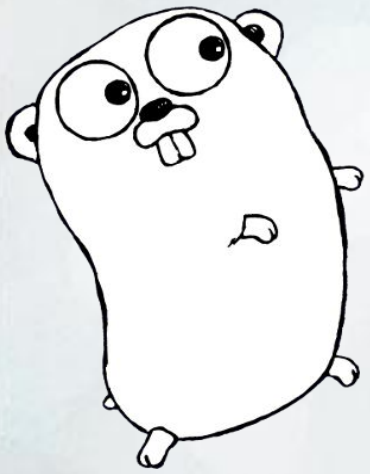
GO语言中国用户组 http://golang-china.org/

GO集成C/C++代码

chaishushan@gmail.com



内容大纲

- **如何编写包**
  - **包代码**
  - **Makefile文件**
  - **使用包**
- 用CGO工具编写包
  - 包代码
  - Makefile文件
  - cgo语法
- 用SWIG工具编写包
- 相关资源

如何编写包-HELLO

// file: ./org.golang-china.sz/**hello/hello.go**

```
package hello

import "fmt"

func PrintHello() {
  fmt.Printf("Hello, 世界\n")
}
```

如何编写包-MAKEFILE文件

```
# file: ./org.golang-china.sz/hello/Makefile

include $(GOROOT)/src/Make.inc

TARG=org.golang-china.sz/hello
GOFILES=hello.go

include $(GOROOT)/src/Make.pkg
```

如何编写包-使用包

```
// file: ./org.golang-china.sz/hello/helloapp.go

package main

import "org.golang-china.sz/hello"

func main() {
    print("hello,")
    hello.PrintHello()
}
```

如何编写包-完善MAKEFILE

```
# file: ./org.golang-china.sz/hello/Makefile

include $(GOROOT)/src/Make.inc

TARG=mypkg/hello
GOFILES=hello.go

include $(GOROOT)/src/Make.pkg

# Simple test programs
%: install %.go
   $(GC) $*.go
   $(LD) -o $@ $*.$O
```

如何编写包-编译运行

cd org.golang-china.sz/hello
make nuke
make helloapp
./helloapp

也可以基于gotest编写包测试，详细信息可参考文档。

内容大纲

- 如何编写包
  - 包代码
  - Makefile文件
  - 使用包
- **用CGO工具编写包**
  - **包代码**
  - **Makefile文件**
  - **cgo语法**
- 用SWIG工具编写包
- 相关资源

用C语言编写包-工具

- GC中的**CGO**支持用**C语言**开发包
- **GCCGO**支持链接C/C++开发的动态库(?)
- **SWIG-2.0.1**开始支持GO语言(GC和GCCGO)
- 最新进展:**Roadmap**和**Release History**

用C语言编写包-代码

```
//file: ./org.golang-china.sz/hello2/hello.go

package hello2

/*
// 这行是注释
#include <stdio.h>
*/
import "C"

func PrintHello() {
  C.puts(C.CString("Hello, world\n"))
}
```

用C语言编写包-MAKEFILE

```
# file: ./org.golang-china.sz/hello2/Makefile

include $(GOROOT)/src/Make.inc

TARG=org.golang-china.sz/hello2
CGOFILES=hello2.go

include $(GOROOT)/src/Make.pkg

# Simple test programs
%: install %.go
   $(GC) $*.go
   $(LD) -o $@ $*.$O
```

用C语言-导入C头文件

```
// #include <stdio.h>
// #include <errno.h>
import "C"
```

C语言中的类型和变量将通过**伪包"C"**来访问，如：

- 类型：**C.**size_t
- 变量：**C.**stdout
- 函数：**C.**putchar

用C语言编写包-导入第三方C库

```
include $(GOROOT)/src/Make.inc

TARG=pkgname
GOFILES=gofiles.go
CGOFILES=cgofiles.go

CGO_CFLAGS=-I/home/rsc/gmp32/include
CGO_LDFLAGS=-L/home/rsc/gmp32/lib -lgmp

include $(GOROOT)/src/Make.pkg
```

