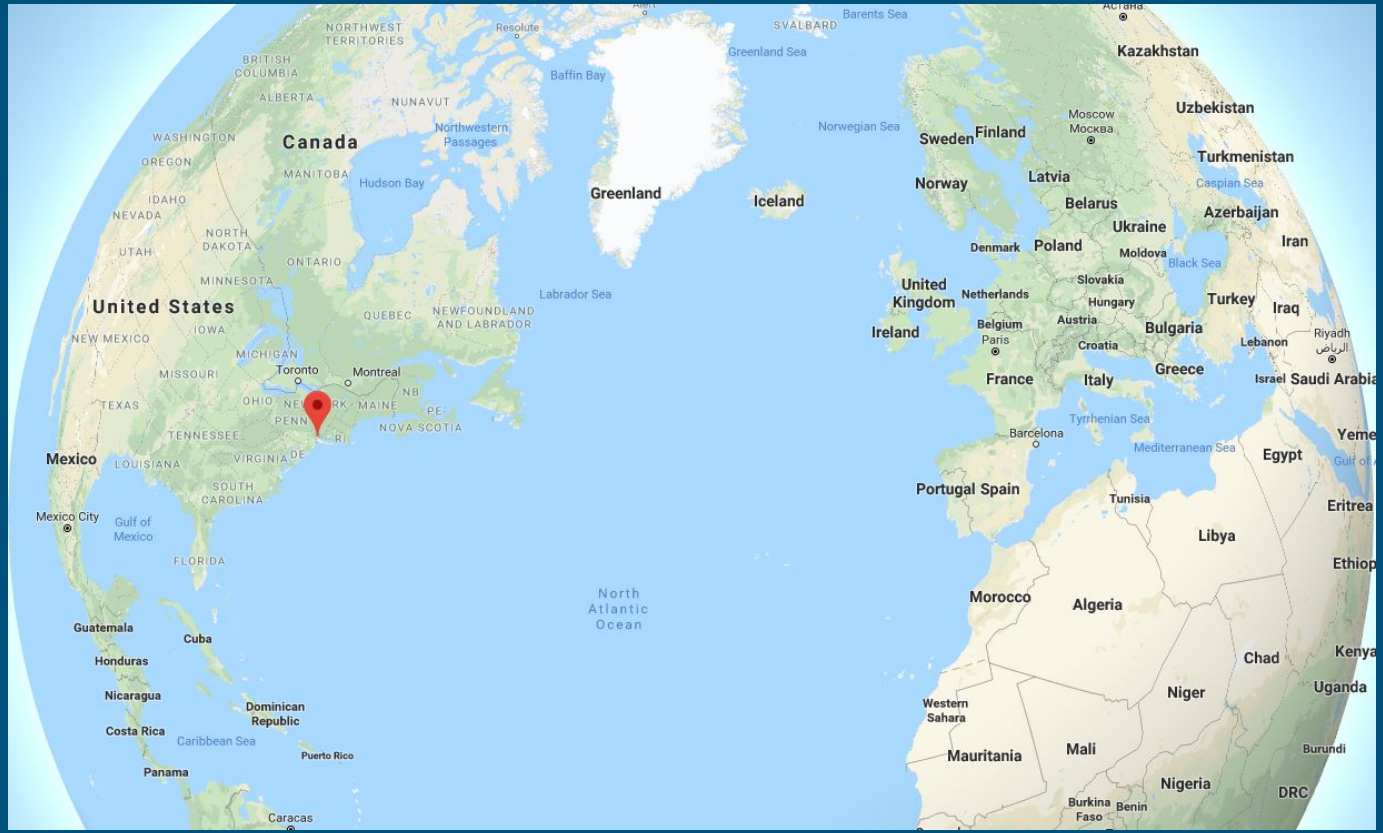
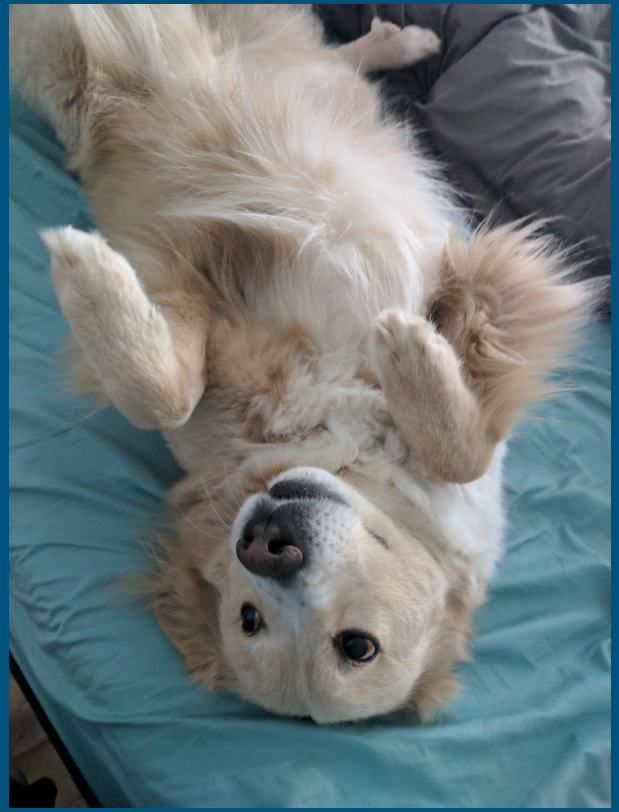
Foreign Functions for Fun and Profit

