

Go и мир системного программирования

Алексей Веселовский

C++ Russia 2022



Кто я?

Кто я?

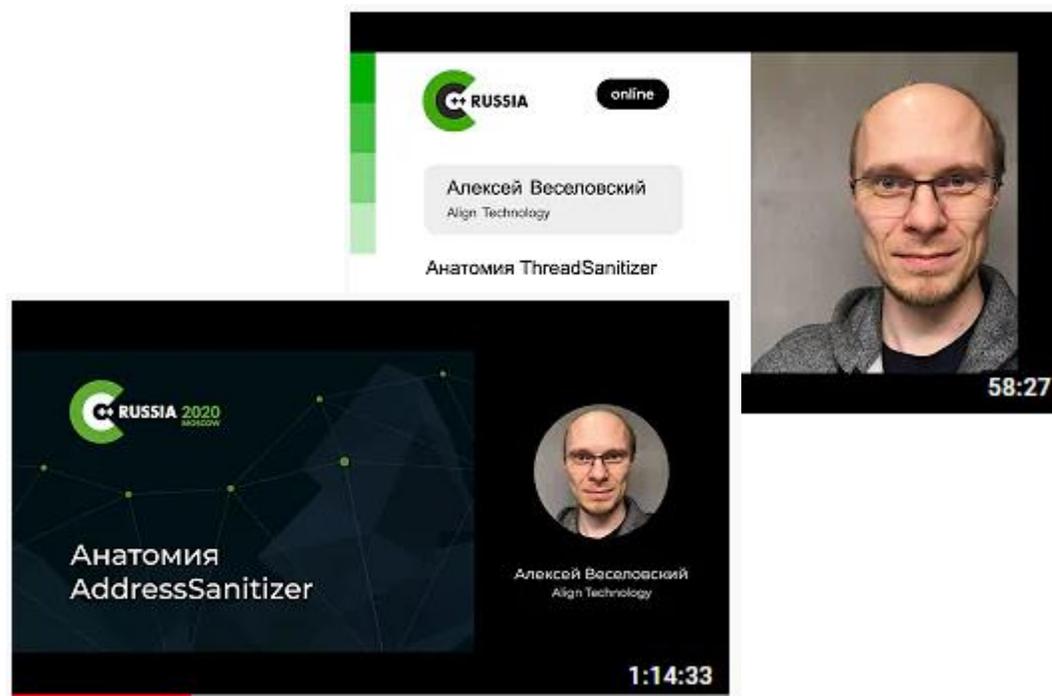
- C++ программист с каким-то опытом (15+ лет)

Кто я?

- С++ программист с каким-то опытом (15+ лет)
- Писал на Go в “продакшн” когда это не было модным (2011-2014)

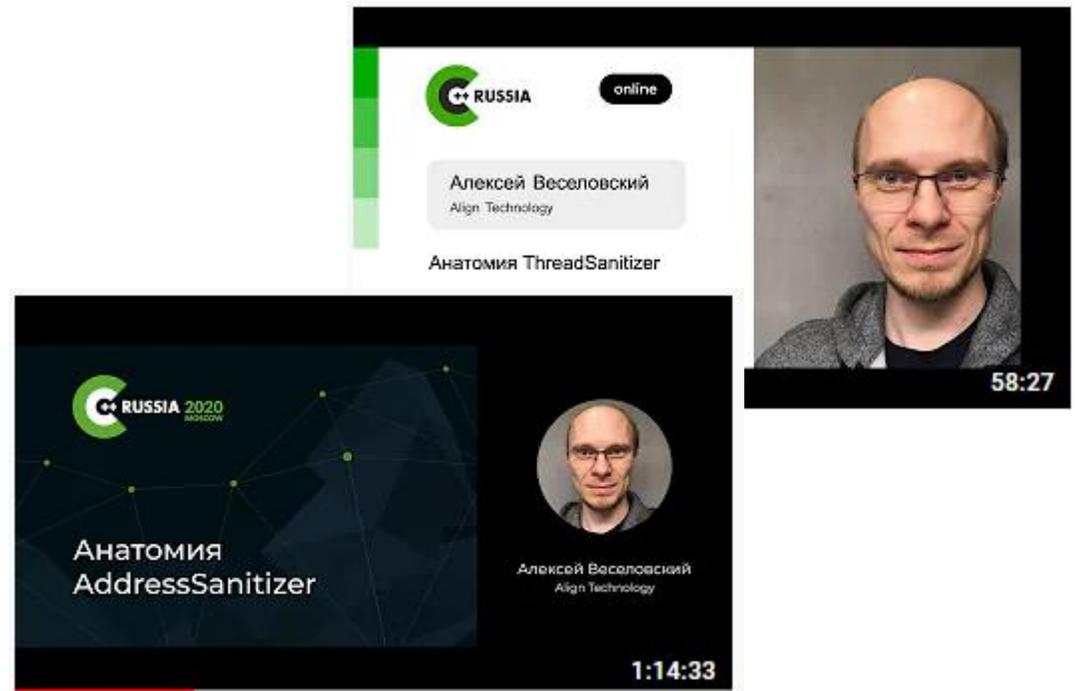
Кто я?

- C++ программист с каким-то опытом (15+ лет)
- Писал на Go в “продакшн” когда это не было модным (2011-2014)
- Доклады по ASan и TSan



Кто я?

- С++ программист с каким-то опытом (15+ лет)
- Писал на Go в “продакшн” когда это не было модным (2011-2014)
- Доклады по ASan и Tsan
- Иногда желаю странного



Терминология

Терминология

1. Высокоуровневое программирование

Терминология

1. Высокоуровневое программирование
 - Возможность абстрагирования от команд исполнителя

Терминология

1. Высокоуровневое программирование
 - Возможность абстрагирования от команд исполнителя
 - Возможность и интенсивное использование новых абстракций в ходе разработки ПО

Терминология

1. Высокоуровневое программирование
 - Возможность абстрагирования от команд исполнителя
 - Возможность и интенсивное использование новых абстракций в ходе разработки ПО
2. Низкоуровневое программирование

Терминология

1. Высокоуровневое программирование
 - Возможность абстрагирования от команд исполнителя
 - Возможность и интенсивное использование новых абстракций в ходе разработки ПО
2. Низкоуровневое программирование
 - Написание ПО в терминах команд исполнителя

