Rust in the Linux Kernel

My journey to a ArchLinux kernel with support for Rust modules

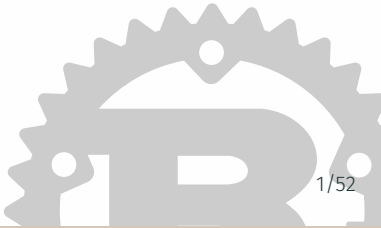
Raphael Nestler (@rnestler)

2023-07-03

Rust Zürichsee Meetup

println!("{:?}", rnestler)

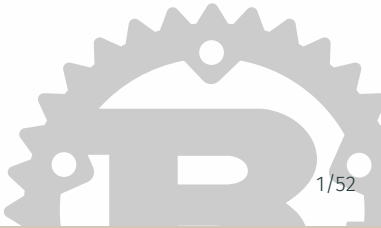
Hi! I'm Raphael (@rnestler).



println!("{:?}", rnestler)

Hi! I'm Raphael (@rnestler).

I live in Rapperswil

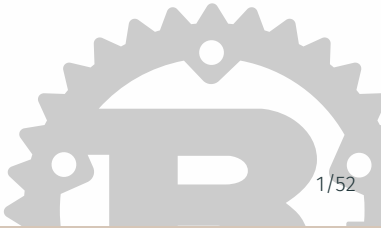


println!("{:?}", rnestler)

Hi! I'm Raphael (@rnestler).

I live in Rapperswil

I work at Renuo (`https://renuo.ch`).



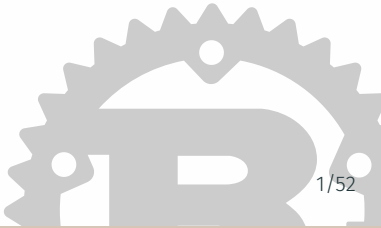
println!("{:?}", rnestler)

Hi! I'm Raphael (@rnestler).

I live in Rapperswil

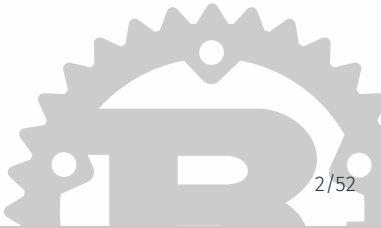
I work at Renuo (`https://renuo.ch`).

I'm a founding member of Coredump hackerspace (`https://coredump.ch`).



Disclaimer

- I use Rust since early 2015
- I have some experience in Embedded Rust
- But I’m not a Linux kernel developer!



Outline

1. Quickstart
2. A brief history of Rust in Linux
3. My Journey
4. Some Outlook

Quickstart

Install the kernel

```
# Edit /etc/makepkg.conf and make sure it uses
# all your cores by adding the following:
MAKEFLAGS="-j8"
$ git clone https://aur.archlinux.org/linux-rust.git
$ cd linux-rust
$ makepkg # this takes some minutes
$ sudo pacman -U linux-rust-headers-6.3.9.*pkg.tar.zst
$ sudo pacman -U linux-rust-6.3.9*.pkg.tar.zst
# Reboot into the new kernel. You may need to configure
# your bootloader for that.
```

Compile and load the out-of-tree module

```
$ git clone git@github.com:Rust-for-Linux/rust-out-of-tree-module.git
$ cd rust-out-of-tree-module
$ make LLVM=1
# Or if you don't run the Rust kernel yet
$ make KDIR=/lib/modules/6.3.9-arch1-1-rust/build LLVM=1
$ dmesg --follow # in a seperate terminal
$ sudo insmod rust_out_of_tree.ko
```

A brief history of Rust in Linux

Early History of Rust in Linux

- Oldest reference I found is from 2013![1] (maintained until 2019)

[1] https://github.com/tsgates/rust.ko

Early History of Rust in Linux

- Oldest reference I found is from 2013![1] (maintained until 2019)
- Writing Linux Kernel Modules in Safe Rust (Linux Security Summit 2019)[2] [3] (maintained until 2021)

[1] https://github.com/tsgates/rust.ko
[2] https://lssna19.sched.com/event/RHaT
[3] https://github.com/fishinabarrel/linux-kernel-module-rust

How did it look like?

