Rust for Linux 驱动开发技术分享

萧络元

2023.02.05

提纲

- **Rust for Linux**
  - 相关背景
  - Rust支持代码合并情况
  - 现有的Rust for Linux 设备驱动
- **Rust驱动开发**
  - Intel E1000网卡驱动开发案例
  - 驱动相关的接口trait实现
  - 非OS依赖的独立e1000驱动开发
- **驱动的测试展示**

s Dirk Hohndel (left) talks to Linus Torvalds at the virtual Open Source S
Credit: Linux Foundation

背景 - Rust for Linux

先看看Linus大佬对 Rust for Linux 怎么看?

Linux怎么看

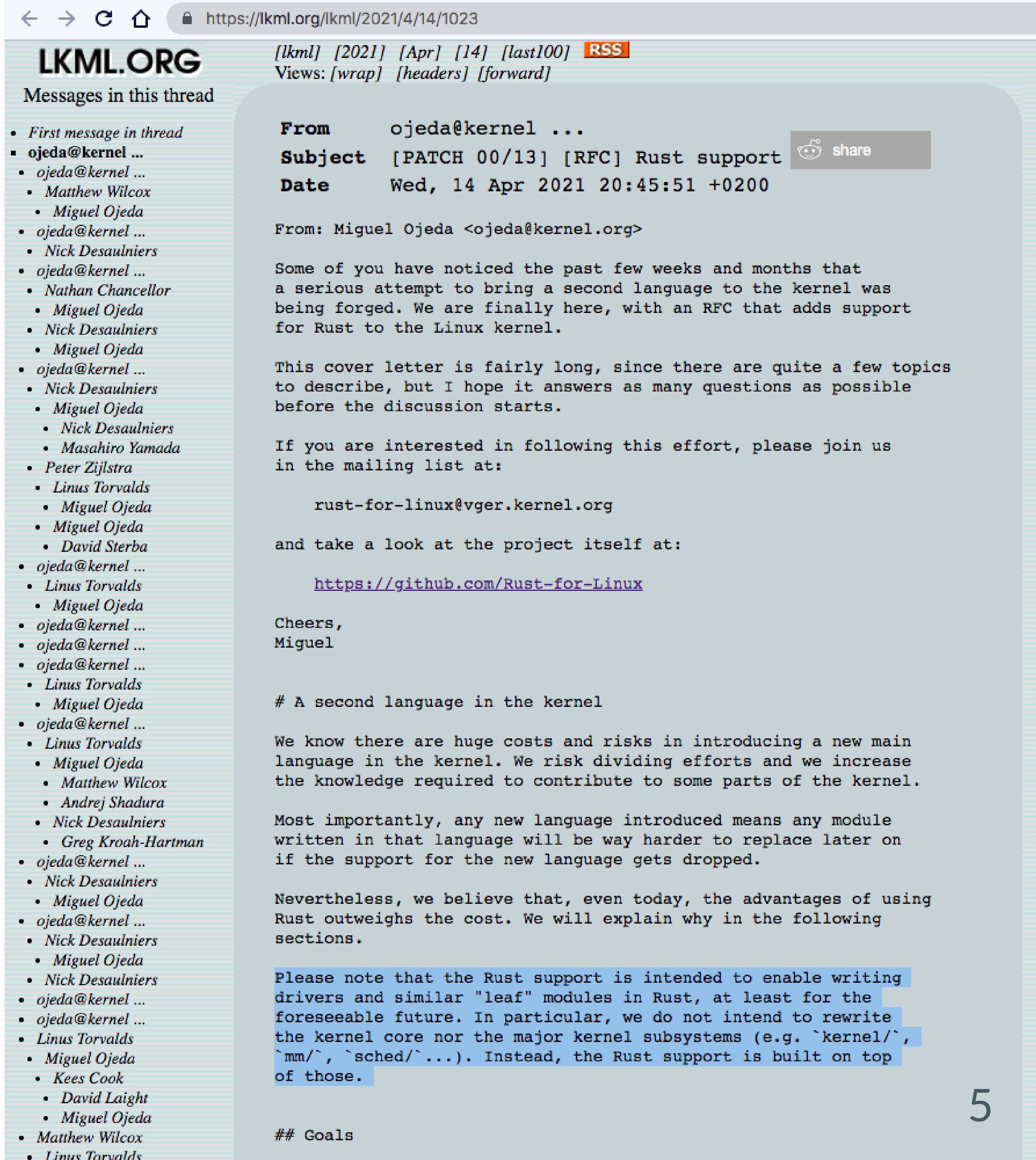
Linus与VMware首席开源官Dirk Hohndel
在2020年6月的开源峰会的对话表示：

- 对于Driver这些非Kernel中心组件，考虑用C語言以外的编程語言開發，如Rust，重点驱动模块

> “Linus Torvalds said that for things "not very central to the kernel itself", like drivers, the kernel team is looking at "having interfaces to do those, for example, in Rust. ”

Rust support mail list

2021年4月的Linux内核邮件列表中，提交了
Rust support [RFC]



https://lkml.org/lkml/2021/4/14/1023

LKML.ORG

[lkml] [2021] [Apr] [14] [last100] RSS
Views: [wrap] [headers] [forward]

