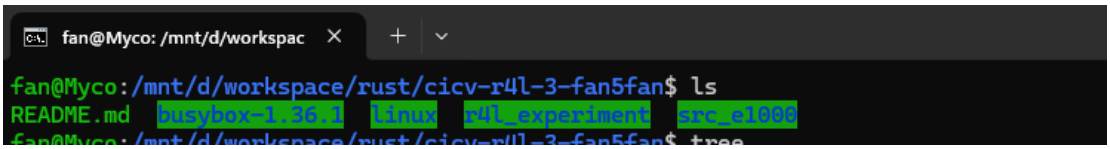
第三期训练营第二阶段 rust for linux 作业

一、环境配置

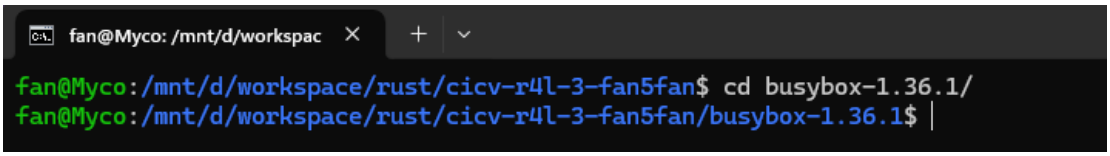
1、下载作业源码

git clone https://github.com/cicvedu/cicv-r4l-3-fan5fan.git

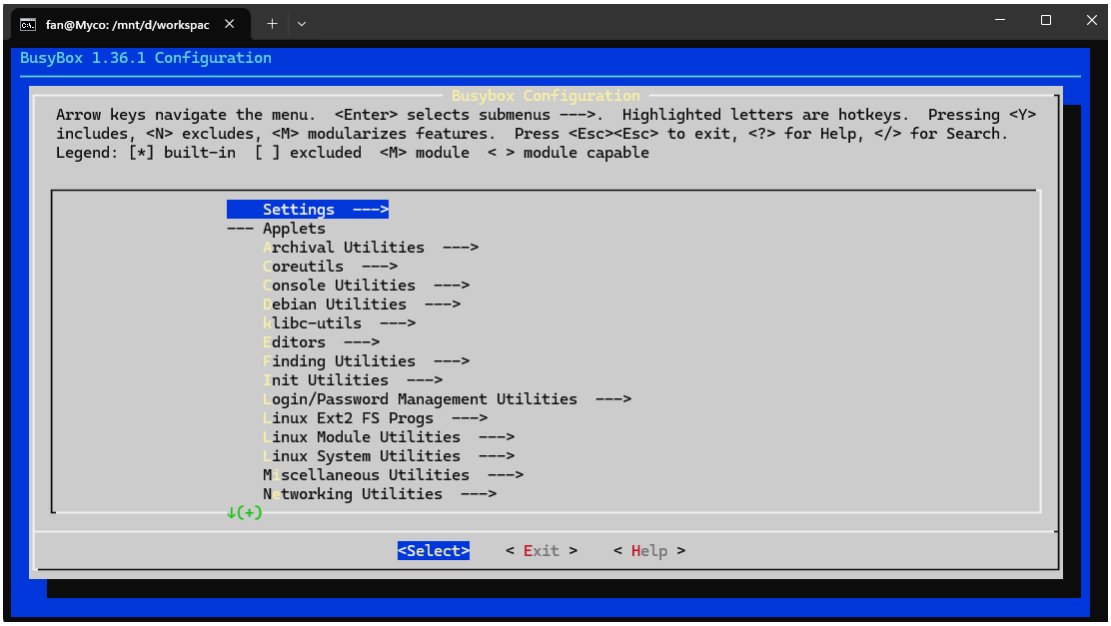


2、编译 BusyBox

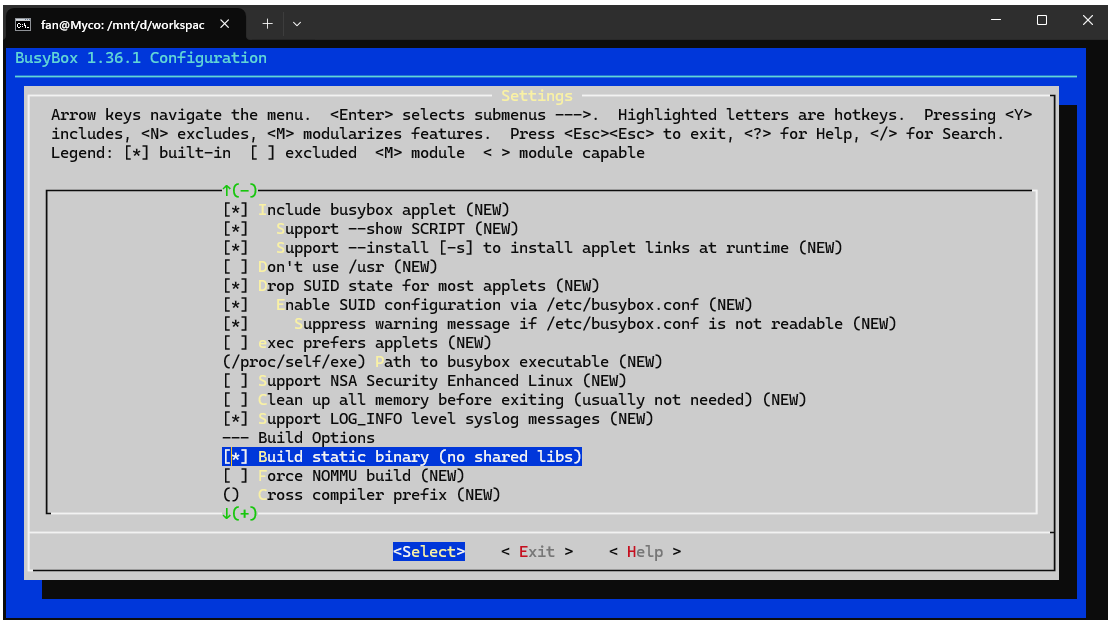
2.1、cd cicv-r4l-3-fan5fan/busybox-1.36.1/



2.2、make menuconfig

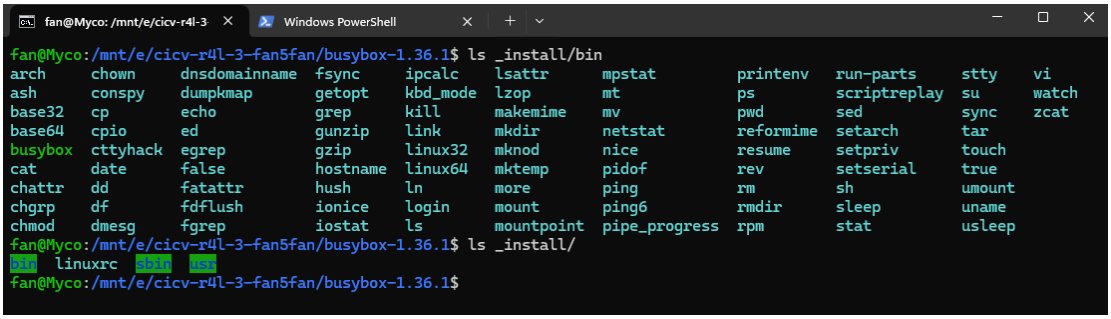


2.3、启动静态链接



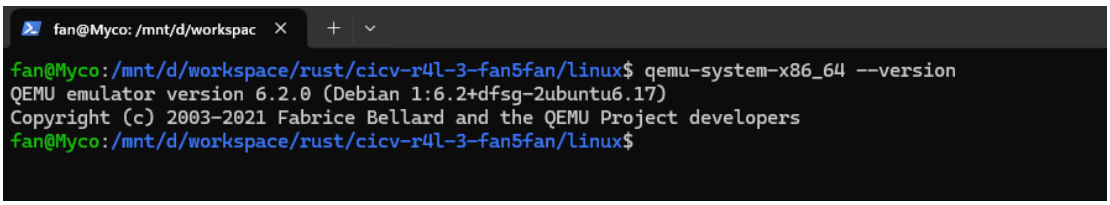
2.4、编译、安装

```
make install -j$(nproc)
```