When and how to use CGo

github.com/liztio/cgo-demo // #FOSDEM 2019 // @stillinbeta

github.com/liztio/cgo-demo

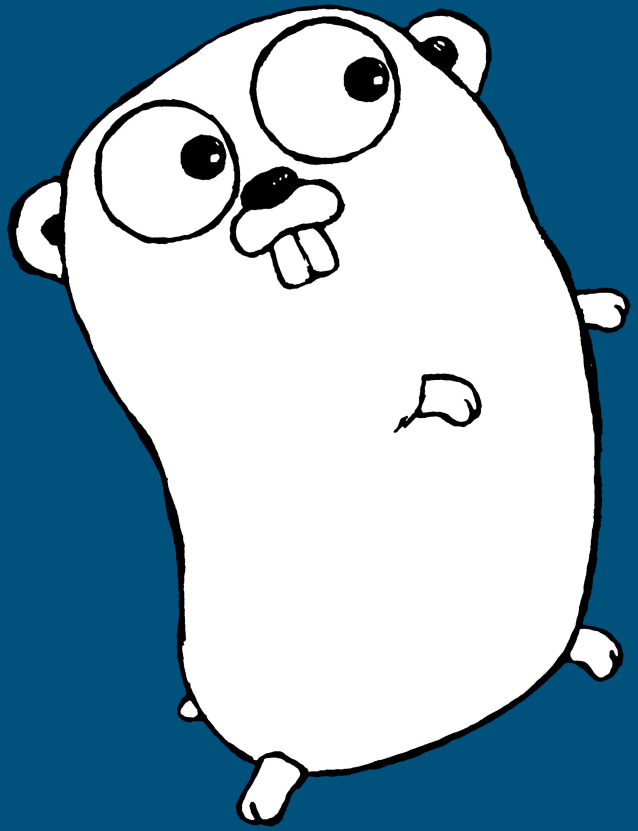

Barents Sea
SVALBARD
Greenland Sea
Kazakhstan
NORTHWEST TERRITORIES
Resolute
BRITISH COLUMBIA
ALBERTA
NUNAVUT
Baffin Bay
Uzbekistan
Moscow
Москва
Norwegian Sea
Sweden
Finland
Canada
Northwestern Passages
WASHINGTON
OREGON
MANITOBA
Hudson Bay
Turkmenistan
Caspian Sea
Norway
Latvia
Greenland
Iceland
IDAHO
NEVADA
Belarus
Azerbaijan
NORTH DAKOTA
Ukraine
UTAH
ONTARIO
Denmark
Poland
Iran
Moldova
Black Sea
MINNESOTA
United States
Labrador Sea
United Kingdom
Netherlands
Slovakia
Turkey
QUEBEC
NEWFOUNDLAND AND LABRADOR
Hungary
Iraq
IOWA
NEW MEXICO
Belgium
Austria
Bulgaria
Ireland
Paris
Croatia
Lebanon
Riyadh
MICHIGAN
Toronto
Montreal
MISSOURI
Greece
France
Italy
Israel
Saudi Arabia
TEXAS
OHIO
MAINE
NB
PE
NOVA SCOTIA
TENNESSEE
Tyrrhenian Sea
Barcelona
Mediterranean Sea
Egypt
Mexico
LOUISIANA
VIRGINIA
SOUTH CAROLINA
Portugal
Spain
Tunisia
Eritrea
Mexico City
Gulf of Mexico
Libya
FLORIDA
Morocco
Algeria
North Atlantic Ocean
Guatemala
Cuba
Honduras
Chad
Kenya
Nicaragua
Niger
Uganda
Dominican Republic
Western Sahara
Costa Rica
Caribbean Sea
Mauritania
Mali
Burundi
Puerto Rico
Panama
Nigeria
DRC
Burkina Faso
Benin
Caracas



@stillinbeta


vmware®

recurse.com



Renee French. (http://reneefrench.blogspot.com/)

Full disclosure?

...I don’t actually like Go very much.

but I use it a lot

Why *not* Cgo?

cgo is not Go - Dave Cheney

bit.ly/cgo-not-go

