THE BITS BETWEEN THE BITS

HOW WE GET TO `main()`

MATT GODBOLT
@MATTGODBOLT

CPPCON 2018

"Good, but non-essential" — progarchives.com

A PROGRAM

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

A PROGRAM

```c
int main() {}
```

```
$ gcc -Os empty.c -o c/empty
$ g++ -Os empty.cpp -o cpp/empty
```

A PROGRAM

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

```
$ gcc -Os empty.c -o c/empty
$ g++ -Os empty.cpp -o cpp/empty

$ ls -l c/empty cpp/empty
```

A PROGRAM

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

```
$ gcc -Os empty.c -o c/empty
$ g++ -Os empty.cpp -o cpp/empty

$ ls -l c/empty cpp/empty
7976 c/empty
```

A PROGRAM

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

```
$ gcc -Os empty.c -o c/empty
$ g++ -Os empty.cpp -o cpp/empty

$ ls -l c/empty cpp/empty
7976 c/empty
7976 cpp/empty
```

WHAT'S IN IT?

WHAT'S IN IT?

```
$ objdump --no-show-raw-insn -dC cpp/empty
```

WHAT'S IN IT?

```
$ objdump --no-show-raw-insn -dC cpp/empty
```

```
research/out/empty/c++/none/dynamic/empty:     file format elf64-x86-64

Disassembly of section .init:

00000000004003b0 <_init>:
  4003b0: sub    $0x8,%rsp
  4003b4: mov    0x200c3d(%rip),%rax        # 600ff8 <__gmon_start__>
  4003bb: test   %rax,%rax
  4003be: je     4003c2 <_init+0x12>
  4003c0: callq  *%rax
  4003c2: add    $0x8,%rsp
  4003c6: retq

Disassembly of section .text:
```

WHAT'S IN IT?

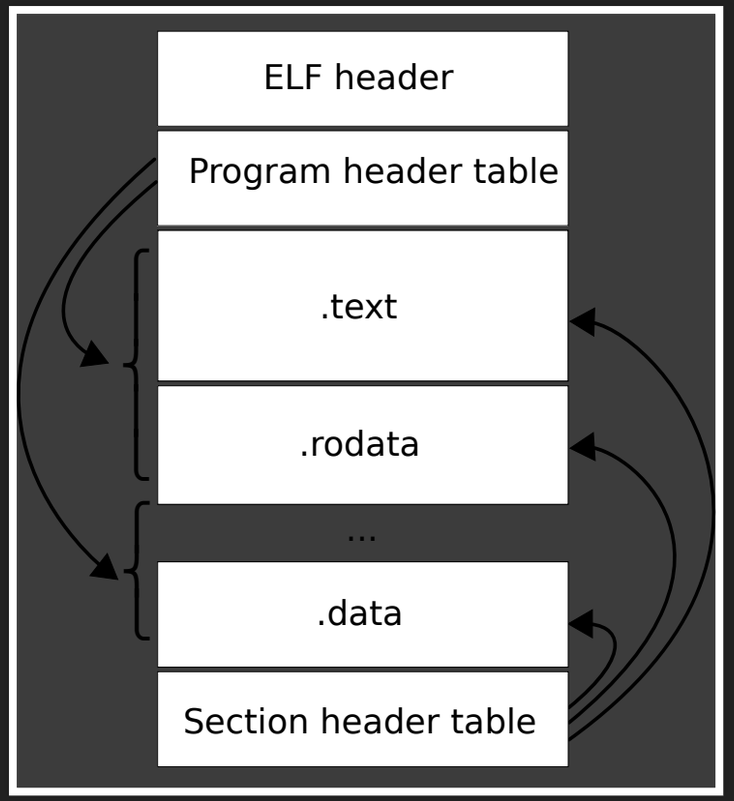
```
$ readelf -a cpp/empty
```

WHAT'S IN IT?

```
$ readelf -a cpp/empty
```

```
ELF Header:
  Magic:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Class:                             ELF64
  Data:                              2's complement, little endian
  Version:                           1 (current)
  OS/ABI:                            UNIX - System V
  ABI Version:                       0
  Type:                              EXEC (Executable file)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Entry point address:               0x4003e0
  Start of program headers:          64 (bytes into file)
  Start of section headers:          6248 (bytes into file)
  Flags:                             0x0
  Size of this header:               64 (bytes)
```

