cgo

package standard library

Version: go1.20.1 Latest | Published: Feb 14, 2023 | License: BSD-3-Clause | Imports: 5 | Imported by: 346

Details Valid go.mod file ? Redistributable license ? Tagged version ? Stable version ? Learn more

Repository cs.opensource.google/go/go

Links Report a Vulnerability

Documentation

Rendered for linux/amd64

Overview

Package cgo contains runtime support for code generated by the cgo tool. See the documentation for the cgo command for details on using cgo.

Index

type Handle
- func NewHandle(v any) Handle
- func (h Handle) Delete()
- func (h Handle) Value() any

type Incomplete

Constants

This section is empty.

Variables

This section is empty.

Functions

This section is empty.

Types

type Handle

```
type Handle uintptr
```

Handle provides a way to pass values that contain Go pointers (pointers to memory allocated by Go) between Go and C without breaking the cgo pointer passing rules. A Handle is an integer value that can represent any Go value. A Handle can be passed through C and back to Go, and Go code can use the Handle to retrieve the original Go value.

The underlying type of Handle is guaranteed to fit in an integer type that is large enough to hold the bit pattern of any pointer. The zero value of a Handle is not valid, and thus is safe to use as a sentinel in C APIs.

For instance, on the Go side:

```
package main

/*
#include <stdint.h> // for uintptr_t

extern void MyGoPrint(uintptr_t handle);
void myprint(uintptr_t handle);
*/
import "C"
import "runtime/cgo"

//export MyGoPrint
func MyGoPrint(handle C.uintptr_t) {
    h := cgo.Handle(handle)
    val := h.Value().(string)
    println(val)
    h.Delete()
}

func main() {
    val := "hello Go"
    C.myprint(C.uintptr_t(cgo.NewHandle(val)))
    // Output: hello Go
}
```

and on the C side:

```
#include <stdint.h> // for uintptr_t

// A Go function
extern void MyGoPrint(uintptr_t handle);

// A C function
void myprint(uintptr_t handle) {
    MyGoPrint(handle);
}
```

Some C functions accept a void* argument that points to an arbitrary data value supplied by the caller. It is not safe to coerce a cgo.Handle (an integer) to a Go unsafe.Pointer, but instead we can pass the address of the cgo.Handle to the void* parameter, as in this variant of the previous example:

```
package main

/*
extern void MyGoPrint(void *context);
static inline void myprint(void *context) {
    MyGoPrint(context);
}
*/
import "C"
import (
    "runtime/cgo"
    "unsafe"
)

//export MyGoPrint
func MyGoPrint(context unsafe.Pointer) {
    h := *(*cgo.Handle)(context)
    val := h.Value().(string)
    println(val)
    h.Delete()
}

func main() {
    val := "hello Go"
    h := cgo.NewHandle(val)
    C.myprint(unsafe.Pointer(&h))
    // Output: hello Go
}
```

func NewHandle

```
func NewHandle(v any) Handle
```

NewHandle returns a handle for a given value.

The handle is valid until the program calls Delete on it. The handle uses resources, and this package assumes that C code may hold on to the handle, so a program must explicitly call Delete when the handle is no longer needed.

The intended use is to pass the returned handle to C code, which passes it back to Go, which calls Value.

func (Handle) Delete

```
func (h Handle) Delete()
```

Delete invalidates a handle. This method should only be called once the program no longer needs to pass the handle to C and the C code no longer has a copy of the handle value.

The method panics if the handle is invalid.

func (Handle) Value

```
func (h Handle) Value() any
```

Value returns the associated Go value for a valid handle.

The method panics if the handle is invalid.

type Incomplete

added in go1.20

```
type Incomplete struct {
    // contains filtered or unexported fields
}
```

Incomplete is used specifically for the semantics of incomplete C types.

Source Files

View all

callbacks.go
callbacks_traceback.go
cgo.go
handle.go
iscgo.go
linux.go
mmap.go
setenv.go
sigaction.go

Why Go

Use Cases

Case Studies

Get Started

Playground

Tour

Stack Overflow

Help

Packages

Standard Library

About Go Packages

About

Download

Blog

Issue Tracker

Release Notes

Brand Guidelines

Code of Conduct

Connect

Twitter

GitHub

Slack

r/golang

Meetup

Golang Weekly

Copyright

Terms of Service

Privacy Policy

Report an Issue