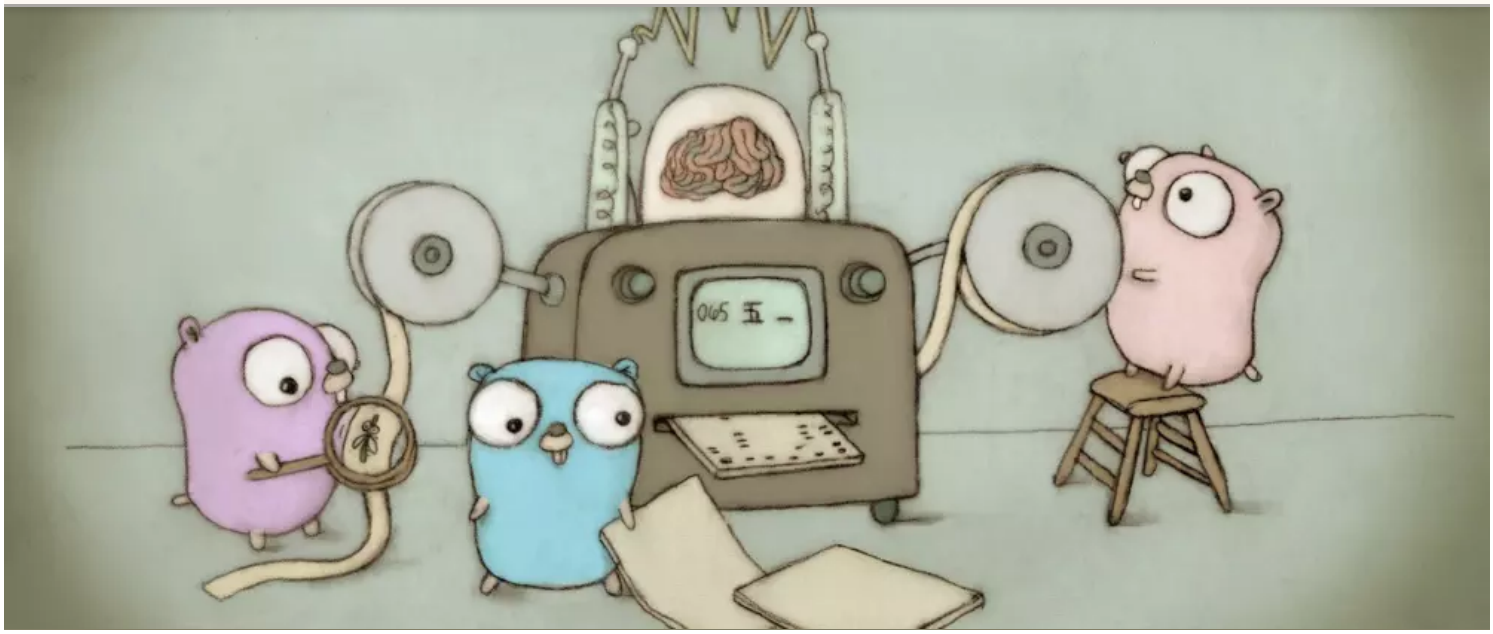

Call Go function from C function

Yasuhiro Matsumoto Nov 17 '17 · 3 min read

#go

Cgo is useful to embed C function into Go.

```
package main

/*
void doSomething(int *p) {
  *p = 123;
}
*/
import "C"

func main() {
    var n C.int
    C.doSomething(&n)
    println(n)
```

no problem. So in your first look, you might think you will not met any issue in future.

Please imagine the case when you want to call APIs which take a callback function and pointer of user_data.

APIs

```
typedef void (*callback)(void *);
void register_callback(callback cb, void *user_data);
void wait_event();
```

As you guess, `register_callback` registers a callback function, `wait_event` waits for some event, and calls the callback function when an event occured.

Go

```
func my_callback(v unsafe.Pointer) {
}

func main() {
    user_data := "my userdata"
    C.register_event(/* ? */, /* ? */)
    C.wait_event()
}
```

`register_event`? Unfortunately, type of my_callback in Go is not same as C.callback. In short, you can't write below.

```
C.register_event(my_callback, &user_data) // compilation errors
```

To make a function which is possible to be called from C, hack is required.

