Deep in the Wild

An intro to development of the Wild linker

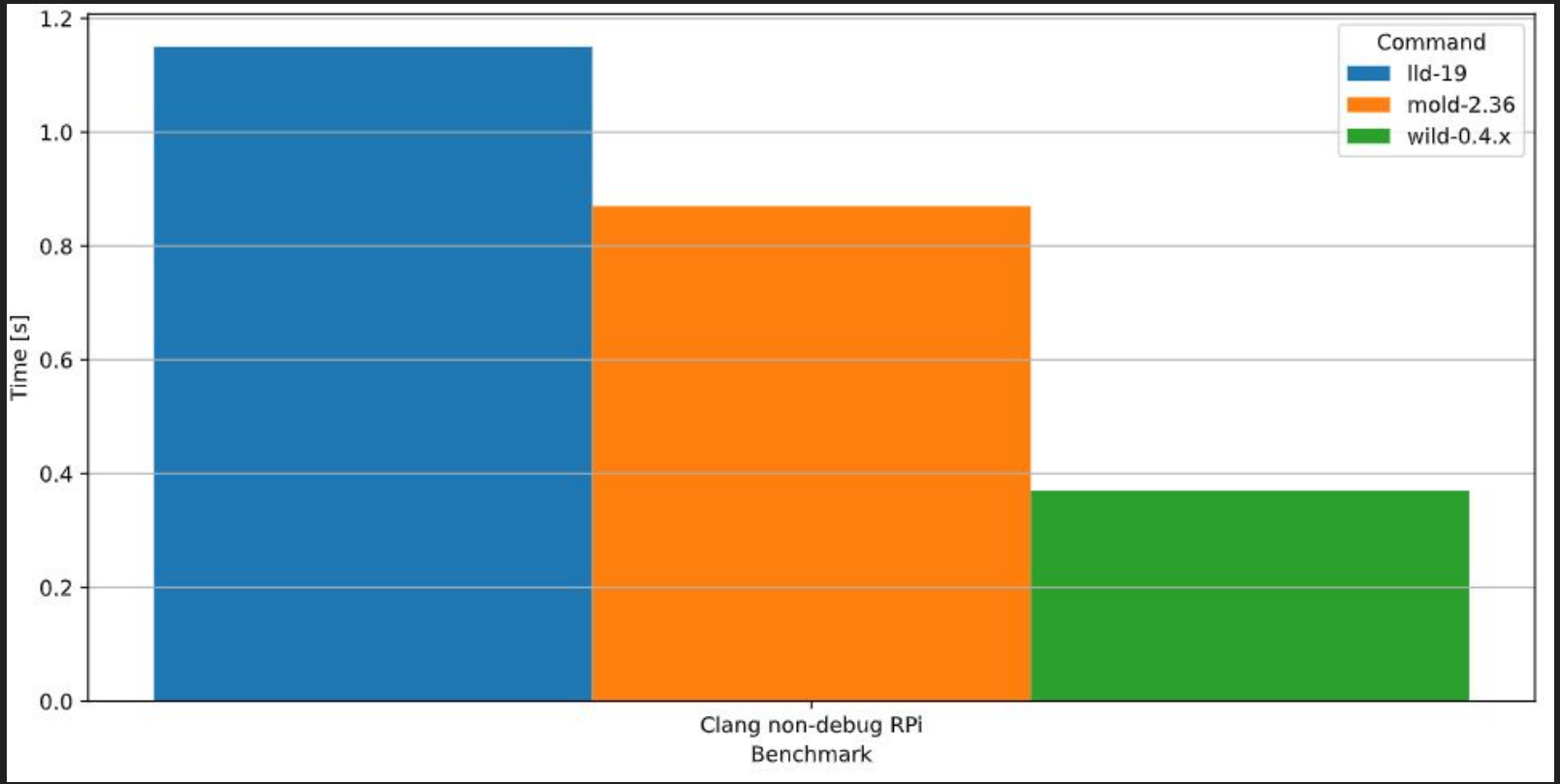
First, some background

- Linker: Combines all the bits of your program into an executable.
- Relocation: A reference to something. A placeholder, left by the compiler for the linker to fill in.

Why linkers are cool

- They write the executables that we run.
- They decide, within reason, where everything goes.
- When they get things wrong, things fail spectacularly.
- They give you an excuse to look at low-level details, step through assembly code.
- Performance is very important and optimising things is fun.

Benchmark linking clang on Raspberry Pi-5



Symbol resolution algorithm

- Phase 1: Read names of symbol definitions, hash and bucket by hash.
- Phase 2: Build a hashmap for each bucket.
- Phase 3: Read names of undefined symbols, hash, then lookup in appropriate bucket.

Mapping input sections to output sections

- Until recently: if secname.starts_with(b”.text”) …
- Now uses a hash of the first 4 bytes of the input section name to locate starting offset in a hashbrown::HashTable.

Debugging: Different classes of failures

- Linker gives internal or unexpected error.
- Generated binary segfaults at the point where the linker did the wrong thing.
- Generated binary segfaults some time later.
- Generated binary misbehaves.