Why *not Cgo*

- More complicated compilation
- Less portable
- No cross compilation
- More segfaults
- No backtraces
- Less tooling

Why Cgo

- ...sometimes it’s the only way

Calling C from Go

addlib.h

```c
#include <stdint.h>

uint32_t add_one(uint32_t val);
```

addlib.c

```c
#include "addlib.h"

uint32_t add_one(uint32_t val)
{
  return val + 1;
}
```

uint32_t

Unsigned integer 32 bit type

```
import "C"
```

```
// #include "addlib.h"
import "C"
```

```
added := C.add_one(num)
```

main.go

```
package main

// #include "addlib.h"
import "C" // “C” always gets its own line
import "fmt"

func main() {
    num := 10

    added := C.add_one(num)
    fmt.Printf("%d + 1 is %d!\n", num, added)
}
```

```
$ go run github.com/liztio/cgo-demo/cmd/c_from_go
```

```
$ go run github.com/liztio/cgo-demo/cmd/c_from_go
# github.com/liztio/cgo-demo/cmd/c_from_go
cmd/c_from_go/main.go:10:27: cannot use num (type
int) as type _Ctype_uint in argument to
_Cfunc_add_one
```

```
added := C.add_one(C.uint32_t(num))
```

```
$ go run github.com/liztio/cgo-demo/cmd/c_from_go
10 + 1 is 11!
```

Calling Go from C

```
//export Greet
func Greet(chars *C.char) {
    str := C.GoString(chars)
    fmt.Printf("Hello from Go, %s!\n", str)
}
```

```
//export Greet
func Greet(chars *C.char) {
    str := C.GoString(chars)
    fmt.Printf("Hello from Go, %s!\n", str)
}
```

```go
//export Greet
func Greet(chars *C.char) {
    str := C.GoString(chars)
    fmt.Printf("Hello from Go, %s!\n", str)
}
```

