INLINING AND LTO FOR RUST KERNEL MODULES

Kangrejos 2023

Andreas Hindborg

Samsung GOST

AGENDA

- Quick `blk-mq` **update**
- Inlining and LTO
  - Context
  - Inlining **Rust**
  - Integer Overflow Checks
  - Inlining **C**

`blk-mq` STATUS

- **Goal**: Provide `blk-mq` Rust API
- `rnull` Rust `null_blk` **replacement**
  - Status: In active **development**
  - Goal: New **RFC** in 2023, feature parity in 2024
  - Latest **benchmark** results available at https://rust-for-linux.com/null-block-driver
  - Code: https://github.com/metaspace/linux/tree/null_blk-next-for-6.6
- `rnvme` Rust `nvme` replacement
  - Status: **Maintained**, rebased on `rust-next` + common base with `null_blk`
  - Latest **benchmark** results available at https://rust-for-linux.com/nvme-driver
  - Code: https://github.com/metaspace/linux/tree/nvme

```
16c865453c8f * null_blk-next-for-6.6 metaspace/null_blk-next-for-6.6 null_blk: Propagate result
8e49d9f2c7f6 * rust: inline a number of short functions
25aa570fa5ea * rust: add null block driver
403945e0c06e * RUST: implement `ForeignOwnable` for `Pin`
b1395f13df0a * rust: lock: implement `IrqSaveBackend` for `SpinLock`
c654077c1bf3 * rust: lock: add support for `Lock::lock_irqsave`
0d84c89b7283 * rust: apply cache line padding for `SpinLock`
9af29a5d21c1 * RUST: add `module_params` macro
9936ca52131b * rust: add radix tree abstraction
f0ab846d0d1e * rust: add improved version of `ForeignOwnable::borrow_mut`
7353a81d5665 | * metaspace/nvme gandalf/nvme-next-for-6.6 rust: wip: Add nvme driver.
b55fb28a0da7 | * rust: update blk-mq for nvme
62af8b3d3723 | * RUST: add `module_params` macro
da701a8fcd84 | * rust: enable allocator API in modules for Box::try_new()
dad672eb54e1 | * rust: add device::Data
dac700359bac | * rust: add revocable objects
5e0cc0dbcc6d | * rust: add revocalbe mutex
0b92f190cd5a | * rust: add rcu abstraction
1bc904b89d65 | * rust: add atomic optional
40434ecd7604 | * rust: add num_possible_cpus
46e084b40ecc | * rust: add atomic allocated box
3a683582f103 | * rust: allow allocation with gfp_t flags
b998f700139f | * rust: add dma pool and coherent allocator
e838ea29eef9 | * rust: device: add dma map functions
5c92ed5ec619 | * rust: device: add print functions
42421184d909 | * rust: pci: add IRQ and misc methods
9479a64cd649 | * rust: add io_mem
7a4e8986c545 | * rust: add pci_device_id abstraction
c2300da95453 | * rust: add initial PCI support
83afcd52b651 | * rust: add irq abstractions
370e98a06160 | * rust: add driver abstraction
639f85b58c44 | * rust: device: Add a stub abstraction for devices
0496c9663cfc | * rust: device: Add a minimal RawDevice trait
             |/
fc76a6d68300 * blk_mq-next-6.6 rust: block: introduce `kernel::block::bio` module
9beb2b204522 * rust: block: introduce `kernel::block::mq` module
1fec729cda12 * rust: add `pages` module for handling page allocation
```

INLINING AND LTO

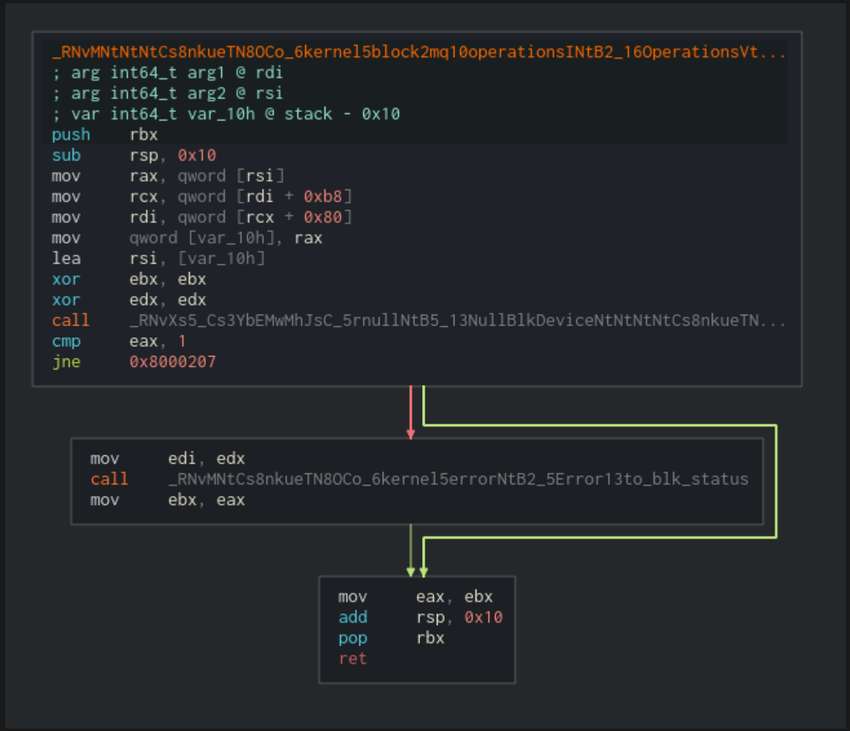
THE CODE

```
fn queue_rq(
    _hw_data: <Self::HwData as ForeignOwnable>::Borrowed<'_>,
    queue_data: <Self::QueueData as ForeignOwnable>::Borrowed<'_>,
    rq: &mq::Request<Self>,
    _is_last: bool,
) -> Result {
    rq.start();
    if *memory_backed.read() {
        let mut tree = queue_data.lock_irqsave();

        let mut sector = rq.sector();
        for bio in rq.bio_iter() {
            for mut segment in bio.segment_iter() {
                Self::transfer(rq.command(), &mut tree, sector, &mut segment)?;
                sector += segment.len() >> 9; // TODO: SECTOR_SHIFT
            }
        }
    }
    rq.end_ok();
    Ok(())
}
```