THE ELF FILE FORMAT



By Surueña [CC BY-SA 3.0], from Wikimedia Commons

SECTIONS

- `.text` — code
- `.rodata` — read-only data
- `.data`— read/write data
- `.bss` — zero-initialised data

HOW WE GET TO `main()`

A (slightly) more interesting program

```
struct Foo {
  static int numFoos;
  Foo() {
    numFoos++;
  }
  ~Foo() {
    numFoos--;
  }
};
int Foo::numFoos;
```

```
Foo globalFoo;

int main() {
  std::cout << "numFoos = "
      << Foo::numFoos << "\n";
}
```

WHAT DOES IT PRINT?

```
$ g++ -O0 -g global.cpp -o global
$ ./global
```

WHAT DOES IT PRINT?

```
$ g++ -O0 -g global.cpp -o global
$ ./global
numFoos = 1
```

CODE ARCHAEOLOGY - PART 1

CALL STACK

```
#0  Foo::Foo (this=0x601050 <global>) at global.cpp:6
#1  0x000000000040079d in __static_initialization_and_destruction_0 (
    __initialize_p=1, __priority=65535) at global.cpp:14
#2  0x00000000004007b3 in _GLOBAL__sub_I_global.cpp(void) () at global.cpp:18
#3  0x000000000040082d in __libc_csu_init ()
#4  0x00007ffff70c1b28 in __libc_start_main (main=0x400702 <main()>, argc=1,
    ... at ../csu/libc-start.c:266
#5  0x000000000040064a in _start ()
```

CALL STACK

```
#0  Foo::Foo (this=0x601050 <global>) at global.cpp:6
#1  0x000000000040079d in __static_initialization_and_destruction_0 (
    __initialize_p=1, __priority=65535) at global.cpp:14
#2  0x00000000004007b3 in _GLOBAL__sub_I_global.cpp(void) () at global.cpp:18
#3  0x000000000040082d in __libc_csu_init ()
#4  0x00007ffff70c1b28 in __libc_start_main (main=0x400702 <main()>, argc=1,
    ... at ../csu/libc-start.c:266
#5  0x000000000040064a in _start ()
```

CALL STACK

```
#0  Foo::Foo (this=0x601050 <global>) at global.cpp:6
#1  0x000000000040079d in __static_initialization_and_destruction_0 (
    __initialize_p=1, __priority=65535) at global.cpp:14
#2  0x00000000004007b3 in _GLOBAL__sub_I_global.cpp(void) () at global.cpp:18
#3  0x000000000040082d in __libc_csu_init ()
#4  0x00007ffff70c1b28 in __libc_start_main (main=0x400702 <main()>, argc=1,
    ... at ../csu/libc-start.c:266
#5  0x000000000040064a in _start ()
```

CALL STACK

```
#0  Foo::Foo (this=0x601050 <global>) at global.cpp:6
#1  0x000000000040079d in __static_initialization_and_destruction_0 (
    __initialize_p=1, __priority=65535) at global.cpp:14
#2  0x00000000004007b3 in _GLOBAL__sub_I_global.cpp(void) () at global.cpp:18
#3  0x000000000040082d in __libc_csu_init ()
#4  0x00007ffff70c1b28 in __libc_start_main (main=0x400702 <main()>, argc=1,
    ... at ../csu/libc-start.c:266
#5  0x000000000040064a in _start ()
```

CALL STACK

```
#0  Foo::Foo (this=0x601050 <global>) at global.cpp:6
#1  0x000000000040079d in __static_initialization_and_destruction_0 (
    __initialize_p=1, __priority=65535) at global.cpp:14
#2  0x00000000004007b3 in _GLOBAL__sub_I_global.cpp(void) () at global.cpp:18
#3  0x000000000040082d in __libc_csu_init ()
#4  0x00007ffff70c1b28 in __libc_start_main (main=0x400702 <main()>, argc=1,
    ... at ../csu/libc-start.c:266
#5  0x000000000040064a in _start ()
```

WHERE DO THESE FUNCTIONS COME FROM?