Messages in this thread

- First message in thread
- ojeda@kernel ...
  - ojeda@kernel ...
    - Matthew Wilcox
      - Miguel Ojeda
  - ojeda@kernel ...
    - Nick Desaulniers
  - ojeda@kernel ...
    - Nathan Chancellor
      - Miguel Ojeda
    - Nick Desaulniers
      - Miguel Ojeda
  - ojeda@kernel ...
    - Nick Desaulniers
      - Miguel Ojeda
        - Nick Desaulniers
        - Masahiro Yamada
    - Peter Zijlstra
      - Linus Torvalds
        - Miguel Ojeda
      - Miguel Ojeda
        - David Sterba
  - ojeda@kernel ...
    - Linus Torvalds
      - Miguel Ojeda
  - ojeda@kernel ...
  - ojeda@kernel ...
  - ojeda@kernel ...
    - Linus Torvalds
      - Miguel Ojeda
  - ojeda@kernel ...
    - Linus Torvalds
      - Miguel Ojeda
        - Matthew Wilcox
        - Andrej Shadura
      - Nick Desaulniers
        - Greg Kroah-Hartman
  - ojeda@kernel ...
    - Nick Desaulniers
      - Miguel Ojeda
  - ojeda@kernel ...
    - Nick Desaulniers
      - Miguel Ojeda
    - Nick Desaulniers
  - ojeda@kernel ...
  - ojeda@kernel ...
  - Linus Torvalds
    - Miguel Ojeda
      - Kees Cook
        - David Laight
        - Miguel Ojeda
  - Matthew Wilcox
    - Linus Torvalds

From ojeda@kernel ...
Subject [PATCH 00/13] [RFC] Rust support
Date Wed, 14 Apr 2021 20:45:51 +0200

share

```
From: Miguel Ojeda <ojeda@kernel.org>

Some of you have noticed the past few weeks and months that
a serious attempt to bring a second language to the kernel was
being forged. We are finally here, with an RFC that adds support
for Rust to the Linux kernel.

This cover letter is fairly long, since there are quite a few topics
to describe, but I hope it answers as many questions as possible
before the discussion starts.

If you are interested in following this effort, please join us
in the mailing list at:

    rust-for-linux@vger.kernel.org

and take a look at the project itself at:

    https://github.com/Rust-for-Linux

Cheers,
Miguel


# A second language in the kernel

We know there are huge costs and risks in introducing a new main
language in the kernel. We risk dividing efforts and we increase
the knowledge required to contribute to some parts of the kernel.

Most importantly, any new language introduced means any module
written in that language will be way harder to replace later on
if the support for the new language gets dropped.

Nevertheless, we believe that, even today, the advantages of using
Rust outweighs the cost. We will explain why in the following
sections.

Please note that the Rust support is intended to enable writing
drivers and similar "leaf" modules in Rust, at least for the
foreseeable future. In particular, we do not intend to rewrite
the kernel core nor the major kernel subsystems (e.g. `kernel/`,
`mm/`, `sched/`...). Instead, the Rust support is built on top
of those.


## Goals
```

Rust support 邮件列表

2021年4月的Rust support [RFC]写道：

"(Rust) 因为 memory safety 以及该系统编程语言上相比于 C 有更高级的特征与功能，使得开发过程中更容易编程以及维护"

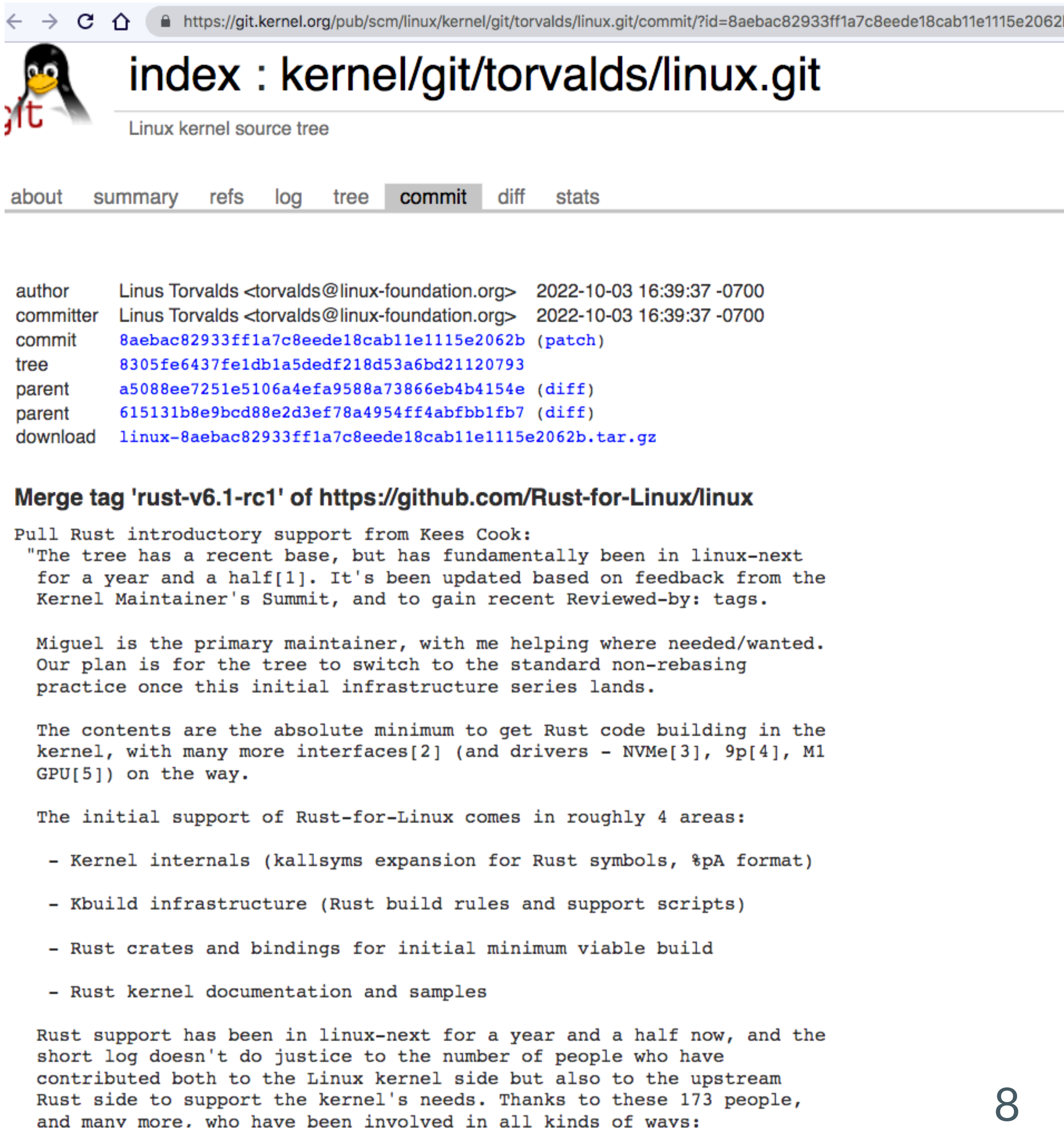
"Rust支持旨在支持在 Rust 中编写驱动程序和类似的“叶”周边模块，至少在可预见的未来。特别是，我们不打算重写内核核心或主要内核子系统, 例如 `kernel`、`mm`、`sched` "