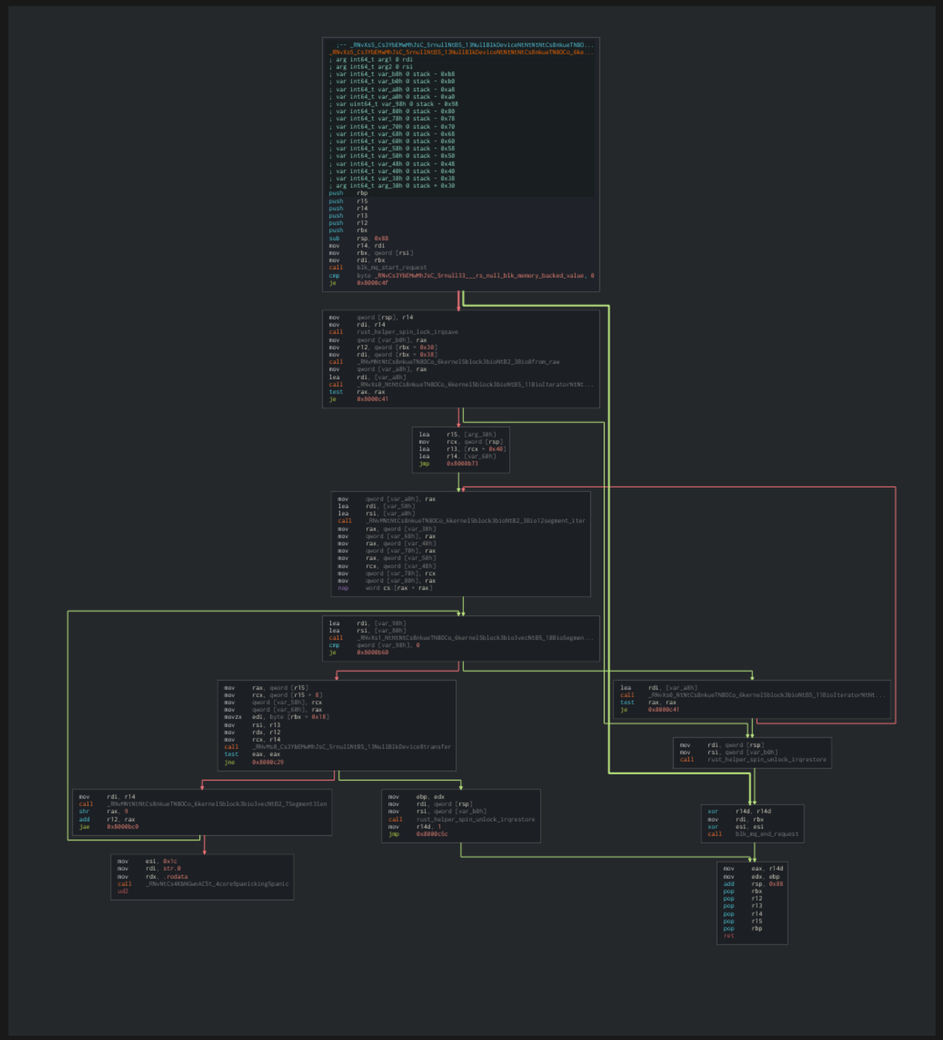
C CALLBACK

```
unsafe extern "C" fn queue_rq_callback(
    hctx: *mut bindings::blk_mq_hw_ctx,
    bd: *const bindings::blk_mq_queue_data,
) -> bindings::blk_status_t {
    let rq = unsafe { (*bd).rq };

    let hw_data = unsafe { T::HwData::borrow((*hctx).driver_data) };

    let queue_data = unsafe { (*(*hctx).queue).queuedata };

    let queue_data = unsafe { T::QueueData::borrow(queue_data) };

    let ret = T::queue_rq(
        hw_data,
        queue_data,
        &unsafe { Request::from_ptr(rq) },
        unsafe { (*bd).last },
    );
    if let Err(e) = ret {
        e.to_blk_status()
    } else {
        bindings::BLK_STS_OK as _
    }
}
```

extern "C" fn queue_rq_callback()



fn queue_rq()



