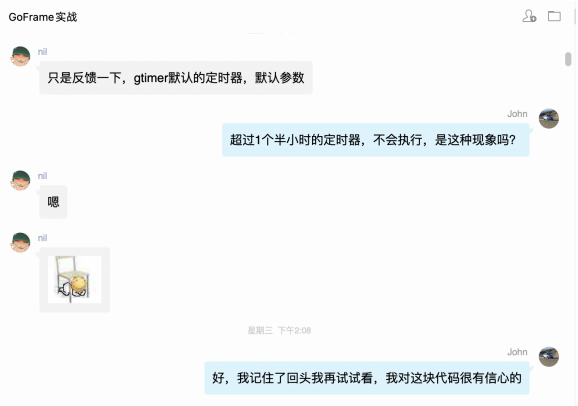
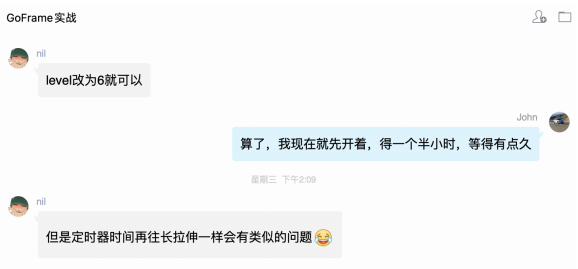
GoFramegtimerbug

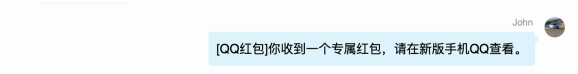
Author

Content Menu

QQ







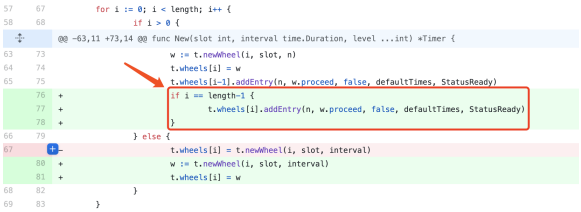
QQ...5

`gtimerbugbug`

https://github.com/gogf/gf/commit/93a648ba15c5cfa45c856f4b26316c44dcb59d34

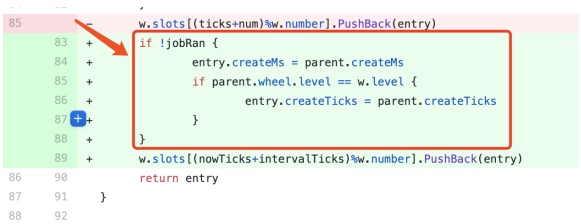
> bug

1



```
@@ -63,11 +73,14 @@ func New(slot int, interval time.Duration, level ...int) *Timer {
                        w := t.newWheel(i, slot, n)
                        t.wheels[i] = w
                        t.wheels[i-1].addEntry(n, w.proceed, false, defaultTimes, StatusReady)
+                       if i == length-1 {
+                               t.wheels[i].addEntry(n, w.proceed, false, defaultTimes, StatusReady)
+                       }
                } else {
-                       t.wheels[i] = t.newWheel(i, slot, interval)
                        w := t.newWheel(i, slot, interval)
                        t.wheels[i] = w
                }
        }
```

2""""""



```
-       w.slots[(ticks+num)%w.number].PushBack(entry)
+       if !jobRan {
+               entry.createMs = parent.createMs
+               if parent.wheel.level == w.level {
+                       entry.createTicks = parent.createTicks
+               }
+       }
+       w.slots[(nowTicks+intervalTicks)%w.number].PushBack(entry)
        return entry
}
```

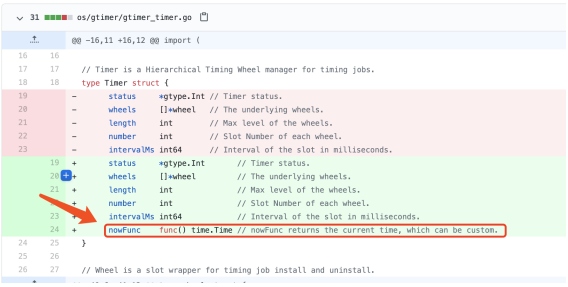
`gtimer` -gtimer ""**SLOT+**SLOTSLOT""`createMscreateTicks`

