Table of Contents

Welcome

A bunch of programming documents.

Go Documents

Go Standard library Translation

errors

本文是 Go 标准库中 errors 包文档的翻译， 原文地址为：
https://golang.org/pkg/errors/

概述

errors 包实现了用于处理错误的函数。

示例：

```
package main

import (
    "fmt"
    "time"
)

// MyError 是一个包含了时间和消息的错误实现
type MyError struct {
    When time.Time
    What string
}

func (e MyError) Error() string {
    return fmt.Sprintf("%v: %v", e.When, e.What)
}

func oops() error {
    return MyError{
        time.Date(1989, 3, 15, 22, 30, 0, 0, time.UTC),
        "the file system has gone away",
    }
}

func main() {
    if err := oops(); err != nil {
        fmt.Println(err)
    }
}
```

示例执行结果：

```
1989-03-15 22:30:00 +0000 UTC: the file system has gone away
```

New 函数

```
func New(text string) error
```

根据给定的文本返回一个错误。

示例：

```
package main

import (
    "errors"
    "fmt"
)

func main() {
    err := errors.New("emit macho dwarf: elf header corrupted")
    if err != nil {
        fmt.Print(err)
    }
}
```

示例执行结果：

```
emit macho dwarf: elf header corrupted
```

fmt 包的 Errorf 函数可以让用户使用该包的格式化功能来创建描述错误的消息。

示例：

```
package main

import (
    "fmt"
)

func main() {
    const name, id = "bimmler", 17
    err := fmt.Errorf("user %q (id %d) not found", name, id)
    if err != nil {
        fmt.Print(err)
    }
}
```

示例执行结果：

```
user "bimmler" (id 17) not found
```

Go Gotchas

This collection of Go gotchas and pitfalls will help you find and fix similar problems in your own code.

Assignment to entry in nil map

Why does this program panic?

```
var m map[string]float64
m["pi"] = 3.1416
```

```
# Output
panic: assignment to entry in nil map
```

Answer

You have to initialize the map using the make function (or a map literal) before you can add any elements:

```
m := make(map[string]float64)
m["pi"] = 3.1416
```

Invalid memory address or nil pointer dereference

Why does this program panic?

```
package main

import (
    "math"
    "fmt"
)

type Point struct {
    X, Y float64
}

func (p *Point) Abs() float64 {
    return math.Sqrt(p.X*p.X + p.Y*p.Y)
}

func main() {
    var p *Point
    fmt.Println(p.Abs())
}
```

```
panic: runtime error: invalid memory address or nil pointer dereference
[signal SIGSEGV: segmentation violation code=0x1 addr=0x0 pc=0x499043]

goroutine 1 [running]:
main.(*Point).Abs(...)
    /tmp/sandbox466157223/prog.go:13
main.main()
    /tmp/sandbox466157223/prog.go:18 +0x23
```

Answer

The uninitialized pointer p in the main function is nil, and you can't follow the nil pointer.

> If x is nil, an attempt to evaluate *x will cause a run-time panic.
>
> — The Go Programming Language Specification: Address operators

You need to create a Point

```
func main() {
    var p *Point = new(Point)
    fmt.Println(p.Abs())
}
```

Since methods with pointer receivers take either a value or a pointer, you could also skip the pointer altogether:

```
func main() {
    var p Point // has zero value Point{X:0, Y:0}
    fmt.Println(p.Abs())
}
```

Array won't change

Why does the array value stick?

```
package main

import "fmt"

func Foo(a [2]int) {
    a[0] = 6
}

func main() {
    a := [2]int{1, 2}
    Foo(a) // Try to change a[0].
    fmt.Println(a) // Output: [1 2]
}
```

Answer

> - Arrays in Go are **values**
> - When you pass an array to a function, **the array is copied.**

If you want to Foo to update the elements of a function, use a **Slice** instead.

```
package main

import "fmt"

func Foo(a []int) {
    if len(a) > 0 {
        a[0] = 6
    }
}

func main() {
    a := []int{1, 2}
    Foo(a) // Change a[0].
    fmt.Println(a)  // Output: [6 2]
}
```

A slice does not store any data, it just describes a section of an underlying array.

When you change an element of a slice, you modify the corresponding element of its underlying array, and other slices that share the same underlying array will see the change.

How does characters add up?

Why doesn’t these print statements give the same result?

```
fmt.Println("H" + "i")
fmt.Println('H' + 'i')

// Output:
// Hi
// 177
```

Answer

The rune literals 'H' and 'i' are integer values identifying Unicode code points: 'H' is 72 and 'i' is 105.

You can turn a code point into a string with a conversion.

```
fmt.Println(string(72) + string('i')) // "Hi"
```

You can also use the **fmt.Sprintf** function.

```
s := fmt.Sprintf("%c%c, world!", 72, 'i')
fmt.Println(s)// "Hi, world!"
```

What happened to ABBA?