WHERE DO THESE FUNCTIONS COME FROM?

https://godbolt.org/z/cppcon2018_1

WHO CALLS THIS FUNCTION?

WHO CALLS THIS FUNCTION?

By Google Inc. - https://chromium.googlesource.com/chromium/src/+/master/ui/webui/resources/images/google_logo.svg, Public Domain, Link

LIBC SPELUNKING

LIBC SPELUNKING

```
// Paraphrased from glibc/csu/elf-init.c
typedef void (*init_func)(int, char **, char **);

extern init_func __init_array_start[];
extern init_func __init_array_end[];

int __libc_csu_init(int argc, char **argv, char **envp) {
  const size_t size = __init_array_end - __init_array_start;
  for (size_t i = 0; i < size; i++)
    (*__init_array_start[i])(argc, argv, envp);
}
```

LIBC SPELUNKING

```
// Paraphrased from glibc/csu/elf-init.c
typedef void (*init_func)(int, char **, char **);

extern init_func __init_array_start[];
extern init_func __init_array_end[];

int __libc_csu_init(int argc, char **argv, char **envp) {
  const size_t size = __init_array_end - __init_array_start;
  for (size_t i = 0; i < size; i++)
    (*__init_array_start[i])(argc, argv, envp);
}
```

LIBC SPELUNKING

```
// Paraphrased from glibc/csu/elf-init.c
typedef void (*init_func)(int, char **, char **);

extern init_func __init_array_start[];
extern init_func __init_array_end[];

int __libc_csu_init(int argc, char **argv, char **envp) {
  const size_t size = __init_array_end - __init_array_start;
  for (size_t i = 0; i < size; i++)
    (*__init_array_start[i])(argc, argv, envp);
}
```

LIBC SPELUNKING

```c
// Paraphrased from glibc/csu/elf-init.c
typedef void (*init_func)(int, char **, char **);

extern init_func __init_array_start[];
extern init_func __init_array_end[];

int __libc_csu_init(int argc, char **argv, char **envp) {
  const size_t size = __init_array_end - __init_array_start;
  for (size_t i = 0; i < size; i++)
    (*__init_array_start[i])(argc, argv, envp);
}
```

LIBC SPELUNKING

```
// Paraphrased from glibc/csu/elf-init.c
typedef void (*init_func)(int, char **, char **);

extern init_func __init_array_start[];
extern init_func __init_array_end[];

int __libc_csu_init(int argc, char **argv, char **envp) {
  const size_t size = __init_array_end - __init_array_start;
  for (size_t i = 0; i < size; i++)
    (*__init_array_start[i])(argc, argv, envp);
}
```

OK BUT...

`__init_array_start`?
`__init_array_end`?

OK BUT...

`__init_array_start`?
`__init_array_end`?

https://godbolt.org/z/cppcon2018_2

WHAT'S GOING ON HERE?

```
.section .init_array,"aw"
.align   8
.quad    _GLOBAL__sub_I_Foo::numFoos
.text
```

THE LINKER

Matt Godbolt [CC BY-SA 3.0]

THE LINKER

What does it do?