3、安装 Qemu

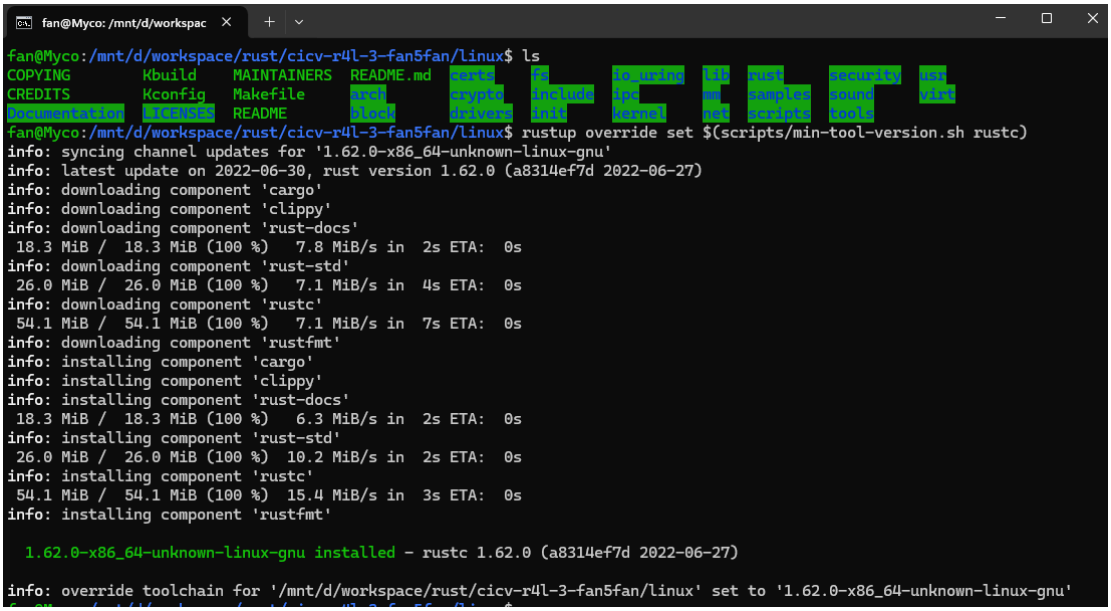
sudo apt install qemu-system-x86

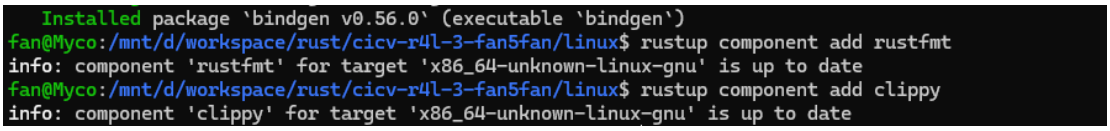


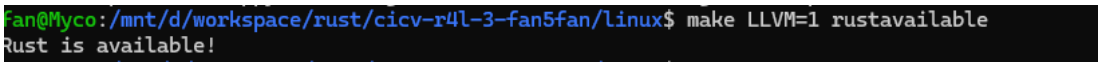
4、安装 rust

```
curl --proto '=https' --tlsv1.2 -sSf https://sh.rustup.rs | sh
```