Rust support RFC: Why Rust ?

- 安全子集(safe subset)中不存在未定义行为，包括内存安全和数据竞争；
- 更加严格的类型检测系统能够进一步减少逻辑错误；
- 明确区分 safe 和 unsafe 代码；
- 更加面向未来的语言：sum 类型、模式匹配、泛型、RAII、生命周期、共享及专属引用、模块与可见性等等；
- 可扩展的独立标准库；
- 集成的开箱可用工具：文档生成、代码格式化、linter 等，这些都基于编译器本身。

Rust for Linux 支持情况

- 支持以 `LLVM/Clang` 编译内核
- 在2022.10.16发布了 `Linux6.1-rc1` 开发版本，正式包含Rust代码
- 当前已经发布到Linux6.2-rc6



https://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git/commit/?id=8aebac82933ff1a7c8eede18cab11e1115e2062

index : kernel/git/torvalds/linux.git

Linux kernel source tree

author	Linus Torvalds <torvalds@linux-foundation.org>	2022-10-03 16:39:37 -0700
committer	Linus Torvalds <torvalds@linux-foundation.org>	2022-10-03 16:39:37 -0700
commit	8aebac82933ff1a7c8eede18cab11e1115e2062b (patch)	
tree	8305fe6437fe1db1a5dedf218d53a6bd21120793	
parent	a5088ee7251e5106a4efa9588a73866eb4b4154e (diff)	
parent	615131b8e9bcd88e2d3ef78a4954ff4abfbb1fb7 (diff)	
download	linux-8aebac82933ff1a7c8eede18cab11e1115e2062b.tar.gz	

Merge tag 'rust-v6.1-rc1' of https://github.com/Rust-for-Linux/linux

```
Pull Rust introductory support from Kees Cook:
 "The tree has a recent base, but has fundamentally been in linux-next
  for a year and a half[1]. It's been updated based on feedback from the
  Kernel Maintainer's Summit, and to gain recent Reviewed-by: tags.

  Miguel is the primary maintainer, with me helping where needed/wanted.
  Our plan is for the tree to switch to the standard non-rebasing
  practice once this initial infrastructure series lands.

  The contents are the absolute minimum to get Rust code building in the
  kernel, with many more interfaces[2] (and drivers - NVMe[3], 9p[4], M1
  GPU[5]) on the way.

  The initial support of Rust-for-Linux comes in roughly 4 areas:

   - Kernel internals (kallsyms expansion for Rust symbols, %pA format)

   - Kbuild infrastructure (Rust build rules and support scripts)

   - Rust crates and bindings for initial minimum viable build

   - Rust kernel documentation and samples

  Rust support has been in linux-next for a year and a half now, and the
  short log doesn't do justice to the number of people who have
  contributed both to the Linux kernel side but also to the upstream
  Rust side to support the kernel's needs. Thanks to these 173 people,
  and many more, who have been involved in all kinds of ways:
```

Rust for Linux 支持情况

Linux Merge rust-v6.1-rc1 代码情况：

- Rust-for-Linux的初始支持大致有4个方面：
  - Kernel内部（Rust符号的kallsyms扩展，%pA format)
  - Kbuild基础设施（Rust构建规则和支持脚本）
  - 用于初始最小可行构建的Rust crates和bindings, 包括对设备驱动的最小支持，如NVMe, 9pfs等
  - Rust kernel文档和samples样例

Rust for Linux 支持情况

Linux Merge rust-v6.2 代码情况：

- 字符串和格式化的新类型： 如 'CStr','Formatter';
- 打印，错误代码error codes；
- alloc crate：用于'RawVec'和'Vec'的新的构造函数；
- 新的过程宏，断言：'#[vtable]'和'concat_idents!', 以及'build_assert!'等；
- 新的类型'Opaque'和'Either'；
- Debugging 调试：新的宏'dbg';

