主流编程语言的异常处理机制

2019-04-24

学习编程语言应该从语言特性入手，而不是编程语言本身。这里尝试对各种编程语言的异常和错误处理机制做一个横向的、简单的了解。涉及到的编程语言包括C、C++、Go、Java、Scala、Kotlin、Ruby、Rust、JavaScript、PHP、Python、Lisp。

C

C语言没有异常捕获机制。程序在发生错误时会设置一个错误代码errno，该变量是全局变量。C语言提供了perror()和strerror()函数来显示与errno相关的描述信息。perror()函数可以直接调用，入参是一个字符串，输出入参：错误文本。strerror()函数入参是一个数字（错误码），返回一个指针，指针指向错误码对应的文本。

```
#include <stdio.h>
#include <errno.h>
#include <string.h>

void main ()
{
  // 打开一个不存在的文件，会发生错误
  fopen ("unexist.txt", "rb");

  // 2
  printf("%d\n", errno);

  // No such file or directory
  perror("");

  // No such file or directory
  printf("%s\n", strerror(errno));
}
```

C++

C++支持异常捕获机制。C++可以抛出或捕获两种内容，一种是int或char*之类的内容，程序可以捕获并抛出，这一点和Java相比有差异，因为Java并不支持直接抛出基本类型的异常：

```
#include <iostream>
#include <exception>
using namespace std;
int main () {
    try
    {
        throw "error";
    }
    catch(const char* msg)
    {
        cout << msg << endl;
    }
}

// error
```

另一种内容就是类，可以是内置的标准异常类，或是自定义的异常类：

```
#include <iostream>
#include <exception>
using namespace std;
int main () {
    try
    {
```

```
        throw exception();
    }
    catch(std::exception& e)
    {
        cout << e.what() << endl;
    }
}

// std::exception
```

Go

Go语言作为非OOP派系的编程语言，并不支持try-catch的语法，但仍然具有类似抛出和捕获的特性。Go语言有3个错误相关的关键字，panic()、recover()和defer。可以理解为，panic()函数抛出异常，recover()函数捕获异常，defer关键字定义最后也就是finally执行的内容：

```
package main
import "fmt"

func main() {
  defer func() {
    err := recover()
    fmt.Println(err)
  }()
  panic("error")
}

// error
```

Java

Java是纯粹的OOP语言，仅支持对象的抛出和捕获：

```
public class ErrorTest {
    public static void main(String[] args) {
        try {
            throw new Exception();
        } catch (Exception e) {
            System.out.println(e);
        }
    }
}

// java.lang.Exception
```

Scala

Scala和Java是一个流派，同样仅支持对象的抛出和捕获，除了语法上和Java稍有差异，概念上基本是一jian样rong的：

```
object ErrorTest {
  def main(args: Array[String]): Unit = {
    try {
      throw new Exception()
    } catch {
      case e: Exception => print(e)
    }
  }
}

// java.lang.Exception
```

另外，Scala抛出的是Java的异常，也许Scala不能算作是独立的编程语言，而是依附于Java、为Java提供语法糖的编程语言。这一点值得深入思考和探究。

Kotlin

Kotlin和Scala是一种性质的语言，默认抛出的同样是Java的异常：

```
fun main(args: Array<String>) {
  try {
    throw Exception()
  } catch (e: Exception) {
    print(e)
  }
}

// java.lang.Exception
```

Ruby

Ruby使用关键字raise和rescue代替try和catch来实现异常的抛出和捕获。Ruby同样支持try-catch关键字，这里暂不讨论，因为我没搞清楚它的用法。

```
begin
  raise "error"
  rescue Exception => e
    puts e
end

// error
```

Rust

Rust没有try-catch的语法，也没有类似Go的错误处理函数，而是用对错误处理进行过包装的`Option<T>`或Option的加强版`Result<T, E>`进行错误处理。Rust的模式匹配和Scala类似：

```
fn main() {
  match find() {
    None => println!("none"),
    Some(i) => println!("{}", i),
  }
}

fn find() -> Option<usize> {
  if 1 == 1 {
    return Some(1);
  }
  None
}

// 1
```

JavaScript

脚本语言在变量类型上不做强制约束，捕获时也就不能按照异常类型来做区分。抛出错误的内容还是相对自由的：

```
try {
  throw 1
} catch (e) {
  console.log(e)
}

// 1

try {
  throw new Error('')
} catch (e) {
```

```
  console.log(e)
}

// Error
```

PHP

PHP的try-catch和Java类似，并没有特殊之处：

```
<?php
  try {
      throw new Exception("error");
  } catch (Exception $e) {
      echo $e->getMessage();
  }
```

Python

Python在语法上能找到Ruby的影子，raise触发异常，execpt捕获异常：

```
try:
  raise
except:
  print("error")
```

Lisp

Lisp整体较复杂，Lisp捕获处理异常的内容暂时留坑。以下是Common Lisp触发错误的情形之一，declare会声明函数入参类型，传入错误参数将引发错误：

```
(defun df (a b)
  (declare (double-float a b))
  (* a b))

(df "1" 3)

// *** - *: "1" is not a number
```

后续

原先想梳理这些语言的大部分异常和错误处理相关概念，然而真正开始后发现比较困难，并且之前我没能区分”exception”和”checked exception”，以致从立意到标题到内容可能都有偏差。这次就先提及”exception”，之后讨论关于”checked exception”的内容。