Терминология

1. Высокоуровневое программирование

- Возможность абстрагирования от команд исполнителя
- Возможность и интенсивное использование новых абстракций в ходе разработки ПО

2. Низкоуровневое программирование

- Написание ПО в терминах команд исполнителя
- Отсутствие, либо крайне скудные возможности введения собственных абстракций

Терминология

1. Высокоуровневое программирование

- Возможность абстрагирования от команд исполнителя
- Возможность и интенсивное использование новых абстракций в ходе разработки ПО

2. Низкоуровневое программирование

- Написание ПО в терминах команд исполнителя
- Отсутствие, либо крайне скудные возможности введения собственных абстракций

```
MOV EAX, [EBX]
MOV [ESI+EAX], CL
MOV DS, DX
```

Терминология

1. Высокоуровневое программирование

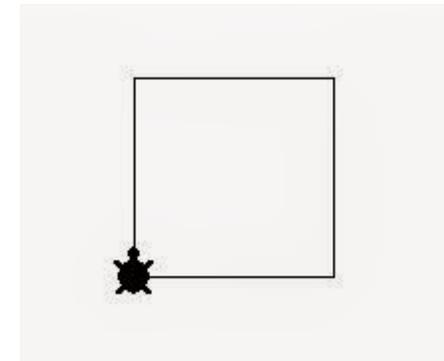
- Возможность абстрагирования от команд исполнителя
- Возможность и интенсивное использование новых абстракций в ходе разработки ПО

2. Низкоуровневое программирование

- Написание ПО в терминах команд исполнителя
- Отсутствие, либо крайне скудные возможности введения собственных абстракций

```
MOV EAX, [EBX]
MOV [ESI+EAX], CL
MOV DS, DX
```

```
вперед 100
направо 90
вперед 100
направо 90
вперед 100
направо 90
вперед 100
направо 90
```



Терминология

1. Высокоуровневое программирование

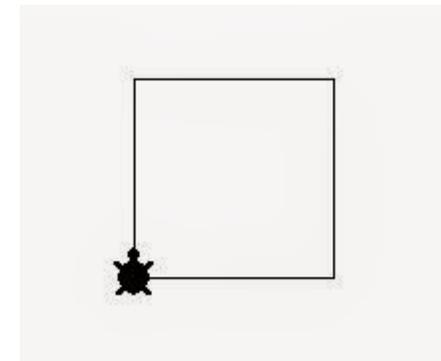
- Возможность абстрагирования от команд исполнителя
- Возможность и интенсивное использование новых абстракций в ходе разработки ПО

2. Низкоуровневое программирование

- Написание ПО в терминах команд исполнителя
- Отсутствие, либо крайне скудные возможности введения собственных абстракций

```
MOV EAX, [EBX]
MOV [ESI+EAX], CL
MOV DS, DX
```

```
вперед 100
направо 90
вперед 100
направо 90
вперед 100
направо 90
вперед 100
направо 90
```



Терминология

Системное программирование/системное ПО:

Терминология

Системное программирование/системное ПО:

- Такое ПО, которое обеспечивает и “облегчает” функционирование прикладного ПО в данной Системе.

Терминология

Системное программирование/системное ПО:

- Такое ПО, которое обеспечивает и “облегчает” функционирование прикладного ПО в данной Системе.
- Не решает прикладных задач

Терминология

Системное программирование/системное ПО:

- Такое ПО, которое обеспечивает и “облегчает” функционирование прикладного ПО в данной Системе.
- Не решает прикладных задач
- Зачастую не несёт прямого business value 😊

Терминология

Системное программирование/системное ПО:

- Такое ПО, которое обеспечивает и “облегчает” функционирование прикладного ПО в данной Системе.
- Не решает прикладных задач
- Зачастую не несёт прямого business value 😊

Система может быть любой:

Терминология

Системное программирование/системное ПО:

- Такое ПО, которое обеспечивает и “облегчает” функционирование прикладного ПО в данной Системе.
- Не решает прикладных задач
- Зачастую не несёт прямого business value 😊

Система может быть любой:

- Операционная система

Терминология

Системное программирование/системное ПО:

- Такое ПО, которое обеспечивает и “облегчает” функционирование прикладного ПО в данной Системе.
- Не решает прикладных задач
- Зачастую не несёт прямого business value 😊

Система может быть любой:

- Операционная система
- Браузер

Терминология

Системное программирование/системное ПО:

- Такое ПО, которое обеспечивает и “облегчает” функционирование прикладного ПО в данной Системе.
- Не решает прикладных задач
- Зачастую не несёт прямого business value 😊

Система может быть любой:

- Операционная система
- Браузер
- CAD-система

Терминология

Системное программирование/системное ПО:

- Такое ПО, которое обеспечивает и “облегчает” функционирование прикладного ПО в данной Системе.
- Не решает прикладных задач
- Зачастую не несёт прямого business value 😊

Система может быть любой:

- Операционная система
- Браузер
- CAD-система
- SCADA

Терминология

Системное программирование/системное ПО:

- Такое ПО, которое обеспечивает и “облегчает” функционирование прикладного ПО в данной Системе.
- Не решает прикладных задач
- Зачастую не несёт прямого business value 😊

Система может быть любой:

- Операционная система
- Браузер
- CAD-система
- SCADA
- Черепашка

Терминология

Системное программирование/системное ПО:

- Такое ПО, которое обеспечивает и “облегчает” функционирование прикладного ПО в данной Системе.
- Не решает прикладных задач
- Зачастую не несёт прямого business value 😊

Система может быть любой:

- Операционная система
- Браузер
- CAD-система
- SCADA
- Черепашка
- JVM

Терминология

Системное программирование/системное ПО:

- Такое ПО, которое обеспечивает и “облегчает” функционирование прикладного ПО в данной Системе.
- Не решает прикладных задач
- Зачастую не несёт прямого business value 😊

Система может быть любой:

- **Операционная система**
- Браузер
- CAD-система
- SCADA
- Черепашка
- JVM

Терминология

Системное программирование/системное ПО:

- Такое ПО, которое обеспечивает и “облегчает” функционирование прикладного ПО в данной Системе.
- Не решает прикладных задач
- Зачастую не несёт прямого business value 😊

Система может быть любой:

- **Операционная система => Linux**
- Браузер
- CAD-система
- SCADA
- Черепашка
- JVM

