# Table of Contents


- Introduction 1.1
- Node 1.2
- Linux 1.3
- Go 1.4
  - Go Documents 1.4.1
    - errors 1.4.1.1
  - Go Gotchas 1.4.2
    - Assignment to entry in nil map 1.4.2.1
    - Invalid memory address or nil pointer dereference 1.4.2.2
  - Go Blog 1.4.3
- 算法 1.5
  - 冒泡算法 1.5.1

# Introduction

A bunch of programming documents.

# Node相关的一些笔记

# Linux

不能折叠的吗？

# Go

A bunch of Go learning stuffs.

# Go Doc

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 awa
```

## 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 corrupt
    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
[signal SIGSEGV: segmentation violation code=0x1 addr=0x0 p

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())
}
```

# Go Blog

Some Go Learning notes.

# 算法

一些常用的算法信息

# 冒泡算法

其中的一些描述