Rust for Linux 设备驱动

Linux上的Rust驱动程序：

Western Digital 首席工程师 Andreas Hindborg 在2022 Linux Plumbers Summit上
展示了用户可以使用 Rust 编写一流的驱动程序，即适用于 Linux 的 SSD NVM-Express (NVMe) 驱动程序。

- Linux (PCI) NVMe driver in Rust
  https://lpc.events/event/16/contributions/1180/attachments/1017/1961/deck.pdf
  https://github.com/metaspace/linux/commits/nvme

Rust for Linux 设备驱动

Linux上的Rust驱动程序：

- Asahi Lina Linux组织开发的Apple's M1 GPU驱动 https://github.com/AsahiLinux/linux/commits/gpu/rust-wip
- 对async异步内核编程的支持，当前状态已经用于允许异步网络TCP连接支持；https://github.com/Rust-for-Linux/linux/commit/08358e5a955f662daf154a89f3ba43a3f18228a2

Rust for Linux 设备驱动

- Google Android 团队正试验移植Rust Binder IPC 驱动，其它 Android 厂商也表示有兴趣开发 Rust 驱动 https://github.com/Rust-for-Linux/linux/commit/d8dcaf80c9535056ea541d8dd876edad52c538a9
- 9p fs server 方便地从虚拟主机与Host机进行文件系统访问 https://kangrejos.com/Async Rust and 9p server.pdf https://github.com/wedsonaf/linux/commits/9p

Rust for Linux 项目

代码仓库：

```
https://github.com/Rust-for-Linux
```

架构支持	条件
`x86`	仅限x86_64
`riscv`	仅限riscv64
`arm`	仅限armv6和兼容版
`arm64`	无限制
`powerpc`	仅限ppc64le

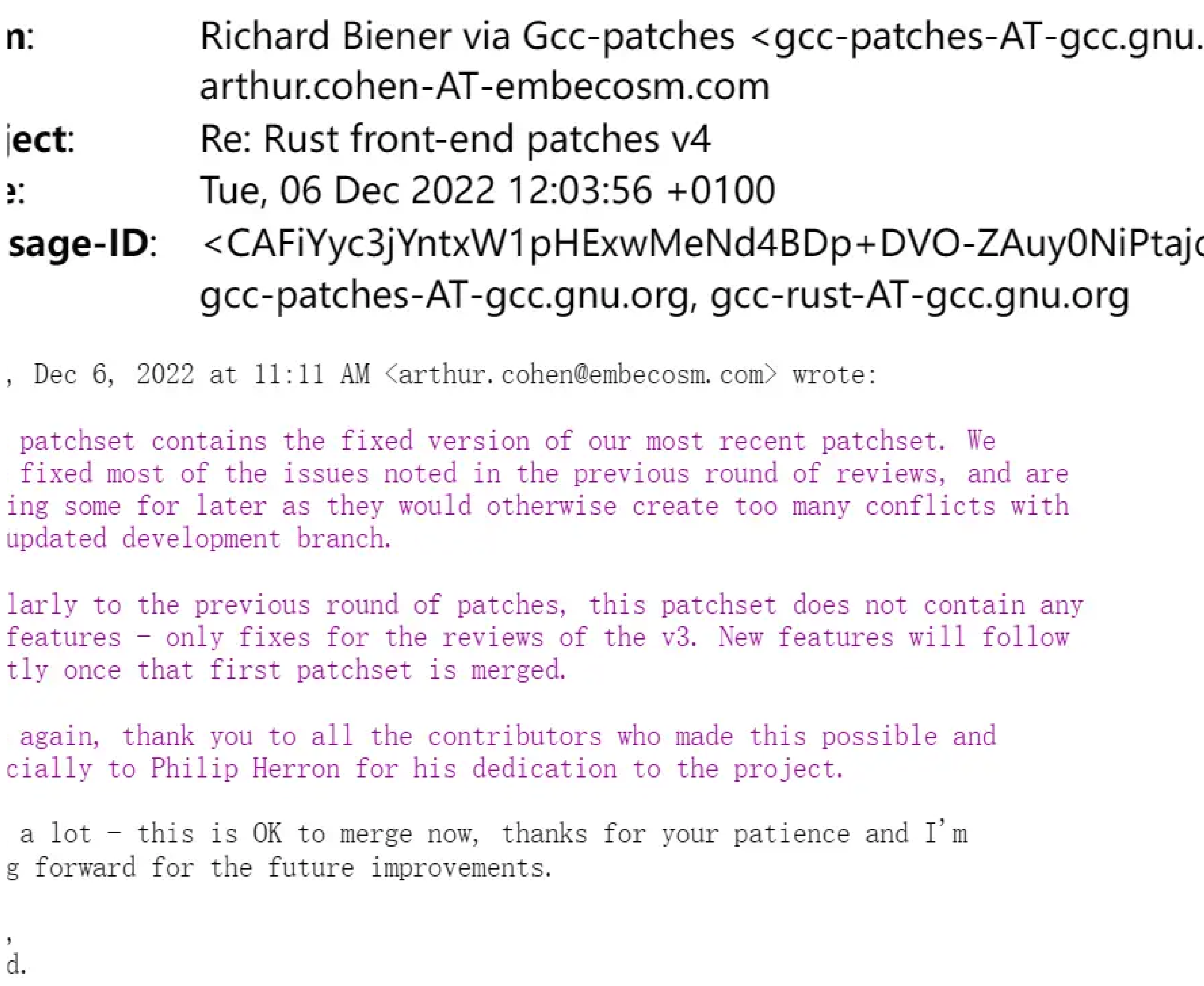
Rust for Linux 项目

- 支持以 `LLVM/Clang` 编译内核
- Rust编译器环境：
  - rustc
  - rust-src
  - rust-bindgen 用于解析Kernel中的C代码
- 开启Rust support内核配置选项 `CONFIG_RUST`
- 命令参数

```
make LLVM=1   #或在无完整LLVM工具链时： make CC=clang
```