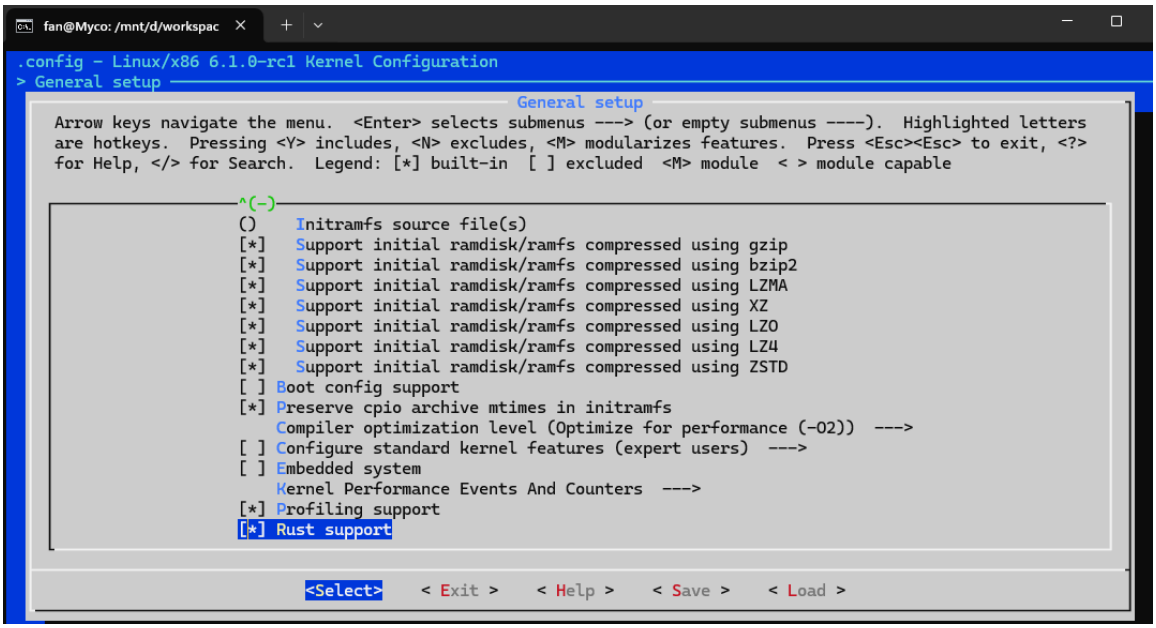
5、配置 linux 文件夹

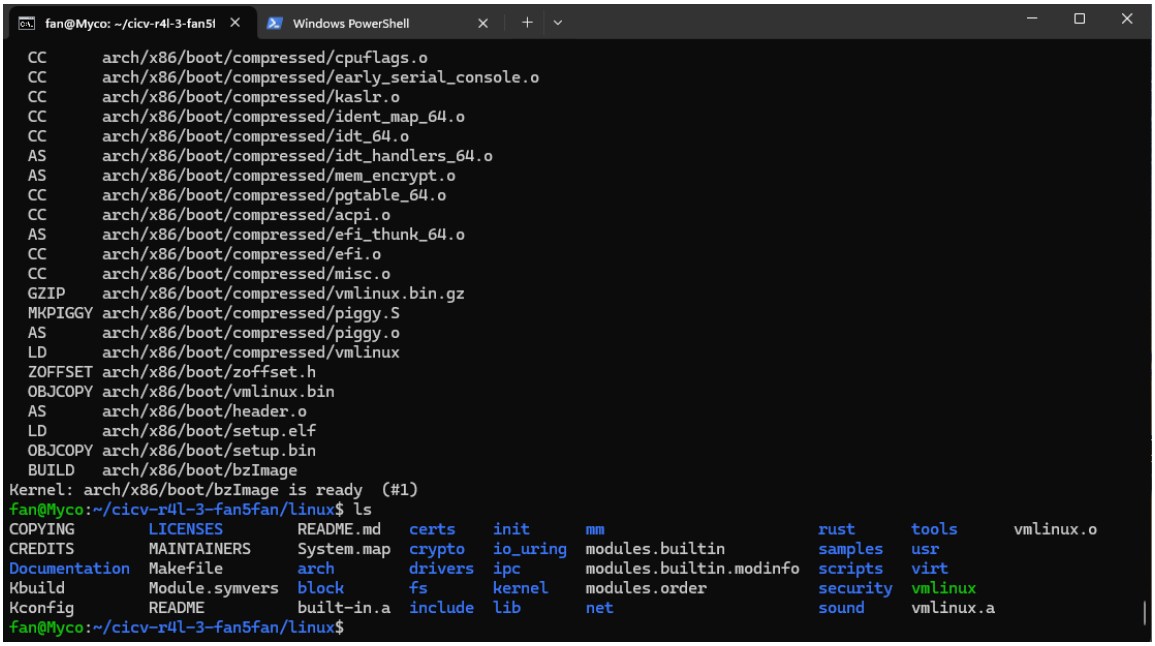






二、作业一 编译 Linux 内核





注：用 WSL 编译内核时，不能将文件放在 Windows 的文件系统内，必须放在 WSL 里面，不然会编译不过

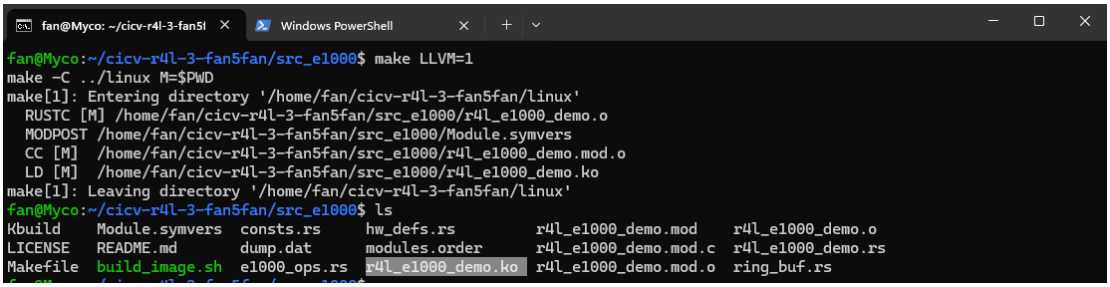
三、作业二 对 Linux 内核进行一些配置

1、操作步骤

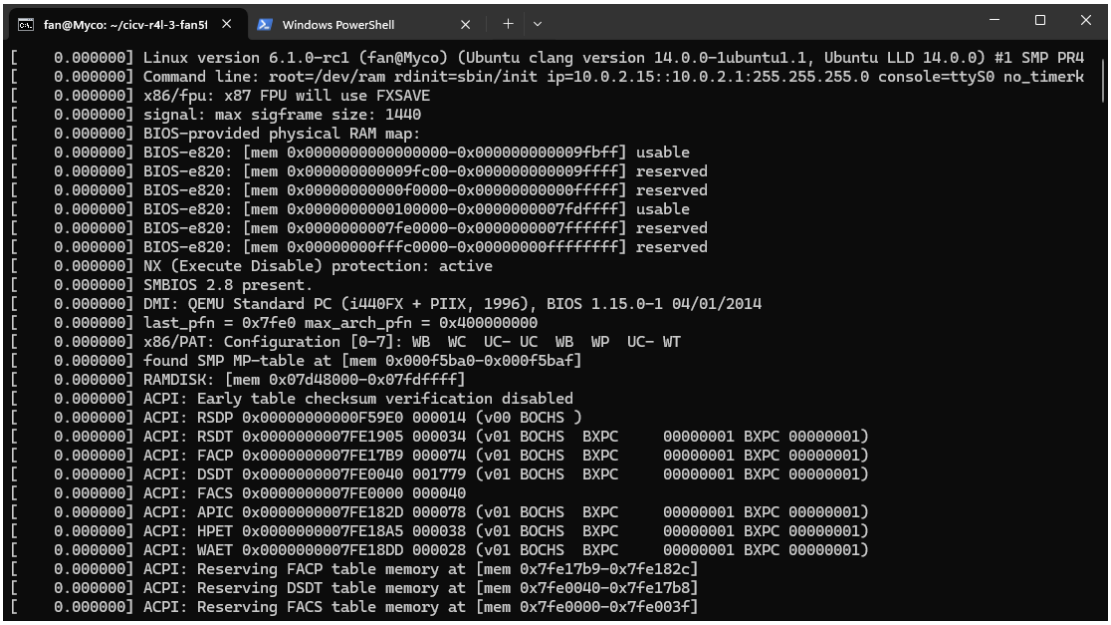
1.0、进入源码文件夹

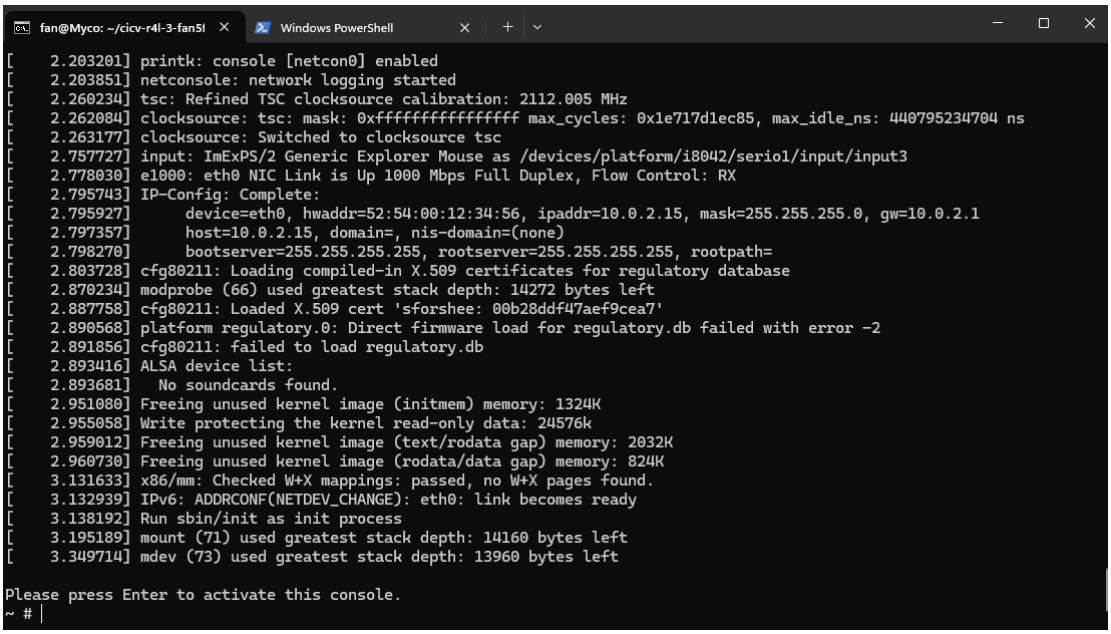
cd cicv-r4l-3-fan5fan/src_e1000