用C语言-代码中导入库

```
// #cgo CFLAGS: -DPNG_DEBUG=1
// #cgo linux CFLAGS: -DLINUX=1
// #cgo LDFLAGS: -lpng
// #include <png.h>
import "C"
```

通过cgo参数可以从代码设置编译参数。

用C语言-标准类型

- C.char, C.**schar**(signed char), C.**uchar**(unsigned char)
- C.short, C.ushort (unsigned short)
- C.**int**, C.**uint** (unsigned int)
- C.**long**, C.**ulong** (unsigned long)
- C.longlong(long long), C.ulonglong(unsigned long long)
- C.**float**, C.**double**

用C语言-字符串和指针

```
package stdio
/*
#include <stdio.h>
char* greeting = "hello, world";
*/
import "C"
import "unsafe"

type File C.FILE
var Greeting = C.GoString(C.greeting)

func (f *File) WriteString(s string) {
    p := C.CString(s)
    C.fputs(p, (*C.FILE)(f))
    // 字符串需要是手工释放
    C.free(unsafe.Pointer(p))
}
```

用C语言-数组和指针

数组转为指针是必须先取第一个元素的地址，然后作类型转换：

```
// c-life.c
void Step(int x, int y, int* a, int* n) {
    // ...
}

// life.go
func Run(gen, x, y int, a []int) {
    n := make([]int, x*y)
    for i := 0; i < gen; i++ {
        C.Step(C.int(x), C.int(y),
            (*C.int)(unsafe.Pointer(&a[0])),
            (*C.int)(unsafe.Pointer(&n[0])))
        copy(a, n)
    }
}
```

用C语言-STRUCT/UNION/ENUM/宏

```
/*
#define MYCONST 1024
typedef struct { int s1; int s1; } Struct1;
typedef union { int u2; int u2; } Union1;
typedef enum { E1 = 0x0000, E2 = 0x8000 } Enum1;
*/
import "C"

func Foo(size C.CvSize) {
    var myStruct C.Struct1
    var myUnion  C.Union1
    var myEnum   C.Enum1

    myStruct.s1 = C.MYCONST
    myUnion.u1 = C.(20)
    myEnum = C.E1
}
```

用C语言-define的限制

https://code.google.com/p/go/source/detail?r=bcbfbe066b

Log message

cgo: Only allow **numeric / string / character type constants** for references to **#defined** things.

Fixes issue 520 .

R=rsc, rsaarelm
CC=golang-dev
http://codereview.appspot.com/186138

Committer: Russ Cox <rsc@golang.org>

golang-nuts讨论：CGO and #define

用C语言-_cgo_export.h

```
/* Created by cgo - DO NOT EDIT. */

typedef unsigned int uint;
typedef signed char schar;
typedef unsigned char uchar;
typedef unsigned short ushort;
typedef long long int64;
typedef unsigned long long uint64;
typedef __SIZE_TYPE__ uintptr;

typedef struct { char *p; int n; } GoString;
typedef void *GoMap;
typedef void *GoChan;
typedef struct { void *t; void *v; } GoInterface;
```

用C语言-函数参数和返回值

在返回函数值同时可以返回error值：

```
n, err := C.atoi("abc")
if err != nil {
    return err
}
```

不支持参数可变顶C函数，如printf；
详情见(Issue975)：
http://code.google.com/p/go/issues/detail?id=975

用C语言-C回调GO函数

life.go

```
//export GoStart
// Double return value is just for testing.
func GoStart(x, y C.int) (int, int) {
    return int(0), int(100)
}
```

是否支持从另一个C线程回调GO函数?

c-life.c

```
#include "_cgo_export.h"
void Step(int x, int y)
{
    // GO函数多个返回值
    struct GoStart_return r;
    r = GoStart(x, y);
    assert(r.r0 == 0 && r.r1 == 100);
}
```

用C语言例子:GO-GTK

https://github.com/mattn/go-gtk

内容大纲

- 如何编写包
  - 包代码
  - Makefile文件
  - 使用包
- 用CGO工具编写包
  - 包代码
  - Makefile文件
  - cgo语法
- **用SWIG工具编写包**
- 相关资源