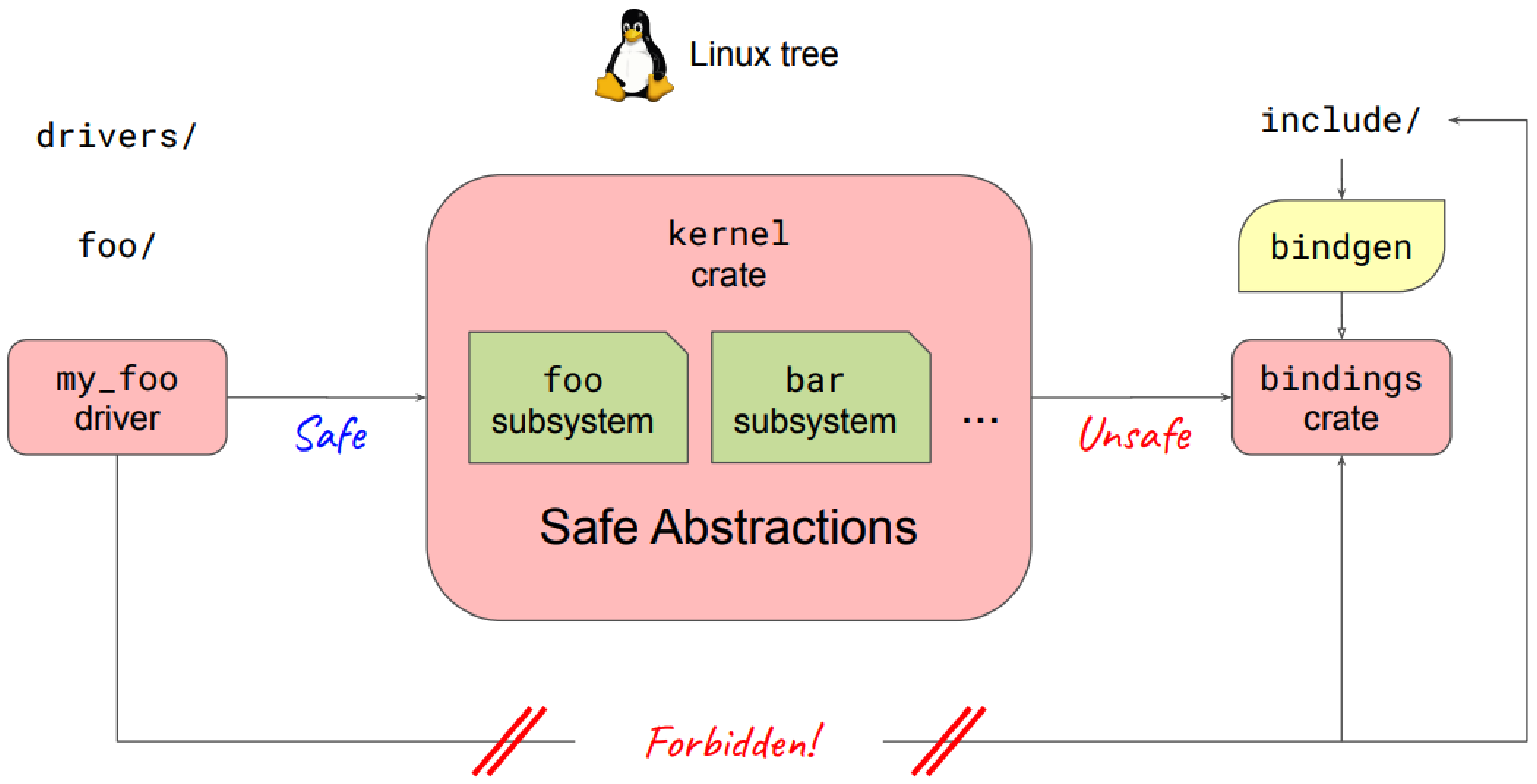
GCC前端支持Rust

- 在2022年12月GCC的Rust前端项目Gccrs已被批准合并到GCC主干
- GCC将能 编译Rust源代码 ;
- 按照计划，GCC 13 将于 2023 年 4 月左右发布稳定版，其对Rust支持达到beta级;



```
n:          Richard Biener via Gcc-patches <gcc-patches-AT-gcc.gnu.
            arthur.cohen-AT-embecosm.com
ject:       Re: Rust front-end patches v4
e:          Tue, 06 Dec 2022 12:03:56 +0100
sage-ID:    <CAFiYyc3jYntxW1pHExwMeNd4BDp+DVO-ZAuy0NiPtajc
            gcc-patches-AT-gcc.gnu.org, gcc-rust-AT-gcc.gnu.org

, Dec 6, 2022 at 11:11 AM <arthur.cohen@embecosm.com> wrote:

 patchset contains the fixed version of our most recent patchset. We
 fixed most of the issues noted in the previous round of reviews, and are
ing some for later as they would otherwise create too many conflicts with
updated development branch.

larly to the previous round of patches, this patchset does not contain any
features - only fixes for the reviews of the v3. New features will follow
tly once that first patchset is merged.

 again, thank you to all the contributors who made this possible and
cially to Philip Herron for his dedication to the project.

 a lot - this is OK to merge now, thanks for your patience and I'm
g forward for the future improvements.

,
d.
```