Loop Header


mov qword [rsp], r14
mov rdi, r14
call rust_helper_spin_lock_irqsave
mov qword [var_b0h], rax
mov r12, qword [rbx + 0x30]
mov rdi, qword [rbx + 0x38]
call _RNvMNtNtCs8nkueTN8OCo_6kernel5block3bioNtB2_3Bio8from_raw
mov qword [var_a8h], rax
lea rdi, [var_a8h]
call _RNvXs0_NtNtCs8nkueTN8OCo_6kernel5block3bioNtB5_11BioIteratorNtNt...
test rax, rax
je 0x8000c41
lea r15, [arg_30h]
mov rcx, qword [rsp]
lea r13, [rcx + 0x40]
lea r14, [var_60h]
jmp 0x8000b73
mov qword [var_a0h], rax
lea rdi, [var_50h]
lea rsi, [var_a0h]
call _RNvMNtNtCs8nkueTN8OCo_6kernel5block3bioNtB2_3Bio12segment_iter
mov rax, qword [var_38h]
mov qword [var_68h], rax
mov rax, qword [var_40h]
mov qword [var_70h], rax
mov rax, qword [var_50h]
mov rcx, qword [var_48h]
mov qword [var_78h], rcx
mov qword [var_80h], rax
nop word cs:[rax + rax]
lea rdi, [var_98h]
lea rsi, [var_80h]
call _RNvXs1_NtNtNtCs8nkueTN8OCo_6kernel5block3bio3vecNtB5_18BioSegmen...
cmp qword [var_98h], 0
je 0x8000b60

ISSUES PT 1

- `queue_rq()` not inlined into `queue_rq_callback()`
- `kernel::block::bio::Bio::segment_iter()` is not inline

.

`Bio::segment_iter()` is not in our module:

```
shell# objdump -t linux-build/drivers/block/rnull_mod.ko | rustfilt | grep segment_iter
```

```
*UND*  0000000000000000 <kernel::block::bio::Bio>::segment_iter
```

.

It is in the kernel object:

```
shell# objdump -t linux-build/vmlinux | rustfilt | grep segment_iter
```

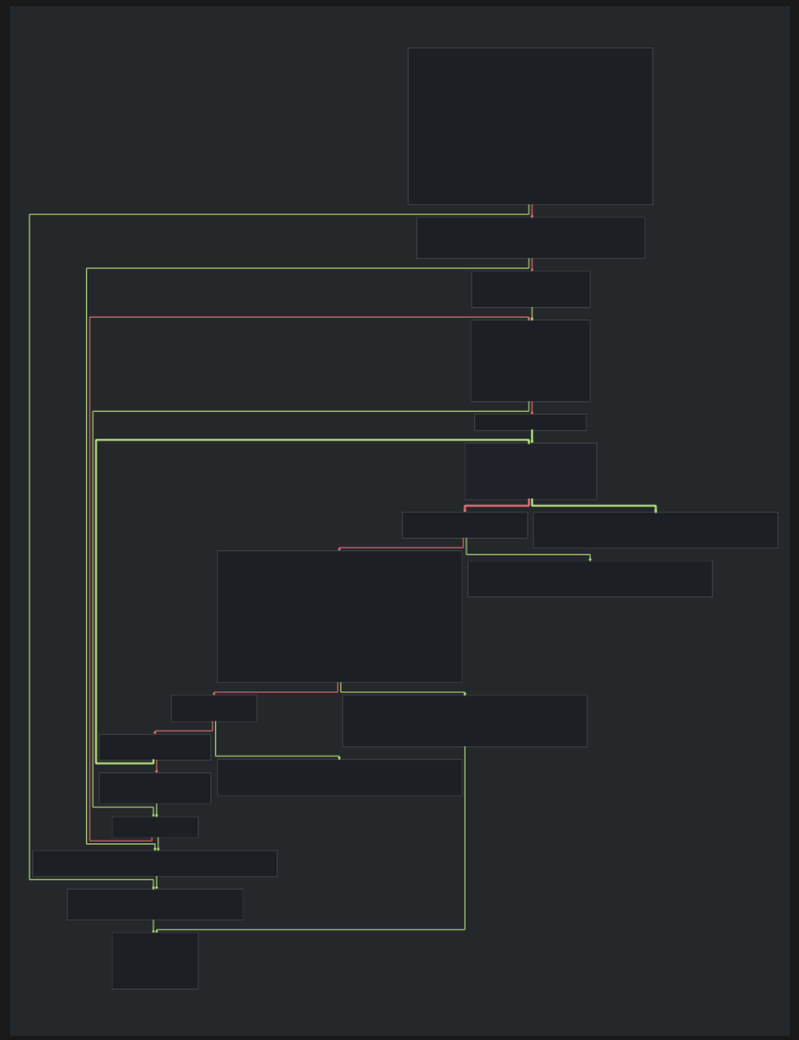
```
F .text  0000000000000020 <kernel::block::bio::Bio>::segment_iter
```

