指针：*const T和*mut T

- 引用的底层实现
- 常见错误：
  - “*pointer = data”=“{ ptr::drop_in_place(pointer); pointer.write(data) }”
  - 指针不能假定地址对齐。需要读写不一定地址对齐的指针需要考虑read_unaligned方法
  - Rust的借用规则包括引用的存在性。即使不满足规则的引用存在而不使用也是UB！
- 示例：浏览std::ptr的官方文档
- 拓展：非稳定功能strict_provenance

常用的unsafe结构：ManuallyDrop<T>

- 手动控制对象的drop时机
- 特性：
  - 和T可以safe地互相转换
  - unsafe其一：ManuallyDrop::drop：手动释放内部数据，需要保证其不再被使用
  - unsafe其二：ManuallyDrop::take：手动移动内部数据而不消耗外部所有权，同样需要保证内部数据不再被使用
- 示例：浏览ManuallyDrop的官方文档
- 可以注意到，unsafe的约束范围可以划到其他safe的区域（非局部性）。

常用的unsafe结构：NonNull<T>

- 存储不可能为空的指针
- 特性：
  - 不可能为空：在safe层面将“不可能为空”这个约束交给编译器满足
  - Rust的布局优化：Option<NonNull<T>> = *mut T = *const T
- 示例：浏览NonNull的官方文档

常用的unsafe结构：MaybeUninit<T>

- 存储可能未初始化的数据
- 特性：
  - MaybeUninit::uninit方法：可以编译期初始化成未初始化的值
  - 未初始化的一个含义是不保证内部bit固定，因此POD类型也有对应的未初始化
  - 保证与T内存布局一致
  - 与ManuallyDrop一样，不会drop内部可能初始化的数据
- 示例：浏览MaybeUninit的官方文档

常用unsafe：其他函数&方法

- core::ptr
- core::mem
- 智能指针的into_raw、from_raw
- union

- 最后重要的提示：请确认清楚自己在干什么，造成了什么样的影响！

感谢聆听

Thanks for listening