Go Çalımları

Oğuzhan Yılmaz
Gopher

About me

- Developer > Sysop > Developer
- MaestroPanel - Web Hosting Control Panel
- Bloket - Deep Packet Inspector
- Masomo - HeadBall2, Basketball Arena
- Mage Games - ...

custom iota

```
package main

type Level byte

func (l Level) Name() string {
    return [...]string{"Gold", "Silver", "Platinum"}[l]
}

const (
    Gold Level = iota
    Silver
    Platinum
)

func main() {
    println(Gold.Name())
    println(Silver.Name())
    println(Platinum.Name())
}
```

Run

bitmask iota

```
type States uint8

const (
    Jump     States = 1 << iota // 1 << 0 00000001
    Freeze                      // 1 << 1 00000010
    Fire                        // 1 << 2 00000100
    IsGround                    // 1 << 3 00001000
)

func main() {
    input := Jump | Fire // 00000101

    fmt.Printf("%d=%08b\n", input, input)

    if input&Jump == 1 {
        fmt.Println("Jump is set")
    }

    if input&IsGround != 0 {
        fmt.Println("Ground is set")
    }
}
```

Run

thread safe map

```
func main() {
    var list = struct {
        sync.RWMutex
        m map[string]int
    }{m: map[string]int{"a": 1, "b": 2, "c": 3}}

    go func() {
        list.RLock()
        println("rlock a: ", list.m["a"])
        list.RUnlock()
    }()

    go func() {
        list.Lock()
        list.m["a"] = 5
        list.Unlock()
    }()

    fmt.Scanf("...")
}
```

Run

embed

```
var data []byte

func init() {
    var err error
    data, err = os.ReadFile("data.txt")
    if err != nil {
        panic(err)
    }
}
```

```
package main

import _ "embed"

//go:embed data.txt
var data []byte

func main() {}
```

empty struct

```
package main

import "os"

type State struct{}

func (State) Load() (any, error) {
    return os.ReadFile("game.dat")
}

func (State) Save(s any) error {
    return os.WriteFile("game.dat", s.([]byte), 0644)
}

func main() {
    var stor = State{}
    stor.Save([]byte("state-data"))
}
```

unkeyed fields

```go
package main

import (
    "fmt"
)

type Character struct {
    Level  int16
    Health int16
    _      struct{}
}

func main() {

    //c := Character{1, 100}
    c := Character{Level: 1, Health: 100}
    fmt.Printf("%+v\n", c)
}
```

Run

non-comparable struct

```
package main

type Character struct {
    _      [0]func()
    Level  float64
    Health float64
}

func main() {
    c1 := Character{Level: 1, Health: 100}
    c2 := Character{Level: 1, Health: 100}
    println(c1 == c2)
}
```

Run

field alignment

```go
type Character1 struct {
    isAlive  bool    // 1 byte   (7 byte)
    Health   int64   // 8 bytes  (0 byte)
    isBanned bool    // 1 bytes  (7 byte)
    Name     string  // 16 bytes (0 byte)
    Coin     float32 // 4 bytes  (4 byte)
}

type Character2 struct {
    Name     string  // 16 bytes (0 byte)
    Health   int64   // 8 bytes  (0 byte)
    Coin     float32 // 4 bytes  (2 byte)
    isAlive  bool    // 1 byte
    isBanned bool    // 1 bytes
}

func main() {
    c1, c2 := Character1{}, Character2{}
    println("Character 1:", unsafe.Sizeof(c1))
    println("Character 2:", unsafe.Sizeof(c2))
}
```

Run

interface embedding

```
package main

type Jumper interface{ Jump() }
type Shooter interface{ Shot() }

type Character interface {
    Jumper
    Shooter
}

type Rocket struct{}

func (Rocket) Jump() { println("jumping") }
func (Rocket) Shot() { println("shotting") }

func main() {
    var char Character = &Rocket{}
    char.Jump()
    char.Shot()
}
```

Run

anonymous interface

```
package main

import "encoding/base64"

func main() {
    var data = []byte("hello")

    var encoder interface {
        EncodeToString([]byte) string
    } = base64.StdEncoding

    encoded := encoder.EncodeToString(data)
    print(encoded)
}
```

Run

auto-check interface

```
package main

type Character interface {
    Level(int)
}

type Rocket struct{}

func (c *Rocket) Level(l int) { println("Level:", l) }

var _ Character = (*Rocket)(nil)

func main() {
    rocket := new(Rocket)

    if c, ok := any(rocket).(Character); ok {
        c.Level(10)
    }
}
```

Run

two-stage defer

```go
package main

import (
    "fmt"
    "time"
)

func metric() func() {
    begin := time.Now()

    return func() {
        end := time.Now().Sub(begin)
        fmt.Printf("execution time: %v", end)
    }
}

func main() {
    defer metric()()
    time.Sleep(time.Second)
}
```

Run

build arguments

```
go build -ldflags "-X main.version=1.0.0 -X main.buildTime=202402171520" .
```

```
package main

var (
    version   string
    buildTime string
)

func main() {
    println("Version:", version)
    println("Build Time:", buildTime)
}
```

GODEBUG=gctrace=1 go run .

```
gc 1          : First GC run since program started.
@0.009s       : Nine milliseconds since the program started.
1%            : One percent of the programs time has been spent in GC.

// wall-clock
0.059ms       : **STW** Sweep termination.
0.17ms        : Mark/Scan - Assist Time (GC performed in line with allocation).
0.005ms       : Mark/Scan - Background GC time.
0.24ms        : Mark/Scan - Idle GC time.
0.12ms        : **STW** Mark termination.

// CPU time
0.17ms        : **STW** Sweep termination.
0.17+0+0ms    : Mark/Scan - Assist Time (GC performed in line with allocation).
0.36ms        : Mark/Scan - Background GC time.
0.067ms       : Mark/Scan - Idle GC time.
0.38ms        : **STW** Mark termination.

5MB           : Heap size at GC start.
5MB           : Heap size at GC end.
3MB           : Live Heap.
4MB           : Goal heap size.
8P            : Number of logical processors.
```

GODEBUG=inittrace=1 go run .

```
init encoding/binary @0.82 ms, 0.002 ms clock, 16 bytes, 1 allocs
init math @0.32 ms, 0 ms clock, 0 bytes, 0 allocs
init strconv @0.36 ms, 0.006 ms clock, 32 bytes, 2 allocs
init sync @0.40 ms, 0.003 ms clock, 16 bytes, 1 allocs
init unicode @0.44 ms, 0.065 ms clock, 23320 bytes, 24 allocs
init reflect @0.55 ms, 0.001 ms clock, 0 bytes, 0 allocs
init io @0.59 ms, 0.002 ms clock, 144 bytes, 9 allocs
```

GOMEMLIMIT

```
GOMEMLIMIT=24MiB
```

private repo

```
GOPRIVATE=*.gokonf.com
```

```
package main

import "gokonf.com/backend/math"

func main() {
    println(math.Sum(1, 2))
}
```

Thank you

Oğuzhan Yılmaz
Gopher
@c1982