FIX PT 1

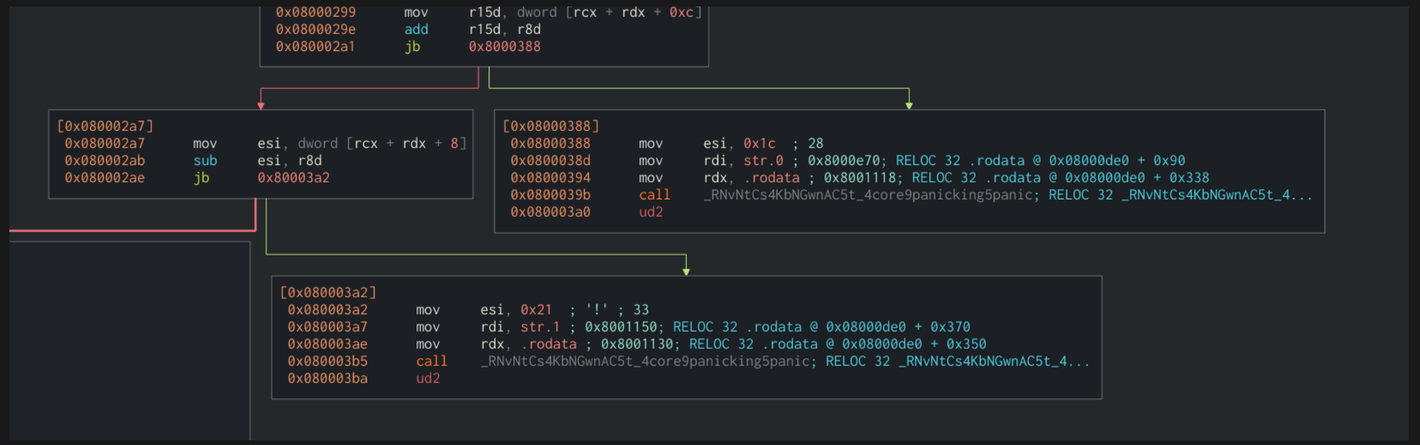
```
modified   drivers/block/rnull.rs
@@ -135,7 +135,7 @@ fn new_request_data(
         Ok(())
     }

+    #[inline(always)]
     fn queue_rq(
         _hw_data: <Self::HwData as ForeignOwnable>::Borrowed<'_>,
         queue_data: <Self::QueueData as ForeignOwnable>::Borrowed<'_>,
modified   rust/kernel/block/bio.rs
@@ -25,13 +25,13 @@ pub struct Bio<'a>(
 impl<'a> Bio<'a> {
     /// Returns an iterator over segments in this `Bio`. Does not consider
     /// segments of other bios in this bio chain.
+    #[inline(always)]
     pub fn segment_iter(&'a self) -> BioSegmentIterator<'a> {
         BioSegmentIterator::new(self)
     }
```

Note: We are building crates with `-Clto=n`. This disables LTO across codegen units and across crates in the dependency graph when building a crate. We are also building with `-Ccodegen-units=1` so this only has an effect on cross crate LTO.

extern "C" fn queue_rq_callback()