Strings in C

Strings in C

F	O	S	D	E	M	\x00
46	4F	53	44	45	4d	00

Strings in Go

```go
//export Greet
func Greet(chars *C.char) {
    str := C.GoString(chars)
    fmt.Printf("Hello from Go, %s!\n", str)
}
```

greet.go

```
package main

import "C"
import "fmt"

//export Greet
func Greet(chars *C.char) {
    str := C.GoString(chars)
    fmt.Printf("Hello from Go, %s!\n", str)
}

func main() {} // required for main package
```

main.c

```c
#include "_cgo_export.h"

int main() {
  char *name = "FOSDEM";

  Greet(name);

  return 0; // exit code
}
```

_cgo_export.h

```
/* Code generated by cmd/cgo; DO NOT EDIT. */

/* package main */
#line 1 "cgo-builtin-prolog"

#include <stddef.h> /* for ptrdiff_t below */

#ifndef GO_CGO_EXPORT_PROLOGUE_H
#define GO_CGO_EXPORT_PROLOGUE_H

typedef struct { const char *p; ptrdiff_t n; }
_GoString_;

#endif

/* Start of preamble from import "C" comments.
*/
/* End of preamble from import "C" comments.
*/
/* Start of boilerplate cgo prologue.  */
```

_cgo_export.h

```
/* don't worry about it */
```

A brief introduction to Makefiles

A brief introduction to Makefiles

```
all: output

output: input1 input2 input3
    command --output $@ --inputs $^
```

A brief introduction to Makefiles

```
all: output

output: input1 input2 input3
    command --output output --inputs input1 input2 input3
```

Our Makefile

```
_cgo_export.h: greet.go
    go tool cgo -exportheader $@ $^

greet.a: greet.go
    go build -buildmode=c-archive $^

gofromc: main.c _cgo_export.h greet.a
    $(CC) -o $@ $^ -lpthread
```

Our Makefile

```
_cgo_export.h: greet.go
	go tool cgo -exportheader $@ $^
```

Our Makefile

```
greet.a: greet.go
	go build -buildmode=c-archive $^
```

Our Makefile

```
gofromc: main.c _cgo_export.h greet.a
	$(CC) -o $@ $^ -lpthread
```

$ make

```
go tool cgo -exportheader _cgo_export.h greet.go

go build -buildmode=c-archive greet.go

cc -o gofromc main.c _cgo_export.h greet.a -lpthread
```

Let’s try it out!

```
$ ./gofromc
Hello from Go, FOSDEM!
```

Pointers and Memory

Go Memory

- `new() / &value`
- Garbage Collected

C Memory

- `malloc()`
- Manually Freed with free()

Allocating in Go

```
type Point struct {
    x, y float32
}

func getPoint() *Point {
    return &Point{
        x: 10,
        y: 12,
    }
}
```

Allocating in C

```
typedef struct {
  float x;
  float y;
} Point;

Point *getPoint() {
  Point *pt = malloc(sizeof(Point));
  pt->x = 10;
  pt->y = 12;
  return pt;
}
```

Go

```go
func usePoint(pt *Point) {
    fmt.Printf("x: %f, y: %f\n", pt.x, pt.y)
    // Goes out of scope
}
```

C

```c
void usePoint(Point *pt) {
  printf("x: %f, y: %f\n", pt->x, pt->y);
  free(pt);
}
```

What happens if you forget to free?

```
$ ./gofromc
x: 10.000000, y: 12.000000
```