- Resolves references between `.o` files
- Determines the layout of an executable
- Writes metadata

A MORE REPRESENTATIVE PROGRAM

```cpp
// hello.cpp
extern const char *getMessage();
void greet() {
  std::cout << getMessage() << "\n";
}
int main() {
  greet();
}
```

```cpp
// message.cpp
const char *getMessage() {
  return "Hello world";
}
```

```
$ g++ -Os -o hello.o hello.cpp
$ g++ -Os -o message.o message.cpp
$ g++ -Os -o hello message.o hello.o
$ ./hello
Hello world
```

OBJECT FILES

```
$ file hello hello.o message.o
hello:     ELF 64-bit LSB executable, x86-64,
           dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2,
           for GNU/Linux 3.2.0, not stripped
hello.o:   ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), not stripped
message.o: ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), not stripped
```

WHAT'S IN AN OBJECT FILE?

```
$ objdump -dC hello.o
```

WHAT'S IN AN OBJECT FILE?

```
$ objdump -dC hello.o
```

```
research/out/hello/c++/none/dynamic/hello.o:     file format elf64-x86-64

Disassembly of section .text:

0000000000000000 <greet()>:
   0: 55                    push   %rbp
   1: 48 89 e5              mov    %rsp,%rbp
   4: e8 00 00 00 00        callq  9 <greet()+0x9>
   9: 48 89 c6              mov    %rax,%rsi
   c: 48 8b 05 00 00 00 00  mov    0x0(%rip),%rax        # 13 <greet()+0x13>
  13: 48 89 c7              mov    %rax,%rdi
  16: e8 00 00 00 00        callq  1b <greet()+0x1b>
  1b: 48 8d 35 00 00 00 00  lea    0x0(%rip),%rsi        # 22 <greet()+0x22>
  22: 48 89 c7              mov    %rax,%rdi
  25: e8 00 00 00 00        callq  2a <greet()+0x2a>
```

RELOCATIONS

Matt Godbolt [CC BY-SA 3.0]

WHAT'S IN AN OBJECT FILE?

```
$ objdump --reloc -dC hello.o
```

```
research/out/hello/c++/none/dynamic/hello.o:     file format elf64-x86-64

Disassembly of section .text:

0000000000000000 <greet()>:
   0: 55                    push   %rbp
   1: 48 89 e5              mov    %rsp,%rbp
   4: e8 00 00 00 00        callq  9 <greet()+0x9>
      5: R_X86_64_PLT32 getMessage()-0x4
   9: 48 89 c6              mov    %rax,%rsi
   c: 48 8b 05 00 00 00 00  mov    0x0(%rip),%rax        # 13 <greet()+0x13>
      f: R_X86_64_REX_GOTPCRELX std::cout-0x4
  13: 48 89 c7              mov    %rax,%rdi
  16: e8 00 00 00 00        callq  1b <greet()+0x1b>
      17: R_X86_64_PLT32  std::basic_ostream<char, std::char_traits<char> >& std:…
```

RELOCATIONS

- Different types
- Used within same object file

SYMBOLS

Matt Godbolt [CC BY-SA 3.0]

SYMBOLS

```
$ objdump --syms -C hello.o
```

SYMBOLS

```
$ objdump --syms -C hello.o
```

```
research/out/hello/c++/none/dynamic/hello.o:     file format elf64-x86-64

SYMBOL TABLE:
00000000 l    df *ABS*  00000000 hello.cpp
00000000 l    d  .text  00000000 .text
00000000 l    d  .data  00000000 .data
00000000 l    d  .bss 00000000 .bss
00000000 l    d  .rodata  00000000 .rodata
00000000 l     O .rodata  00000001 std::piecewise_construct
00000000 l     O .bss 00000001 std::__ioinit
0000003d l     F .text  00000049 __static_initialization_and_destruction_0(int, i…
00000086 l     F .text  00000015 _GLOBAL__sub_I_hello.cpp
00000000 l    d  .init_array  00000000 .init_array
00000000 l    d  .note.GNU-stack  00000000 .note.GNU-stack
00000000 l    d  .eh_frame  00000000 .eh_frame
```