用SWIG编写包-C代码

```
/* File : example.c */
/* A global variable */
double Foo = 3.0;

int gcd(int x, int y) {
    int g = y;
    while (x > 0) {
        g = x;
        x = y % x;
        y = g;
    }
    return g;
}
```

用SWIG编写包-接口文件

```
/* File: example.i */
%module example

extern int gcd(int x, int y);
extern double Foo;
```

用SWIG编写包-演示代码

```
/* File: runme.go */
package main

import "fmt"
import "./example"

func main() {
    // Call our gcd() function
    x, y := 42, 105
    g := example.Gcd(x, y)
    fmt.Println("The gcd of", x, "and", y, "is", g)

    fmt.Println("Foo =", example.GetFoo())
    example.SetFoo(3.1415926)
    fmt.Println("Foo =", example.GetFoo())
}
```

用SWIG编写包-Makefile

```
include $(GOROOT)/src/Make.inc

TARG=pkgname
SWIGFILES=example.i

include $(GOROOT)/src/Make.pkg
```

用SWIG编写包-编译过程

1. 运行 **swig -go example.i**，生成3个文件：**example.go, example_gc.c, example_wrap.c**
2. 编译 example.go：**6g example.go**
3. 编译 example_gc.c：**6c example_gc.c**
4. 将以上2个目标文件打包为 example.a：**gopack grc example.a example.6 example_gc.6**
5. 编译 example_wrap.c（-fpic选项生存位置无关代码）：**gcc -c -O -fpic example_wrap.c**
6. 编译原始的C代码：**gcc -c -O -fpic example.c**
7. 将gcc生存顶目标文件做成共享库：**gcc -shared -o example.so example_wrap.o example.o**
8. 编译GO代码：**6g runme.go**
9. 链接：**6l -o runme runme.6**

用SWIG编写包-GC编译器

% swig -go interface.i
% gcc -fpic -c interface_wrap.c
% gcc -shared interface_wrap.o $(OBJS) -o nterfacemodule.so
% 6g interface.go
% 6c interface_gc.c
% gopack grc interface.a interface.6 interface_gc.6
% 6l program.6

用SWIG编写包-GCCGO编译器

```
% swig -go interface.i
% gcc -c interface_wrap.c
% gccgo -c interface.go
% gccgo program.o interface.o interface_wrap.o
```

用SWIG编写包-更多的例子

- http://www.swig.org/Doc2.0/Go.html#Go
- swig-dir/Examples/**go/index.html**

21 SWIG and Go

This chapter describes SWIG's support of Go. For more information on the Go programming language see golang.org.

21.1 Overview

Go is a compiled language, not a scripting language. However, it does not support direct calling of functions written in C/C++. The cgo
to generate wrappers to call C code from Go, but there is no convenient way to call C++ code. SWIG fills this gap.

There are (at least) two different Go compilers. One is the gc compiler, normally invoked under the names 6g, 8g, or 5g. The other is t
which is a frontend to the gcc compiler suite. The interface to C/C++ code is completely different for the two Go compilers. SWIG suppo
a command line option.

Because Go is a type-safe compiled language, SWIG's runtime type checking and runtime library are not used with Go. This should be born
reading the rest of the SWIG documentation.

21.2 Running SWIG with Go

To generate Go code, use the `-go` option with SWIG. By default SWIG will generate code for the gc compilers. To generate code for gccgo,

内容大纲

- 如何编写包
  - 包代码
  - Makefile文件
  - 使用包
- 用CGO工具编写包
  - 包代码
  - Makefile文件
  - cgo语法
- 用SWIG工具编写包
- **相关资源**

相关资源-网站

- http://golang.org/
- http://go-lang.cat-v.org/
- http://golang-china.org/
- http://go-lang.cat-v.org/library-bindings
- QQ群：**1023-1985-4**

相关资源-中文文档

- https://golang-china.googlecode.com/

Go语言文档

[Go语言中文小组] 翻译整理
2010-12-15

欢迎参与GO的文档翻译和推广!

http://golang-china.org
http://golang.org