What happens if you forget to free?

```
$ valgrind --leak-check=full ./gofromc
==7974== Memcheck, a memory error detector
==7974== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==7974== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==7974== Command: ./gofromc
==7974==
x: 10.000000, y: 12.000000
==7974==
==7974== HEAP SUMMARY:
==7974==     in use at exit: 8 bytes in 1 blocks
==7974==   total heap usage: 2 allocs, 1 frees, 1,032 bytes allocated
==7974==
==7974== 8 bytes in 1 blocks are definitely lost in loss record 1 of 1
==7974==    at 0x4C2FB0F: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==7974==    by 0x10869B: getPoint (in /home/liz/src/github.com/liztio/cgo-demo/src/gofromc/gofromc)
==7974==    by 0x10871C: main (in /home/liz/src/github.com/liztio/cgo-demo/src/gofromc/gofromc)
==7974==
==7974== LEAK SUMMARY:
==7974==    definitely lost: 8 bytes in 1 blocks
==7974==    indirectly lost: 0 bytes in 0 blocks
==7974==      possibly lost: 0 bytes in 0 blocks
==7974==    still reachable: 0 bytes in 0 blocks
==7974==         suppressed: 0 bytes in 0 blocks
==7974==
==7974== For counts of detected and suppressed errors, rerun with: -v
==7974== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

What happens if you forget to free?

```
$ valgrind --leak-check=full ./gofromc
==7974== Memcheck, a memory error detector
==7974== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.
==7974== Using Valgrind-3.13.0 and LibVEX; rerun with -h for copyright info
==7974== Command: ./gofromc
==7974==
x: 10.000000, y: 12.000000
==7974==
==7974== HEAP SUMMARY:
==7974==     in use at exit: 8 bytes in 1 blocks
==7974==   total heap usage: 2 allocs, 1 frees, 1,032 bytes allocated
==7974==
==7974== 8 bytes in 1 blocks are definitely lost in loss record 1 of 1
==7974==    at 0x4C2FB0F: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==7974==    by 0x10869B: getPoint (in /home/liz/src/github.com/liztio/cgo-demo/src/gofromc/gofromc)
==7974==    by 0x10871C: main (in /home/liz/src/github.com/liztio/cgo-demo/src/gofromc/gofromc)
==7974==
==7974== LEAK SUMMARY:
==7974==    definitely lost: 8 bytes in 1 blocks
==7974==    indirectly lost: 0 bytes in 0 blocks
==7974==      possibly lost: 0 bytes in 0 blocks
==7974==    still reachable: 0 bytes in 0 blocks
==7974==         suppressed: 0 bytes in 0 blocks
==7974==
==7974== For counts of detected and suppressed errors, rerun with: -v
==7974== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

What happens if you forget to free?

```
$ valgrind --leak-check=full ./gofromc
==7974== 8 bytes in 1 blocks are definitely lost in loss record 1 of 1
==7974==    at 0x4C2FB0F: malloc (in
/usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==7974==    by 0x10869B: getPoint (in
/home/liz/src/github.com/liztio/cgo-demo/src/gofromc/gofromc)
==7974==    by 0x10871C: main (in
/home/liz/src/github.com/liztio/cgo-demo/src/gofromc/gofromc)
```

Passing Pointers between C and Go

Okay

- Passing Go pointers to C
- Passing C pointers to Go
- Storing Go pointers in C memory before returning

- A bunch of weird special cases involving Java and Darwin

Invalid

- Passing Go pointers with nested pointers to C
- Storing Go pointers in Go memory from C
- Storing Go pointers in C memory
- Storing Go pointers in C memory after returning
- Go Functions called by C returning Go pointers

tl;dr

- Pass pointers carefully
- Read the Cgo docs (golang.org/cmd/cgo)

How about something nontrivial?

How about a little magick?

```c
#include <wand/magick_wand.h>

int main() {
    MagickWand *mw = NULL;

    MagickWandGenesis();

    /* Create a wand */
    mw = NewMagickWand();
    /* Read the input image */
    MagickReadImage(mw,"logo:");
    /* write it */
    MagickWriteImage(mw,"logo.jpg");

    /* Tidy up */
    if(mw) mw = DestroyMagickWand(mw);
    MagickWandTerminus();
}
```

