GHC Runtime Linker

by Example

DNS is likely some of the most broadly deployed, yet most poorly understood software components deployed in the world today. What else is as common but less understood?

Timothy Perrett @timperrett
2:12am - 19 Apr 2019

Linkers twitter.com/timperrett/sta…

Gabriel Gonzalez @GabrielG439
1:32pm - 19 Apr 2019

Linking

+

Runtime

=

?

GHCi ⬅ Byte Code ⬅ Source Code

↖? Compiled Code ↙

Do we even need both?

```haskell
module FibSlow where

fib :: Int -> Int
fib 0 = 1
fib 1 = 1
fib n = fib (n - 1) + fib (n - 2)
```

GHCi

```
λ> :set +s
λ> :load *haskell/FibSlow
[1 of 1] Compiling Fib     ( haskell/FibSlow.hs, interpreted )
λ> fib 35
9227465
(7.14 secs, 4,897,229,624 bytes)
λ> :load haskell/FibSlow
λ> fib 35
9227465
(0.50 secs, 2,150,044,856 bytes)
```

GHCi

```
λ> :set +s
λ> :load *haskell/FibSlow
[1 of 1] Compiling Fib    ( haskell/FibSlow.hs, interpreted )
λ> fib 35
9227465
(7.14 secs, 4,897,229,624 bytes)
λ> :load haskell/FibSlow
λ> fib 35
9227465
(0.50 secs, 2,150,044,856 bytes)
```

Plan

1

Linkers

Simple in theory, complex in practice.

2

Linking in runtime

There's more than one way to do it[2]

[2] **Two** ways to be precise: dynamic and static.

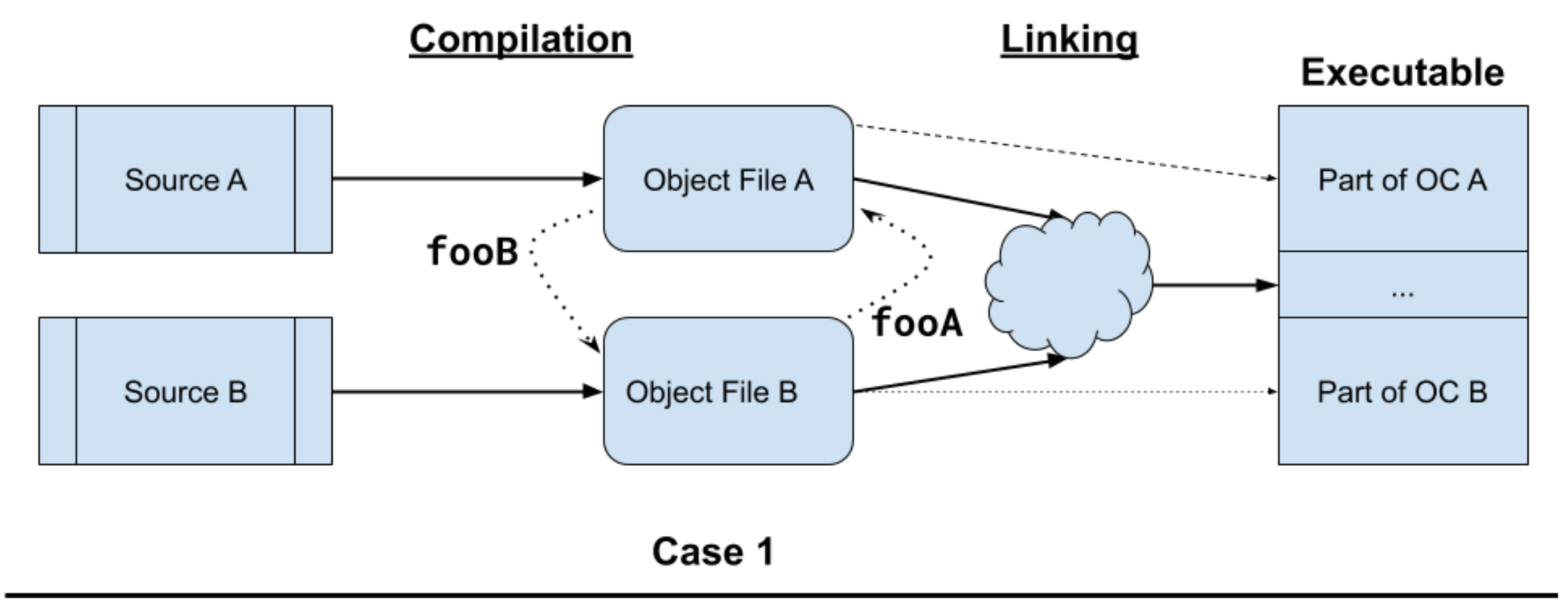
3



... or in other words: pros, cons, and challanges.

