Unigornel
Initializing Go in Mini-OS (Part 1)

Henri Verroken

March 1, 2016

Go Runtime Initialization - Linux[2]

- Entrypoint dependent on Go usage
  - For standalone binary: `_rt0_amd64_linux` calls `_rt0_go`
  - For c-archive: `_rt0_amd64_linux_lib` does some things, than calls `_rt0_go`
- Entrypoint must be called
  - For standalone binary: `_rt0_amd64_linux` is ELF entrypoint
  - For c-archive: `_rt0_amd64_linux_lib` is called by `__libc_csu_init`[1]
- Mini-OS?

[1] http://dbp-consulting.com/tutorials/debugging/linuxProgramStartup.html

[2] http://blog.altoros.com/golang-internals-part-5-runtime-bootstrap-process.html

Go Runtime Initialization - Mini-OS

- Mini-OS does not call constructors from c-archive
- `_rt0_amd64_netbsd_lib` must be called manually
- After OS initialization

```
[...]
#include "sum.h" // generated by cgo

extern void _rt0_amd64_netbsd_lib(void);

// function of the primary mini-os thread
int app_main(start_info_t *si) {
    _rt0_amd64_netbsd_lib();

    [...]
}
```

Go Runtime Initialization - Mini-OS

Linking did not go as expected

```
$ cd minios
$ make GOARCHIVE=go/src/sum/sum.a \
    GOINCLUDE=go/src/sum/
[...]
ld -m elf_x86_64 -T [...]/minios-x86_64.lds \
    [...]/mini-os.o -o [...]/mini-os
mini-os.o: In function ‘app_main’:
go_main.c:12: undefined reference
    to ‘_rt0_amd64_netbsd_lib’
[...]
```

Go Runtime Initialization - Mini-OS

But the symbol is there?

```
$ nm sum.a | grep rt0_amd64_netbsd_lib
0000000000004e610 t _rt0_amd64_netbsd_lib
0000000000004e640 t _rt0_amd64_netbsd_lib_go
0000000000000000 t _rt0_amd64_netbsd_lib.ptr
```

Symbol `_rt0_amd64_netbsd_lib` is not global!

Compare with a global symbol (see `man nm`)

```
$ nm sum.a | grep Sum
0000000000000000 T Sum
```

Go Runtime Initialization - Mini-OS

- Need to make symbol global
- Approach 1: Edit Go runtime to export symbol
  - Not a good idea
- Approach 2: Rewrite ELF[3]
  - Very handy utility `objcopy`
  - Has option `--globalize-symbol=`*`symbol`*

```
$ S=_rt0_amd64_netbsd_lib
$ objcopy --globalize-symbol=$S sum.a
$ nm sum.a | grep rt0_amd64_netbsd_lib
000000000004e610 T _rt0_amd64_netbsd_lib
000000000004e640 t _rt0_amd64_netbsd_lib_go
0000000000000000 t _rt0_amd64_netbsd_lib.ptr
```

Works!

[3] https://github.ugent.be/unigornel/minios/commit/52c42d1f720aa22665da18f3e8caa0169c911ac9

Let's crash it!

- Previous version crashes
  - In `Sum`-crosscall
  - With uninitialized Go runtime
  - Using uninitialized `%fs`-segment resulted in crash
- This version[4]
  - Crashes on Ubuntu Wily (gcc 5.2.1/4.9.3)
  - Runs on Debian Jessie (gcc 4.9.2)

[4] To reproduce: Mini-OS at commit 52c42d1f720aa22665da18f3e8caa0169c911ac9, Go at commit 871141c521e89845044d2b758d4160f619aff877

Let's crash it![5]

```
Breakpoint 1, [...] at 0x6505f: x_cgo_sys_thread_create
(gdb) disas
Dump [...] for function x_cgo_sys_thread_create:
   0x65050 <+0>:   sub    $0x18,%rsp
   0x65054 <+4>:   mov    %rdi,%rdx
   0x65057 <+7>:   mov    %rsi,%rcx
   0x6505a <+10>:  mov    %rsp,%rdi
   0x6505d <+13>:  xor    %esi,%esi
=> 0x6505f <+15>:  mov    %fs:0x28,%rax
   0x65068 <+24>:  mov    %rax,0x8(%rsp)
   0x6506d <+29>:  xor    %eax,%eax
   0x6506f <+31>:  callq  0x9f3f <pthread_create>
   0x65074 <+36>:  test   %eax,%eax
   0x65076 <+38>:  jne    0x6508d
                            <x_cgo_sys_thread_create+61>
```

[5] On Ubuntu Wily with gcc 5.2.1/4.9.3

Let's run it![6]

```
Breakpoint 1, [...] at 0x64fc0: x_cgo_sys_thread_create
(gdb) disas
Dump [...] for function x_cgo_sys_thread_create:
=> 0x64fc0 <+0>:   sub    $0x18,%rsp
   0x64fc4 <+4>:   mov    %rdi,%rdx
   0x64fc7 <+7>:   mov    %rsi,%rcx
   0x64fca <+10>:  lea    0x8(%rsp),%rdi
   0x64fcf <+15>:  xor    %esi,%esi
   0x64fd1 <+17>:  callq  0x9f13 <pthread_create>
   0x64fd6 <+22>:  test   %eax,%eax
   0x64fd8 <+24>:  jne    0x64fdf
                            <x_cgo_sys_thread_create+31>
```

[6] On Debian Jessie with gcc 4.9.2

Why the difference?

?

Table of Contents