- export Go function to be called from C
- pass valid pointer from Go

To export Go function, put comment line `//export FuncName` above function.

```
//export hello
func hello() {
    // do something
}
```

Wow this is simplify. Thus, it is possible make callback proxy.

```
package main

/*
#include <myapi.h>
```

```
}
*/
import "C"
import (
    "fmt"
)

func my_callback(v string) {
    fmt.Println("hello", v)
}

func main() {
    C._register_callback(...)
    C.wait_event()
}

//export cb_proxy
func cb_proxy(v unsafe.Pointer) {
    // call my_callback
}
```

`_register_callback` is wapper of `register_callback` to call `cb_proxy`. `wait_event` will invoke callback internally with taking argument user_data. user_data should be passed from `main`. However, one another issue you meet.

Rules for passing pointers between Go and C

Unfortunately again, Go can't pass pointer which is allocated in Go into C function.

But that's possible. See https://github.com/mattn/go-pointer.

```
var s string
C.pass_pointer(pointer.Save(&s))
v := *(pointer.Restore(C.get_from_pointer()).(*string))
```

As the trick, go-pointer allocate 1-byte dummy memory for passing to C. And it is pointed to the value. i.e. this is unique key of map which is related on the real Go pointer.
`pointer.Save(&v)` store Go pointer into the map and return C pointer which is allocated as dummy.

```
package main

/*
#include <myapi.h>

void cb_proxy(void *v);

static void _register_callback(void *user_data) {
  register_callback(cb_proxy, user_data);
}
*/
import "C"
import (
    "fmt"
    "unsafe"

    "github.com/mattn/go-pointer"
)

type Callback struct {
    Func     func(string)
    UserData string
}
```

```
func main() {
    C._register_callback(pointer.Save(&Callback{
        Func:     my_callback,
        UserData: "my-callback",
    }))
    C.wait_event()
}

//export cb_proxy
func cb_proxy(v unsafe.Pointer) {
    cb := pointer.Restore(v).(*Callback)
    cb.Func(cb.UserData)
}
```

`_register_callback` pass `cb_proxy`. And `cb_proxy` restore the poinetr as `*Callback`. And it call original `Callback.Func` with `Callback.UserData`. You can pass callback function to C from Go.

You can try this behavior on this code.

```
package main

/*
typedef void (*callback)(void *);

static callback _cb;
static void *_user_data;
static void register_callback(callback cb, void *user_data) {
    _cb = cb;
    _user_data = user_data;
```

```
void cb_proxy(void *v);

static void _register_callback(void *user_data) {
  register_callback(cb_proxy, user_data);
}
*/
import "C"
import (
    "fmt"
    "unsafe"

    "github.com/mattn/go-pointer"
)

type Callback struct {
    Func     func(string)
    UserData string
}

func my_callback(v string) {
    fmt.Println("hello", v)
}

func main() {
    C._register_callback(pointer.Save(&Callback{
        Func:     my_callback,
        UserData: "my-callback",
    }))
    C.wait_event()
}

//export cb_proxy
func cb_proxy(v unsafe.Pointer) {
    cb := pointer.Restore(v).(*Callback)
    cb.Func(cb.UserData)
}
```

DEV search

@mattn mattn_jp mattn mattn.kaoriya.net

Add to the discussion

PREVIEW SUBMIT

code of conduct - report abuse

Classic DEV Post from Dec 25 '18

5 Books Which Will Improve Your Career

Emma Wedekind ✨

5 Books Which Will Change Your Career

370 17

Another Post You Might Like

Tracking Service with Go and Redis

Douglas Makey Mendez Molero

Tracking Service

75 4

23 2 2

End Users

Nick Parsons

Migrating from Python to Go for 300+ Million End Users

98 4

A Conservative Backend

Matt Reyer - Mar 14

Testing is Not for Beginners

Jon Calhoun - Mar 13

Functional Programming in Go With dcode

Aaron Schlesinger - Mar 11

You think you understand key-value pairs

Ishan Khare - Mar 8