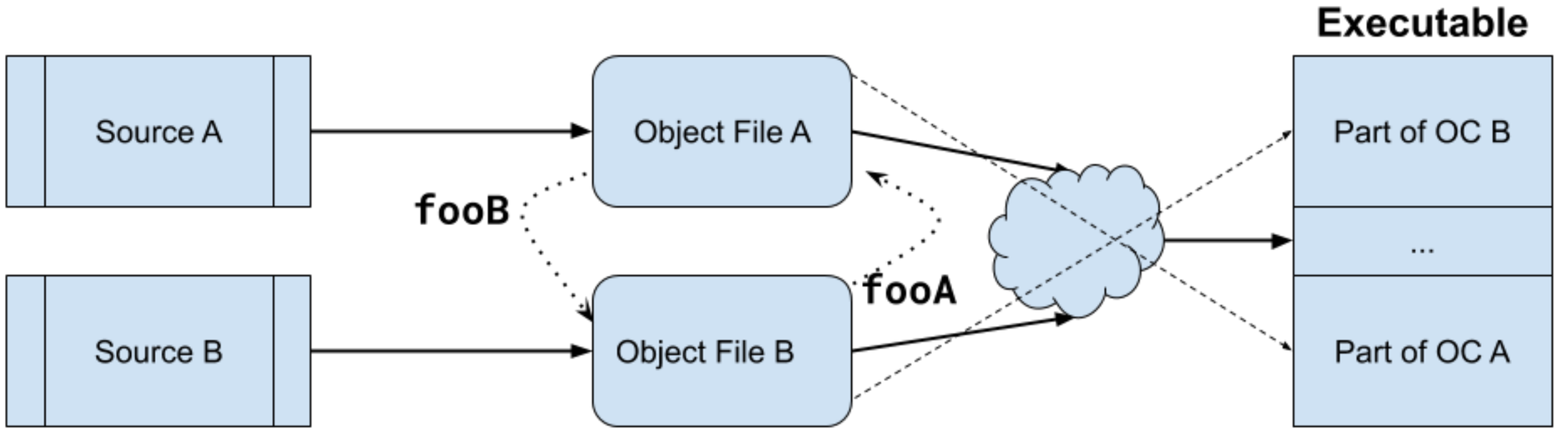
1

ld = link editor


Compilation
Linking
Executable
Source A
Source B
Object File A
Object File B
fooB
fooA
Part of OC A
...
Part of OC B

Case 1

Case 2


Executable
Source A
Source B
Object File A
Object File B
fooB
fooA
Part of OC B
...
Part of OC A

file:c/magic.c

```c
int MAGIC = 42;

int magic(int x)
{
        return x + MAGIC;
}
```

```
c/magic.o:
(__TEXT,__text) section
_magic:
       0:   55  pushq   %rbp
       1:   48 89 e5    movq    %rsp, %rbp
       4:   03 3d 00 00 00 00   addl    _MAGIC(%rip), %edi
       a:   89 f8   movl    %edi, %eax
       c:   5d  popq    %rbp
       d:   c3  retq

SYMBOL TABLE:
0000000000000010 g       __DATA,__data  _MAGIC
0000000000000000 g     F __TEXT,__text  _magic

Contents of (__DATA,__data) section
0000000000000010    2a 00 00 00
```

```
c/magic.o:
(__TEXT,__text) section
_magic:
       0:   55  pushq   %rbp
       1:   48 89 e5    movq    %rsp, %rbp
       4:   03 3d 00 00 00 00   addl    _MAGIC(%rip), %edi
       a:   89 f8   movl    %edi, %eax
       c:   5d  popq    %rbp
       d:   c3  retq

SYMBOL TABLE:
0000000000000010 g       __DATA,__data  _MAGIC
0000000000000000 g     F __TEXT,__text  _magic

Contents of (__DATA,__data) section
0000000000000010    2a 00 00 00
```

```
c/magic.o:
(__TEXT,__text) section
_magic:
       0:   55  pushq   %rbp
       1:   48 89 e5    movq    %rsp, %rbp
       4:   03 3d 00 00 00 00   addl    _MAGIC(%rip), %edi
       a:   89 f8   movl    %edi, %eax
       c:   5d  popq    %rbp
       d:   c3  retq

SYMBOL TABLE:
0000000000000010 g       __DATA,__data  _MAGIC
0000000000000000 g     F __TEXT,__text  _magic

Contents of (__DATA,__data) section
0000000000000010    2a 00 00 00
```

```
4:  03 3d 00 00 00 00   addl    _MAGIC(%rip), %edi
```