https://imagemagick.org/MagickWand/

```
package main

// #cgo pkg-config: MagickWand
// #include <wand/MagickWand.h>
import "C"

func main() {
    C.MagickWandGenesis()
    /* Create a wand */
    mw := C.NewMagickWand()
    defer func() {
        /* Tidy up */
        C.DestroyMagickWand(mw)
        C.MagickWandTerminus()
    }()
    /* Read the input image */
    C.MagickReadImage(mw, C.CString("logo:"))
    /* write it */
    C.MagickWriteImage(mw,
C.CString("logo.jpg"))
}
```

```
// #cgo pkg-config: MagickWand
```

what's this?

```
$ pkg-config MagickWand --cflags
-fopenmp -DMAGICKCORE_HDRI_ENABLE=0 -DMAGICKCORE_QUANTUM_DEPTH=16 -fopenmp -DMAGICKCORE_
$ pkg-config MagickWand --libs
-lMagickWand-6.Q16 -lMagickCore-6.Q16
```

```
// #cgo CFLAGS: -I/usr/include/ImageMagick-6 -I/usr/include/x86_64-linux-gnu/ImageMagick-6
// #cgo LDFLAGS: -lMagickWand-6.Q16 -lMagickCore-6.Q16
```

Expands to

```
defer func() {
    /* Tidy up */
    C.DestroyMagickWand(mw)
    C.MagickWandTerminus()
}()
```

C `defer` anyone?

Full resizing demo in the Github Repo!

`github.com/liztio/cgo-demo/cmd/imagemagick`

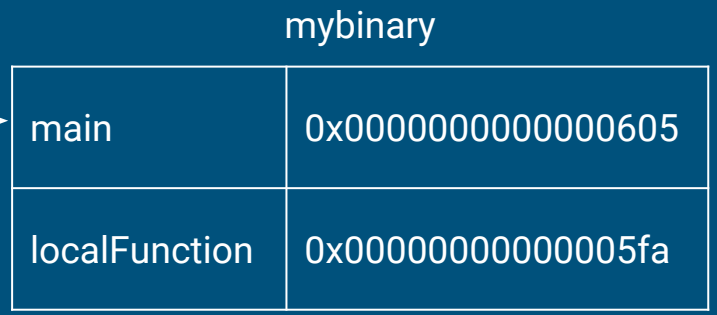
Dynamic Libraries

“Go binaries are static”

Windows	OSX	Linux
.dll	.dylib	.so

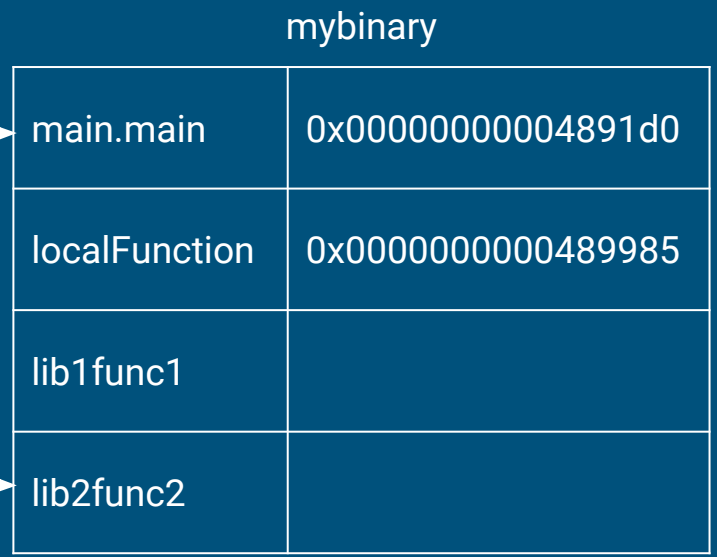
```
int localFunc() {
  return 0;
}

int main() {
  return localFunc();
}
```