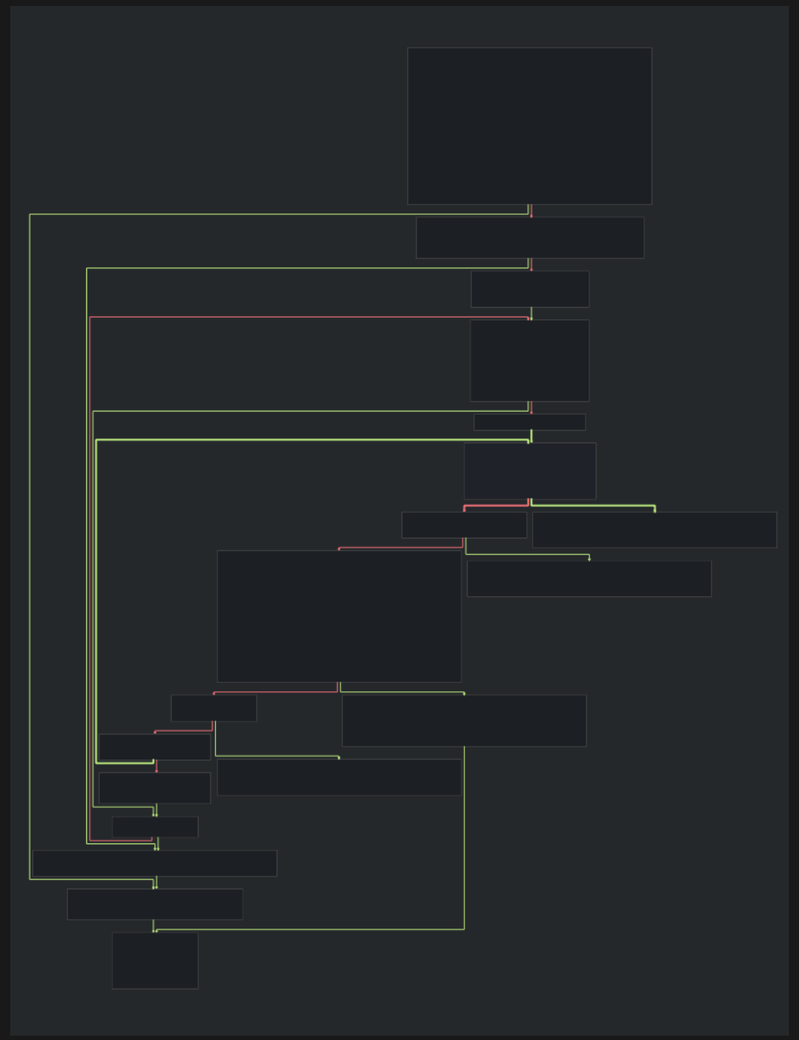


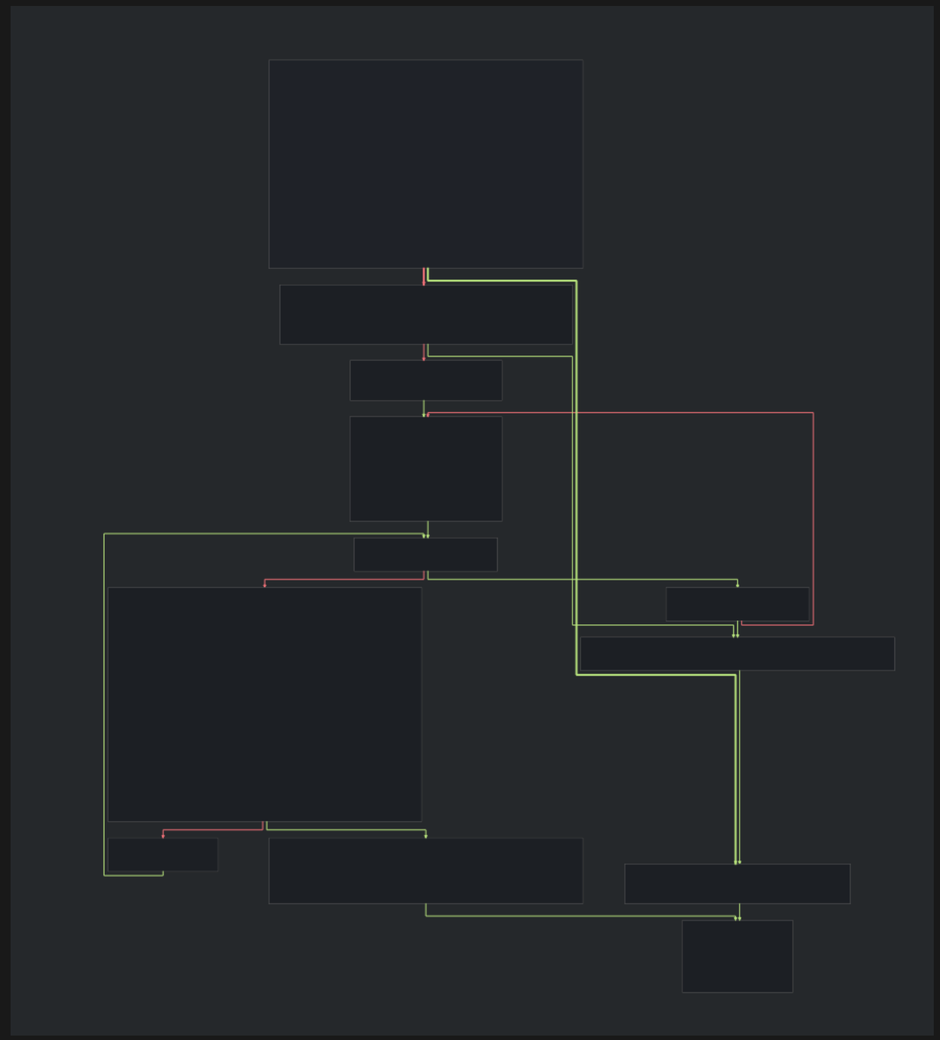
PANIC PADS


0x08000299 mov r15d, dword [rcx + rdx + 0xc]
0x0800029e add r15d, r8d
0x080002a1 jb 0x8000388
[0x080002a7]
0x080002a7 mov esi, dword [rcx + rdx + 8]
0x080002ab sub esi, r8d
0x080002ae jb 0x80003a2
[0x08000388]
0x08000388 mov esi, 0x1c ; 28
0x0800038d mov rdi, str.0 ; 0x8000e70; RELOC 32 .rodata @ 0x08000de0 + 0x90
0x08000394 mov rdx, .rodata ; 0x8001118; RELOC 32 .rodata @ 0x08000de0 + 0x338
0x0800039b call _RNvNtCs4KbNGwnAC5t_4core9panicking5panic; RELOC 32 _RNvNtCs4KbNGwnAC5t_4...
0x080003a0 ud2
[0x080003a2]
0x080003a2 mov esi, 0x21 ; '!' ; 33
0x080003a7 mov rdi, str.1 ; 0x8001150; RELOC 32 .rodata @ 0x08000de0 + 0x370
0x080003ae mov rdx, .rodata ; 0x8001130; RELOC 32 .rodata @ 0x08000de0 + 0x350
0x080003b5 call _RNvNtCs4KbNGwnAC5t_4core9panicking5panic; RELOC 32 _RNvNtCs4KbNGwnAC5t_4...
0x080003ba ud2


0x0800034d call rust_helper_spin_unlock_irqrestor
0x08000352 mov edi, r15d
0x08000355 call _RNvMNtCs8nkueTN8OCo_6kernel5errc
0x0800035a mov ebp, eax
0x0800035c jmp 0x8000377
[0x08000320]
0x08000320 mov eax, dword [var_50h]
0x08000324 test eax, eax
0x08000326 jne 0x8000280
[0x080003bc]
0x080003bc mov esi, 0x1c ; 28
0x080003c1 mov rdi, str.0 ; 0x8000e70; RELOC 32 .rodata @ 0x08000de0 + 0x90
0x080003c8 mov rdx, .rodata ; 0x8001178; RELOC 32 .rodata @ 0x08000de0 + 0x398
0x080003cf call _RNvNtCs4KbNGwnAC5t_4core9panicking5panic; RELOC 32 _RNvNtCs4KbNGwnAC5t_4...
0x080003d4 ud2
[0x0800032c]
0x0800032c mov r14, qword [var_80h]
0x08000331 mov rdx, qword [var_68h]
0x08000336 lea rdi, [var_70h]
0x0800033b jmp 0x8000240