Kenel Module from the slides of the LSS talk

```
struct HelloWorldModule;
impl KernelModule for HelloWorldModule {
  fn init() -> KernelResult<Self> {
    println!("Hello world!");
    Ok(HelloWorldModule)
  }
}
kernel_module!(HelloWorldModule, license: "GPL");
```

Plans for mainline

- July 2020: Discussion on LKML for a session at LPC [4]
- July/August 2020: Session about obstacles to accepting Rust upstream[5]

[4] https://lore.kernel.org/lkml/CANiq72=rFzxMyxDNkobdnMZohT_qT0KfGCincYBteyoJbtr2Gw@mail.gmail.com/

[5] https://lwn.net/Articles/829858/

Mainlining it

- March 2021: Initial Rust support lands in Linux-Next[6]
- April 2021: Official RFC for Rust support[7]

[6] https://www.phoronix.com/news/Rust-Hits-Linux-Next
[7] https://lkml.org/lkml/2021/4/14/1023

Mainlining it

- March 2021: Initial Rust support lands in Linux-Next[6]
- April 2021: Official RFC for Rust support[7]
- June 2021: Goolge starts funding ”Rust for Linux”[8]
- July 2021 - September 2022: V2 - V10 of the patch set

[6] https://www.phoronix.com/news/Rust-Hits-Linux-Next
[7] https://lkml.org/lkml/2021/4/14/1023
[8] https://www.phoronix.com/news/Google-Wants-Rust-In-Kernel

Mainlining it

- October 2022: Rust support merged for Linux 6.1[9]
- December 2022: Linux 6.1 Released with initial Rust support![10]

[9] https://www.phoronix.com/news/Rust-Is-Merged-Linux-6.1
[10] https://www.phoronix.com/news/Linux-6.1-Released

Mainlining it

- October 2022: Rust support merged for Linux 6.1[9]
- December 2022: Linux 6.1 Released with initial Rust support![10]
- December 2022: I start playing around with it!

[9] https://www.phoronix.com/news/Rust-Is-Merged-Linux-6.1
[10] https://www.phoronix.com/news/Linux-6.1-Released

My Journey

My Journey

December 2022 — The naïve start

Install and try it

It should be very straightforward

1. Verify that the ArchLinux kernel has `CONFIG_HAVE_RUST` set[11]
2. Install Linux 6.1 from the testing repository
3. Clone `https://github.com/Rust-for-Linux/rust-out-of-tree-module`
4. Compile and Run

[11] `https://gitlab.archlinux.org/archlinux/packaging/packages/linux/-/blob/6.1.arch1-1/config#L786`

Install and try it

It should be very straightforward

1. Verify that the ArchLinux kernel has `CONFIG_HAVE_RUST` set[11]
2. Install Linux 6.1 from the testing repository
3. Clone `https://github.com/Rust-for-Linux/rust-out-of-tree-module`
4. Compile and Run
5. Profit?

[11] `https://gitlab.archlinux.org/archlinux/packaging/packages/linux/-/blob/6.1.arch1-1/config#L786`

Well...

```
rust-out-of-tree-module (git)-[main] % make
make -C /lib/modules/`uname -r`/build M=$PWD
make[1]: Entering directory '/usr/lib/modules/6.1.0-arch1-1/buil
  RUSTC [M] ~/projects/github/Rust-for-Linux/rust-out-of-tree-mo
error: target file "./rust/target.json" does not exist

make[2]: *** [scripts/Makefile.build:307: ~/projects/github/Rust
make[1]: *** [Makefile:1992: ~/projects/github/Rust-for-Linux/ru
make[1]: Leaving directory '/usr/lib/modules/6.1.0-arch1-1/build
make: *** [Makefile:6: default] Error 2
```