What’s up with strings.TrimRight?

```
fmt.Println(strings.TrimRight("ABBA", "BA")) // Output: ""
```

Answer

The Trim, TrimLeft and TrimRight functions strip all Unicode code points contained in a **cutset**. In this case, all trailing A:s and B:s are stripped from the string, leaving the empty string.

To strip a trailing string, use **strings.TrimSuffix**.

```
fmt.Println(strings.TrimSuffix("ABBA", "BA")) // Output: "AB"
```

Where is my copy?

Why does the copy disappear?

```
var src, dst []int
src = []int{1, 2, 3}
copy(dst, src) // Copy elements to dst from src.
fmt.Println("dst:", dst)

// Output:
// dst: []
```

Answer

The number of elements copied by the copy function is the **minimum of len(dst) and len(src)**. To make a full copy, you must allocate a big enough destination slice.

```
var src, dst []int
src = []int{1, 2, 3}

dst = make([]int, len(src)) // Update Here

copy(dst, src) // Copy elements to dst from src.
fmt.Println("dst:", dst)

// Output:
// dst: [1 2 3]
```

The return value of the copy function is the number of elements copied. See Copy function for more about the built-in copy function in Go.

Using append

You could also use the append function to make a copy by appending to a nil slice.

```
var src, dst []int
src = []int{1, 2, 3}
dst = append(dst, src...)
fmt.Println("dst:", dst)

// Output:
// dst: [1 2 3]
```

Note that the capacity of the slice allocated by append may be a bit larger than len(src).

Why doesn’t append work every time?

What’s up with the append function?

```
a := []byte("ba")

a1 := append(a, 'd')
a2 := append(a, 'g')

fmt.Println(string(a1)) // bag
fmt.Println(string(a2)) // bag
```

Answer

If there is room for more elements, append reuses the underlying array. Let's take a look:

```
a := []byte("ba")
fmt.Println(len(a), cap(a)) // 2 32
```

This means that the slices **a, a1 and a2** will refer to the **same underlying array** in our example.

To avoid this, we need to use two separate byte arrays.

```
const prefix = "ba"

a1 := append([]byte(prefix), 'd')
a2 := append([]byte(prefix), 'g')

fmt.Println(string(a1)) // bad
fmt.Println(string(a2)) // bag
```

Go Blog

Some Go Learning notes.

- A Quick Introduction to Elasticsearch for Node Developers

2020

记录每一周的阅读记录及链接

Weekly 01

Book

- 《三体III-死神永生》
- 《Javascript 设计模式与开发实践》

Blog

- A Quick Introduction to Elasticsearch for Node Developers

Blockchain

- An Introduction to Binance Smart Chain (BSC)
- Adding Binance Smart Chain and JNTR to your MetaMask

Weekly 02

Book

- 《三体III-死神永生》
- 《Javascript 设计模式与开发实践》

Blog

- How to be a Better Software Engineer: Book Recommendations
- Best Practices Every Node Developer Should Follow
- Redis + NodeJS 实现一个能处理海量数据的异步任务队列系统
- GitHub Actions 入门教程
- Creating Fast APIs In Go Using Fiber

Tutorial

- basic bash guide

Library

- oclif 命令行工具框架
- Fiber - Go web framework
- Go cobra

Blockchain

- The Best Way To Learn Blockchain Programming
- How to Build Blockchain App - Ethereum Todo List 2019
- Getting Up to Speed on Ethereum

Speech

A collection of excellent english speeches.

Don't You Quit !

Every champion has felt it. Every victorious person has felt it. The urge to quit. Don't you give up on your dream. I don't care if you don't have the money, if you don't have help, and you don't have the family for it, and if you don't have the friends for it. Don't you give up on your dream. Don't you do it.

每个冠军，每个胜利者，都曾经想要放弃。不要放弃你的梦想！不管你有没有钱，不管有没有人帮助你，不管你的家人是否支持你，不管你的朋友是否支持你，你都不要放弃梦想！千万不要！

It may take you twice as long. You may have to take courses and classes. You might not read as fast. You might not move as quick. You might not have as much. But don't you quit. 可能你需要多花很多时间，可能你需要多参加很多课程，可能你读得没别人快，可能你的行动不如别人敏捷，可能你拥有的没有别人多，但是你一定不要放弃。

If you lay dead even the animals won't bite you. The risk of being bitten is the cost of getting up. And you have to decide. Are you so concerned about being bitten that you're willing to spend the rest of your life lying dead? Or is there something pumping down inside of you that is saying bite me or not, I'm getting up.

如果你混吃等死，连野兽都不屑于啃你一口。如果你害怕被吃掉，那么你就永远爬不起来。你要自己做决定，你真的那么害怕被吃掉吗？害怕到你宁愿混吃等死，也不愿意起来奋斗吗？又或者，其实你身上有一股冲劲，不管会不会被吃掉，我都要站起来！

I'm going to be the best me. I'm gonna do all that I can do. I'm going for it. Anybody can do anything they set their mind to. But it depends on how bad you want it. With enough persistence most things that seem impossible become possible. You just have to show up or show at so often. Time after time you gotta wake up and you got a drive. You got to be driven to get something done now.

