Unigornel
The Go Compiler - Linking with Mini-OS

Henri Verroken

February 16, 2016

Components of the Go compiler

- Go
  - gc-compiler
  - `go build` (entrypoint in `cmd/go/build.go`)
- **Cgo**
  - **Interface with C** (Mini-OS)
  - Automatically used by `go build` if needed
  - Manually using `go tool cgo` (`cmd/cgo/main.go`)
- Gccgo
  - Frontend to GCC to compile Go code
  - Fully independent from gc-compiler

Cgo

- Interface with C
- All important information in `cmd/cgo/doc.go`
- Used when importing `C`-pseudopackage (`import "C"`)
- Uses specified C compiler (normally GCC)
- Internal and external linking

Cgo - Linking

- Internal linking
  - gc has linker with minimal support for ELF.
  - Not suited for Mini-OS
- **External linking**
  - **Build a c-archive**
    - Static library
    - See also `cmd/cgo/doc.go`
  - **Manually link C-code and Go c-archive using GCC**
  - Example at GitHub[1]

[1] `https://github.ugent.be/unigornel/playground/tree/cgo_external`

Cgo - Example

```
//- test.go
// go build -buildmode=c-archive \
//    -o test.a test.go
//
// Generates test.a (static library) and test.h
package main

import "C"

func main() {
}

//export Sum
func Sum(a, b int) int {
        return a + b
}
```

Cgo - Example

- C-archive
  - Contains exported `Sum`-symbol (see next slide)
  - Contains full Go runtime
  - Go runtime automatically started when `Sum` is called.
- Header file
  - Declares Go types (`GoInt, GoUInt, GoInt8, ...`)
  - Declares exported functions
    - `extern GoInt Sum(GoInt, GoInt);`

Cgo - Example

```
//- _cgo_export.c
/* Created by cgo - DO NOT EDIT. */
#include "_cgo_export.h"
[...]
GoInt Sum(GoInt p0, GoInt p1) {
        _cgo_wait_runtime_init_done();
        struct {
                GoInt p0; GoInt p1; GoInt r0;
        } [...] a;
        a.p0 = p0;
        a.p1 = p1;
        crosscall2(_cgoexp_[...]_Sum, &a, 24);
        return a.r0;
}
```

Cgo - Example

```
//- main.c
// gcc main.c test.a -lpthread -o test.out
// ./test.out
#include <stdio.h>

#include "testing.h"

int main() {
    GoInt r;

    printf("Hello from C!\n");
    r = Sum(3, 5);
    printf("Result %lld\n", r);
    return 0;
}
```

Cgo - Example

```
$ go build -buildmode=c-archive -o test.a test.g
$ gcc main.c test.a -lpthread -o test.out
$ ./test.out
Hello from C!
Result 8
```

Mini-OS

- General idea:
  - Build Go c-archive
  - Link with c-archive when building Mini-OS
- Problems
  - Linux specific code
  - System calls
  - **Pthreads**, `fprintf`, `abort`, **...**

Dumb approach[2]

Just link!

```
$ nm -u testing.a
go.o:
    [...]
000000.o:
    [...]
000001.o:
    U abort
    U fprintf
    U fputc
    U free
    U fwrite
    U _GLOBAL_OFFSET_TABLE_
    U malloc
    U pthread_attr_destroy
    U pthread_attr_getstacksize
```

[2] https://unigornel.org/doku.php?id=minios:link_with_go

Dumb approach (continued)

```
U pthread_attr_init
    U pthread_cond_broadcast
    U pthread_cond_wait
    U pthread_create
    U pthread_mutex_lock
    U pthread_mutex_unlock
    U pthread_sigmask
    U setenv
    U sigfillset
    U stderr
    U strerror
    U unsetenv
    U vfprintf
```

Approach 2: Edit Cgo[3][4]

- Delete all references to pthreads
- Stub `fprintf`, `abort`, `fwrite`, ...
  - `runtime/cgo/gcc_libinit.c`
  - `runtime/cgo/gcc_linux_amd64.c`
- Failed to compile Go after changes
  - Go built-in linker (`ld`) could not read generated ELF

[3] `https://github.ugent.be/unigornel/go/tree/unigornel_fail`

[4] `https://unigornel.org/doku.php?id=minios:link_with_go`

Approach 3: Dive deep into Cgo

- Use `auditd/auditctl` to audit commands used by Cgo
  - How is `000001.o`[5] generated? (`cmd/link/internal/ld/lib.go:691`)[6]
  - Replace `000001.o`
  - ...

[5] `000001.o` contains references to system specific functions

[6] `cd go/srccmd/link/internal; ack tmpdir`

Table of Contents