INTEGER OVERFLOW DETECTION





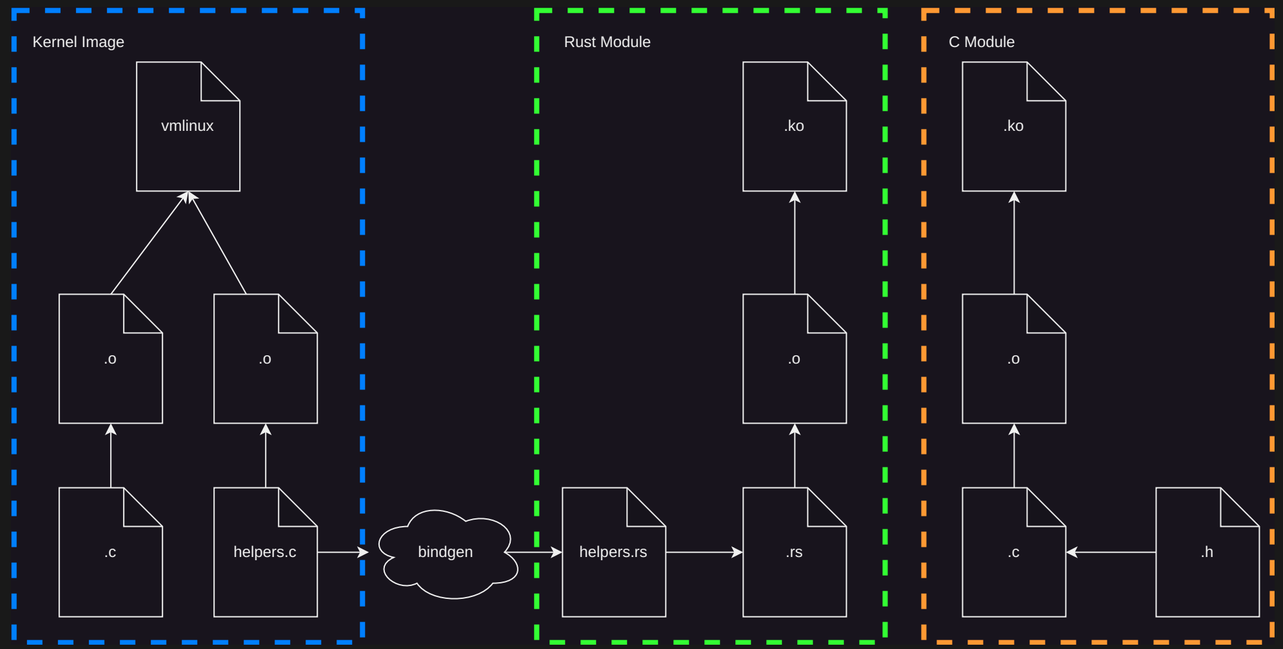
LOOP BODY

```
[0x080002b4]
 0x080002b4      mov     r13d, r15d
 0x080002b7      shr     r13d, 0xc
 0x080002bb      shl     r13, 6
 0x080002bf      add     r13, qword [rcx + rdx]
 0x080002c3      cmp     eax, esi
 0x080002c5      cmovb   esi, eax
 0x080002c8      and     r15d, 0xfff
 0x080002cf      mov     r14d, 0x1000
 0x080002d5      sub     r14d, r15d
 0x080002d8      cmp     esi, r14d
 0x080002db      cmovb   r14d, esi
 0x080002df      lea     rsi, [var_58h]
 0x080002e4      mov     edx, r14d
 0x080002e7      call    rust_helper_bio_advance_iter_single; RELOC 32 rust_helper_bio_advance_ite...
 0x080002ec      mov     qword [var_40h], r13
 0x080002f1      mov     dword [var_38h], r14d
 0x080002f6      mov     dword [var_34h], r15d
 0x080002fb      movzx   edi, byte [rbx + 0x18] ; uint64_t arg1
 0x080002ff      mov     rsi, rbp   ; int64_t arg2
 0x08000302      mov     rdx, r12   ; int64_t arg3
 0x08000305      lea     rcx, [var_40h] ; int64_t arg4
 0x0800030a      call    _RNvMs0_Cs3YbEMwMhJsC_5rnullNtB5_13NullBlkDevice8transfer ; sym._RNvMs0_C...
 0x0800030f      test    eax, eax
 0x08000311      jne     0x8000340
```