SYMBOLS

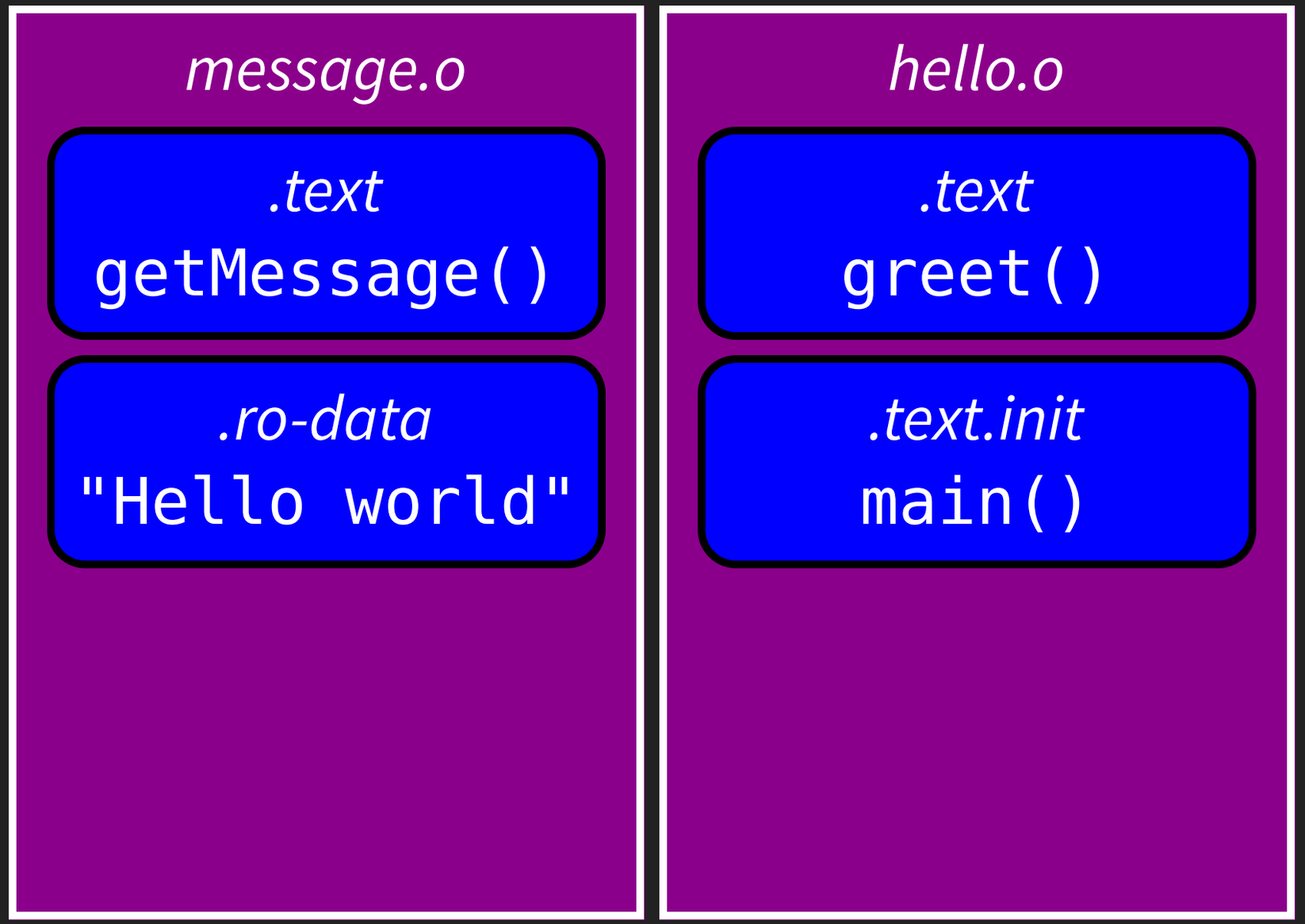
```
$ objdump --syms -C message.o
```

```
research/out/hello/c++/none/dynamic/message.o:     file format elf64-x86-64

SYMBOL TABLE:
00000000 l    df *ABS*  00000000 message.cpp
00000000 l    d  .text  00000000 .text
00000000 l    d  .data  00000000 .data
00000000 l    d  .bss 00000000 .bss
00000000 l    d  .rodata  00000000 .rodata
00000000 l    d  .note.GNU-stack  00000000 .note.GNU-stack
00000000 l    d  .eh_frame  00000000 .eh_frame
00000000 l    d  .comment 00000000 .comment
00000000 g     F .text  00000008 getMessage()
```

LINKER...