Bytes	Value	Meaning
1	`03`	`ADD` opcode
2	`3D = b00_111_101`	addressing mode + register
3-6 (4)	0 stub	32bit offset from `%rip`

```
4:  03 3d 00 00 00 00    addl    _MAGIC(%rip), %edi
                           ==
4:  03 3d ?? ?? ?? ??    addl    _MAGIC(%rip), %edi
```

```
c/magic.o:
(__TEXT,__text) section
_magic:
       0:   55  pushq   %rbp
       1:   48 89 e5    movq    %rsp, %rbp
       4:   03 3d 00 00 00 00   addl    _MAGIC(%rip), %edi
       a:   89 f8   movl    %edi, %eax
       c:   5d  popq    %rbp
       d:   c3  retq

SYMBOL TABLE:
0000000000000010 g       __DATA,__data  _MAGIC
0000000000000000 g     F __TEXT,__text  _magic

Relocation information (__TEXT,__text) 1 entries
address  pcrel length extern type    scattered symbolnum/value
00000006 True  long   True   SIGNED  False     _MAGIC
```

file:c/array_magic.c

```c
extern int magic(int);

void array_magic(int* arr, int len)
{
        for (int i = 0; i < len; i++) {
                arr[i] = magic(arr[i]);
        }
}
```

```
void array_magic(int* arr, int len); // rdi <- arr; esi <- len


(__TEXT,__text) section
_array_magic:
...
       a:   85 f6   testl   %esi, %esi
       c:   7e 27   jle 0x35
       e:   49 89 ff    movq    %rdi, %r15
      11:   41 89 f6    movl    %esi, %r14d
      14:   31 db   xorl    %ebx, %ebx
...
      20:   41 8b 3c 9f     movl    (%r15,%rbx,4), %edi
      24:   e8 00 00 00 00  callq   _magic
      29:   41 89 04 9f     movl    %eax, (%r15,%rbx,4)
      2d:   48 ff c3    incq    %rbx
      30:   49 39 de    cmpq    %rbx, %r14
      33:   75 eb   jne 0x20
      35:   48 83 c4 08     addq    $8, %rsp

SYMBOL TABLE:
0000000000000000 g     F __TEXT,__text  _array_magic
0000000000000000           *UND*  _magic
Relocation information (__TEXT,__text) 1 entries
address  pcrel length extern type    scattered symbolnum/value
00000025 True  long   True   BRANCH  False     _magic
```

```
void array_magic(int* arr, int len); // rdi <- arr; esi <- len


(__TEXT,__text) section
_array_magic:
...
       a:   85 f6   testl   %esi, %esi
       c:   7e 27   jle 0x35
       e:   49 89 ff    movq    %rdi, %r15
      11:   41 89 f6    movl    %esi, %r14d
      14:   31 db   xorl    %ebx, %ebx
...
      20:   41 8b 3c 9f     movl    (%r15,%rbx,4), %edi
      24:   e8 00 00 00 00  callq   _magic
      29:   41 89 04 9f     movl    %eax, (%r15,%rbx,4)
      2d:   48 ff c3    incq    %rbx
      30:   49 39 de    cmpq    %rbx, %r14
      33:   75 eb   jne 0x20
      35:   48 83 c4 08     addq    $8, %rsp

SYMBOL TABLE:
0000000000000000 g     F __TEXT,__text  _array_magic
0000000000000000         *UND*  _magic
Relocation information (__TEXT,__text) 1 entries
address  pcrel length extern type    scattered symbolnum/value
00000025 True  long   True   BRANCH  False     _magic
```

```
c:  7e 27    jle 0x35
e:  49 89 ff    movq    %rdi, %r15
...
35: 48 83 c4 08     addq    $8, %rsp
...
```

→ `7e` is opcode for `JLE rel8` instruction

→ `JLE` takes a **relative offset** from the program counter

→ `e + 27 = 35`

```
24: e8 00 00 00 00  callq   _magic
    ^  ^----------
    |            |
    |            32bit 0-stub
    |
    near call, 32bit displacement relative to next instr.

Relocation information (__TEXT,__text) 1 entries
address  pcrel length extern type    scattered symbolnum/value
00000025 True  long   True   BRANCH  False     _magic
```