RTF README

> *The kernel tree (KDIR) requires the Rust metadata to be available. These are generated during the kernel build, but may not be available for installed/distributed kernels (the scripts that install/distribute kernel headers etc. for the different package systems and Linux distributions are not updated to take into account Rust support yet).*

README from `https://github.com/Rust-for-Linux/rust-out-of-tree-module`

Trying it in Kernel

1. Clone https://github.com/Rust-for-Linux/linux
2. RTFM[12]
3. Verify that we can build that kernel and build the out-of-tree module against that.

[12] `https://www.kernel.org/doc/html/latest/rust/quick-start.html`

Well...

```
Rust-for-Linux/linux (git)-[rust] % make LLVM=1 rustavailable
***
*** Rust compiler 'rustc' is too new. This may or may not work.
***   Your version:     1.66.0
***   Expected version: 1.62.0
***
***
*** Rust bindings generator 'bindgen' is too new. This may or may not work.
***   Your version:     0.63.0
***   Expected version: 0.56.0
***
Rust is available!
```

We can fix that!

```
rustup override set $(scripts/min-tool-version.sh rustc)
rustup component add rust-src
cargo install --locked --version $(scripts/min-tool-version.sh bindgen)
↪  bindgen
```

Trying it in Kernel

1. Switch some stuff in `make menuconfig`[13]
2. Compile the kernel `make LLVM=1` → works
3. Switch to the kernel sources matching my running kernel and compile again → works

[13] In hindsight this was an important hint

Now it just *has* to work, right?

Now it just *has* to work, right?

```
rust-out-of-tree-module (git)-[main] % make KDIR=../linux LLVM=1
...
error: proc macro panicked
  --> ~/github/Rust-for-Linux/rust-out-of-tree-module/rust_out_of_tree.rs:7:1
   |
7  | / module! {
8  | |     type: RustOutOfTree,
9  | |     name: "rust_out_of_tree",
10 | |     author: "Rust for Linux Contributors",
11 | |     description: "Rust out-of-tree sample",
12 | |     license: "GPL",
13 | | }
   | |_^
   |
   = help: message: Expected byte string
```

We can fix that!

Just change every string to `b"string"` and fix the other compile errors [14]

[14] `https://github.com/Rust-for-Linux/rust-out-of-tree-module/pull/3`

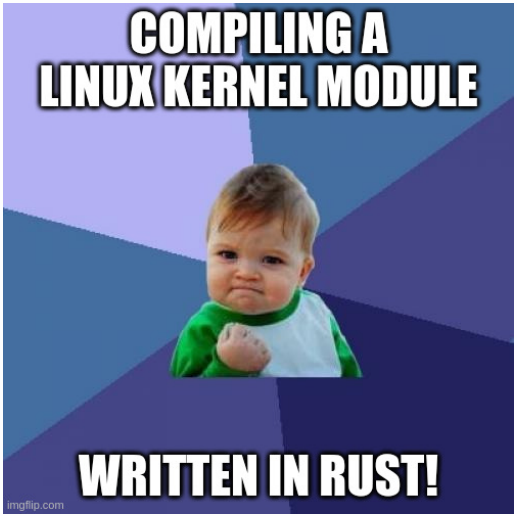
We can fix that!

Just change every string to `b"string"` and fix the other compile errors [14]

```
Rust-for-Linux/rust-out-of-tree-module (git)-[main] % make KDIR=../linux LLV
make -C ../linux M=$PWD
make[1]: Entering directory '~/projects/github/Rust-for-Linux/linux'
  RUSTC [M] ~/projects/github/Rust-for-Linux/rust-out-of-tree-module/rust_ou
  MODPOST ~/projects/github/Rust-for-Linux/rust-out-of-tree-module/Module.sy
  LD [M]  ~/projects/github/Rust-for-Linux/rust-out-of-tree-module/rust_out_
make[1]: Leaving directory '~/projects/github/Rust-for-Linux/linux'
```

[14] https://github.com/Rust-for-Linux/rust-out-of-tree-module/pull/3

It compiles!


