Go channel 拷贝问题

作者：Allenxuxu
原文链接：https://ld246.com/article/1566389261378
来源网站：链滴

Go 的 channel 使用非常方便，但是总听说 channel 会拷贝传递的数据，生怕频繁拷贝影响效率。

究竟是怎么个拷贝法呢，下面会有两个 demo 验证下。

先说结论：

Go channel 的发送接收数据的拷贝和 Go 的函数传参道理是一样的，都是默认的值拷贝。

如果你传递一个值，那么 Go 会复制一份新的；如果传递一个指针，则会拷贝这个指针，不会去拷贝个指针所指的变量（这一点 C++ 选手可能会理解比较深）。

所以，如果你需要通过 channel 传递一个很大的 struct ，那么应该传递 指针。但是，要非常注意通过 channel 发送后，不要修改这个指，这会导致线程间潜在的竞争。

下面是两个验证的小 demo：

- 通过 channel 传递指针

```
package main

import (
    "fmt"
    "time"
)

func recv(ch <-chan *int) {
    time.Sleep(1 * time.Second)

    out := <-ch
    fmt.Println("recv : ", out, *out)
}
```

```
func main() {
    i := 1
    ch := make(chan *int, 2)

    fmt.Println("i    : ", &i, i)
    go recv(ch)
    ch <- &i

    i = 2

    time.Sleep(2 * time.Second)
    fmt.Println("i    : ", &i, i)
}
```

输出：

```
i    :  0xc000084000 1
recv :  0xc000084000 2
i    :  0xc000084000 2
```

上面的代码通过 channel 发送了 *int 的数据，在接收的协程中先 sleep 1 秒钟让别的协程去更改传的值。

从打印结果可以看出，通过 channel 接收的数据，只是拷贝了对象的地址而已。

- 通过 channel 传递值

```
package main

import (
    "fmt"
    "time"
)

func recv(ch <-chan int) {
    time.Sleep(1 * time.Second)

    out := <-ch
    fmt.Println("recv : ", &out, out)
}

func main() {
    i := 1
    ch := make(chan int, 2)

    fmt.Println("i    : ", &i, i)
    go recv(ch)
    ch <- i

    i = 2

    time.Sleep(2 * time.Second)
    fmt.Println("i    : ", &i, i)
```

```
}
```

输出：

```
i    : 0xc00008e000 1
recv :  0xc00007e008 1
i    : 0xc00008e000 2
```