`magic.o` and `array_magic.o` combined

```
(__TEXT,__text) section
_magic:
       0:   55  pushq   %rbp
       1:   48 89 e5    movq    %rsp, %rbp
       4:   03 3d ?? ?? ?? ??   addl    _MAGIC(%rip), %edi
       a:   89 f8   movl    %edi, %eax
       c:   5d  popq    %rbp
       d:   c3  retq
_array_magic:
       e:   55  pushq   %rbp
...
      2e:   41 8b 3c 9f     movl    _array_magic(%r15,%rbx,4), %edi
      32:   e8 ?? ?? ?? ??  callq   _magic
      37:   41 89 04 9f     movl    %eax, _array_magic(%r15,%rbx,4)
...
      4d:   c3  retq
Contents of (__DATA,__data) section
0000000000000050    2a 00 00 00
```

2

Runtime Linking in GHC

-dynamic~

~ *dynamic way = haskell libraries and RTS are linked dynamically*

```haskell
module Foo (foo) where

foo x = x * 3
```

```
$ ghc -c Foo.hs # produces Foo.o
$ ghci
λ> :l Foo
... # Will Foo.o get used?
```

```haskell
module Foo (foo) where

foo x = x * 3
```

```
$ ghc -c Foo.hs
$ ghci
λ> :l Foo
[1 of 1] Compiling Foo                ( hs/Foo.hs, interpreted )
Ok, one module loaded.
```

-dynamic by default

```
$ objdump -dylibs-used {path-to-ghc-bin}
{path-to-ghc-bin}:
    /usr/lib/libSystem.B.dylib
    @rpath/libHShaskeline-0.7.4.3-ghc8.6.5.dylib
    @rpath/libHSghc-8.6.5-ghc8.6.5.dylib
...
    @rpath/libHSrts_thr-ghc8.6.5.dylib
```

```
$ ghc -c -dynamic Foo.hs
$ ghci
λ> :l Foo
Ok, one module loaded.
Collecting type info for 1 module(s) ...
```

diff -u static/Foo.s dyn/Foo.s

```
-static/Foo.s:
+dyn/Foo.s:
...
-    leaq    _stg_ap_pp_info(%rip), %rax
+    movq    _stg_ap_pp_info(%rip), %rax
...
-    jmp _base_GHCziNum_zt_info
+    jmpq    *_base_GHCziNum_zt_info(%rip)
...
```

diff -u <(objdump -r static/Foo.o) <(objdump -r dyn/Foo.o)

```
- X86_64_RELOC_BRANCH _base_GHCziNum_zt_info
- X86_64_RELOC_SIGNED _stg_ap_pp_info
...
+ X86_64_RELOC_GOT _base_GHCziNum_zt_info@GOTPCREL
+ X86_64_RELOC_GOT_LOAD _stg_ap_pp_info@GOTPCREL
```

$ find [ghci]

./compiler/ghci

./libraries/ghci

```
compiler/ghci
├── Linker.hs
├── LinkerTypes.hs
...
```

	files	blank	comment	code
`Linker.hs`	1	262	422	1002
Total	11	1031	1735	4703

file:Linker.hs

```
dynLinkObjs hsc_env pls objs = do
        -- Load the object files and link them
        let (objs_loaded', new_objs) = rmDupLinkables (objs_loaded pls) objs
            pls1                     = pls { objs_loaded = objs_loaded' }
            unlinkeds                = concatMap linkableUnlinked new_objs
            wanted_objs              = map nameOfObject unlinkeds

        if interpreterDynamic (hsc_dflags hsc_env)
            then do pls2 <- dynLoadObjs hsc_env pls1 wanted_objs
                    return (pls2, Succeeded)
            else do mapM_ (loadObj hsc_env) wanted_objs
                    -- Link them all together
                    ok <- resolveObjs hsc_env
...
```

file:Linker.hs

```
dynLoadObjs _       pls                                  []   = return pls
dynLoadObjs hsc_env pls@PersistentLinkerState{..} objs = do
    let dflags = hsc_dflags hsc_env
    ...
    (soFile, libPath , libName) <- newTempLibName
                                       dflags
                                       TFL_CurrentModule (soExt platform)

    let dflags2 = dflags {..., outputFile = Just soFile }
    -- link all "loaded packages" so symbols in those can be resolved
    linkDynLib dflags2 objs pkgs_loaded

    -- if we got this far, extend the lifetime of the library file
    changeTempFilesLifetime dflags TFL_GhcSession [soFile]
    m <- loadDLL hsc_env soFile
    case m of
        Nothing -> return $! pls { temp_sos = (libPath, libName) : temp_sos }
        Just err -> panic ("Loading temp shared object failed: " ++ err)
```

