

☰ 它叫Optional, 却必不可少

◀ 返回视频

Optional关键实现技术模拟 ▶

(/series/optional-is-not-an-option)

(https://www.boxueio.com/series/optional-is-not-an-option/ebook/139)

为什么“哨兵值”没有解决错误处理问题

[⌕ Back to series \(/series/optional-is-not-an-option\)](#)

在编程中，无论是编写还是调用函数，一个最普遍的情况，就是在某些情况下，函数并不总是可以返回我们期望的值。

🔍 字号

● 字号

🖌 默认主题

🖌 金色主题

🖌 暗色主题

需要你小心处理的“哨兵值”

例如，在C++里，当我们打开一个文件时，并不总是可以获得文件的句柄：

```
int fd = open("someFile", O_RDWR);

if (fd != -1) {
    // Some file operations here
}
```

在这里，尽管 open 返回一个 int 表示打开的文件句柄，但它也用了一个整数 -1 来表示任意一种无法成功打开文件的错误。

或者，当我们在STL中查找元素时：

```
auto numbers = { 1, 2, 3 };
auto iteratorOf4 = std::find(numbers.begin(), numbers.end(), 4);

if (iteratorOf4 != numbers.end()) {
    // 4 is found in numbers
}
```

这次，尽管 numbers.end() 也是一个合法的迭代器对象，但是，它表达的含义却是“要查找的值不存在”，我们并不能读取这个迭代器指向的值。

观察这两个例子你就会发现，它们的一个共性就是都使用了一个同类型的特殊值来表示某种特殊的含义，我们通常管这样的值叫做“哨兵值（sentinal value）”。

然而，这个哨兵值就像个潘多拉的盒子一样，为解决错误情况提供了一种方式的同时，也为程序带来无尽的bug。究竟为什么会如此呢？

第一个原因，这种错误处理的方式是被动的，无论是 open 还是 std::find 你从它们各自的签名以及调用上，根本无法得知它有可能发生错误，以及对应的错误处理方式。你总是以一个会正常执行的方式来调用函数，然后通过查阅文档得知对应的错误处理方式。这样一来，怕麻烦不去处理的开发者，有之；粗心大意写错文档的开发者，有之；你又如何相信这样的方式可以安全的处理所有的错误呢？

第二个更重要的原因，是哨兵值的方式，使我们无法通过编译器来强制错误处理的行为。因为这些“哨兵值”的类型，和正常情况下函数返回的类型是一样的。因此，当它们悄无声息的混入正常业务逻辑代码的时候，编译器对此毫无感知。就像我们之前看到过的一样，-1 也是一个整数，numbers.end() 也是一个合法的迭代器，它们只有在你的程序崩溃之后，才会显出原形。

小心也没办法处理的“哨兵值”

对于我们之前提到的几个例子，如果你坚持认为，小心驶得万年船。对“哨兵值”谨慎处理能相当大程度上避免这种方式的弊端，那么，我们来看下面这个Objective-C的例子，即便你小心处理，也是个错：

```
NSString *tmp = nil;

if ([tmp rangeOfString:@"Swift"].location != NSNotFound) {
    // Will print out for nil string
    NSLog(@"Something about swift");
}
```

在我们的例子里，尽管 tmp 的值是 nil，但调用 tmp 的 rangeOfString 方法却是合法的，它会返回一个值为0的 NSRange，因此，location 的值也是0。

但是，NSNotFound 的值却是 NSIntegerMax。于是，尽管 tmp 的值为 nil，我们还可以在控制台看到_Something about swift_这样的输出。

怎么样？现在你应该彻底对这个“哨兵值”没什么好感了吧。

What's next?

既然“哨兵值”不是一个好方法，又该如何解决函数有可能返回错误的情况呢？在下一节，我们就来介绍 Swift 的方法，通过把不同的结果放在一个 enum 里，Swift 可以通过编译器，强制我们明确处理函数返回的异常情况。

◀ 返回视频

Optional 关键实现技术模拟 ▶

(/series/optional-is-not-an-option)

(https://www.boxueio.com/series/optional-is-not-an-option/ebook/139)



职场漂泊的你，每天多学一点。

从开发、测试到运维，让技术不再成为你成长的绊脚石。我们用打磨产品的精神去传播知识，把最新的移动开发技术，通过简单的图表，清晰的视频，简明的文字和切实可行的例子一一向你呈现。让学习不仅是一种需求，也是一种享受。

泊学动态

一个工作十年PM终创业的故事（二）(https://www.boxueio.com/after-the-full-upgrade-to-swift3)
Mar 4, 2017

人生中第一次创业的"10有" (https://www.boxueio.com/founder-chat)
Jan 9, 2016

猎云网采访报道泊学 (http://www.lieyunwang.com/archives/144329)
Dec 31, 2015

What most schools do not teach (https://www.boxueio.com/what-most-schools-do-not-teach)
Dec 21, 2015

一个工作十年PM终创业的故事（一）(https://www.boxueio.com/founder-story)
May 8, 2015

泊学相关

关于泊学 >

加入泊学 >

泊学用户隐私以及服务条款 (HTTPS://WWW.BOXUEIO.COM/TERMS-OF-SERVICE)

版权声明 (HTTPS://WWW.BOXUEIO.COM/COPYRIGHT-STATEMENT)

联系泊学

Email: 10[AT]boxue.io (mailto:10@boxue.io)
QQ: 2085489246