COMPILING A
LINUX KERNEL MODULE
WRITTEN IN RUST!
imgflip.com

Now load it!

```
$ sudo insmod rust_out_of_tree.ko
insmod: ERROR: could not insert module rust_out_of_tree.ko: Invalid
↪  module format
$ dmesg | grep rust_out_of_tree
[24593.691184] rust_out_of_tree: version magic '6.1.0-arch1 SMP preempt
↪  mod_unload ' should be '6.1.0-arch1-1 SMP preempt mod_unload '
```

Now load it!

```
$ sudo insmod rust_out_of_tree.ko
insmod: ERROR: could not insert module rust_out_of_tree.ko: Invalid
↪  module format
$ dmesg | grep rust_out_of_tree
[24593.691184] rust_out_of_tree: version magic '6.1.0-arch1 SMP preempt
↪  mod_unload ' should be '6.1.0-arch1-1 SMP preempt mod_unload '
```

Alright I made a tiny mistake in defining the kernel version...

Now load it!

```
$ sudo insmod rust_out_of_tree.ko
insmod: ERROR: could not insert module rust_out_of_tree.ko: Unknown
↪  symbol in module

$ dmesg | grep rust_out_of_tree
[25300.463334] rust_out_of_tree: Unknown symbol _RNvNtNtCsfATHBUcknU9_6kerne
[25300.463353] rust_out_of_tree: Unknown symbol _RNvNtCsfATHBUcknU9_6kernel5
[25300.463371] rust_out_of_tree: Unknown symbol __rust_dealloc (err -2)
[25300.463386] rust_out_of_tree: Unknown symbol __rust_realloc (err -2)
[25300.463400] rust_out_of_tree: Unknown symbol __rust_alloc (err -2)
...
```

Now load it!

```
$ sudo insmod rust_out_of_tree.ko
insmod: ERROR: could not insert module rust_out_of_tree.ko: Unknown
↪  symbol in module

$ dmesg | grep rust_out_of_tree
[25300.463334] rust_out_of_tree: Unknown symbol _RNvNtNtCsfATHBUcknU9_6kerne
[25300.463353] rust_out_of_tree: Unknown symbol _RNvNtCsfATHBUcknU9_6kernel5
[25300.463371] rust_out_of_tree: Unknown symbol __rust_dealloc (err -2)
[25300.463386] rust_out_of_tree: Unknown symbol __rust_realloc (err -2)
[25300.463400] rust_out_of_tree: Unknown symbol __rust_alloc (err -2)
...
```

Hmm...

Investigating

Try to use the ArchLinux config file directly and recompile

```
% make KDIR=../linux LLVM=1
make -C ../linux M=$PWD
make[1]: Entering directory '~/projects/github/Rust-for-Linux/linux'
  MODPOST .../Module.symvers
ERROR: modpost: "_RNvNtNtCsfATHBUcknU9_6kernel5print14format_strings4INFO" [
ERROR: modpost: "_RNvNtCsfATHBUcknU9_6kernel5print11call_printk" [.../rust_o
ERROR: modpost: "__rust_dealloc" [.../rust_out_of_tree.ko] undefined!
ERROR: modpost: "__rust_realloc" [.../rust_out_of_tree.ko] undefined!
ERROR: modpost: "__rust_alloc" [.../rust_out_of_tree.ko] undefined!
```