file:Linker.hs

```
dynLoadObjs _       pls                                  []   = return pls
dynLoadObjs hsc_env pls@PersistentLinkerState{..} objs = do
    let dflags = hsc_dflags hsc_env
    ...
    (soFile, libPath , libName) <- newTempLibName
                                     dflags
                                     TFL_CurrentModule (soExt platform)

    let dflags2 = dflags {..., outputFile = Just soFile }
    -- link all "loaded packages" so symbols in those can be resolved
    linkDynLib dflags2 objs pkgs_loaded

    -- if we got this far, extend the lifetime of the library file
    changeTempFilesLifetime dflags TFL_GhcSession [soFile]
    m <- loadDLL hsc_env soFile
    case m of
        Nothing -> return $! pls { temp_sos = (libPath, libName) : temp_sos }
        Just err -> panic ("Loading temp shared object failed: " ++ err)
```

file:Linker.hs

```
dynLoadObjs _       pls                                  []   = return pls
dynLoadObjs hsc_env pls@PersistentLinkerState{..} objs = do
    let dflags = hsc_dflags hsc_env
    ...
    (soFile, libPath , libName) <- newTempLibName
                                     dflags
                                     TFL_CurrentModule (soExt platform)

    let dflags2 = dflags {..., outputFile = Just soFile }
    -- link all "loaded packages" so symbols in those can be resolved
    linkDynLib dflags2 objs pkgs_loaded

    -- if we got this far, extend the lifetime of the library file
    changeTempFilesLifetime dflags TFL_GhcSession [soFile]
    m <- loadDLL hsc_env soFile
    case m of
        Nothing -> return $! pls { temp_sos = (libPath, libName) : temp_sos }
        Just err -> panic ("Loading temp shared object failed: " ++ err)
```

Very nice!

Where's the catch?

Example

```haskell
-- file: Foo.hs
module Foo (foo) where


foo a = a * 3
```

```haskell
-- file: Bar.hs
module Bar (bar) where
import Foo

bar a = even $ foo a + 1
```

```
$ ghc --interactive +RTS -Dl

initLinker: start
...
Prelude> :l Foo Bar
Ok, two modules loaded.
Prelude Foo> :m Foo Bar
Prelude Foo Bar>
```

```
Prelude Foo Bar> foo 2
addDLL: dll_name = '[tmp-forlder]/libghc_1.dylib'
internal_dlopen: dll_name = '[tmp-folder]/libghc_1.dylib'
...
```

```
Prelude Foo Bar> bar 3
addDLL: dll_name = '[tmp-folder]/libghc_3.dylib'
internal_dlopen: dll_name = '[tmp-folder]/libghc_3.dylib'
...
```

```
$ PAT='.*F.*\(Foo_foo\|Bar_bar\).*'

$ objdump -t libghc_1.dylib | grep $PAT
l     F __TEXT,__text   _dsp__Foo_foo_info_dsp
g     F __TEXT,__text   _Foo_foo_info

$ objdump -t libghc_3.dylib | grep $PAT
l     F __TEXT,__text   _dsp__Bar_bar_info_dsp
g     F __TEXT,__text   _Bar_bar_info
```

```
Prelude Foo Bar> :l Bar
Ok, two modules loaded.
Prelude Bar> bar 4
addDLL: dll_name = '[tmp-folder]/libghc_5.dylib'
internal_dlopen: dll_name = '[tmp-folder]/libghc_5.dylib'
...
```

```
$ objdump -t libghc_5.dylib | grep $PAT
l     F __TEXT,__text   _dsp__Foo_foo_info_dsp
l     F __TEXT,__text   _dsp__Bar_bar_info_dsp
g     F __TEXT,__text   _Bar_bar_info
g     F __TEXT,__text   _Foo_foo_info
```

```
Prelude Bar> :l Bar
Ok, two modules loaded.
Prelude Bar> bar 5
addDLL: dll_name = '[tmp-folder]/libghc_7.dylib'
internal_dlopen: dll_name = '[tmp-folder]/libghc_7.dylib'
...
```

```
$ objdump -t libghc_7.dylib | grep $PAT
l     F __TEXT,__text   _dsp__Foo_foo_info_dsp
l     F __TEXT,__text   _dsp__Bar_bar_info_dsp
g     F __TEXT,__text   _Bar_bar_info
g     F __TEXT,__text   _Foo_foo_info
```

```
$ lsof -c ghc | grep libghc_
CMD     PID   FD      TYPE    NAME
ghc     91710 txt       REG    [tmp-folder]/libghc_1.dylib
ghc     91710 txt       REG    [tmp-folder]/libghc_3.dylib
ghc     91710 txt       REG    [tmp-folder]/libghc_5.dylib
ghc     91710 txt       REG    [tmp-folder]/libghc_7.dylib
```

➡ **Takes time** to find what it needs.