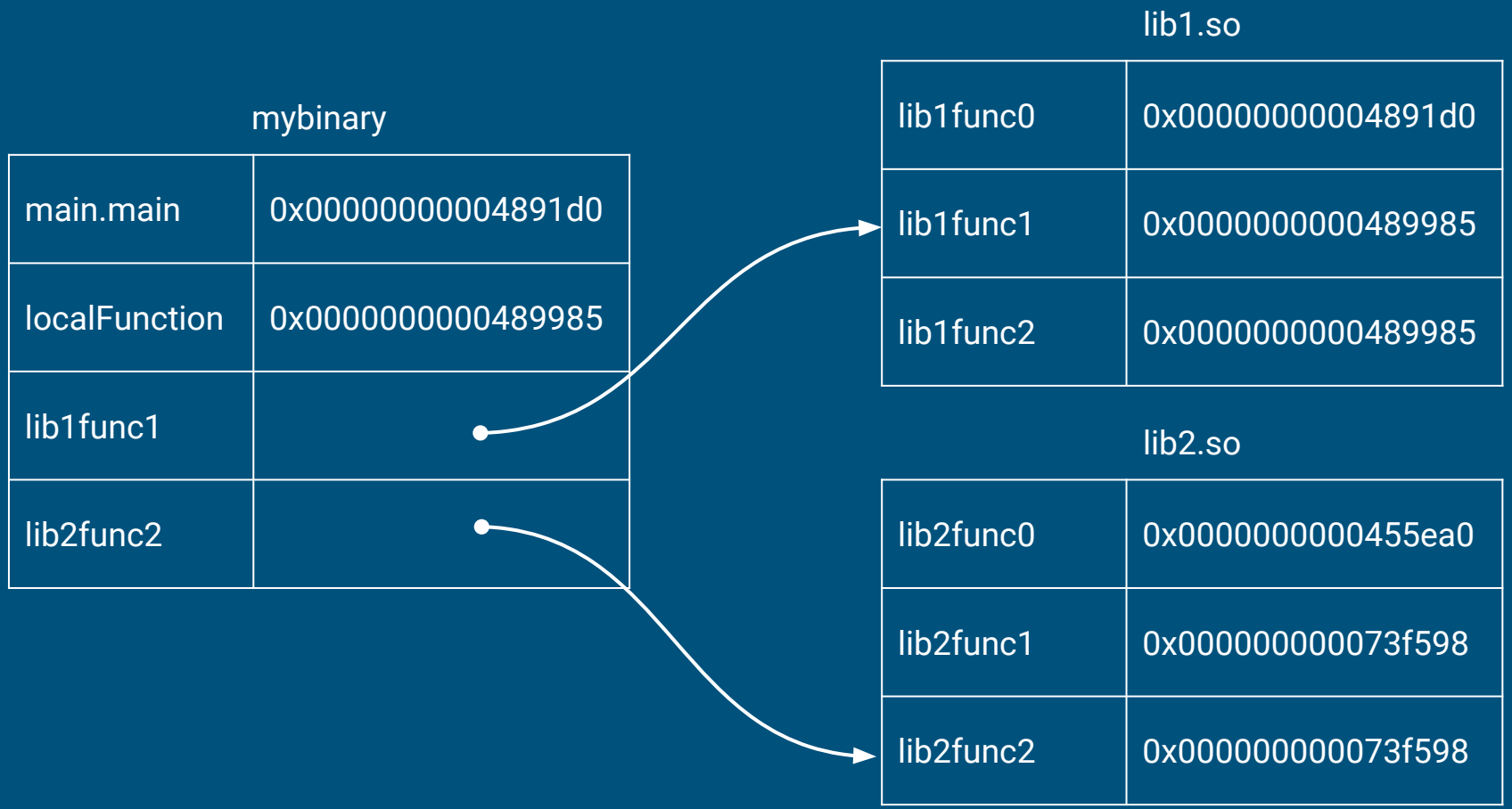
```
int localFunc() {
  return 0;
}

int main() {
  return localFunc();
}
```



```c
#include <lib1.h>
#include <lib2.h>

void localFunction() {
  char *str = lib1Func1();
  lib2Func2(str);
}

int main() {
  localFunction()
}
```

```c
#include <lib1.h>
#include <lib2.h>

void localFunction() {
  char *str = lib1Func1();
  lib2Func2(str);
}

int main() {
  localFunction()
}
```




