常用标准库

时间

时间是非常重要的，离开了时间，几乎没有那个生产环境数据能够有意义。

在Go语言中，时间定义为Time结构体。

```
var t = time.Now()
fmt.Printf("%T, %[1]v\n", t)
// time.Time, 2022-10-06 19:35:00.1963422 +0800 CST m=+0.001534601
fmt.Printf("%T, %[1]v\n", t.UTC()) // UTC时间
// time.Time, 2022-10-06 11:35:00.1963422 +0000 UTC
```

m=+0.001530201为单调时间，利用的是晶体振荡器的间隔时间，很多时间函数计算都舍弃了它。如果不是非常精准的时间间隔计算，请忽略它。

时间格式化

Go语言没有采用%Y%m%d这样的格式化符号，它很特别。

```
var t = time.Now()
fmt.Printf("%T, %[1]v\n", t)
fmt.Println(t.Format("0102 030405 06 pm -0700"))
fmt.Println(t.UTC().Format("0102 030405 06 pm"))
```

记住一个字符串"010203040506pm-0700"，即 1月2日下午3点4分5秒06年西7区，改成我们习惯的格式符 2006/01/02 15:04:05 -0700，也不是特别好记，那就背吧。

```
https://golang.google.cn/pkg/time/

Year: "2006" "06"
Month: "Jan" "January" "01" "1"
Day of the week: "Mon" "Monday"
Day of the month: "2" "_2" "02"
Day of the year: "__2" "002"
Hour: "15" "3" "03" (PM or AM)
Minute: "4" "04"
Second: "5" "05"
AM/PM mark: "PM"
```

```
var t = time.Now()
fmt.Printf("%T, %[1]v\n", t)
fmt.Println(t.Format("2006/01/02 15:04:05 -0700")) // 带时区
fmt.Println(t.UTC().Format("2006/01/02 15:04:05")) // 不带时区
```

时间解析

```go
if t, err := time.Parse(
    "2006/01/02 15:04:05 -0700", // 格式字符串
    "2008/09/08 20:36:50 +0800", // 时间字符串
); err == nil {
    fmt.Println(t) // 2008-09-08 20:36:50 +0800 CST
}
```

时间属性

```go
// 月份的定义
type Month int

const (
    January Month = 1 + iota
    February
    March
    April
    May
    June
    July
    August
    September
    October
    November
    December
)
// 可以看出月份是1到12的整数

type Weekday int

const (
    Sunday Weekday = iota
    Monday
    Tuesday
    Wednesday
    Thursday
    Friday
    Saturday
)
```

```go
if t, err := time.Parse(
    "2006/01/02 15:04:05 -0700", // 格式字符串
    "2008/09/08 20:36:50 +0800", // 时间字符串
); err == nil {
    // 2008-09-08 20:36:50 +0800 CST
    // 时间戳
    fmt.Println(t.Unix(), t.UnixMilli(), t.UnixMicro(), t.UnixNano())
    // 年月日
    fmt.Println(
        t.Year(), t.Month(), t.Month().String(), // 英文月份，默认走String方法
        int(t.Month()),       // 数字月份 9
        t.Day(), t.YearDay(), // YearDay本年的第几天
```

```
        t.Weekday(),
    )
    // 时分秒
    fmt.Println(t.Hour(), t.Minute(), t.Second(), t.Nanosecond()) //
Nanosecond纳秒
    // 星期
    fmt.Println(t.Weekday(), int(t.Weekday())) // Weekday
    fmt.Println(t.ISOWeek())                   // 年的第几周
}
```

小数部分匹配，可以使用0或者9，

```
time.Parse(
        "2006/01/02 15:04:05.000000 -0700", // 格式字符串
        "2008/09/08 20:36:50.123456 +0800", // 时间字符串
    )

time.Parse(
        "2006/01/02 15:04:05.9 -0700",      // 格式字符串
        "2008/09/08 20:36:50.123456 +0800", // 时间字符串
    )

0需要和小数部分个数匹配，9不需要
```

时区

```
if t, err := time.Parse(
    "2006/01/02 15:04:05", // 格式字符串
    "2008/09/08 20:36:50", // 时间字符串
); err == nil {
    fmt.Println(t)       // 2008-09-08 20:36:50 +0000 UTC
    fmt.Println(t.Local()) // 2008-09-09 04:36:50 +0800 CST
}
```

如果没有时区，表示UTC，可以简单认为是零时区时间。

注意，这里可能导致时间错误，给出的时间，心里想的是东八区的时间，但是Go语言却认为是零时区的，如果再转换到东八区，就差了8个小时了。

```
tz, _ := time.LoadLocation("Asia/Shanghai") // 使用名字
if t, err := time.ParseInLocation(
    "2006/01/02 15:04:05", // 格式字符串
    "2008/09/08 20:36:50", // 时间字符串
    tz,
); err == nil {
    fmt.Println(t)         // 2008-09-08 20:36:50 +0800 CST
    fmt.Println(t.Local()) // 2008-09-08 20:36:50 +0800 CST
}
```

时间运算

```
时间 + 时间 = ?
时间 - 时间 = 时间差、时间增量
时间 ± 时间增量 = 时间
```

```go
tz, _ := time.LoadLocation("Asia/Shanghai") // 使用名字
s1 := "2022/09/08 20:36:50"
s2 := "2022/09/08 21:40:51"
t1, _ := time.ParseInLocation("2006/01/02 15:04:05", s1, tz)
t2, _ := time.ParseInLocation("2006/01/02 15:04:05", s2, tz)
fmt.Println(t1)
fmt.Println(t2)
```

```go
// 时间差
delta := t2.Sub(t1) // t2 - t1
fmt.Printf("delta: %v, %[1]T\n", delta) // Duration类型
fmt.Println(delta.Seconds()) // 共差多少秒

// 构造Duration
ns3 := time.Duration(3)              // 3纳秒
s3 := time.Duration(3 * time.Second) // 3秒
h3 := time.Duration(3 * time.Hour)   // 3小时
fmt.Println(ns3, s3, h3)
```

```go
// 时间偏移
t3 := t2.Add(h3)
fmt.Println(t3)
t4 := t2.Add(-h3)
fmt.Println(t4)

fmt.Println(t3.After(t4)) // t3是否在t4之后吗？true
```