Rust drivers 与 kernel crate



Rust for Linux 设备驱动开发

- kernel crate
  - 为内核驱动开发者提供了公共接口支持的库，封装了kernel C API;
  - 提供了module宏和pr_info宏，Module trait定义，以及对Vec的支持等
  - 它须包含no_std属性，表示其不会链接标准库中的内容;
  - 所有kernel Rust代码依赖这个关键性crate;

Rust for Linux 设备驱动开发

为方便编写Linux驱动， Linux中提供了Rust相关的基础设施，并抽象出了接口、宏等。

- module!{} 宏

这个宏可以被认为是 Rust 内核模块的入口，因为在其中定义了一个内核模块所需的所有信息，包括：Author、License、Description 等。其中最重要的是 type 字段，在其中需要指定内核模块结构的名字，如 `RustE1000dev`.

Rust for Linux 设备驱动开发

- module_init() 与 module_exit()
  在使用 C 编写的内核模块中，这两个宏定义了模块的入口函数与退出函数。
  在 Rust 编写的内核模块中，对应的功能由 trait KernelModule 和 trait Drop 来实现。
  - `trait KernelModule` 中定义init()函数，会在模块驱动初始化时被调用；
    `impl kernel::Module for RustE1000dev`
  - `trait Drop` 是 Rust 的内置 trait，其中定义的 drop() 函数会在变量生命周期结束时被调用，这里是在驱动卸载时;

Rust for Linux 设备驱动开发

Intel E1000网卡驱动开发为案例

- 参考linux rust samples样例代码，如rust_miscdev
- 分析提供的Rust抽象接口,需要实现的接口trait
  - driver::Registration
  - pci::Adapter
  - net::DeviceOperations

Rust for Linux 设备驱动开发

Intel E1000网卡驱动开发为案例

注册一个 PCI 驱动

Kernel 提供:

- kernel::driver::Registration<T: kernel::driver::DriverOps>
- impl kernel::driver::DriverOps for kernel::pci::Adapter<T:kernel::pci::Driver>

PCI 驱动的实现:

impl kernel::pci::Driver for E1000Driver

Rust for Linux 设备驱动开发

编写驱动代码的框架

```
use kernel::{pci, define_pci_id_table};
struct E1000;
impl pci::Driver for E1000Driver {
  /// 用于保存PCI驱动数据，如E1000网卡结构体
  type Data = Box<DrvData>;
  /// PCI 设备初始化
  fn probe(_dev: &mut pci::Device, _id_info: Option<&Self::IdInfo>) -> Result {
      Ok(())
  }
  define_pci_id_table! {u32, [
      (pci::DeviceId::new(0x010800, 0xffffff), None),
      (pci::DeviceId::with_class(0x010802, 0xfffff), Some(0x10)),
  ]}
}
```

Rust for Linux 设备驱动开发

Intel E1000网卡驱动开发为案例

接口trait net::DeviceOperations

```
/// 对应于kernel的结构体struct net_device_ops
pub trait DeviceOperations {
    /// 数据的指针类型用于保存驱动定义的数据
    type Data: PointerWrapper + Send + Sync = ();

    /// 对应struct net_device_ops的`ndo_open`
    fn open(dev: &Device, data: <Self::Data as PointerWrapper>::Borrowed<'_>) -> Result;

    /// 对应struct net_device_ops的`ndo_start_xmit`
    fn start_xmit(skb: &SkBuff, dev: &Device, data: <Self::Data as PointerWrapper>::Borrowed<'_>) -> NetdevTx;
    ...
}
```

Rust for Linux 设备驱动开发

实现网络设备操作接口

```
impl net::DeviceOperations for E1000 {
    // 注意该网卡数据结构类型，保存了网络设备收发环形缓存、中断irq等重要数据
    type Data = Box<DevData>;
    fn open(netdev: &net::Device, devdata: &DevData) -> Result { ... }
    ...
}
struct DevData {
    dev: Arc<device::Device>,
    tx_ring: Pin<Box<SpinLock<Box<Ring<usize>>>>>, // 网络包的发送ring缓存
    rx_ring: Pin<Box<SpinLock<Box<Ring<usize>>>>>, // 网络包的接收ring缓存
    irq: AtomicPtr<irq::Registration<E1000>>,      // 网卡设备中断
}
```

Rust for Linux 设备驱动开发

- 驱动代码的编译与链接

Rust for Linux内核模块文件会首先被编译成 .o 目标文件，之后由内核链接器将这些.o文件和自动生成的模块目标文件.mod.o一起链接成为.ko文件。
这个.ko文件符合动态库 ELF 文件格式，能够被内核识别并加载。

```
make modules LLVM=1 -C $(KDIR) ARCH=$(ARCH) M=$(PWD)
```

```
insmod rust_e1000_for_linux.ko
```