`gtimerbug`

1

`bug`

https://github.com/gogf/gf/commit/93a648ba15c5cfa45c856f4b26316c44dcb59d34



```
31 os/gtimer/gtimer_timer.go
@@ -16,11 +16,12 @@ import (

  // Timer is a Hierarchical Timing Wheel manager for timing jobs.
  type Timer struct {
-       status     *gtype.Int // Timer status.
-       wheels     []*wheel   // The underlying wheels.
-       length     int        // Max level of the wheels.
-       number     int        // Slot Number of each wheel.
-       intervalMs int64      // Interval of the slot in milliseconds.
+       status     *gtype.Int       // Timer status.
+       wheels     []*wheel         // The underlying wheels.
+       length     int              // Max level of the wheels.
+       number     int              // Slot Number of each wheel.
+       intervalMs int64            // Interval of the slot in milliseconds.
+       nowFunc    func() time.Time // nowFunc returns the current time, which can be custom.
  }

  // Wheel is a slot wrapper for timing job install and uninstall.
```

```
// Copyright GoFrame Author(https://goframe.org). All Rights Reserved.
//
// This Source Code Form is subject to the terms of the MIT License.
// If a copy of the MIT was not distributed with this file,
// You can obtain one at https://github.com/gogf/gf.

package gtimer

import (
        "github.com/gogf/gf/container/gtype"
        "github.com/gogf/gf/test/gtest"
        "testing"
        "time"
)

func TestTimer_Proceed(t *testing.T) {
        gtest.C(t, func(t *gtest.T) {
                index := gtype.NewInt()
                slice := make([]int, 0)
                timer := doNewWithoutAutoStart(10, 60*time.Millisecond, 6)
                timer.nowFunc = func() time.Time {
                        return time.Now().Add(time.Duration(index.Add(1))
* time.Millisecond * 60)
                }
                timer.AddOnce(2*time.Second, func() {
                        slice = append(slice, 1)
                })
                timer.AddOnce(1*time.Minute, func() {
                        slice = append(slice, 2)
                })
                timer.AddOnce(5*time.Minute, func() {
                        slice = append(slice, 3)
                })
                timer.AddOnce(1*time.Hour, func() {
                        slice = append(slice, 4)
                })
                timer.AddOnce(100*time.Minute, func() {
                        slice = append(slice, 5)
                })
                timer.AddOnce(2*time.Hour, func() {
                        slice = append(slice, 6)
                })
                timer.AddOnce(1000*time.Minute, func() {
                        slice = append(slice, 7)
                })
                timer.AddOnce(1100*time.Minute, func() {
                        slice = append(slice, 8)
                })
                timer.AddOnce(1200*time.Minute, func() {
                        slice = append(slice, 9)
                })
                for i := 0; i < 2000000; i++ {
                        timer.wheels[0].proceed()
                        time.Sleep(10 * time.Microsecond)
                }
                time.Sleep(time.Second)
                t.Assert(slice, []int{1, 2, 3, 4, 5, 6, 7, 8, 9})
        })
}
```

2

3

buggtimer

```go
// addEntryByParent adds a timing job with parent entry.
func (w *wheel) addEntryByParent(interval int64, parent *Entry) *Entry {
        num := interval / w.intervalMs
        if num == 0 {
                num = 1
        }
        nowMs := time.Now().UnixNano() / 1e6
        ticks := w.ticks.Val()
        entry := &Entry{
                wheel:          w,
                job:            parent.job,
                times:          parent.times,
                status:         parent.status,
                create:         ticks,
                interval:       num,
                singleton:      parent.singleton,
                createMs:       nowMs,
                intervalMs:     interval,
                rawIntervalMs: parent.rawIntervalMs,
        }
        w.slots[(ticks+num)%w.number].PushBack(entry)
        return entry
}
```

num是什么鬼?

WTF?

WTF?

numticksCreateintervalEntry

```
// Entry is the timing job entry to wheel.
type Entry struct {
+       name          string
        wheel         *wheel      // Belonged wheel.
        job           JobFunc     // The job function.
        singleton     *gtype.Bool // Singleton mode.
        status        *gtype.Int  // Job status.
        times         *gtype.Int  // Limit running times.
-       create        int64       // Timer ticks when the job installed.
-       interval      int64       // The interval ticks of the job.
+       intervalTicks int64       // The interval ticks of the job.
+       createTicks   int64       // Timer ticks when the job installed.
        createMs      int64       // The timestamp in milliseconds when job installed.
        intervalMs    int64       // The interval milliseconds of the job.
-       rawIntervalMs int64       // Raw input interval in milliseconds.
}
```

```
+ func (w *wheel) addEntryByParent(jobRan bool, nowMs, interval int64, parent *Entry) *Entry {
+       intervalTicks := interval / w.intervalMs
+       if intervalTicks == 0 {
+               intervalTicks = 1
        }
-       nowMs := time.Now().UnixNano() / 1e6
-       ticks := w.ticks.Val()
+       nowTicks := w.ticks.Val()
        entry := &Entry{
+               name:          parent.name,
                wheel:         w,
                job:           parent.job,
                times:         parent.times,
                status:        parent.status,
-               create:        ticks,
-               interval:      num,
+               intervalTicks: intervalTicks,
                singleton:     parent.singleton,
+               createTicks:   nowTicks,
                createMs:      nowMs,
                intervalMs:    interval,
-               rawIntervalMs: parent.rawIntervalMs,
        }
```

根据时间计算出的时间轮刻度值

当前时间轮的刻度值

1.
2.
3. CI