- Reads all the inputs
- Identifies symbols
- Applies relocations


message.o
.text
getMessage()
.ro-data
"Hello world"
hello.o
.text
greet()
.text.init
main()

message.o

.text
`getMessage()`

.ro-data
`"Hello world"`

hello.o

.text
`greet()`

.text.init
`main()`

hello

Program Headers

.text
```
greet() {}
getMessage()
main()
```

.ro-data
`"Hello world"`

LINKER SCRIPTS

```
$ g++ -o /dev/null -x c /dev/null -Wl,--verbose
```

LINKER SCRIPTS

```
$ g++ -o /dev/null -x c /dev/null -Wl,--verbose
```

```
GNU ld (GNU Binutils for Ubuntu) 2.30
  Supported emulations:
   elf_x86_64
   elf32_x86_64
   elf_i386
   elf_iamcu
   i386linux
   elf_l1om
   elf_k1om
   i386pep
   i386pe
using internal linker script:
==================================================
/* Script for -pie -z combreloc -z now -z relro: position independent executable,…
/* Copyright (C) 2014-2018 Free Software Foundation, Inc.
```

```
  .init_array     :
  {
    PROVIDE_HIDDEN (__init_array_start = .);
    KEEP (*(SORT_BY_INIT_PRIORITY(.init_array.*) SORT_BY_INIT_PRIORITY(.ctors.*)))
    KEEP (*(.init_array EXCLUDE_FILE (
            *crtbegin.o *crtbegin?.o *crtend.o *crtend?.o ) .ctors))
    PROVIDE_HIDDEN (__init_array_end = .);
  }
```

```
  .init_array     :
  {
    PROVIDE_HIDDEN (__init_array_start = .);
    KEEP (*(SORT_BY_INIT_PRIORITY(.init_array.*) SORT_BY_INIT_PRIORITY(.ctors.*)))
    KEEP (*(.init_array EXCLUDE_FILE (
           *crtbegin.o *crtbegin?.o *crtend.o *crtend?.o ) .ctors))
    PROVIDE_HIDDEN (__init_array_end = .);
  }
```

```
  .init_array     :
  {
    PROVIDE_HIDDEN (__init_array_start = .);
    KEEP (*(SORT_BY_INIT_PRIORITY(.init_array.*) SORT_BY_INIT_PRIORITY(.ctors.*)))
    KEEP (*(.init_array EXCLUDE_FILE (
            *crtbegin.o *crtbegin?.o *crtend.o *crtend?.o ) .ctors))
    PROVIDE_HIDDEN (__init_array_end = .);
  }
```

```
  .init_array     :
  {
    PROVIDE_HIDDEN (__init_array_start = .);
    KEEP (*(SORT_BY_INIT_PRIORITY(.init_array.*) SORT_BY_INIT_PRIORITY(.ctors.*)))
    KEEP (*(.init_array EXCLUDE_FILE (
            *crtbegin.o *crtbegin?.o *crtend.o *crtend?.o ) .ctors))
    PROVIDE_HIDDEN (__init_array_end = .);
  }
```

NOW WE KNOW!

- Compiler:

NOW WE KNOW!

- Compiler:
  - "static init" function for each TU
  - pointer to this function into `init_array`

NOW WE KNOW!

- Compiler:
  - "static init" function for each TU
  - pointer to this function into `init_array`
- Linker:
  - gathers all init_arrays together
  - script defines symbols pointing at begin and end of `init_array`

NOW WE KNOW!