mybinary
main.main
0x00000000004891d0
localFunction
0x0000000000489985
lib1func1
lib2func2
lib1.so
lib1func0
0x00000000004891d0
lib1func1
0x0000000000489985
lib1func2
0x0000000000489985
lib2.so
lib2func0
0x0000000000455ea0
lib2func1
0x000000000073f598
lib2func2
0x000000000073f598

```
LDD(1)    Linux Programmer's Manual    LDD(1)

NAME
       ldd - print shared object dependencies

SYNOPSIS
       ldd [option]... file...
```

```
$ ldd ~/bin/ginkgo
  linux-vdso.so.1 (0x00007ffc71493000)
  libpthread.so.0 =>
/lib/x86_64-linux-gnu/libpthread.so.0
(0x00007fa2d36c4000)
  libc.so.6 =>
/lib/x86_64-linux-gnu/libc.so.6
(0x00007fa2d32d3000)
  /lib64/ld-linux-x86-64.so.2
(0x00007fa2d38e3000)
```

```
$ ldd ~/bin/ginkgo
  linux-vdso.so.1 (0x00007ffc71493000)
  libpthread.so.0 =>
/lib/x86_64-linux-gnu/libpthread.so.0
(0x00007fa2d36c4000)
  libc.so.6 =>
/lib/x86_64-linux-gnu/libc.so.6
(0x00007fa2d32d3000)
  /lib64/ld-linux-x86-64.so.2
(0x00007fa2d38e3000)
```

```
// #cgo LDFLAGS: -laddlib
// #include "addlib.h"
import "C"
```

```
$ go install github.com/liztio/cgo-demo/cmd/addbin
```

```
$ export ADDLIB_DIR=/home/liz/src/github.com/liztio/cgo-demo/src/addlib
$ export CGO_CFLAGS=-I$ADDLIB_DIR
$ export CGO_LDFLAGS=-L$ADDLIB_DIR
$ go install github.com/liztio/cgo-demo/cmd/addbin
```

```
$ export ADDLIB_DIR=/home/liz/src/github.com/liztio/cgo-demo/src/addlib
$ export CGO_CFLAGS=-I$ADDLIB_DIR
$ export CGO_LDFLAGS=-L$ADDLIB_DIR
$ go install github.com/liztio/cgo-demo/cmd/addbin
```

```
$ ldd addbin
  linux-vdso.so.1 (0x00007ffd2b966000)
  libaddlib.so => not found
  libpthread.so.0 =>
/lib/x86_64-linux-gnu/libpthread.so.0
(0x00007f252a003000)
  libc.so.6 =>
/lib/x86_64-linux-gnu/libc.so.6
(0x00007f2529c12000)
  /lib64/ld-linux-x86-64.so.2
(0x00007f252a222000)
```

```
$ ldd addbin
  linux-vdso.so.1 (0x00007ffd2b966000)
  libaddlib.so => not found
  libpthread.so.0 =>
/lib/x86_64-linux-gnu/libpthread.so.0
(0x00007f252a003000)
  libc.so.6 =>
/lib/x86_64-linux-gnu/libc.so.6
(0x00007f2529c12000)
  /lib64/ld-linux-x86-64.so.2
(0x00007f252a222000)
```

```
$ addbin
addbin: error while loading shared libraries:
libaddlib.so: cannot open shared object file:
No such file or directory
```

```
$ LD_LIBRARY_PATH=$ADDLIB_DIR addbin
Go says: 11 + one is 12
```

Questions?

Please play with the code!
github.com/liztio/cgo-demo