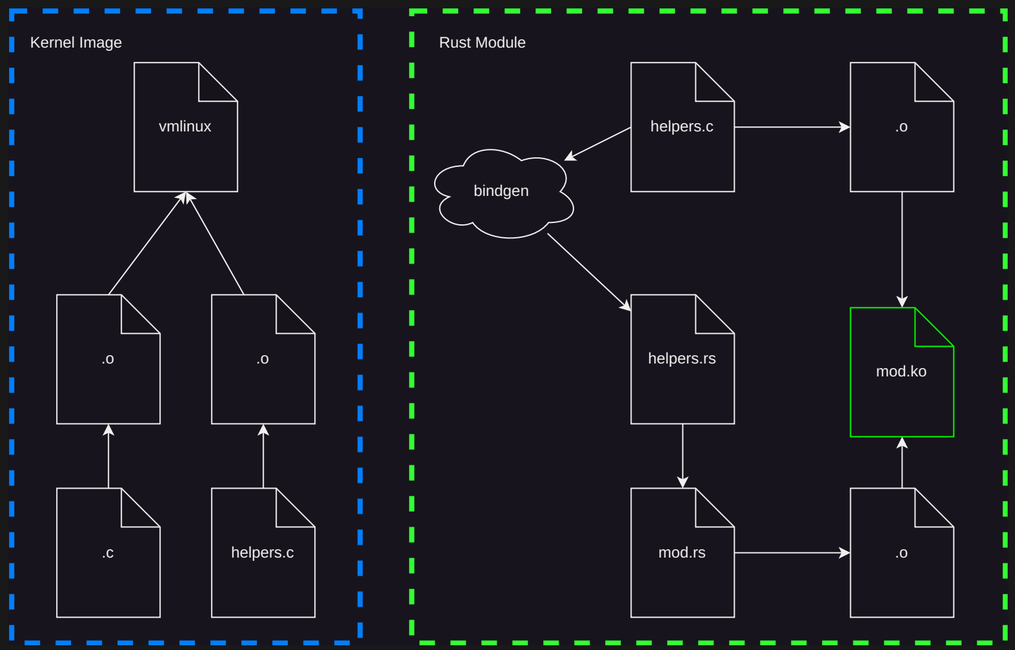
```
shell# objdump -t linux-build/drivers/block/rnull_mod.ko | rustfilt | grep rust_helper_bio_ad
```

```
*UND*  0000000000000000 rust_helper_bio_advance_iter_single
```

HELPERS


Kernel Image
vmlinux
.o
.o
.c
helpers.c
bindgen
Rust Module
.ko
.o
helpers.rs
.rs
C Module
.ko
.o
.c
.h

HELPERS




```c
// rnull-helpers.c:
__attribute__((always_inline))
void rust_helper_bio_advance_iter_single(const struct bio *bio,
                                         struct bvec_iter *iter, unsigned int bytes)
{
    bio_advance_iter_single(bio, iter, bytes);
}
```

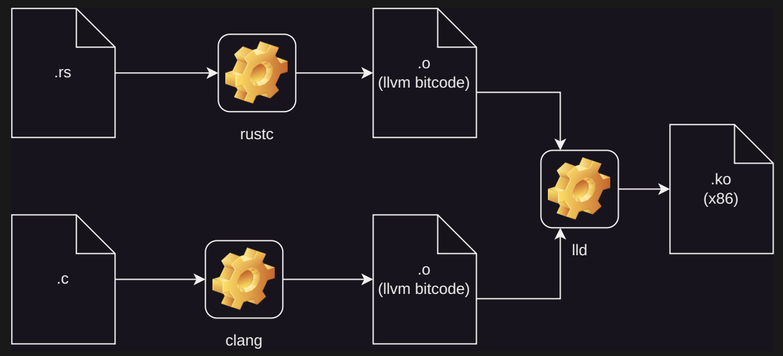
LOOP BODY

```
mov     rcx, qword [var_60h]
mov     rdi, qword [rcx]            ; uint64_t arg1
mov     rcx, qword [rdi + 0x50]
mov     edx, dword [var_4ch]
shl     rdx, 4
mov     esi, dword [var_48h]
mov     r15d, dword [rcx + rdx + 0xc]
add     r15d, esi
mov     r13d, r15d
shr     r13d, 0xc
shl     r13, 6
add     r13, qword [rcx + rdx]
mov     ecx, dword [rcx + rdx + 8]
sub     ecx, esi
cmp     eax, ecx
cmovb   ecx, eax
and     r15d, 0xfff
mov     r14d, 0x1000
sub     r14d, r15d
cmp     ecx, r14d
cmovb   r14d, ecx
lea     rsi, [var_58h]              ; int64_t arg2
mov     edx, r14d                   ; int64_t arg3
call    rust_helper_bio_advance_iter_single ; sym.rust_helper_bio_advance...
mov     qword [var_40h], r13
mov     dword [var_38h], r14d
mov     dword [var_34h], r15d
movzx   edi, byte [rbx + 0x18]      ; uint64_t arg1
mov     rsi, rbp                    ; int64_t arg2
mov     rdx, r12                    ; int64_t arg3
lea     rcx, [var_40h]              ; int64_t arg4
call    _RNvMs0_Cs3YbEMwMhJsC_5rnullNtB5_13NullBlkDevice8transfer ; sym._...
test    eax, eax
jne     0x8000579
```

```
shell# objdump -t linux-build/drivers/block/rnull_mod.ko | rustfilt | grep rust_helper_bio_ad
```

```
F .text  0000000000000055 rust_helper_bio_advance_iter_single
```

LINK TIME OPTIMIZATION WITH LLVM




```
diff --git a/Makefile b/Makefile
index 0c7f6240df2e..a4f8ee7eaa24 100644
--- a/Makefile
+++ b/Makefile
@@ -987,6 +988,7 @@ export CC_FLAGS_SCS
 endif

 ifdef CONFIG_LTO_CLANG
+KBUILD_RUSTFLAGS += -Clinker_plugin_lto
 ifdef CONFIG_LTO_CLANG_THIN
 CC_FLAGS_LTO    := -flto=thin -fsplit-lto-unit
 KBUILD_LDFLAGS += --thinlto-cache-dir=$(extmod_prefix).thinlto-cache
```

IT’S NOT WORKING!

```
file linux-build/drivers/block/rnull-helpers.o
drivers/block/rnull-helpers.o: LLVM IR bitcode
```

.

```
file drivers/block/rnull.o
drivers/block/rnull.o: LLVM IR bitcode
```

.