1.1、make

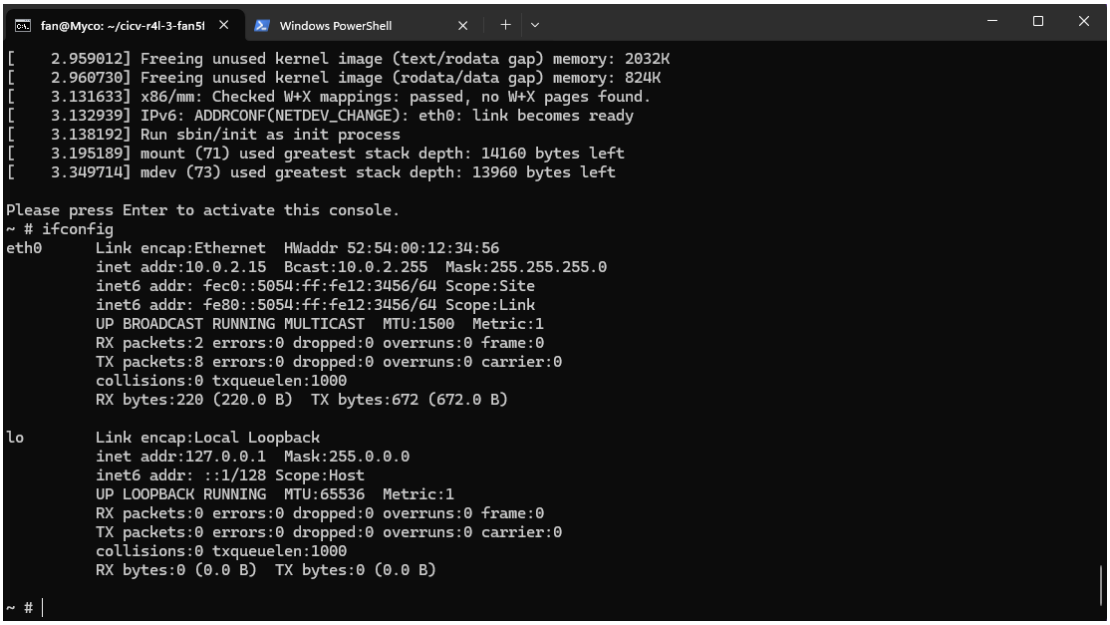


1.2、执行 ./build_image.sh

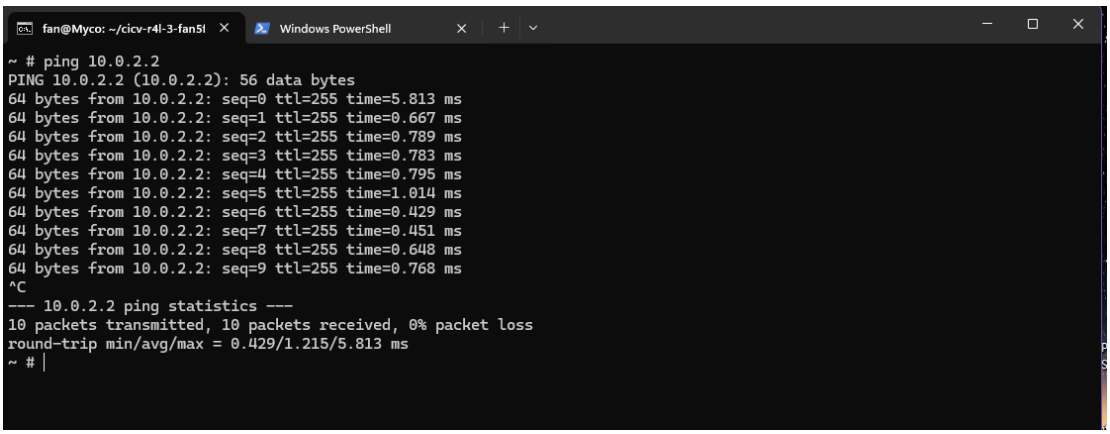




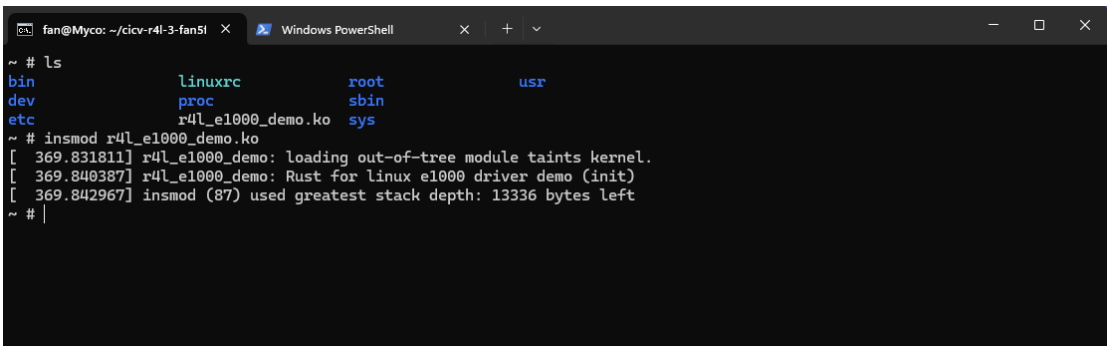
1.3、执行 ifconfig



1.4、执行 ping

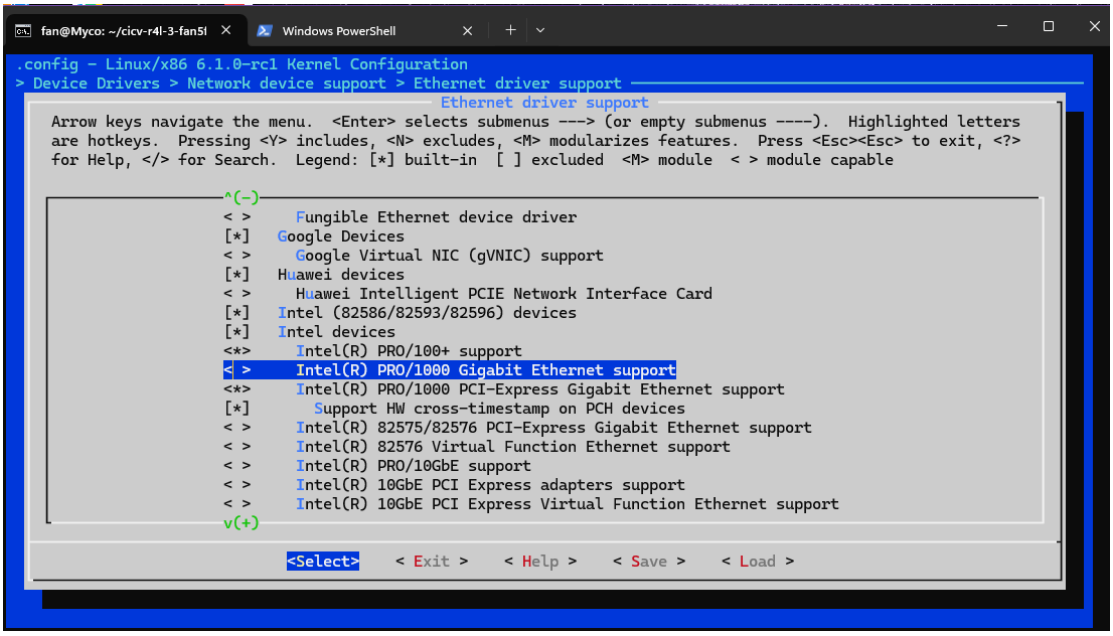


1.5、安装 r4l_e1000_demo.ko

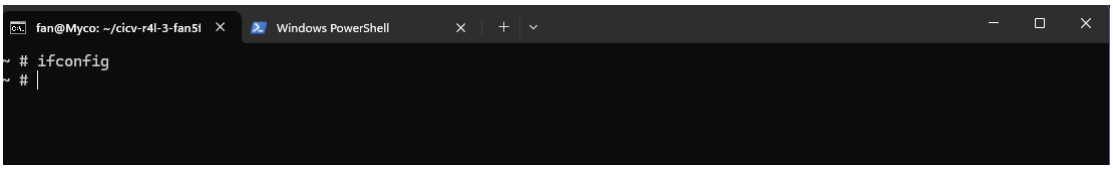


1.6、退出 Qemu

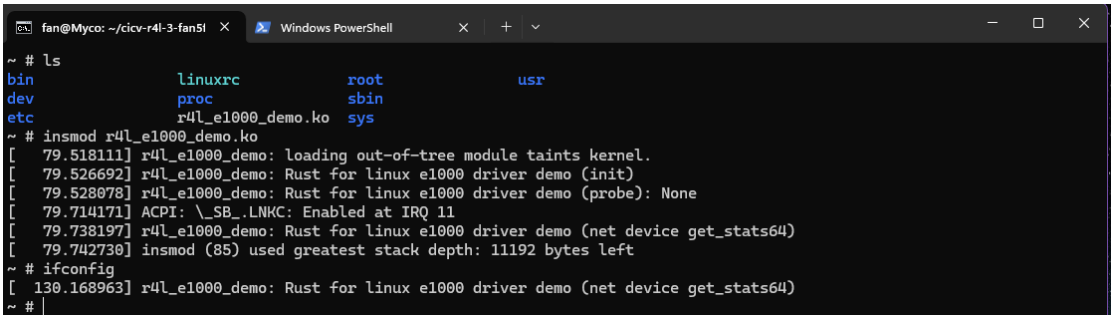