Терминология

Системное программирование/системное ПО:

- Такое ПО, которое обеспечивает и “облегчает” функционирование прикладного ПО в данной Системе.
- Не решает прикладных задач
- Зачастую не несёт прямого business value 😊

Система может быть любой:

- **Операционная система => Linux**
- Браузер
- CAD-система
- SCADA
- Черепашка
- JVM

Терминология

Системное программирование/системное ПО:

- Такое ПО, которое обеспечивает и “облегчает” функционирование прикладного ПО в данной Системе.
- Не решает прикладных задач
- Зачастую не несёт прямого business value 😊

Система может быть любой:

- **Операционная система => Linux**
- Браузер
- CAD-система
- SCADA
- Черепашка
- JVM

Терминология

Системное программирование/системное ПО:

- Такое ПО, которое обеспечивает и “облегчает” функционирование прикладного ПО в данной Системе.
- Не решает прикладных задач
- Зачастую не несёт прямого business value 😊
- От производительности и качества системного ПО зависит всё

Система может быть любой:

- **Операционная система => Linux**
- Браузер
- CAD-система
- SCADA
- Черепашка
- JVM

Мотивация

Мотивация

1. Это круто

Мотивация

1. Это круто
2. У Go рантайм написан на Go

Мотивация

1. Это круто
2. У Go рантайм написан на Go
3. Часто Go упоминают в ряду системных ЯП

Мотивация

1. Это круто
2. У Go рантайм написан на Go
3. Часто Go упоминают в ряду системных ЯП
4. ASan/TSan для Go не на Go...

Мотивация

1. Это круто
2. У Go рантайм написан на Go
3. Часто Go упоминают в ряду системных ЯП
4. ASan/TSan для Go не на Go...
5. Это ВОЗМОЖНО [https://wiki.osdev.org/Go Bare Bones](https://wiki.osdev.org/Go_Bare_Bones)

In this tutorial you'll learn how to get started using the Go language to write your own OS. It will be an example of how to create a very minimal system to get text on the screen. It's in no way an example of how you should organize or structure your project.

Мотивация

1. Это круто
2. У Go рантайм написан на Go
3. Часто Go упоминают в ряду системных ЯП
4. ASan/TSan для Go не на Go...
5. Это ВОЗМОЖНО [https://wiki.osdev.org/Go Bare Bones](https://wiki.osdev.org/Go_Bare_Bones)
6. Это НЕВОЗМОЖНО <https://golangdocs.com/system-programming-in-go-interview-questions>

All compiled programs written in Go, include the Go runtime engine, and can't live without it.

In this tutorial you'll learn how to get started using the Go language to write your own OS. It will be an example of how to create a very minimal system to get text on the screen. It's in no way an example of how you should organize or structure your project.

Мотивация

1. Это круто
2. У Go рантайм написан на Go
3. Часто Go упоминают в ряду системных ЯП
4. ASan/TSan для Go не на Go...
5. Это ВОЗМОЖНО [https://wiki.osdev.org/Go Bare Bones](https://wiki.osdev.org/Go_Bare_Bones)
6. Это НЕВОЗМОЖНО
<https://golangdocs.com/system-programming-in-go-interview-questions>

All compiled programs written in Go, include the Go runtime engine, and can't live without it.

In this tutorial you'll learn how to get started using the Go language to write your own OS. It will be an example of how to create a very minimal system to get text on the screen. It's in no way an example of how you should organize or structure your project.

7. Это круто!

Отрываем Runtime...

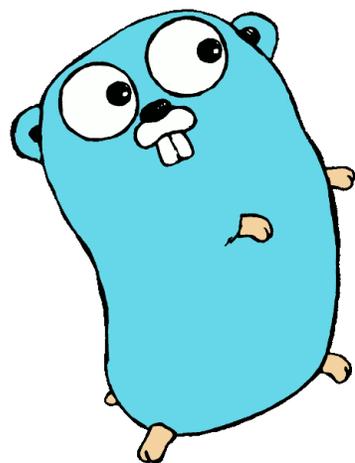
- Зачем?

Отрываем Runtime...

- Зачем?
- Чтобы всё контролировать (heap and so on)

Отрываем Runtime...

Малыш Go спешит на встречу
в друзьями, чтобы
поиграть в новую игру



Отрываем Runtime...

Попытка #1 – ищем опцию у линковщика

Отрываем Runtime...

Попытка #1 – ищем опцию `y go build`, `y go tool link..`

Отрываем Runtime...

Попытка #1 – ищем опцию `y go build`, `y go tool link`..

Официальной опции нет

Отрываем Runtime...

Попытка #1 – ищем опцию `y go build`, `y go tool link`..

Официальной опции нет, но может есть неофициальная?

Идём в исходники...

Отрываем Runtime...

Попытка #1 – ищем опцию `y go build`, `y go tool link`..

Официальной опции нет, но может есть неофициальная?

Идём в исходники... Опцию не находим, но..

Отрываем Runtime...

Попытка #1 – ищем опцию `y go build`, `y go tool link`..

Официальной опции нет, но может есть неофициальная?

Идём в исходники... Опцию не находим, но..

<https://github.com/golang/go/blob/master/src/cmd/go/internal/load/pkg.go>

```
2491 // LinkerDeps returns the list of linker-induced dependencies for main package p.
2492 func LinkerDeps(p *Package) []string {
2493     // Everything links runtime.
2494     deps := []string{"runtime"}
```

Отрываем Runtime...

Попытка #2 – ok google, есть примеры mini runtime для Go?

Отрываем Runtime...

Попытка #2 – ok google, есть примеры mini runtime для Go?

[a minimal container runtime written in Go, mainly for learning and fun](#)

[www.reddit.com › golang › comments › minict_a_minimal_container_runt...](#)

Jan 2, 2021 · I wrote a small container runtime in Golang named Minict, mainly to learn Go (as it is not the language I use on a daily basis) and the behind- ...

[Minict - a minimal container runtime written in Go, mainly for learning ...](#)

[Elsa is a minimal, fast and secure runtime for Javascript and ... - Reddit](#)

[More results from www.reddit.com](#)

[Create the smallest and secured golang docker image based ... - CH](#)

[chemidy.medium.com › create-the-smallest-and-secured-golang-docker-im...](#)

Mar 1, 2018 · Create the smallest and secured golang docker image based on scratch. When we are building a docker Image, the first idea is using the ...