➡ **Never gives back** what it once received.

-vanilla*

* *vanilla way = haskell libraries and RTS are linked statically*

file:Linker.hs

```
dynLinkObjs hsc_env pls objs = do
        -- Load the object files and link them
        let (objs_loaded', new_objs) = rmDupLinkables (objs_loaded pls) objs
            pls1                     = pls { objs_loaded = objs_loaded' }
            unlinkeds                = concatMap linkableUnlinked new_objs
            wanted_objs              = map nameOfObject unlinkeds

        if interpreterDynamic (hsc_dflags hsc_env)
            then do pls2 <- dynLoadObjs hsc_env pls1 wanted_objs
                    return (pls2, Succeeded)
            else do mapM_ (loadObj hsc_env) wanted_objs
                    -- Link them all together
                    ok <- resolveObjs hsc_env
...
```

```
libraries/ghci/GHCi
├── ObjLink.hs
...
```

	files	blank	comment	code
`ObjLink`	1	27	45	123
Total	12	292	414	1437

```
rts
├── Linker.c
├── LinkerInternals.h
└── linker
    ...
    ├── ELFRelocs
    │   ...
    ├── Elf.c
    ├── Elf.h
    ├── ElfTypes.h
    ├── LoadArchive.c
    ├── M32Alloc.c
    ├── M32Alloc.h
    ├── MachO.c
    ├── MachO.h
    ├── MachOTypes.h
    ├── PEi386.c
    ├── PEi386.h
    ├── PEi386Types.h
    ├── SymbolExtras.c
    ├── SymbolExtras.h
    └── util.h
```

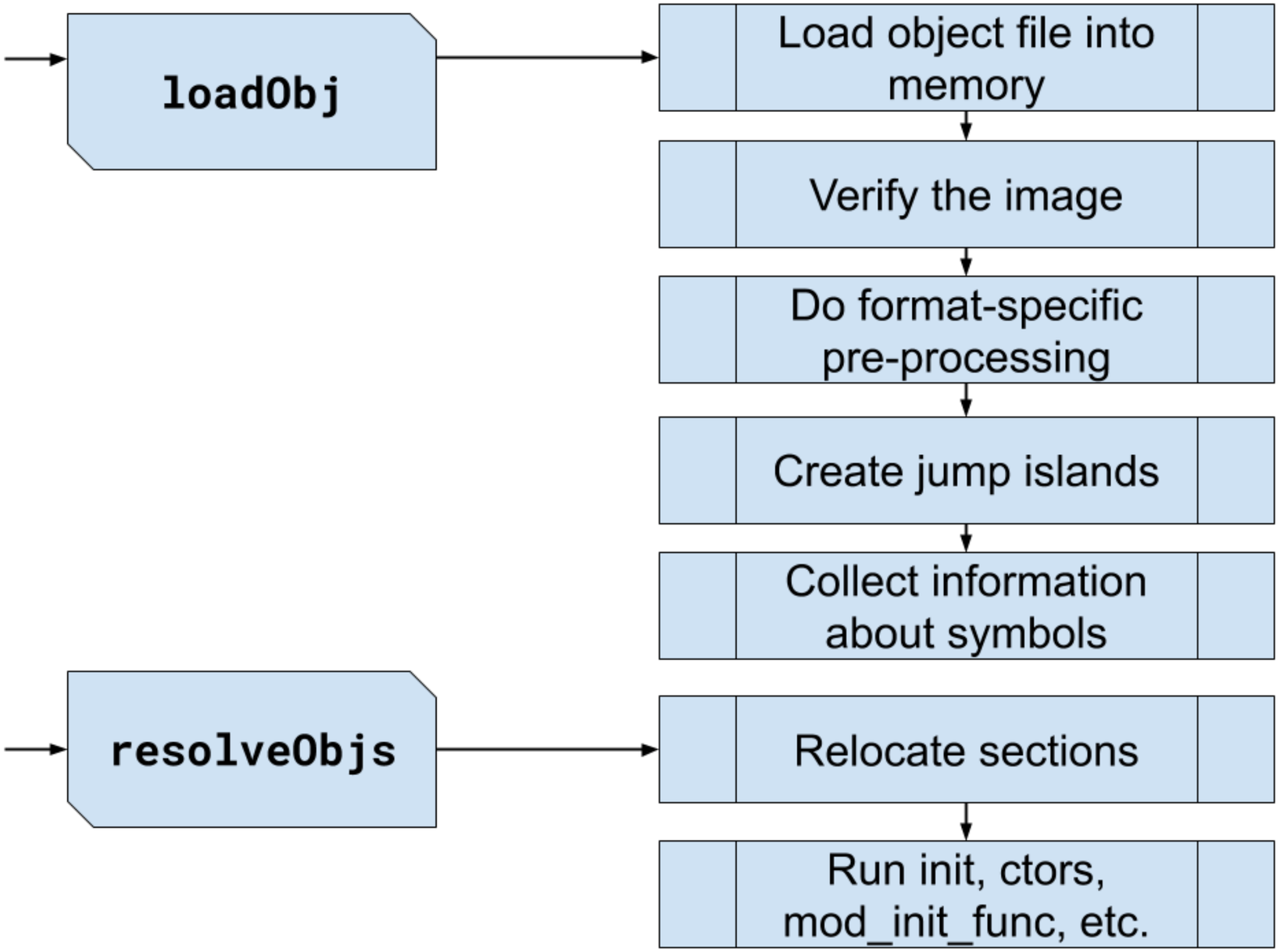
```
├── elf_compat.h
├── elf_got.c
├── elf_got.h
├── elf_plt.c
├── elf_plt.h
├── elf_plt_aarch64.c
├── elf_plt_aarch64.h
├── elf_plt_arm.c
├── elf_plt_arm.h
├── elf_reloc.c
├── elf_reloc.h
├── elf_reloc_aarch64.c
├── elf_reloc_aarch64.h
├── elf_util.c
└── elf_util.h
```