1.7、进入 `linux` 文件夹，重新 `make LLVM=1 menuconfig`，重新编译 `linux` 内核



1.8、重新进入 Qemu，并执行 ifconfig

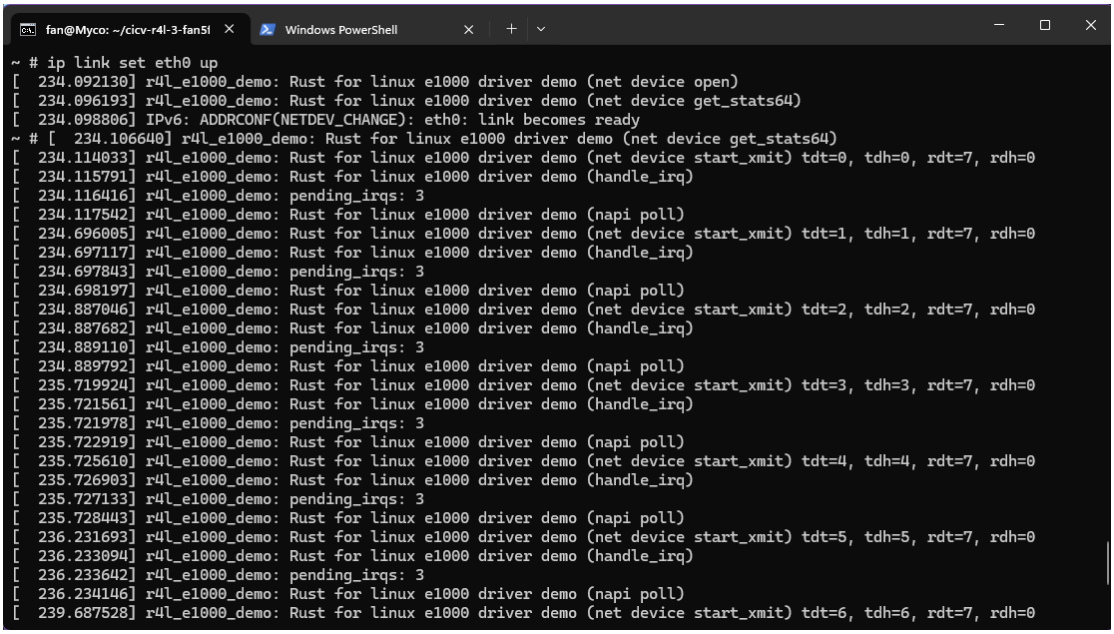


1.9、安装 r4l_e1000_demo.ko

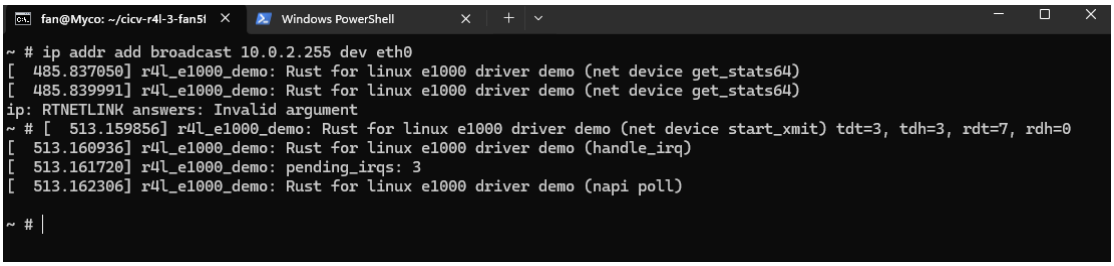


1.10、设置

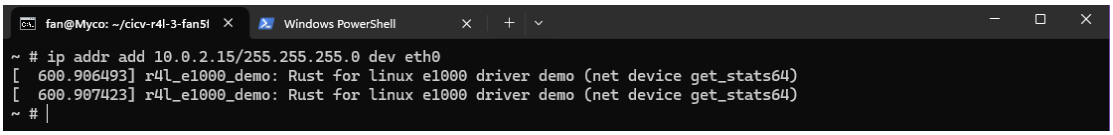
```
ip link set eth0 up
```



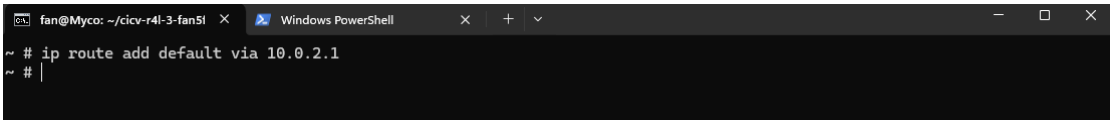
ip addr add broadcast 10.0.2.255 dev eth0