[Go — build a minimal docker image in just three steps - DEV ...](#)

[dev.to › ivan › go-build-a-minimal-docker-image-in-just-three-steps-514i](#)

Sep 9, 2019 · # Let's create a /dist folder containing just the files necessary for runtime. # Later, it will be copied as the / (root) of the output image.

[Build your Go image | Docker Documentation](#)

[docs.docker.com › language › golang › build-images](#)

An image includes everything you need to run an application – the code or binary, runtime, dependencies, and any other file system objects required. To complete ...

Отрываем Runtime...

Попытка #2 – ok google, есть примеры mini runtime для Go?

<https://github.com/malvira/ugo>

 malvira getting there.	84b238b on Jan 21, 2016	 8 commits
 math	This should build a arm binary against the stubbed out runtime. It sh...	7 years ago
 runtime	This should build a arm binary against the stubbed out runtime. It sh...	7 years ago
 README.md	getting there.	7 years ago
 build.sh	getting there.	7 years ago
 clean.sh	get the assembly code in the right spot and start poking values in th...	7 years ago
 header-bin	getting there.	7 years ago
 main.go	getting there.	7 years ago

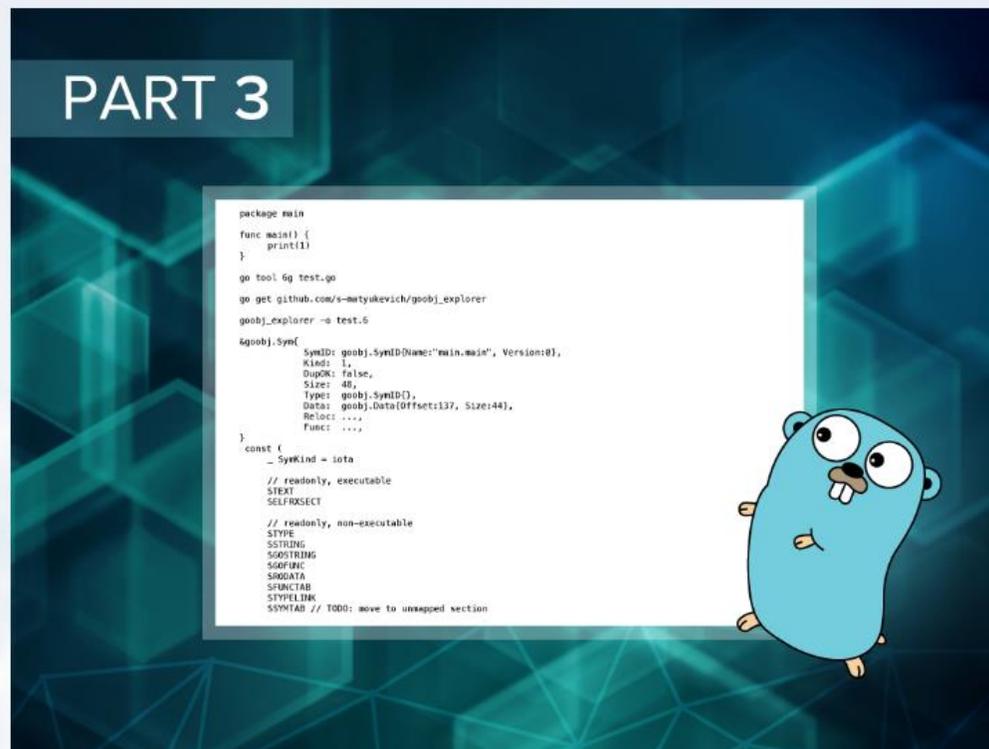
Отрываем Runtime...

Попытка #2 – ok google, а статьи?

7 лет

	malvira getting there.	
	math	This should build a arm
	runtime	This should build a arm
	README.md	getting there.
	build.sh	getting there.
	clean.sh	get the assembly code
	header-bin	getting there.
	main.go	getting there.

Golang Internals, Part 3: The Linker, Object Files, and Relocations



by SIARHEI MATSIUKEVICH
MARCH 11, 2015

7 years ago

Отрываем Runtime...

Попытка #2 – попробуем!

Отрываем Runtime...

Попытка #2 – попробуем!

- `runtime/runtime.go`
- `main.go`

Отрываем Runtime...

Попытка #2 – попробуем!

- runtime/runtime.go
- main.go

1. Компилируем runtime:

```
go tool compile -pack runtime.go => runtime.a
```

2. Компилируем main.go:

```
go tool compile main.go => main.o
```

3. Линкуем!

```
go tool link -w -c -n -v -v -L runtime main.o
```

Отрываем Runtime...

Попытка #2 – попробуем!

- runtime/runtime.go
- main.go

1. Компилируем runtime:

```
go tool compile -pack runtime.go => runtime.a
```

Отрываем Runtime...

Попытка #2 – попробуем!

- runtime/runtime.go
- main.go

1. Компилируем runtime:

```
go tool compile -pack runtime.go => runtime.a
```

2. Компилируем main.go:

```
go tool compile main.go => main.o
```

Отрываем Runtime...

Попытка #2 – попробуем!

- runtime/runtime.go
- main.go

1. Компилируем runtime:

```
go tool compile -pack runtime.go => runtime.a
```

2. Компилируем main.go:

```
go tool compile main.go => main.o
```

3. Линкуем!

```
go tool link -w -c -n -v -v -L runtime main.o
```

Отрываем Runtime...

Попытка #2 – попробуем! Линкуем!

```
go tool link -w -c -n -v -v -L runtime main.o
```

```
34 runtime.functab @ SPCLNTAB
```

```
panic: runtime error: index out of range [0] with length 0
```

```
goroutine 1 [running]:
```

```
cmd/link/internal/ld.(*Link).findfunctab(0xc00014e000?, 0x67610a?, {0xc0006d81b0?,  
0x14?, 0xc00001c195?})
```

```
    /usr/lib/go/src/cmd/link/internal/ld/pcln.go:825 +0x28f
```

```
cmd/link/internal/ld.Main(_, {0x20, 0x20, 0x1, 0x7, 0x10, 0x0, {0x0, 0x0}, {0x67f585,  
...}, ...})
```

```
    /usr/lib/go/src/cmd/link/internal/ld/main.go:325 +0x133e
```

```
main.main()
```

```
    /usr/lib/go/src/cmd/link/main.go:69 +0xfe5
```

Отрываем Runtime...