Investigating

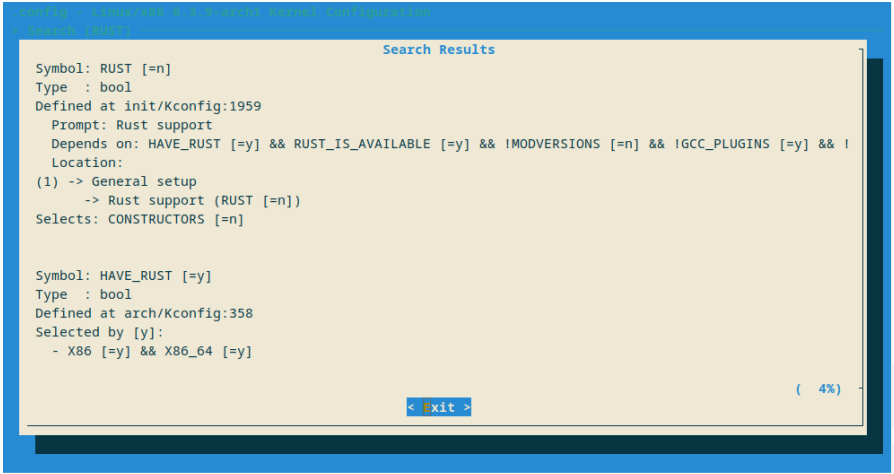
Try to use the ArchLinux config file directly and recompile

```
% make KDIR=../linux LLVM=1
make -C ../linux M=$PWD
make[1]: Entering directory '~/projects/github/Rust-for-Linux/linux'
  MODPOST .../Module.symvers
ERROR: modpost: "_RNvNtNtCsfATHBUcknU9_6kernel5print14format_strings4INFO" [
ERROR: modpost: "_RNvNtCsfATHBUcknU9_6kernel5print11call_printk" [.../rust_o
ERROR: modpost: "__rust_dealloc" [.../rust_out_of_tree.ko] undefined!
ERROR: modpost: "__rust_realloc" [.../rust_out_of_tree.ko] undefined!
ERROR: modpost: "__rust_alloc" [.../rust_out_of_tree.ko] undefined!
```

This looks oddly familiar!

The end

The ArchLinux kernel doesn't have Rust support. make menuconfig reveals, that options which conflict with CONFIG_HAVE_RUST=y are enabled:



What did we learn?

- Rust support needs to compile the kernel with clang/LLVM
- The kernel only attempts to load modules which have the same version magic
- It needs a specific *stable* Rust version (since it uses the `RUSTC_BOOTSTRAP=1` trick to enable unstable features on stable compilers)
- The module interface is (of course) still in flux
- The kernel has automatic configuration variables which detect if stuff can be enabled (`CONFIG_RUST_IS_AVAILABLE`)
- Actual Rust support depends on a combination of multiple variables

My Journey

January 2023 — Compiling our own kernel

Building our own kernel

- The ArchLinux kernel doesn't support Rust modules
- I want to have a bootable / useable kernel $\rightarrow$ take the ArchLinux kernel as base and only do minimal changes
- Alternative: Just use the Rust-for-Linux kernel

Building our own kernel

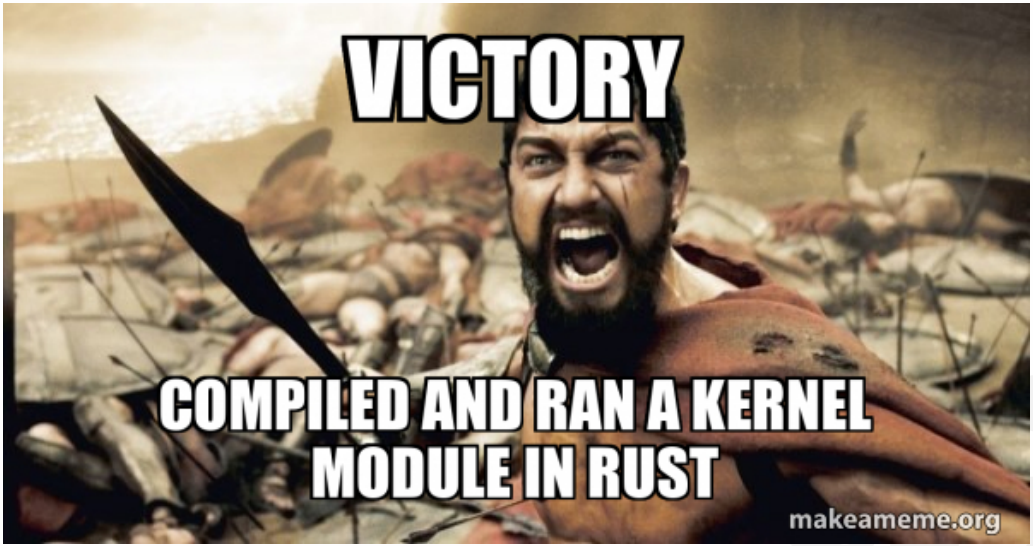
- Create https://aur.archlinux.org/packages/linux-rust
- Take the ArchLinux kernel package
- Run `make menuconfig`
- Enhance the build process with Rust specific stuff
- Build kernel, reboot, profit[15]