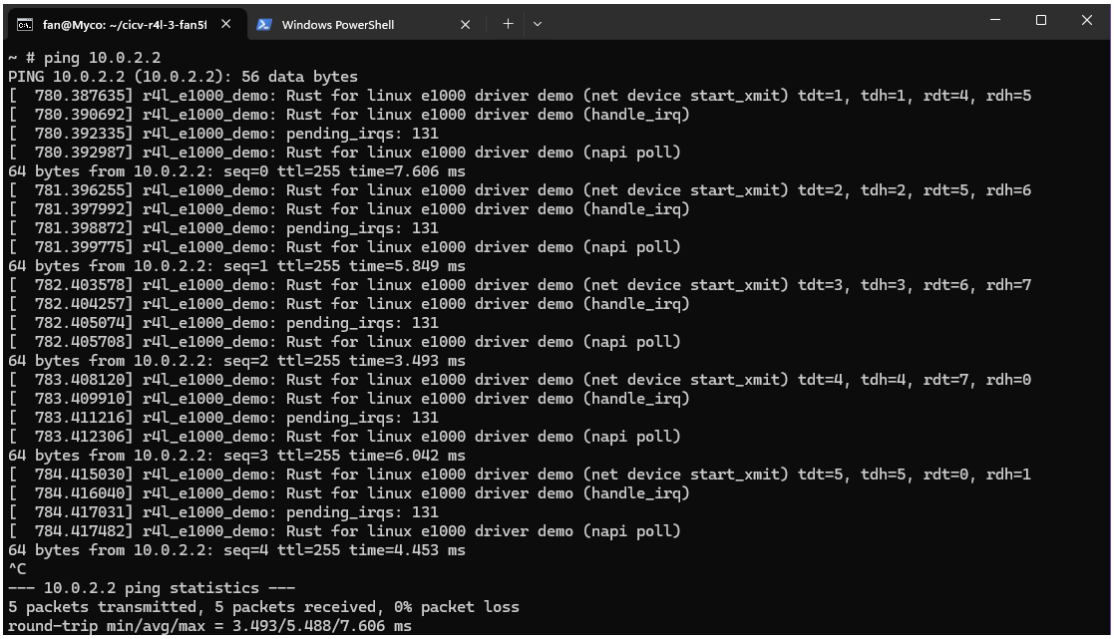
ip addr add 10.0.2.15/255.255.255.0 dev eth0



ip route add default via 10.0.2.1



ping 10.0.2.2



2、问题

2.1、编译成内核模块，是在哪个文件中以哪条语句定义的？

Kbuild 文件中的 obj-m := r4l_e1000_demo.o 这里定义的

2.2、该模块位于独立的文件夹内，却能编译成 Linux 内核模块，这叫做 out-of-tree module，请分析它是如何与内核代码产生联系的？

加载一个 out-of-tree 模块时，它通常有以下几种方式与内核代码发生关联：

1. 直接内核符号引用：模块可能会直接引用内核导出的符号，如函数和变量。
2. 内核服务：模块可能会使用内核提供的服务，如消息传递机制。
3. 设备文件节点：模块可能会创建设备文件节点，通过 /dev 目录与用户空间通信。
4. 模块参数：模块可以定义参数，在加载模块时传入这些参数。

四、作业三

1、在 samples/rust 里添加 rust_helloworld.rs

```
fan@Myco:~/cicv-r4l-3-fan5fan/linux/samples/rust$ cat rust_helloworld.rs
// SPDX-License-Identifier: GPL-2.0
//! Rust minimal sample.

use kernel::prelude::*;

module! {
  type: RustHelloWorld,
  name: "rust_helloworld",
  author: "whocare",
  description: "hello world module in rust",
  license: "GPL",
}

struct RustHelloWorld {}

impl kernel::Module for RustHelloWorld {
  fn init(_name: &'static CStr, _module: &'static ThisModule) -> Result<Self> {
      pr_info!("Hello World from Rust module");
      Ok(RustHelloWorld {})
  }
}

fan@Myco:~/cicv-r4l-3-fan5fan/linux/samples/rust$
```

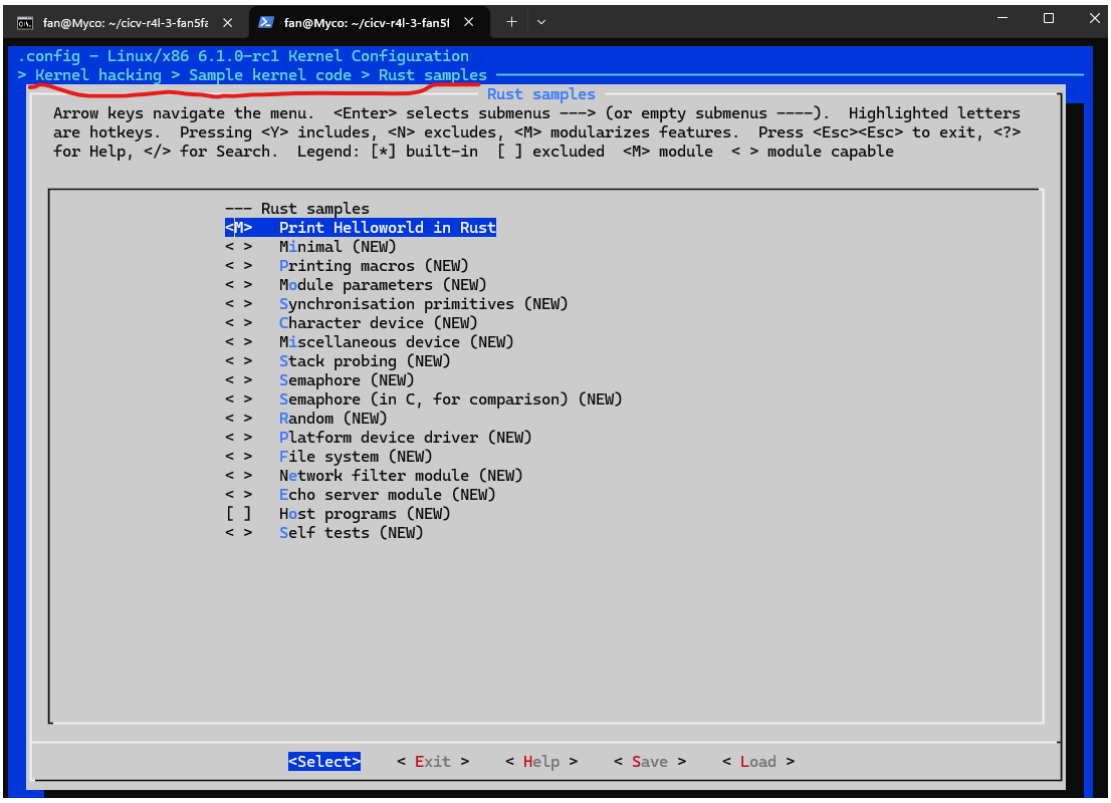
2、修改 Kconfig

```
# SPDX-License-Identifier: GPL-2.0

menuconfig SAMPLES_RUST
        bool "Rust samples"
        depends on RUST
        help
          You can build sample Rust kernel code here.

          If unsure, say N.

if SAMPLES_RUST

config PRINT_HELLOWORLD
        tristate "Print Helloworld in Rust"
        help
          This option builds the Rust helloworld module sample.

          To compile this as a module, choose M here:
          the module will be called rust_helloworld.

          If unsure, say N.

config SAMPLE_RUST_MINIMAL
        tristate "Minimal"
```