Попытка #2 – попробуем! Линкуем! **Паникуем!**

```
go tool link -w -c -n -v -v -L runtime main.o
```

```
34 runtime.functab @ SPCLNTAB
```

```
panic: runtime error: index out of range [0] with length 0
```

```
goroutine 1 [running]:
```

```
cmd/link/internal/ld.(*Link).findfunctab(0xc00014e000?, 0x67610a?, {0xc0006d81b0?,  
0x14?, 0xc00001c195?})
```

```
    /usr/lib/go/src/cmd/link/internal/ld/pcln.go:825 +0x28f
```

```
cmd/link/internal/ld.Main(_, {0x20, 0x20, 0x1, 0x7, 0x10, 0x0, {0x0, 0x0}, {0x67f585,  
...}, ...})
```

```
    /usr/lib/go/src/cmd/link/internal/ld/main.go:325 +0x133e
```

```
main.main()
```

```
    /usr/lib/go/src/cmd/link/main.go:69 +0xfe5
```

Отрываем Runtime...

Попытка #2 – попробуем! Линкуем! **Паникуем!**

Линкер вместо внятной диагностики в любой непонятной ситуации впадает в панику.

Отрываем Runtime...

Попытка #2 – попробуем! Линкуем! **Паникуем!**

GNU ld линкер



golang линкер



Отрываем Runtime...

Попытка #3 – допиливаем рантайм самостоятельно.

Отрываем Runtime...

Попытка #3 – допиливаем рантайм самостоятельно.

```
36 // moduledata records information about the layout of the executable
37 // image. It is written by the linker. Any changes here must be
38 // matched changes to the code in cmd/internal/ld/symtab.go:symtab.
39 // moduledata is stored in read-only memory; none of the pointers here
40 // are visible to the garbage collector.
41 type moduledata struct {
42     pclntable []byte
43     ftab      []functab
44     filetab   []uint32
45     findfunctab uintptr
46     minpc, maxpc uintptr
```

Отрываем Runtime...

Попытка #3 – допиливаем рантайм самостоятельно.

Нужно синхронизировать руками. Каждый релиз.

```
36 // moduledata records information about the layout of the executable
37 // image. It is written by the linker. Any changes here must be
38 // matched changes to the code in cmd/internal/ld/symtab.go:symtab.
39 // moduledata is stored in read-only memory; none of the pointers here
40 // are visible to the garbage collector.
41 type moduledata struct {
42     pclntable []byte
43     ftab      []functab
44     filetab   []uint32
45     findfunctab uintptr
46     minpc, maxpc uintptr
```

Отрываем Runtime...

Попытка #3 – допиливаем рантайм самостоятельно.

За 7 лет в moduledata добавилось полей... И зависимостей.

Добавляем...

Отрываем Runtime...

Попытка #3 – допиливаем рантайм самостоятельно.



Отрываем Runtime...

Попытка #3 – допиливаем рантайм самостоятельно.

Было 132 строчки кода в runtime.go

Отрываем Runtime...

Попытка #3 – допиливаем рантайм самостоятельно.

Было 132 строчки кода в runtime.go

Стало 220 строк

Отрываем Runtime...

Попытка #3 – допиливаем рантайм самостоятельно.

Было 132 строчки кода в runtime.go

Стало 220 строк

А! И не забыть добавить аннотацию `//go:nosplit` к функциям и в `main` и в `runtime`

```
//go:linkname _rt0_arm_linux _rt0_arm_linux
func _rt0_arm_linux() {
|   main()
}
```



```
//go:linkname _rt0_amd64_linux _rt0_amd64_linux
//go:nosplit
func _rt0_amd64_linux() {
|   main()
}
```

Отрываем Runtime...

Попытка #3 – допиливаем рантайм самостоятельно.

Было 132 строчки кода в runtime.go

Стало 220 строк

А! И не забыть добавить анотацию `//go:nosplit` к функциям и в `main` и в `runtime`

```
//go:linkname _rt0_arm_linux _rt0_arm_linux
func _rt0_arm_linux() {
|   main()
}
```



```
//go:linkname _rt0_amd64_linux _rt0_amd64_linux
//go:nosplit
func _rt0_amd64_linux() {
|   main()
}
```

Отрываем Runtime...

Попытка #3 – допиливаем рантайм самостоятельно.

Теперь всё собирается, линкуется.

Отрываем Runtime...

Попытка #3 – допиливаем рантайм самостоятельно.

Теперь всё собирается, линкуется.

Но сегфолтится.

Отрываем Runtime...

Попытка #3 – допиливаем рантайм самостоятельно.

Теперь всё собирается, линкуется.

Но сегфолтится => надо добавить завершение приложения.

Иначе – stack **underflow**

Отрываем Runtime...

Попытка #3 – допиливаем рантайм самостоятельно.

Теперь всё собирается, линкуется.

Но сегфолтится => надо добавить завершение приложения.

Иначе – stack **underflow**

```
//go:linkname _rt0_amd64_linux _rt0_amd64_linux
//go:nosplit
func _rt0_amd64_linux() {
    main()
    Syscall6(60, 0, 0, 0, 0, 0, 0)
}
```

Отрываем Runtime...

Попытка #3 – допиливаем рантайм самостоятельно.

```
//go:linkname _rt0_amd64_linux _rt0_amd64_linux
//go:nosplit
func _rt0_amd64_linux() {
    main()
    Syscall6(60, 0, 0, 0, 0, 0, 0)
}
```

```
TEXT ·Syscall6<ABIInternal>(SB),NOSPLIT,$0
    // a6 already in R9.
    // a5 already in R8.
    MOVQ    SI, R10 // a4
    MOVQ    DI, DX  // a3
    MOVQ    CX, SI  // a2
    MOVQ    BX, DI  // a1
    // num already in AX.
    SYSCALL
    CMPQ    AX, $0xffffffffffff001
    JLS ok
    NEGQ    AX
    MOVQ    AX, CX  // errno
    MOVQ    $-1, AX // r1
    MOVQ    $0, BX  // r2
    RET
ok:
    // r1 already in AX.
    MOVQ    DX, BX // r2
    MOVQ    $0, CX // errno
    RET
```

Отрываем Runtime...