[15] Of course this needed some iterations in the build process

First sucesses!

```
$ make
 ↪  KDIR=~/projects/archpkg/linux-rust/src/archlinux-linux
 ↪  LLVM=1
$ sudo insmod rust_out_of_tree.ko
# dmesg output
[  451.297415] rust_out_of_tree: loading out-of-tree module
 ↪  taints kernel.
[  451.297460] rust_out_of_tree: module verification
 ↪  failed: signature and/or required key missing -
 ↪  tainting kernel
[  451.297724] rust_out_of_tree: Rust out-of-tree sample
 ↪  (init)
```

First sucesses!



A small issue...

- Did you notice our workaround?

A small issue...

- Did you notice our workaround?
- Right: `KDIR=~/projects/archpkg/linux-rust/src/archlinux-linux`
- This meens we need the sources of our kernel build around. Not nice if we want to enable people to install it via package manager

Remember build metadata?

- Remember the quote about build metadata in the readme?
- Let's try to package this correctly![a]

[a] The build metadata goes in the linux-rust-headers package. The name is slightly missleading, because it contains various other build artifacts

Just follow the errors

My strategy:

- Try to build the out-of-tree module
- Check the errors
- Add files to the metadata which should fix the issue
- `makepkg --repackage -f` is our friend

First error: target.json

```
rust-out-of-tree-module (git)-[main] % make LLVM=1
make -C /lib/modules/`uname -r`/build M=$PWD
make[1]: Entering directory '/usr/lib/modules/6.1.5-arch2-1-rust
  RUSTC [M] ~/projects/github/Rust-for-Linux/rust-out-of-tree-mo
error: target file "./rust/target.json" does not exist
```

→ Let's add `install -Dt "$builddir/rust" -m644 scripts/target.json`

Next error: core library

```
make -C /lib/modules/`uname -r`/build M=$PWD
make[1]: Entering directory '/usr/lib/modules/6.1.5-arch2-1-rust
  RUSTC [M] ~/projects/github/Rust-for-Linux/rust-out-of-tree-mo
error[E0463]: can't find crate for `core`
  |
  = note: the `target` target may not be installed
  = help: consider downloading the target with `rustup target ad
  = help: consider building the standard library from source wit
```

→ Let's just add everything from `rust/`![16]

[16] This decision happened after several iterations...

Next error: Rust version

```
error[E0514]: found crate `core` compiled by an
 ↪  incompatible version of rustc
  = note: the following crate versions were found:
          crate `core` compiled by rustc 1.62.0 (a8314ef7d
           ↪  2022-06-27): /usr/lib/modules/6.1.5-arch2-1-
           ↪  rust/build/rust/libcore.rmeta
  = help: please recompile that crate using this compiler
   ↪  (rustc 1.66.0 (69f9c33d7 2022-12-12)) (consider
   ↪  running `cargo clean` first)
```

→ We need to put `rust-toolchain` file into `/usr/lib/modules/...`, not the out-of-tree module folder!