A fair amount of C code!

Language	files	blank	comment	code
C	15	1247	1977	**6810**
C Header	19	217	265	**710**

Linker C interface

```
initLinker_

addLibrarySearchPath
removeLibrarySearchPath
findSystemLibrary

addDLL

loadArchive
loadObj
purgeObj
unloadObj

resolveObjs

lookupSymbol
```


loadObj
Load object file into memory
Verify the image
Do format-specific pre-processing
Create jump islands
Collect information about symbols
resolveObjs
Relocate sections
Run init, ctors, mod_init_func, etc.

What is the problem?[=]

[=] … apart from needing to deal with 10 kLOC of involved C that is hard to get right.


BUGS

Example 1

Debugging a rare[r] segfault

[r] About 0.02% chance of a segfault.

```
int
ocGetNames_MachO(ObjectCode* oc) // oc was mmapped earlier
{
    // within for-loop over object code (oc) sections
    section_64 *section = &oc->info->macho_sections[i];
    // for section of type ZEROFILL
    zeroFillArea = mmap(section->size, MAP_ANONYMOUS, -1, 0);
    // check mmap
    section->offset = zeroFillArea - oc->image;
    //...
}
```

```
int
ocGetNames_MachO(ObjectCode* oc) // oc was mmapped earlier
{
    // within for-loop over object code (oc) sections
    section_64 *section = &oc->info->macho_sections[i];
    // for section of type ZEROFILL
    zeroFillArea = mmap(section->size, MAP_ANONYMOUS, -1, 0);
    section->offset = zeroFillArea - oc->image;
    //...
}

// MacOS SDK/mach-o/loader.h
struct section_64 {
    uint32_t    offset;     /* file offset of this section */
};
```

What if...

the kernel allocates a page before `oc->image`[?]

[?] unsigned integer underflow and segfault down the road

Example 2

Not all numbers are created equal

```
int
ocBuildSegments_MachO(ObjectCode *oc)
{
        size_t size_[xx]Segment = 0; //xx = rx, rw, zerofill

        for (int i = 0; i < oc->n_sections; i++) {
                MachOSection *macho = &oc->info->macho_sections[i];
                size_t alignment = 1 << macho->align;

                size_[xx]Segment = roundUpToAlign(size_[xx]Segment, alignment);
                size_[xx]Segment += macho->size;
        }

        size_compound = roundUpToPage(size_[xx]Segment) + ...;
        mem = mmap(size_compound, MAP_ANON, -1, 0);
        //...
}
```

```
int
ocBuildSegments_MachO(ObjectCode *oc)
{
        size_t size_[xx]Segment = 0; //xx = rx, rw, zerofill

        for (int i = 0; i < oc->n_sections; i++) {
                MachOSection *macho = &oc->info->macho_sections[i];
                size_t alignment = 1 << macho->align;

                size_[xx]Segment = roundUpToAlign(size_[xx]Segment, alignment);
                size_[xx]Segment += macho->size;
        }

        size_compound = roundUpToPage(size_[xx]Segment) + ...;
        mem = mmap(size_compound, MAP_ANON, -1, 0);
        //...
}
```

What if...

all sections in the object file are empty??