非OS依赖的独立Rust设备驱动

驱动程序大多数时候只与设备有关的内存及寄存器打交道，理论上不依赖上层操作系统；

为了增加开发的驱动程序的多平台可移植性；

设计编写了非OS依赖的独立Rust e1000设备驱动：

https://github.com/elliott10/e1000-driver

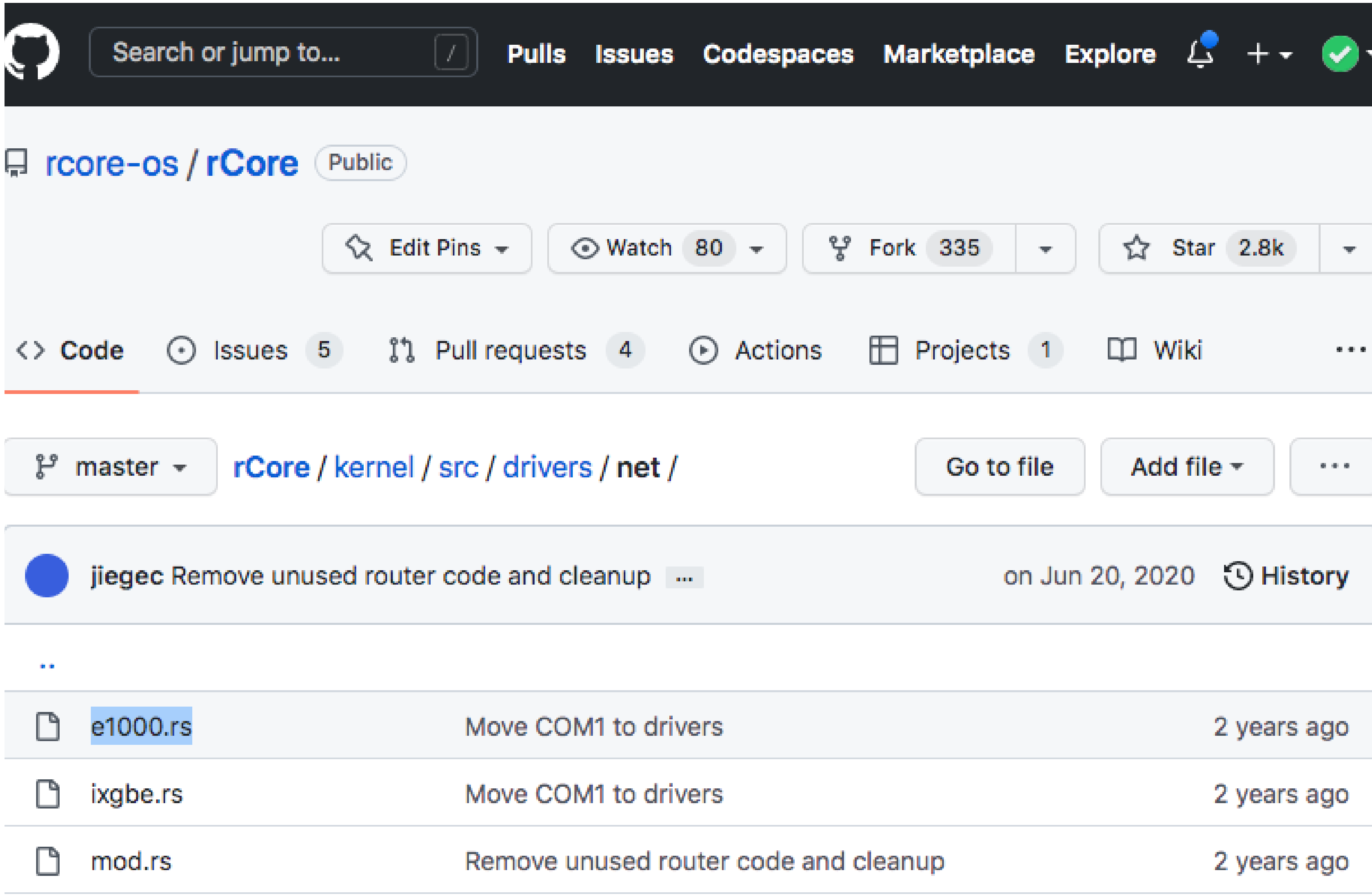
独立Rust设备驱动

定义驱动需要的接口trait

```
/// 驱动需要用到的内核功能
pub trait KernelFunc {
    /// 页大小
    const PAGE_SIZE: usize = 4096;
    /// 物理地址与虚拟地址的转换
    fn phys_to_virt(paddr: usize) -> usize;
    fn virt_to_phys(vaddr: usize) -> usize;
    /// 请求分配中断irq
   fn request_irq(irq: u32, data: Box<u32>) -> Result<u32>;
    /// 申请DMA一致性物理内存接口
    fn dma_alloc_coherent(pages: usize) -> usize;
    /// DMA内存释放
    fn dma_free_coherent(paddr: usize, pages: usize);
}
```