Target tripple what?

```
make -C /lib/modules/`uname -r`/build M=$PWD
make[1]: Entering directory
 ↪  '/usr/lib/modules/6.1.5-arch2-1-rust/build'
  RUSTC [M] /home/roughl/projects/github/Rust-for-
   ↪  Linux/rust-out-of-tree-module/rust_out_of_tree.o
error[E0461]: couldn't find crate `core` with expected
 ↪  target triple target-12809083303779448358
  |
  = note: the following crate versions were found:
          crate `core`, target triple
           ↪  target-3911737072772191946:
           ↪  /usr/lib/modules/6.1.5-arch2-1-
           ↪  rust/build/rust/libcore.rmeta
```

Target tripple what?

- WAT?
- Target tripples define for what you want to compile (`x86_64-unknown-linux-gnu` for example)
- What was that strange number? Don't we have a custom `target.json` anyway?

What did we learn? (part 2)

- Building our own workable kernel was surprisingly easy!
- Packaging the correct build metadata not
- `target.json` is used to describe the kernel environement to the Rust compiler
- The kernel uses it's own Rust library (mostly std libcore, but with custom liballoc for example)
- out-of-tree module builds get executed in the context of the build metadata and are passed the directory of the module.

My Journey

June 2023 — Packing the correct metadata

Motivation

- I did update my kernel a few times to keep it up to date and checked if anything changed with Linux 6.2[17]
- I did propose to do this talk
- I had some nice interactions with Miguel Ojeda from the Rust-for-Linux project!

[17] Here I needed to change back the changes we made to the out-of-tree module `https://github.com/Rust-for-Linux/rust-out-of-tree-module/pull/5`

What I learned growing up with C++/Rust

Never blame the compiler for an issue you have!

What I learned growing up with C++/Rust

Never blame the compiler for an issue you have!

It was the compiler...

- For custom `target.json` the Rust compiler uses a hash to assert that the target it the same (the strange number)
- The Rust compiler decided (probably as a quick hack?) to use the *path* instead of the *content* of the `target.json` to calculate this hash! This was fixed in Rustr 1.63.0[18]

[18] `https://github.com/rust-lang/rust/pull/98225/`

It was the compiler...

- For custom `target.json` the Rust compiler uses a hash to assert that the target it the same (the strange number)
- The Rust compiler decided (probably as a quick hack?) to use the *path* instead of the *content* of the `target.json` to calculate this hash! This was fixed in Rustr 1.63.0[18]
- But we need to use 1.62.0

[18] `https://github.com/rust-lang/rust/pull/98225/`

It was the compiler...

- For custom `target.json` the Rust compiler uses a hash to assert that the target it the same (the strange number)
- The Rust compiler decided (probably as a quick hack?) to use the *path* instead of the *content* of the `target.json` to calculate this hash! This was fixed in Rustr 1.63.0[18]
- But we need to use 1.62.0, right?

[18] `https://github.com/rust-lang/rust/pull/98225/`

Compiling with 1.63.0

Wrong! We only get some minor warnings in the output:

```
*** Rust compiler 'rustc' is too new. This may or may not work.
***   Your version:     1.63.0
***   Expected version: 1.62.0
warning: the feature `nll` has been stable since 1.63.0 and no l
   --> rust/alloc/lib.rs:170:12
    |
170 | #![feature(nll)] // Not necessary, but here to test the `r
    |            ^^^
    |
    = note: `#[warn(stable_features)]` on by default
```

Getting the necessary build metadata

- `rust/target.json` the custom target description
- `rust/*.rmeta` the compiled rust libraries
- `rust/*.so` the pre-compiled procedural macros
- `rust-toolchain` such that we use the same Rust compiler version that was used to compile the kernel

What did we learn? (part 3)

- Sometimes it *is* the compiler
- Just upgrading one version of the Rust compiler was safe (A stabilized feature should probably not change from the last release)

My Journey

June 2023 — There and Back Again

Updating to 6.3

- We did it
- We can compile out-of-tree modules!
- Now we can just update and play around with new features that land for Rust modules!