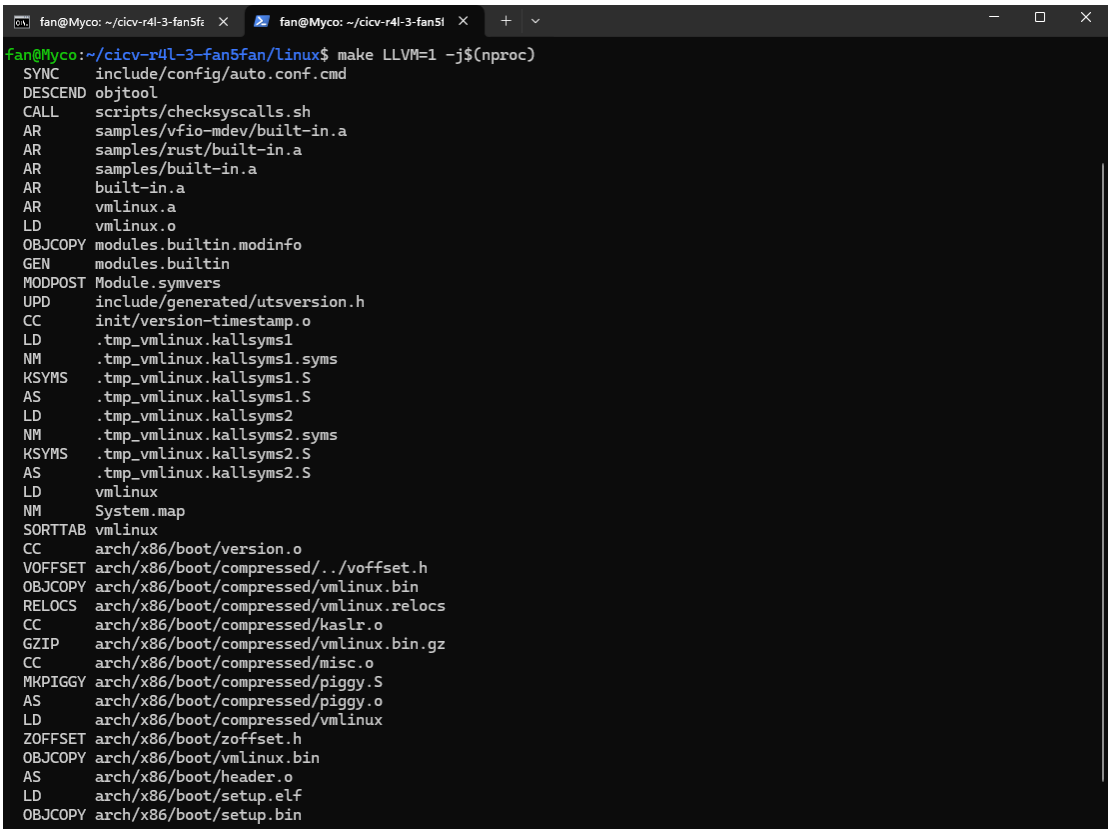
3、修改 Makefile

```
fan@Myco:~/cicv-r4l-3-fan5fan/linux$ cat samples/rust/Makefile
# SPDX-License-Identifier: GPL-2.0

obj-$(CONFIG_SAMPLE_RUST_MINIMAL)               += rust_minimal.o
obj-$(CONFIG_SAMPLE_RUST_PRINT)                 += rust_print.o
obj-$(CONFIG_SAMPLE_RUST_MODULE_PARAMETERS)     += rust_module_parameters.o
obj-$(CONFIG_SAMPLE_RUST_SYNC)                  += rust_sync.o
obj-$(CONFIG_SAMPLE_RUST_CHRDEV)                += rust_chrdev.o
obj-$(CONFIG_SAMPLE_RUST_MISCDEV)               += rust_miscdev.o
obj-$(CONFIG_SAMPLE_RUST_STACK_PROBING)         += rust_stack_probing.o
obj-$(CONFIG_SAMPLE_RUST_SEMAPHORE)             += rust_semaphore.o
obj-$(CONFIG_SAMPLE_RUST_SEMAPHORE_C)           += rust_semaphore_c.o
obj-$(CONFIG_SAMPLE_RUST_RANDOM)                += rust_random.o
obj-$(CONFIG_SAMPLE_RUST_PLATFORM)              += rust_platform.o
obj-$(CONFIG_SAMPLE_RUST_NETFILTER)             += rust_netfilter.o
obj-$(CONFIG_SAMPLE_RUST_ECHO_SERVER)           += rust_echo_server.o
obj-$(CONFIG_SAMPLE_RUST_FS)                    += rust_fs.o
obj-$(CONFIG_SAMPLE_RUST_SELFTESTS)             += rust_selftests.o

obj-$(CONFIG_PRINT_HELLOWORLD)                  += rust_helloworld.o

subdir-$(CONFIG_SAMPLE_RUST_HOSTPROGS)          += hostprogs
fan@Myco:~/cicv-r4l-3-fan5fan/linux$
```