?? GHC-16701 -- `mmap 0 bytes at 0x0`

Bigger issue

GHC-13624[!]

`loadObj()` does not respect alignment

[!] Not fixed yet 😭

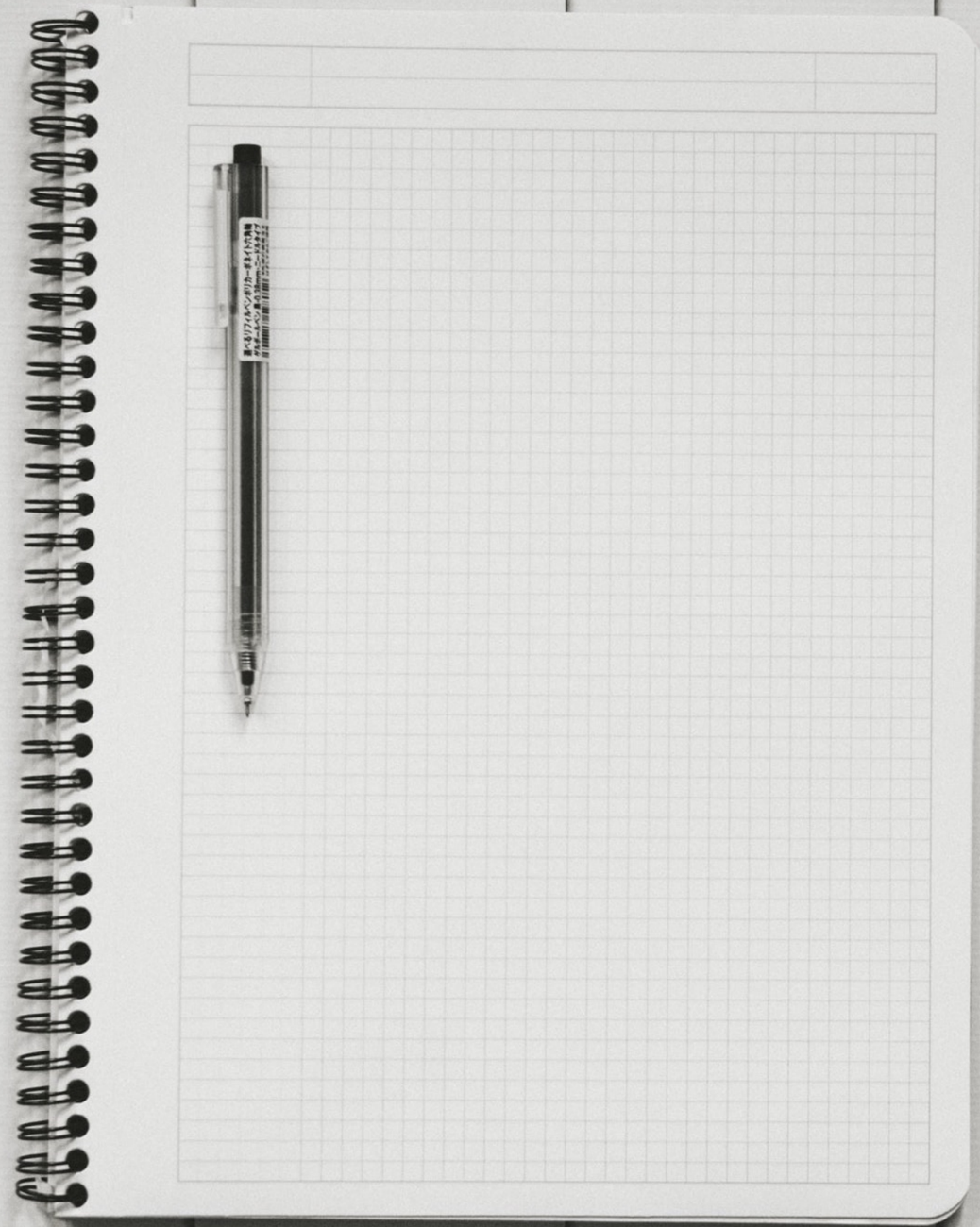
Unlucky SSE instruction

=

SIGBUS 


static
dynamic
3

	Static	Dynamic
⭐	Small overhead	Simple implementation
⭐	Space efficient	Can support more platforms
⭐	Can unload modules	Low maintenance
💩	Complex	Loading is costly
💩	Bugs	Accumulates loaded modules



Recap

Hack on GHC

Help fix those bugs[b]

Fun and rewarding it is!

[b] Looking at you GHC-13624.

Thanks for your attention!

Questions?

```
-- speaker: Artem Pianykh
-- twitter: @artem_pyanykh
```