Assignment 4

主要修改

1. 修改 `pci::Driver` 接口，传递在停止驱动时所需的 `pci_dev`

> project > cicv > cicv-r4l-3-Yang2096 > linux > rust > kernel > pci.rs

```
pub trait Driver {
    /// Data stored on device by driver.
    ///
    /// Corresponds to the data set or retrieved via the kernel's
    /// `pci_{set,get}_drvdata()` functions.
    ///
    /// Require that `Data` implements `PointerWrapper`. We guarantee to
    /// never move the underlying wrapped data structure.
    type Data: PointerWrapper + Send + Sync + driver::DeviceRemoval = ();

    /// The type holding information about each device id supported by the driver.
    type IdInfo: 'static = ();

    /// The table of device ids supported by the driver.
    const ID_TABLE: driver::IdTable<'static, DeviceId, Self::IdInfo>;

    /// PCI driver probe.
    ///
    /// Called when a new platform device is added or discovered.
    /// Implementers should attempt to initialize the device here.
    fn probe(dev: &mut Device, id: Option<&Self::IdInfo>) -> Result<Self::Data>;

    /// PCI driver remove.
    ///
    /// Called when a platform device is removed.
    /// Implementers should prepare the device for complete removal here.
    fn remove(_data: &Self::Data);
    fn remove(pdev: *mut bindings::pci_dev, _data: &Self::Data);
}
```

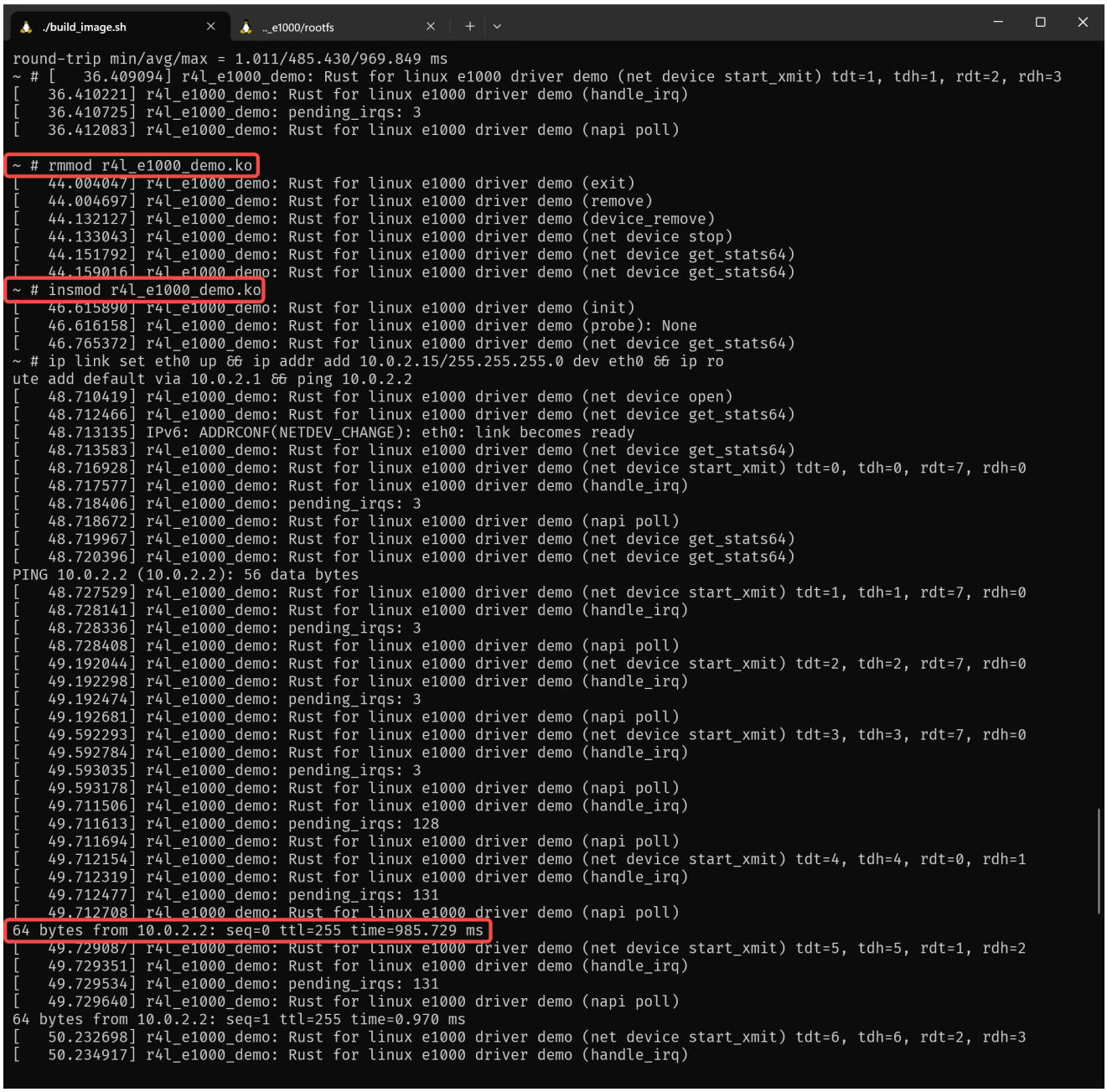
2. 释放 MMIO 资源，停止设备

```
> project > cicv > cicv-r4l-3-Yang2096 > src_e1000 > r4l_e1000_demo.rs
impl pci::Driver for E1000Drv {
    fn probe(dev: &mut pci::Device, id: core::option::Option<&Self::IdInfo>)
        netdev_reg.register(Box::try_new(
            NetDevicePrvData {
                e1000_hw_ops: Arc::try_new(e1000_hw_ops)?,
                napi: napi.into(),
                tx_ring,
                rx_ring,
                irq,
                _irq_handler: AtomicPtr::new(core::ptr::null_mut()),
            }
        )?)?;



        Ok(Box::try_new(
            E1000DrvPrvData{
                // Must hold this registration, or the device will be remove
                _netdev_reg: netdev_reg,
                bars,
            }
        )?)
    }

    fn remove(data: &Self::Data) {
    fn remove(pdev: *mut bindings::pci_dev, data: &Self::Data) {
        pr_info!("Rust for linux e1000 driver demo (remove)\n");
        unsafe {
            pci_release_selected_regions(pdev, data.bars);
            pci_disable_device(pdev);
        };
    }
}
```