独立Rust驱动

操作系统实现该接口trait
可以在不同的OS中运行，例如包括 `rCore/zCore`



rCore / kernel / src / drivers / net /

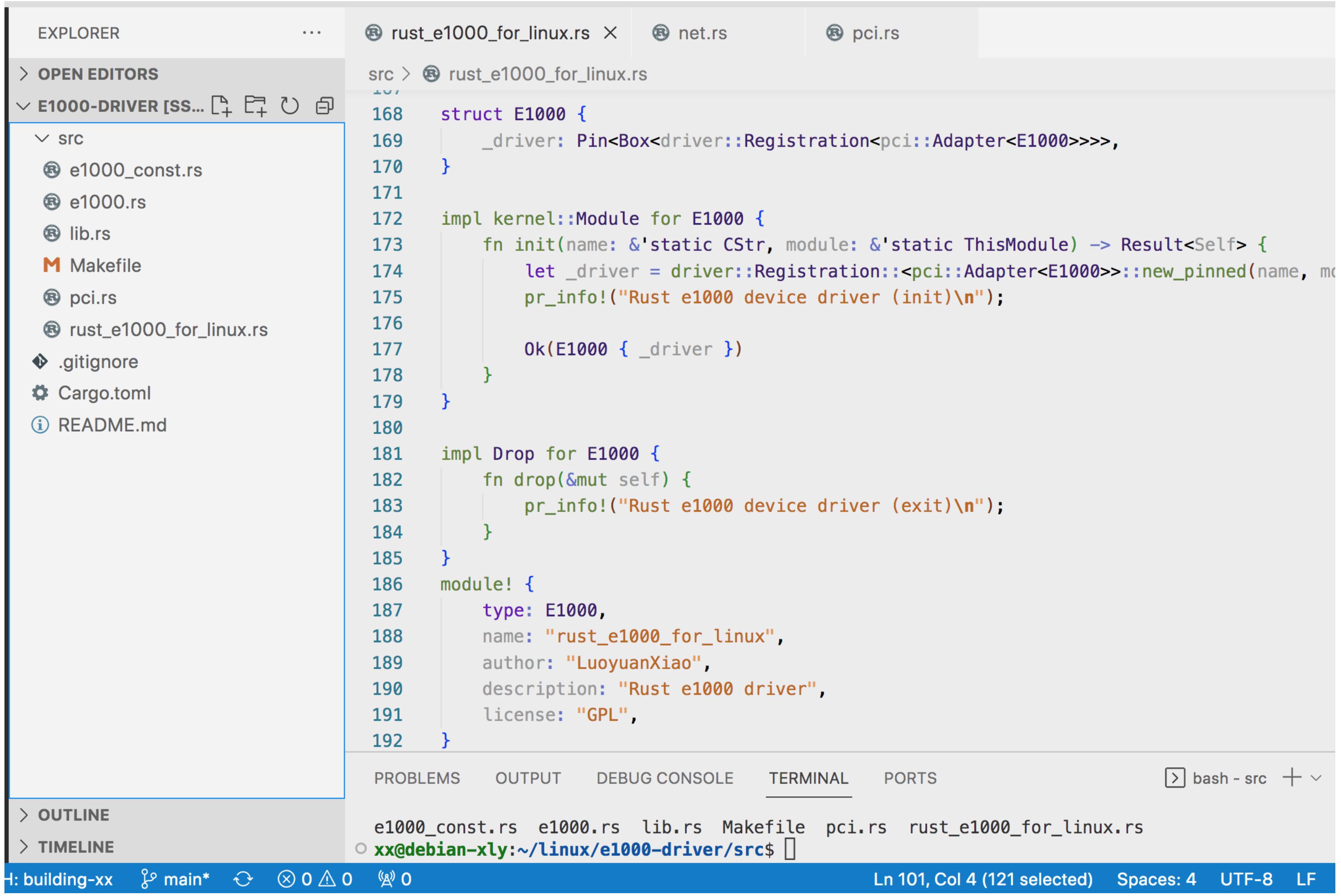
jiegec Remove unused router code and cleanup ... on Jun 20, 2020

..

e1000.rs	Move COM1 to drivers	2 years ago
ixgbe.rs	Move COM1 to drivers	2 years ago
mod.rs	Remove unused router code and cleanup	2 years ago

独立Rust驱动 for Linux

- 在Linux上，独立Rust驱动的接口及函数，填充到Rust驱动框架中



EXPLORER
OPEN EDITORS
E1000-DRIVER [SS...
src
e1000_const.rs
e1000.rs
lib.rs
Makefile
pci.rs
rust_e1000_for_linux.rs
.gitignore
Cargo.toml
README.md
OUTLINE
TIMELINE

rust_e1000_for_linux.rs × | net.rs | pci.rs

src > rust_e1000_for_linux.rs

```
struct E1000 {
    _driver: Pin<Box<driver::Registration<pci::Adapter<E1000>>>>,
}

impl kernel::Module for E1000 {
    fn init(name: &'static CStr, module: &'static ThisModule) -> Result<Self> {
        let _driver = driver::Registration::<pci::Adapter<E1000>>::new_pinned(name, mo
        pr_info!("Rust e1000 device driver (init)\n");

        Ok(E1000 { _driver })
    }
}

impl Drop for E1000 {
    fn drop(&mut self) {
        pr_info!("Rust e1000 device driver (exit)\n");
    }
}
module! {
    type: E1000,
    name: "rust_e1000_for_linux",
    author: "LuoyuanXiao",
    description: "Rust e1000 driver",
    license: "GPL",
}
```

PROBLEMS | OUTPUT | DEBUG CONSOLE | TERMINAL | PORTS | bash - src

```
e1000_const.rs  e1000.rs  lib.rs  Makefile  pci.rs  rust_e1000_for_linux.rs
xx@debian-xly:~/linux/e1000-driver/src$
```

H: building-xx main* ⊗ 0 ⚠ 0 0 Ln 101, Col 4 (121 selected) Spaces: 4 UTF-8 LF

驱动的测试展示

- 独立Rust设备驱动在裸机运行时上收发包测试；
- Rust for Linux驱动程序框架的编译与加载;

谢谢