- Compiler:
  - "static init" function for each TU
  - pointer to this function into `init_array`
- Linker:
  - gathers all init_arrays together
  - script defines symbols pointing at begin and end of `init_array`
- C runtime walks `init_array` and calls each

STUFF TO KNOW

- You can write your own linker scripts
- Linker can discard unused sections: `-Wl,--gc-sections`
- Compiler flags: `-ffunction-sections`, `-fdata-sections`

DYNAMIC LINKING

```
$ ls -l dynamic/hello static/hello
8,688      dynamic/hello*
2,406,632  static/hello*
```

ANOTHER HELLO WORLD

```
$ g++ -Os -o message.o message.cpp
$ g++ -shared -o libhello.so message.o
$ g++ -Os -o hello hello.o
$ g++ -Os -o hello.o hello.cpp -L. -lhello
$ ./hello
Hello world
```

MORE ELF HEADERS

```
$ readelf --dynamic --program-headers dynamic-dso/hello
```

```
Elf file type is EXEC (Executable file)
Entry point 0x4006f0
There are 9 program headers, starting at offset 64

Program Headers:
  Type           Offset             VirtAddr           PhysAddr
                 FileSiz            MemSiz              Flags  Align
  PHDR           0x00000040 0x00400040 0x00400040
                 0x000001f8 0x000001f8  R E    0x8
  INTERP         0x00000238 0x00400238 0x00400238
                 0x0000001c 0x0000001c  R      0x1
      [Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]
  LOAD           0x00000000 0x00400000 0x00400000
                 0x00000ab0 0x00000ab0  R E    0x200000
  LOAD           0x00000da8 0x00600da8 0x00600da8
```

CODE ARCHAEOLOGY - PART 2

Matt Godbolt [CC BY-SA 3.0]

```
getMessage()@plt:
 0x4006b0:  jmpq  *0x200962(%rip) # 0x601018
 0x4006b6:  pushq $0x0
 0x4006bb:  jmpq  0x4006a0
...
 0x601018:  .quad 0x4006b6
```

```
getMessage()@plt:
 0x4006b0:  jmpq  *0x200962(%rip) # 0x601018
 0x4006b6:  pushq $0x0
 0x4006bb:  jmpq  0x4006a0
...
 0x601018:  .quad 0x4006b6
```

```
getMessage()@plt:
 0x4006b0:  jmpq  *0x200962(%rip) # 0x601018
 0x4006b6:  pushq $0x0
 0x4006bb:  jmpq  0x4006a0
...
 0x601018:  .quad 0x4006b6
```

```
getMessage()@plt:
 0x4006b0:  jmpq  *0x200962(%rip) # 0x601018
 0x4006b6:  pushq $0x0
 0x4006bb:  jmpq  0x4006a0
...
 0x601018:  .quad 0x4006b6
```

```
getMessage()@plt:
 0x4006b0:  jmpq  *0x200962(%rip) # 0x601018
 0x4006b6:  pushq $0x0
 0x4006bb:  jmpq  0x4006a0  ; ultimately resolves symbol 0
...
 0x601018:  .quad 0x4006b6
```

```
getMessage()@plt:
 0x4006b0:  jmpq  *0x200962(%rip) # 0x601018
 0x4006b6:  pushq $0x0
 0x4006bb:  jmpq  0x4006a0
...
 0x601018:  .quad 0x7ffff7bd35d5 ; now resolved to getMessage()
```

DEBUGGING

- `LD_BIND_NOW` (and `-Wl,-znow`)
- `ldd` and `LD_DEBUG`
- `LD_PRELOAD`

I WISH I HAD MORE TIME

- Weak references
- ODR violations
- LTO

MORE READING

- Ian Lance Taylor's blog - www.airs.com/blog
- Honza Hubička's blog - hubicka.blogspot.com/

SPECIAL TRAINING EVENT

- Summer 2019 — Denver Area
- Charley Bay, Jason Turner and me together for 3 days
- C++20, error handling and performance
- Check out coloradoplusplus.info for more info