我要做最好的自己，我要付出所有的努力，我要朝着目标去奋斗。有志者，事竟成。但是，这取决于你的愿望有多强烈。只要付出坚持不懈的努力，一切皆有可能。你要让世人看见你的努力，你要一次次地觉醒，你要找到自己的动力，你要让一股力量驱使你去实现梦想。

You got to be willing to stand on your beliefs. Cuz a lot of times you're the only one who can see it. And other people don't see it really. But you gotta really believe in what you've got inside of you that's telling you and driving you to a certain goal. Every minute you decide upon something you know that's what you want. You know you're going to do it. All of these negatives that have been bothering you. They pick up their baggage and get out. They can't live any positive mind. You have to move with courage, with faith, with determination.

你要勇于坚持自己的信念。因为，很多时候，和其他人相比，你才是最清醒的人。但是，你必须对自己的梦想有信心，要对你的目标和动力有信心。每次做决定，你都清楚地知道，这就是你想要的，你知道自己要努力实现它。那些曾经让你心烦意

乱的消极的声音，都被清理得一干二净，因为积极的心态容不得它们存在。你要勇往直前，你要心怀信念，你要意志坚定。

I don't care what it is. I don't care how difficult it is. Listen, you better put some strength in your back. Plant your feet and stand up. Don't you dare sit down. Don't you dare crumble under the weight of it. Don't you dare give in because of the nature. Don't you dare go home and raise the white flag of surrender because whatever is over you defines you and whatever defines you limits you.

我不管你的梦想是什么，我不管它有多难实现。听着，你必须给自己力量。稳住脚跟，站起来！不要再坐下去了！不要再让压力把你压垮了！不要因为自己的惰性而屈服！不要自己认输然后灰溜溜地躲回家去！能承受多大的压力，决定了你是什么样的人。而你是什么样的人，决定了你能成多大的事。

This is not your season to die. This is not your season to quit. There's joy. There's peace. There's breakthrough. There's provision. But you'll never see it if you don't stand up to it. Your pain is going to be a part of your prize, a part of your product. I challenge you to never give up. Never give in. And finally, guys, you gotta want to succeed as bad as you want to breathe.

不要因此一蹶不振，不要因此半途而废。无尽的快乐，内心的平静，巨大的突破，大好的前途，就在前方。但是，如果你不勇敢面对现实，你就永远看不到这些美好的东西。你的痛苦，将会成为你的荣誉，你的成就。你敢不敢坚持下去，永不言弃？朋友们，你要渴望成功，成功和你的呼吸一样重要。

That's just kind of the way that life works sometimes. It's Murphy's Law. When things go wrong, they always seem to happen at once and they just compound on top of each other and it's, it's pretty easy sometimes to feel beaten up. When you're faced with all of those issues and all those problems and they all hit you at the same time that doesn't mean give up. In fact, it means the opposite. It means it's time for you to fight harder to dig in. It means it's time for you to go on the warpath.

有时候生活就是这样的，这就是墨菲定律。一事不顺，诸事不顺，这很容易让你感到沮丧。当你同时面临很多困难，很多问题，你不能放弃。事实上，你更加应该坚持下去。因为这些事情意味着，你要加倍努力，你要埋头苦干。这意味着，你要走上战场去勇敢地搏斗！

Look For The Good In Your Life

Whatever you focus on you will find. If you search for negativity in this world, you will find plenty of it. If you search for hate, anger, violence and sadness you will find it.

你关注什么，你就会收获什么。如果你寻找负能量，你就会发现很多负能量；如果你寻找憎恨、愤怒、暴力和悲伤，你也会找到。

But the same is true on the flip side: If your only intention is to search for the good, you will find only the good. Whatever meaning you give your life, becomes your life.

但是另外一面也是正确的。如果你的目标是寻找美好，那么你只会找到美好。你赋予你的生命什么，你的生命就会成为什么。

It can be a failure or a lesson. Heart break or character building. Life is against you, or making you stronger.

失败还是教训，伤心还是塑造品格，生活是故意刁难你还是让你变得强大，全看你自己的态度。

Because there is no such thing as reality. We choose our own reality by the meaning we give each moment in our lives. Make it your intention to look for the good in your life. To notice the good in others. To be grateful for what you do have. To see challenges as opportunities to show your true character.

因为根本没有“现实”这回事，全凭我们自己的视角。所以，一定要寻找美好的东西，珍惜自己所拥有的。把挑战当作是锻炼自己的机会。

Remember: What you give your attention to will become your experience in life. Practice seeing the good in your life, and in others. Think the best, expect the best and always ask yourself how can this benefit my life. 记住，你的关注点在哪里，你的成长就在哪里。多关注生命中美好的人和事，思考最好的，期待最好的，而且时刻问自己，这件事怎么能让我的生命受益？

Leave who you were. Love who you are and look forward to who you will become.

接纳过去的自己，爱现在的自己，期待将来的自己。