Rust 编程语言入门

杨旭，微软MVP
Rust、Go、C#开发者

9 错误处理

9.1 panic!

不可恢复的错误

Rust 错误处理概述

- Rust 的可靠性：错误处理
  - 大部分情况下：在编译时提示错误，并处理
- 错误的分类：
  - 可恢复
    - 例如文件未找到，可再次尝试
  - 不可恢复
    - bug，例如访问的索引超出范围
- Rust 没有类似异常的机制
  - 可恢复错误：Result<T, E>
  - 不可恢复：panic! 宏

不可恢复的错误与 panic!

- 当 panic! 宏执行：
  - 你的程序会打印一个错误信息
  - 展开（unwind）、清理调用栈（Stack）
  - 退出程序

为应对 panic，展开或中止（abort）调用栈

- 默认情况下，当 panic 发生：
  - 程序展开调用栈（工作量大）
    - Rust 沿着调用栈往回走
    - 清理每个遇到的函数中的数据
  - 或立即中止调用栈:
    - 不进行清理，直接停止程序
    - 内存需要 OS 进行清理
- 想让二进制文件更小，把设置从“展开”改为“中止”：
  - 在 Cargo.toml 中适当的 profile 部分设置：
    - panic = ‘abort’
  - （例子）

不可恢复的错误与 panic!

- （例子）

使用 panic! 产生的回溯信息

- （例子）
- panic! 可能出现在：
  - 我们写的代码中
  - 我们所依赖的代码中
- 可通过调用 panic! 的函数的回溯信息来定位引起问题的代码
- （例子）
- 通过设置环境变量 RUST_BACKTRACE 可得到回溯信息
- （例子）
- 为了获取带有调试信息的回溯，必须启用调试符号（不带 --release）

再见