Попытка #3 – допиливаем рантайм самостоятельно.

Приложение, которое делает НИЧЕГО

```
//go:linkname _rt0_amd64_linux _rt0_amd64_linux
//go:nosplit
func _rt0_amd64_linux() {
    main()
    Syscall6(60, 0, 0, 0, 0, 0, 0)
}
```

```
func main() {}
```

```
TEXT ·Syscall6<ABIInternal>(SB),NOSPLIT,$0
    // a6 already in R9.
    // a5 already in R8.
    MOVQ    SI, R10 // a4
    MOVQ    DI, DX  // a3
    MOVQ    CX, SI  // a2
    MOVQ    BX, DI  // a1
    // num already in AX.
    SYSCALL
    CMPQ    AX, $0xffffffffffff001
    JLS    ok
    NEGQ    AX
    MOVQ    AX, CX // errno
    MOVQ    $-1, AX // r1
    MOVQ    $0, BX // r2
    RET
ok:
    // r1 already in AX.
    MOVQ    DX, BX // r2
    MOVQ    $0, CX // errno
    RET
```

Отрываем Runtime...

Попытка #3 – Hello world

Надо добавить печать

Отрываем Runtime...

Попытка #3 – Hello world

Надо добавить печать

Печать – через системный вызов `write`

Отрываем Runtime...

Попытка #3 – Hello world

Надо добавить печать

Печать – через системный вызов `write`

Удобную функцию печати – чтобы туда строку, а она печатает.

Отрываем Runtime...

Попытка #3 – Hello world

Надо добавить печать

Печать – через системный вызов write

Удобную функцию печати – чтобы туда строку, а она печатает.

Указатель на начало данных добудем через небезопасные касты, а len добудем через функцию len()

```
type _string struct {  
    elements *byte // underlying bytes  
    len      int   // number of bytes  
}
```

Отрываем Runtime...

Попытка #3 – Hello world

Надо добавить печать

Печать – через системный вызов write

Удобную функцию печати – чтобы туда строку, а она печатает.

Указатель на начало данных добудем через небезопасные касты, а len добудем через функцию len()

```
type _string struct {
    elements *byte // underlying bytes
    len      int   // number of bytes
}

//go:nosplit
func print(str string) {
    var s *byte
    s = (**byte)(unsafe.Pointer(&str))
    runtime.Syscall6(1, 1, uintptr(unsafe.Pointer(s)), uintptr(len(str)), 0, 0, 0)
}
```

Отрываем Runtime...

Попытка #3 – Hello world – результат

```
package main

import (
    "runtime"
    "unsafe" // needed for go:linkname
)

//go:nosplit
func print(str string) {
    var s *byte
    s = *(*byte)(unsafe.Pointer(&str))
    s = s
    runtime.Syscall6(1, 1, uintptr(unsafe.Pointer(s)), uintptr(len(str)), 0, 0, 0)
}

//go:nosplit
func main() {
    print("Hello world\n")
}
```

Отрываем Runtime...

Попытка #3 – Hello world – результат

Размер: 22071 байт

```
package main

import (
    "runtime"
    "unsafe" // needed for go:linkname
)

//go:nosplit
func print(str string) {
    var s *byte
    s = (**byte)(unsafe.Pointer(&str))
    s = s
    runtime.Syscall6(1, 1, uintptr(unsafe.Pointer(s)), uintptr(len(str)), 0, 0, 0)
}

//go:nosplit
func main() {
    print("Hello world\n")
}
```

Отрываем Runtime...

Попытка #3 – Hello world – результат

```
package main
```

```
import (  
    "runtime"  
    "unsafe" // needed for go:linkname  
)
```

```
//go:nosplit
```

```
func print(str string) {  
    var s *byte  
    s = *(*byte)(unsafe.Pointer(&str))  
    s = s  
    runtime.Syscall6(1, 1, uintptr(unsafe.Pointer(s)), uintptr(len(str)), 0, 0, 0)  
}
```

```
//go:nosplit
```

```
func main() {  
    print("Hello world\n")  
}
```

Размер: 22071 байт

Размер: 19264 байт stripped

Отрываем Runtime...

Попытка #3 – Hello world – результат

```
package main

import (
    "runtime"
    "unsafe" // needed for go:linkname
)

//go:nosplit
func print(str string) {
    var s *byte
    s = *(*byte)(unsafe.Pointer(&str))
    s = s
    runtime.Syscall6(1, 1, uintptr(unsafe.Pointer(s)), uintptr(len(str)), 0, 0, 0)
}

//go:nosplit
func main() {
    print("Hello world\n")
}
```

```
package main

import "fmt"

func main() {
    fmt.Println("Hello world")
}
```

Размер: 22071 байт

Размер: 19264 байт stripped

Обычный golang hello world:

Размер: 1758436 байт

Размер: 1180440 байт stripped

Напишем ASan-mini

Подробнее как устроен ASan тут: <https://youtu.be/7WyBAUJ8UA8>



Напишем ASan-mini

План:

Напишем ASan-mini

План:

1. Собрать .so (будет libasan.so)

Напишем ASan-mini

План:

1. Собрать .so (будет libasan.so)
2. Попробовать из Си-приложения вызвать какую-нибудь функцию

Напишем ASan-mini

План:

1. Собрать .so (будет libasan.so)
2. Попробовать из Си-приложения вызвать какую-нибудь функцию
3. Реализовать минимальное подмножество функций ASan'a:
 1. malloc
 2. free
 3. __asan_option_detect_stack_use_after_return
 4. __asan_init()
 5. __asan_stack_malloc_1()
 6. __asan_report_store4()
 7. __asan_before_dynamic_init()
 8. __asan_after_dynamic_init()
 9. __asan_unregister_globals()
 10. __asan_register_globals()
 11. __asan_unregister_globals()
 12. __asan_register_globals()
 13. __asan_version_mismatch_check_v8()
4. Попробовать взлететь

Напишем ASan-mini

1. Собрать .so (будет libasan.so)

А не собирается – `buildmode=c-shared` требует чего-то изоциренного от сго подмодуля. А линкер опять паникует.

Напишем ASan-mini

1. Собрать .so (будет libasan.so)

А не собирается – `buildmode=c-shared` требует чего-то изощренного от сго подмодуля. А линкер опять паникует.

