# Поведение for range с каналами в Go

В Go `for v := range ch` используется для последовательного чтения из канала `ch`. Цикл завершится **только** после того, как:

- канал будет **закрыт**, и
- все значения из него будут **прочитаны**

---

# □□ Сценарии поведения

## 1. Канал пуст и **не закрыт**

```
ch := make(chan string)

for v := range ch {
    fmt.Println(v)
}
```

□□ Итог: **блокировка навсегда**, цикл ждёт значения, но никто не пишет.

---

## 2. В канале есть одно значение, канал **не закрыт**

```
ch := make(chan string)

go func() {
    ch <- "hello"
}()
```

```
for v := range ch {
    fmt.Println(v)
}
```

Итог: `hello` выведется, затем цикл **зависнет**, ожидая следующего значения.

---

# 3. В канале есть значение, потом ещё одно, затем закрытие

```
ch := make(chan string)

go func() {
    ch <- "one"
    time.Sleep(1 * time.Second)
    ch <- "two"
    close(ch)
}()

for v := range ch {
    fmt.Println(v)
}
```

Итог:

- `one` будет прочитан
- цикл подождёт
- `two` будет прочитан
- после `close(ch)` — цикл завершится

---

# Правильное завершение горутины через канал и WaitGroup

```
import (
    "fmt"
    "sync"
)

func worker(id int, jobs <-chan int, wg *sync.WaitGroup) {
    defer wg.Done()
    for job := range jobs {
        fmt.Printf("Worker %d processing job %d\n", id, job)
    }
}

func main() {
    jobs := make(chan int)
    var wg sync.WaitGroup

    for w := 1; w <= 2; w++ {
        wg.Add(1)
        go worker(w, jobs, &wg)
    }

    for j := 1; j <= 5; j++ {
        jobs <- j
    }

    close(jobs)  // □□важно закрыть канал
    wg.Wait()    // □□ждём завершения всех воркеров
}
```

---

# □□ Общее поведение `range` по каналам

| Сценарий | Поведение |
|---|---|
| Канал пуст и не закрыт | □ Блокировка |
| Есть 1 значение, канал не закрыт | □ Прочтёт 1, потом зависнет |

| Сценарий | Поведение |
| --- | --- |
| Есть значения, канал закрыт после них |  Прочтёт все, завершит цикл |
| Канал закрыт сразу |  Завершит цикл (если пуст) |

---

# □□ Рекомендации

- □ **Закрывай канал**, если больше не будет отправок
- □ Используй `sync.WaitGroup`, чтобы ждать завершения горутин
- ⚠ Избегай чтения из **открытых, но не используемых** каналов

---

Revision #1
Created 2025-07-14 10:47:35 UTC by Admin
Updated 2025-07-14 10:47:49 UTC by Admin