Updating to 6.3

- We did it
- We can compile out-of-tree modules!
- Now we can just update and play around with new features that land for Rust modules!
- Right?

Updating to 6.3

Wrong!

```
rust-out-of-tree-module (git)-[main] % make LLVM=1
make -C /lib/modules/6.3.8-arch1-1-rust/build M=$PWD
 RUSTC [M] rust-out-of-tree-module/rust_out_of_tree.o
error: could not write output to
 ↪  rust_out_of_tree.rust_out_of_tree.ad9c6f77-
 ↪  cgu.0.rcgu.o: Permission denied

error: aborting due to previous error
```

Updating to 6.3

Asking on the Rust-for-Linux chat[19]

`rustc` writes temporary files to the current directory[20], even if we specify with `--emit` where we want our output file.

We need to pass `--out-dir`

[19] `https://rust-for-linux.zulipchat.com/#narrow/stream/291565-Help/topic/Out.20of.20tree.20module.20for.20mainline.20kernel`

[20] Which is in `/lib/modules/...` for the module build

Updating to 6.3

```
diff --git a/scripts/Makefile.build
 ↪  b/scripts/Makefile.build
index 94d67252df4e..261e51c0af59 100644
--- a/scripts/Makefile.build
+++ b/scripts/Makefile.build
@@ -287,7 +287,7 @@ rust_common_cmd = \
        --extern alloc --extern kernel \
        --crate-type rlib -L $(objtree)/rust/ \
        --crate-name $(basename $(notdir $@)) \
-       --emit=dep-info=$(depfile)
+       --out-dir $(dir $@) --emit=dep-info=$(depfile)
```

What did we learn? (part 4)

- Nobody except me cares for out-of-tree Rust kernel modules so far 😉
- The Rust-for-Linux community is very helpful! 🎉

Some Outlook

Linux 6.4

- Got released recently → I will update the package
- Mainlines a few more things like the ‘pin-init‘ API[21]
- Maybe I’ll actually *start* with playing around with kernel modules in Rust 😉

[21] https://lore.kernel.org/lkml/20230429012119.421536-1-ojeda@kernel.org

Linux 6.5

- Will upgrade rustc to 1.68.2[22]
- Also the start of a new rustc version policy
  This is the first such upgrade, and we will try to update it often from now on, in order to remain close to the latest release, until a minimum version (which is "in the future") can be established.

[22] `https://lore.kernel.org/lkml/20230618161558.1051269-1-ojeda@kernel.org`

Interesting Stuff happening

- Apple M1 GPU driver[23]
- Rust null block driver (for experimenting with the block device API)[24]
- Rust abstractions for network device drivers[25]

[23] https://asahilinux.org/2022/11/tales-of-the-m1-gpu/
[24] https://lore.kernel.org/linux-block/20230503090708.2524310-1-nmi@metaspace.dk/
[25] https://lore.kernel.org/rust-for-linux/01010188843258ec-552cca54-4849-4424-b671-7a5bf9b8651a-000000@us-west-2.amazonses.com/

Thank you!

https://coredump.ch

https://renuo.ch/

Slides: https://github.com/rust-zurichsee/meetups/

Appendix

target.json

```
{
    "arch": "x86_64",
    "data-layout": "e-m:e-p270:32:32-p271:32:32-p272:64:64-
↪   i64:64-f80:128-n8:16:32:64-S128",
    "features": "-3dnow,-3dnowa,-mmx,+soft-
↪   float,+retpoline-external-thunk",
    "llvm-target": "x86_64-linux-gnu",
    "target-pointer-width": "64",
    "emit-debug-gdb-scripts": false,
    "frame-pointer": "may-omit",
    "stack-probes": {"kind": "none"}
}
```