Но нам это никак не мешает – в Linux elf executable мало чем отличается от .so. Просто допаяем напильником.

Напишем ASan-mini

1. Собрать .so (будет libasan.so)

А не собирается – `buildmode=c-shared` требует чего-то изощенного от сго подмодуля. А линкер опять паникует.

Но нам это никак не мешает – в Linux elf executable мало чем отличается от .so. Просто допаяем напильником.

Напишем ASan-mini

1. Собрать .so (будет libasan.so)

А не собирается – `buildmode=c-shared` требует чего-то изощренного от cgo подмодуля. А линкер опять паникует.

Но нам это никак не мешает – в Linux elf executable мало чем отличается от .so. Просто допаяем напильником.

Берем lief: https://lief-project.github.io/doc/latest/tutorials/08_elf_bin2lib.html

И рождаем вот такой питоновский скрипт (заодно и все нужные функции экспортируем как надо):

Напишем ASan-mini

1. Собрать .so (будет libasan.so)

Берем lief: https://lief-project.github.io/doc/latest/tutorials/08_elf_bin2lib.html

И рождаем вот такой питоновский скрипт (заодно и все нужные функции экспортируем как надо):

```
2 binary = lief.parse("./a.out")
3 binary[lief.ELF.DYNAMIC_TAGS.FLAGS_1].remove(lief.ELF.DYNAMIC_FLAGS_1.PIE)
4 export_funcs = ['something',
5                 'malloc',
6                 'free',
7                 '__asan_init',
8                 '#__asan_option_detect_stack_use_after_return',
9                 '__asan_stack_malloc_1',
10                '__asan_report_store4',
11                '__asan_before_dynamic_init',
12                '__asan_after_dynamic_init',
13                '__asan_unregister_globals',
14                '__asan_version_mismatch_check_v8',
15                '__asan_register_globals']
16 for name in export_funcs:
17     for x in binary.exported_functions:
18         if x.name==name:
19             binary.add_exported_function(x.address, name)
20             print(name + " has been added")
21             break
22 binary.write("libasan.so")
```

Напишем ASan-mini

2. Попробовать из Си-приложения вызвать какую-нибудь функцию

```
func Something(a int) int {  
    return a + 100  
}
```

Напишем ASan-mini

2. Попробовать из Си-приложения вызвать какую-нибудь функцию

```
//go:linkname Something something
//go:nosplit
func Something(a int) int {
|   return a + 100
}
```

Напишем ASan-mini

2. Попробовать из Си-приложения вызвать какую-нибудь функцию

```
//go:linkname Something something
//go:nosplit
func Something(a int) int {
    return a + 100
}

#include <stdio.h>

extern int something(int);

int main() {
    printf("%d\n", something(100));
}
```

Напишем ASan-mini

2. Попробовать из Си-приложения вызвать какую-нибудь функцию

```
//go:linkname Something something
//go:nosplit
func Something(a int) int {
    return a + 100
}

#include <stdio.h>

extern int something(int);

int main() {
    printf("%d\n", something(100));
}
```

Вместо значений – мусор.

Напишем ASan-mini

2. Попробовать из Си-приложения вызвать какую-нибудь функцию

Вместо значений – мусор.

У golang другой calling convention.

Напишем ASan-mini

2. Попробовать из Си-приложения вызвать какую-нибудь функцию

Вместо значений – мусор.

У go lang другой calling convention. На самом деле их даже два:
ABI0 и AbiInternal

Напишем ASan-mini

2. Попробовать из Си-приложения вызвать какую-нибудь функцию

Вместо значений – мусор.

У golang другой calling convention. На самом деле их даже два:
ABI0 и AbiInternal

Си-код кладет аргумент в такой регистр, в котором Go-функция его не ожидает.

Напишем ASan-mini

2. Попробовать из Си-приложения вызвать какую-нибудь функцию

Вместо значений – мусор.

У `golang` другой `calling convention`. На самом деле их даже два:
`ABI0` и `AbiInternal`

Си-код кладет аргумент в такой регистр, в котором Go-функция его не ожидает.

Инлайн-ассемблера тоже нет, инлайновых вставок ассемблера тоже нет.

Directed by
ROBERT B. WEIDE

Напишем ASan-mini

2. Попробовать из Си-приложения вызвать какую-нибудь функцию

Просто возьмём и переложим:

```
TEXT ·Adjust1Args<ABIInternal>(SB),NOSPLIT,$0
    MOVQ DI, AX
    RET
```

```
TEXT ·Adjust2Args<ABIInternal>(SB),NOSPLIT,$0
    MOVQ DI, AX
    MOVQ SI, BX
    RET
```

Напишем ASan-mini

2. Попробовать из Си-приложения вызвать какую-нибудь функцию

Просто возьмём и переложим:

```
TEXT ·Adjust1Args<ABIInternal>(SB),NOSPLIT,$0
    MOVQ DI, AX
    RET

TEXT ·Adjust2Args<ABIInternal>(SB),NOSPLIT,$0
    MOVQ DI, AX
    MOVQ SI, BX
    RET
```

```
//go:linkname Something something
//go:nosplit
func Something(a int) int {
    a = (int)(runtime.Adjust1Args())
    return a + 100
}
```

Напишем ASan-mini

3. Реализовать минимальное подмножество функций ASan'a:

1. `malloc`
2. `free`
3. `__asan_option_detect_stack_use_after_return`
4. `__asan_init()`
5. `__asan_stack_malloc_1()`
6. `__asan_report_store4()`
7. `__asan_before_dynamic_init()`
8. `__asan_after_dynamic_init()`
9. `__asan_unregister_globals()`
10. `__asan_register_globals()`
11. `__asan_unregister_globals()`
12. `__asan_register_globals()`
13. `__asan_version_mismatch_check_v8()`

Напишем ASan-mini

3. Реализовать минимальное подмножество функций ASan'a:

1. `malloc`
2. `free`
3. `__asan_option_detect_stack_use_after_return`
4. `__asan_init()`
5. `__asan_stack_malloc_1()`
6. `__asan_report_store4()`
7. `__asan_before_dynamic_init()`
8. `__asan_after_dynamic_init()`
9. `__asan_unregister_globals()`
10. `__asan_register_globals()`
11. `__asan_unregister_globals()`
12. `__asan_register_globals()`
13. `__asan_version_mismatch_check_v8()`