Debugging a linker bug

--- stdout ----------

Input: TokenStream []

Respanned: TokenStream []

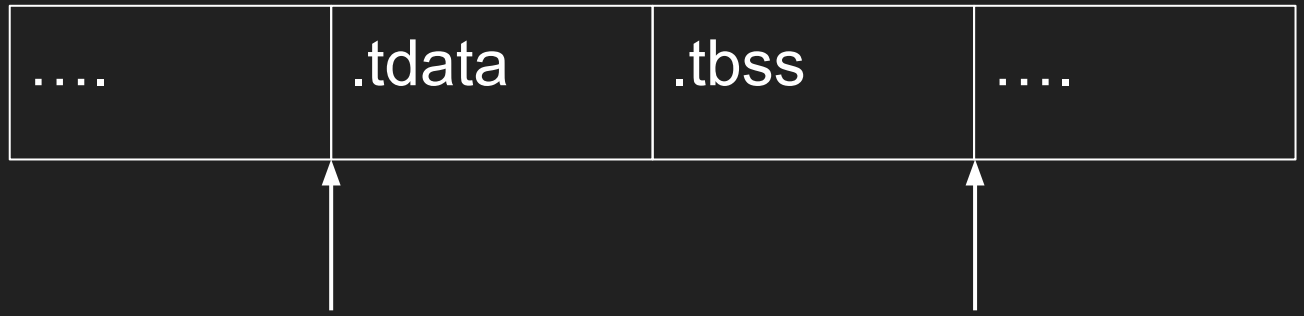
--- stderr ----------

free(): invalid pointer

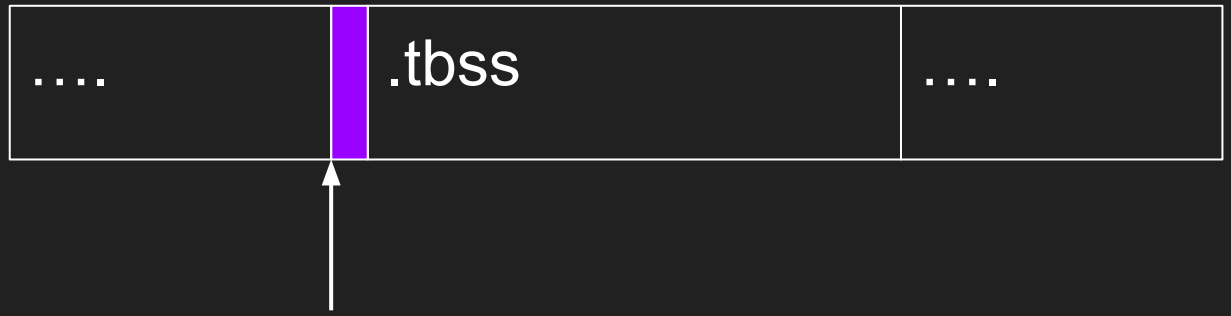
Alignment and padding

- Alignment: A requirement that something be at an address that’s an exact multiple of some power of 2. e.g. 4, 8, 64
- Padding: Unused bytes inserted before something so that it has the correct alignment.

Thread-local storage base address



Thread-local storage base address



Step 1: Run linker-diff

WILD_REFERENCE_LINKER=ld my-build-command

```
rel.R_X86_64_TLSGD.R_X86_64_TLSGD
  `/home/david/work/wild/wild/tests/build/dynamic-bss-only.default-host-1995ada7b02eab67.o` .text set_tvar1
  ORIG 0x002d: [ 66 48 8d 3d 00 00 00 00 ] lea 0x35,%rdi
                           ^^^^^^^^^^^ R_X86_64_TLSGD
  ORIG tvar1 -4
  wild 0x184d: [ 66 48 8d 3d db 11 00 00 ] lea 0x2A30,%rdi
                           ^^^^^^^^^^^ R_X86_64_TLSGD NoOp
  wild GOT->tvar1 +1
  wild TRACE: relocation applied value_flags=ADDRESS | NON_INTERPOSABLE,
  wild TRACE: resolution_flags=DIRECT | GOT_TLS_MODULE, rel_kind=TlsGd,
  wild TRACE: value=0x11db, symbol_name=tvar1
  ld   0x105d: [ 66 48 8d 3d 8b 2f 00 00 ] lea 0x3FF0,%rdi
                           ^^^^^^^^^^^ R_X86_64_TLSGD NoOp
  ld   GOT->tvar1
```

Step 2: Debug the binary with GDB

- If possible, use rr, the replay debugger
- Setting watchpoints on memory addresses is very useful, when it works.

Scripting GDB

```
set pagination off
break _start
run
delete 1
break __tls_get_addr
command 2
     p ((size_t*)$rdi)[0]
     p ((size_t*)$rdi)[1]
     cont
end
set logging file gdb.output
set logging enabled on
cont
```

Saboteur / mutant testing

- Try changing or commenting out random bits of code.
- See what test failure(s) result.

What’s coming up

- Linker script support (Me)
- Getting the rustc test suite to pass when using Wild (Mateusz)
- RISC-V support (Martin)
- Basic incremental linking (Me)
- Some other feature (You?)

Come and help out

- Plenty of technical challenges
- Also, plenty of easier things to solve
- Look for issues tagged good-first-issue
- Try building stuff with Wild and file issues
- If you can’t find anything suitable, please reach out
- Contact me via https://github.com/davidlattimore

Thanks! Questions?

- CodeursenLiberte
- Urgau
- repi
- bes
- bearcove
- Rafferty97
- Kobzol
- jonhoo
- bcmyers
- joshtriplett
- mstange
- acshi
- pmarks
- flba-eb
- marxin
- tommythorn
- binarybana
- rrbutani
- twilco
- yerke
- wezm
- teh
- coastalwhite
- Shnatsel
- teburd
- willstott101
- gendx
- davidcornu
- mati865
- nazar-pc
- teohhanhui
- jkendall327
- drmason13
- jplatte
- ymgyt
- rukai
- tatsuya6502
- dream-dasher
- EdorianDark
- Pratyush
- dralley