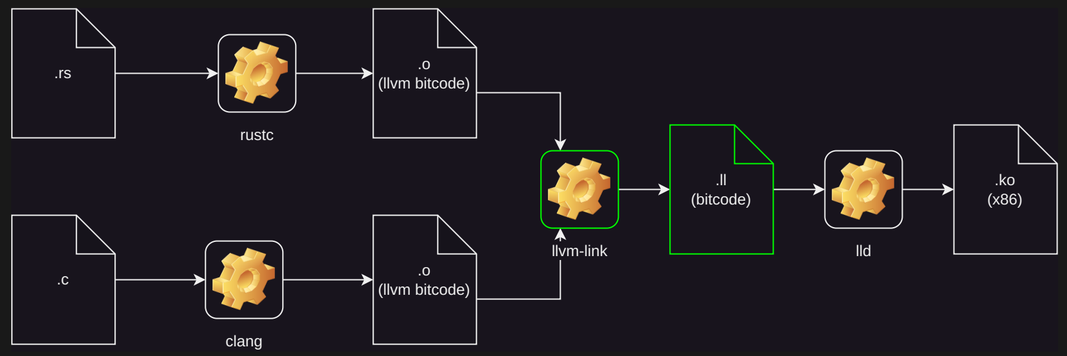
```
file drivers/block/rnull_mod.o
drivers/block/rnull_mod.o: ELF 64-bit LSB relocatable, x86-64, version 1 (SYSV), with debug_info, not stripped
```

.

```
llvm-nm drivers/block/rnull-helpers.o  | grep rust_helper_bio
---------------- T rust_helper_bio_advance_iter_single
```

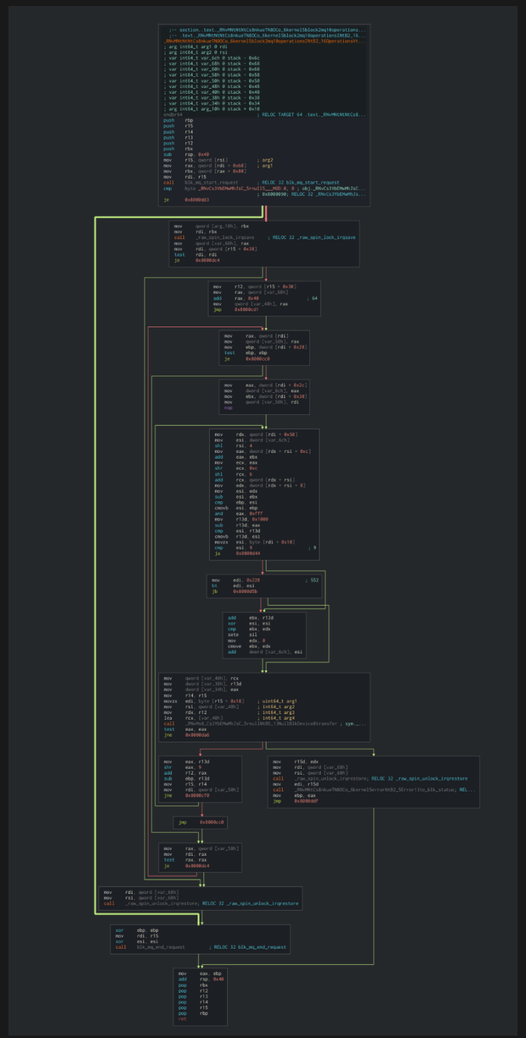
ONE MORE HACK

```
modified   scripts/Makefile.build
@@ -425,7 +425,7 @@ $(obj)/lib.a: $(lib-y) FORCE
    $(call if_changed,ar)

 quiet_cmd_ld_multi_m = LD [M]  $@
-      cmd_ld_multi_m = $(LD) $(ld_flags) -r -o $@ @$(patsubst %.o,%.mod,$@) $(cmd_objtool)
+      cmd_ld_multi_m = llvm-link -o $(patsubst %.o,%.ll.o,$@) $(shell cat $(patsubst %.o,%.mod,$@) | xargs); $(LD) ...

 define rule_ld_multi_m
    $(call cmd_and_savecmd,ld_multi_m)
```



This is broken for modules that link assembly files

SUCCESS!



CAVEATS

- `objcopy --redefine-sym __addsf3=__rust__addsf3 ...` on `core.o` is a problem
- `llvm-link` rule breaks modules that use assembly source files
- Kernel does not boot on real hardware with `CONFIG_LTO_CLANG`
- YMMV regarding performance
  - **No observable performance difference** for `rnull` in virtual machine
  - Tests for `rnvme` pending

QUESTIONS?