Напишем ASan-mini

3. Реализовать минимальное подмножество функций ASan'a:

malloc

```
//go:linkname Malloc malloc
//go:nosplit
func Malloc(l uint64) uintptr {
    l = runtime.Adjust1Args()
    new_addr := memblock
    markShadow(new_addr, l, 0x0)
    if l%8 != 0 {
        markShadow(new_addr+uintptr((l/8)*8), uint64(8), uint8(l%8))
    }
    memblock += uintptr(((l / 8) * 8) + 8)
    markShadow(memblock, 512, 0xff)
    memblock += 512

    return new_addr
}
```

Напишем ASan-mini

3. Реализовать минимальное подмножество функций ASan'a:

free

```
//go:linkname Free free
//go:nosplit
func Free(addr uintptr) {
    i := uint64(0)
    for a := toShadow(addr); *(*uint8)(unsafe.Pointer(addr))=0xff; a++ {
        *(*uint8)(unsafe.Pointer(addr)) = 0xff
    }
}
```

Напишем ASan-mini

3. Реализовать минимальное подмножество функций ASan'a:

`__asan_init()`

```
//go:linkname Asan_init __asan_init
//go:nosplit
func Asan_init() {
    if memblock == 0 {
        mmap(0x7fff8000, 0x8fff6fff-0x7fff8000, PROT_WRITE|PROT_READ, MAP_PRIVATE|MAP_ANONYMOUS|MAP_NORESERVE, 0, 0)
        mmap(0x8fff7000, 0x2008fff6fff-0x8fff7000, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_NORESERVE, 0, 0)
        mmap(0x2008fff7000, 0x10007fff7fff-0x2008fff7000, PROT_WRITE|PROT_READ, MAP_PRIVATE|MAP_ANONYMOUS|MAP_NORESERVE, 0, 0)
        memblock = mmap(0, 1024*1024*1024, PROT_WRITE|PROT_READ, MAP_PRIVATE|MAP_ANONYMOUS|MAP_NORESERVE, 0, 0)
        markShadow(memblock, 512, 0xff)
        memblock += 512
    }
}
```

Напишем ASan-mini

3. Реализовать минимальное подмножество функций ASan'a:

`__asan_report_store4()`

```
//go:linkname Asan_report_store4 __asan_report_store4
//go:nosplit
func Asan_report_store4(addr uintptr) {
    print("store error\n")
    exit(6)
}
```

Напишем ASan-mini

4. Попробовать взлететь

Напишем ASan-mini

4. Попробовать взлететь

Собираем модельное приложение. Запускаем...

Напишем ASan-mini

4. Попробовать взлететь

Собираем модельное приложение. Запускаем...

Падает не доходя до main'a

Напишем ASan-mini

4. Попробовать взлететь

Собираем модельное приложение. Запускаем...

Падает не доходя до main'a

fix:

```
//go:linkname Malloc malloc
//go:nosplit
func Malloc(l uint64) uintptr {
    l = runtime.Adjust1Args()
    if memblock == 0 {
        mmap(0x7fff8000, 0x8fff6fff-0x7fff8000, PROT_WRITE|PROT_READ, MAP_PRIVATE|MAP_ANONYMOUS|MAP_NORESERVE, 0, 0)
        mmap(0x8fff7000, 0x2008fff6fff-0x8fff7000, PROT_NONE, MAP_PRIVATE|MAP_ANONYMOUS|MAP_NORESERVE, 0, 0)
        mmap(0x2008fff7000, 0x10007fff7fff-0x2008fff7000, PROT_WRITE|PROT_READ, MAP_PRIVATE|MAP_ANONYMOUS|MAP_NORESERVE, 0, 0)
        memblock = mmap(0, 1024*1024*1024, PROT_WRITE|PROT_READ, MAP_PRIVATE|MAP_ANONYMOUS|MAP_NORESERVE, 0, 0)
        markShadow(memblock, 512, 0xff)
        memblock += 512
    }
    new_addr := memblock
}

//go:linkname Asan_init __asan_init
//go:nosplit
func Asan_init() {
}
```

Напишем ASan-mini

4. Попробовать взлететь

Пересобираем. Собираем модельное приложение. Запускаем...

```
int main()
{
    uint32_t* res = malloc(8);
    printf("a\n");
    res[-1] = 12;
    printf("b\n");
    return 0;
}
```

Напишем ASan-mini

4. Попробовать взлететь

Пересобираем. Собираем модельное приложение. Запускаем...

```
int main()
{
    uint32_t* res = malloc(8);
    printf("a\n");
    res[-1] = 12;
    printf("b\n");
    return 0;
}
```

```
$ ./capp
a
store error
```

Напишем ASan-mini

5. ASan-mini. Результаты

Golang ASan: 27152 байт

Pure C ASan: 17472 байт

Напишем ASan-mini

5. ASan-mini. Результаты

Golang ASan: 27152 байт

Pure C ASan: 17472 байт

С падением что-то надо делать – mmap в первый malloc может быть слишком поздно.

Напишем ASan-mini

5. ASan-mini. Результаты

Golang ASan: 27152 байт

Pure C ASan: 17472 байт

С падением что-то надо делать – mmap в первый malloc может быть слишком поздно.

```
[valexey@vlinux ~]$ sudo pacman -S gcc-go  
resolving dependencies...  
looking for conflicting packages...  
:: gcc-go and go are in conflict. Remove go? [y/N]
```

Выводы



джентльмен
@sasstemir

Тред про то, что надо знать при ловле уличных котят:

1. Они сильнее, чем ты думаешь
2. Они ловчее и быстрее, чем ты думаешь
3. Укусы не будут слабым
4. Им нечего терять
5. Они все еще высшие хищники
6. Береги глаза
7. Беги

Выводы

1. Можно
2. Больно
3. Но не из за языка, а тулинга и применений.
4. Системный язык должен быть интегрирован в систему.
Вместе со своим тулингом.



джентльмен
@sasstemir

Тред про то, что надо знать при ловле уличных котят:

1. Они сильнее, чем ты думаешь
2. Они ловчее и быстрее, чем ты думаешь
3. Укусы не будут слабым
4. Им нечего терять
5. Они все еще высшие хищники
6. Береги глаза
7. Беги

Спасибо!

Вопросы?