从C++到Rust，错误处理Result，第一集

原创 Ajonbin AJonbin的杂货铺 2024年01月10日 21:54 上海

Rust并没有像C++那样的异常exception。在Rust，函数通过返回Result类型来表明函数运行中是否发生错误。

如果你去看一些开源的Rust的代码，你可以发现几乎所有的Rust程序都大量的使用了Result。

这是和C++很不一样的语法。

即使不懂Rust语法，熟悉C++的人也大致可以猜出大部分Rust代码的含义。引用，函数，borrow和move等等，这些大致也能猜个八九不离十。

但是不讲讲Result的话，就很难读懂代码，就不能理解Rust的错误处理方式，这也就是我把Result的讲解放在语法讲解之前的原因。

首先Result是一个Enum类型，它有两个可能值Ok和Err。

```
pub enum Result<T, E> {
    Ok(T),
    Err(E),
}
```


Result有两个类型参数，T和E。

T是函数成功情况下应该返回的类型，封装为Ok(T)。

E是有错误时返回的错误类型，封装为Err(T)。

来看这个实际的例子，就以文件操作为例。

```
hello_result$ vim src/main.rs

1 use std::fs::File;
2
3 fn main() {
4     println!("Hello, result!");
5
6     let ret_cargo: Result<File, std::io::Error> = File::open("./Cargo.toml");
7
8     match ret_cargo {
9       Ok(f) => println!("Ok(f) -- {:?}", f),
10      Err(e) => println!("Err(e) -- {}", e),
11    };
12
13    let ret_no: Result<File, std::io::Error> = File::open("./no_such_file");
14
15    match ret_no {
16      Ok(f) => println!("Ok(f) -- {:?}", f),
17      Err(e) => println!("Err(e) -- {}", e),
18    };
19 }
```


标记1，用use关键字引入 std库的File模块，类似于C++的#include

标记2，我们调用File模块的open函数打开Cargo.toml文件。这个打开文件操作应该是成功的。

在第6行，我们把open的返回值赋值为变量ret_cargo，open的返回值类型是一个Result<File, std::io::Error>。

这里Result<T,E>中T的类型是File，当打开文件成功后，返回值是Ok(File)类型。

这里Result<T,E>中E的类型是std::io::Error，是一个IO错误，如果打开文件发生错误，则返回值是Err(std::io::Error)。

第8-11行演示了一个标准的处理Result<>的方法，也就是通过match关键字来对Result变量进行匹配。

标记4处，先对Ok(f)进行匹配，如果open函数返回的是Ok()，那么就会执行Ok(f)=>后面的语句，这里就是打印出f的信息。f就是open返回的实际File对象。

标记6处，{:?}是特殊的println格式，是用来打印出f的debug信息。

标记5处，用来匹配Err的情况，由于打开Cargo.toml的是成功的，那么这里就不会匹配到。

在13-18行，我们试图打开一个不存在的文件，这样在匹配结果的过程中，就会匹配到第17行Err(e)，然后执行Err(e) => 之后的语句，打印出错误信息。

好了，来看下输出结果。

正如前面所说的，

在匹配到Ok(f)的情况下，打印出了文件对象f的信息。

在匹配到Err(e)的情况下，打印出了错误信息，文件不存在。