3. 释放 IRQ （通过 `InternalRegistration` 的 `Drop` 实现）

```
> project > cicv > cicv-r4l-3-Yang2096 > src_e1000 > r4l_e1000_demo.rs
impl net::DeviceOperations for NetDevice {

    fn stop(_dev: &net::Device, _data: &NetDevicePrvData) -> Result {
        pr_info!("Rust for linux e1000 driver demo (net device stop)\n");
        _dev.netif_carrier_off();
        _dev.netif_stop_queue();
        // release irq
        let ptr = _data._irq_handler.load(core::sync::atomic::Ordering::Relaxed);
        if !ptr.is_null() {
            let _ = unsafe{ Box::from_raw(ptr) };
        }
        _data.e1000_hw_ops.e1000_reset_hw();
        Ok(())
    }
```

实验记录


./build_image.sh
../e1000/rootfs
round-trip min/avg/max = 1.011/485.430/969.849 ms
~ # [36.409094] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=1, tdh=1, rdt=2, rdh=3
[36.410221] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[36.410725] r4l_e1000_demo: pending_irqs: 3
[36.412083] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
~ # rmmod r4l_e1000_demo.ko
[44.004047] r4l_e1000_demo: Rust for linux e1000 driver demo (exit)
[44.004697] r4l_e1000_demo: Rust for linux e1000 driver demo (remove)
[44.132127] r4l_e1000_demo: Rust for linux e1000 driver demo (device_remove)
[44.133043] r4l_e1000_demo: Rust for linux e1000 driver demo (net device stop)
[44.151792] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
[44.159016] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
~ # insmod r4l_e1000_demo.ko
[46.615890] r4l_e1000_demo: Rust for linux e1000 driver demo (init)
[46.616158] r4l_e1000_demo: Rust for linux e1000 driver demo (probe): None
[46.765372] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
~ # ip link set eth0 up && ip addr add 10.0.2.15/255.255.255.0 dev eth0 && ip ro
ute add default via 10.0.2.1 && ping 10.0.2.2
[48.710419] r4l_e1000_demo: Rust for linux e1000 driver demo (net device open)
[48.712466] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
[48.713135] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[48.713583] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
[48.716928] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=0, tdh=0, rdt=7, rdh=0
[48.717577] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[48.718406] r4l_e1000_demo: pending_irqs: 3
[48.718672] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[48.719967] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
[48.720396] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
PING 10.0.2.2 (10.0.2.2): 56 data bytes
[48.727529] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=1, tdh=1, rdt=7, rdh=0
[48.728141] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[48.728336] r4l_e1000_demo: pending_irqs: 3
[48.728408] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[49.192044] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=2, tdh=2, rdt=7, rdh=0
[49.192298] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[49.192474] r4l_e1000_demo: pending_irqs: 3
[49.192681] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[49.592293] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=3, tdh=3, rdt=7, rdh=0
[49.592784] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[49.593035] r4l_e1000_demo: pending_irqs: 3
[49.593178] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[49.711506] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[49.711613] r4l_e1000_demo: pending_irqs: 128
[49.711694] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[49.712154] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=4, tdh=4, rdt=0, rdh=1
[49.712319] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[49.712477] r4l_e1000_demo: pending_irqs: 131
[49.712708] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=0 ttl=255 time=985.729 ms
[49.729087] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=5, tdh=5, rdt=1, rdh=2
[49.729351] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[49.729534] r4l_e1000_demo: pending_irqs: 131
[49.729640] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=1 ttl=255 time=0.970 ms
[50.232698] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=6, tdh=6, rdt=2, rdh=3
[50.234917] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)