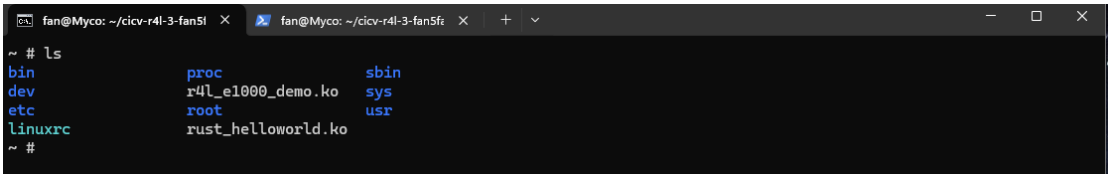
4、`make LLVM=1 menuconfig`



5、`make LLVM=1 -j$(nproc)`

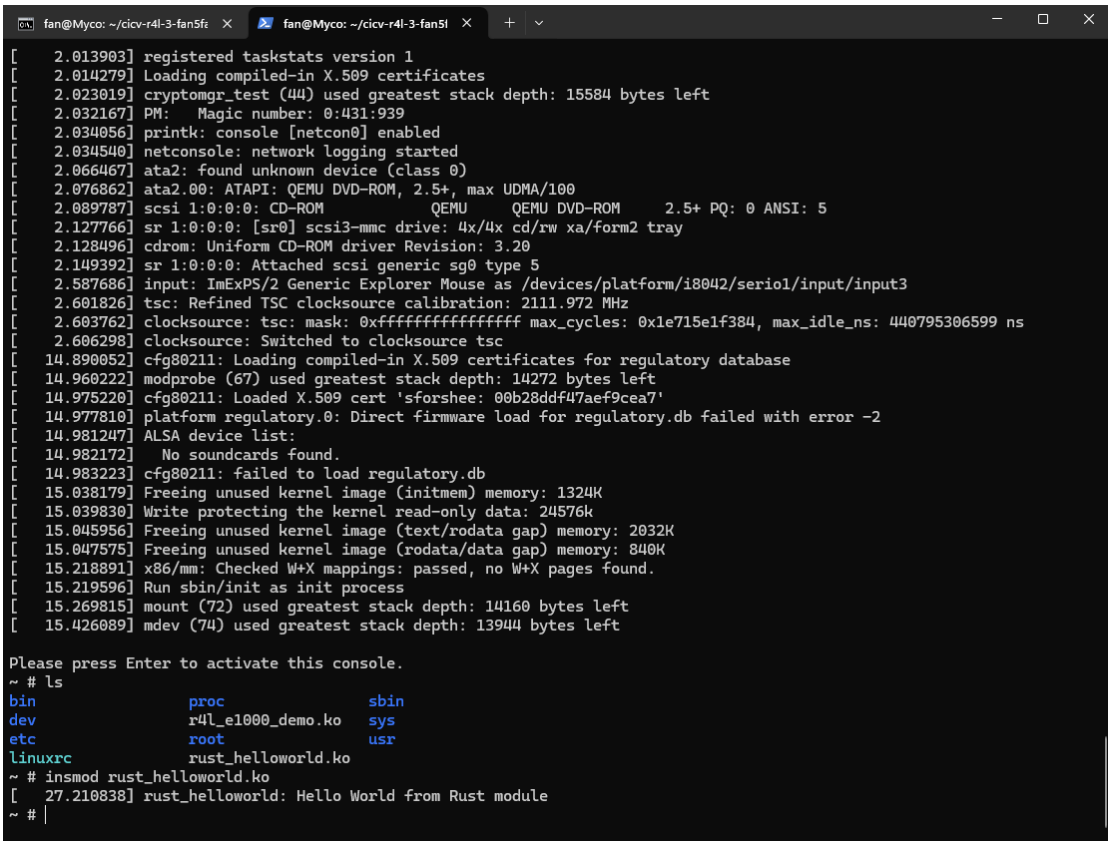


6、复制 rust_helloworld.ko 文件到 src_e1000/rootfs/目录下



7、执行 build_image.sh

8、执行 insmod 加载 rust_helloworld.ko



Makefile 文件添加一行：

```
obj-$(CONFIG_PRINT_HELLOWORLD)          += rust_helloworld.o
```

Kconfig 文件添加：

```
config PRINT_HELLOWORLD
    tristate "Print Helloworld in Rust"
    help
        This option builds the Rust helloworld module sample.
        To compile this as a module, choose M here:
```

the module will be called rust_helloworld.

If unsure, say N.

五、作业四

1、修改 r4l_e1000_demo.rs

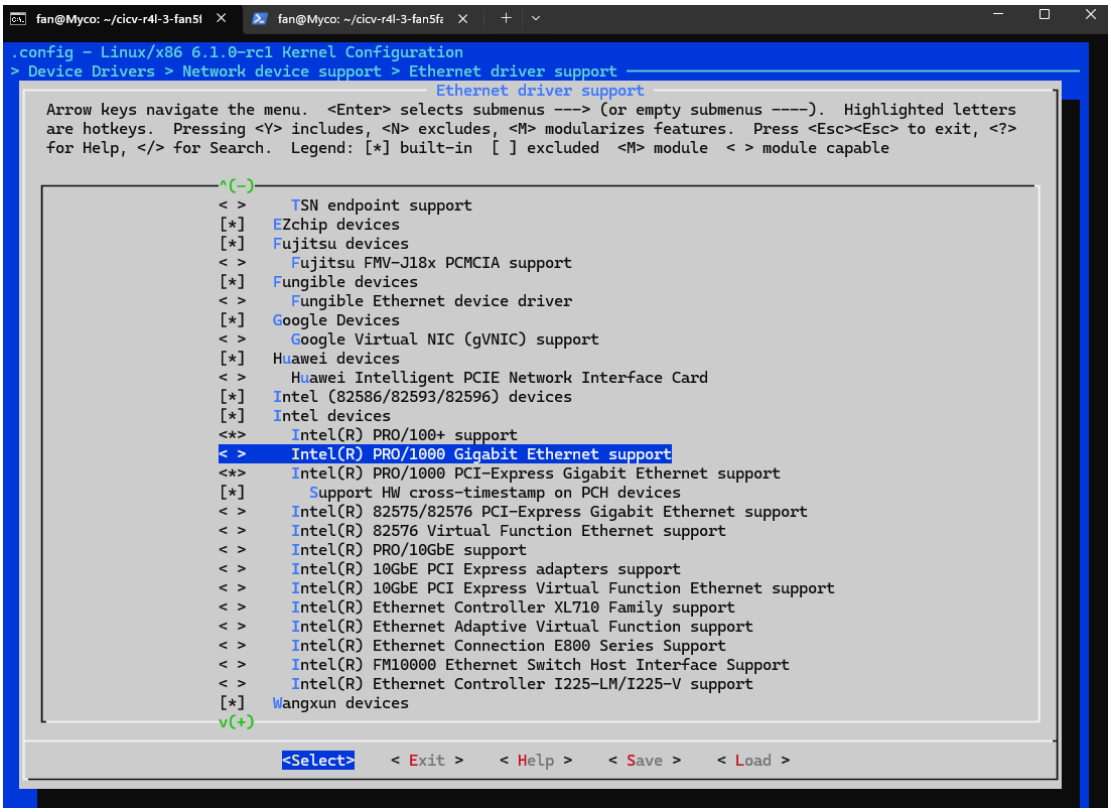
```
fn device_remove(&self) {
    pr_info!("Rust for linux e1000 driver demo (device_remove)\n");

    let netdev = self._netdev_reg.dev_get();
    netdev.netif_carrier_off();
    netdev.netif_stop_queue();
}

fn remove(dev: &mut pci::Device, data: &Self::Data) {
    pr_info!("Rust for linux e1000 driver demo (remove)\n");

    let bars = dev.select_bars((bindings::IORESOURCE_MEM |
bindings::IORESOURCE_IO) as u64);
    dev.release_selected_regions(bars);
    dev.disable_device();
    drop(data);
}
```

2、修改 linux/rust/kernel/pci.rs, 并添加两个函数

```
pub trait Driver {
    fn remove(dev: &mut Device, _data: &Self::Data);
}
impl Device {
    pub fn release_selected_regions(&mut self, bars: i32) {
```

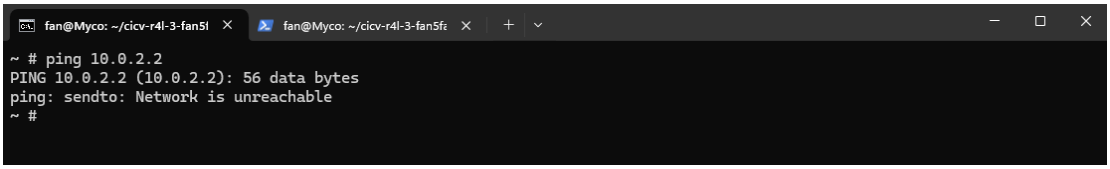
```
            unsafe { bindings::pci_release_selected_regions(self.ptr, bars) };
        }
        pub fn disable_device(&mut self) {
            unsafe { bindings::pci_disable_device(self.ptr) };
        }
    }
```

3、取消选中内置 e1000 网卡驱动，重新编译内核

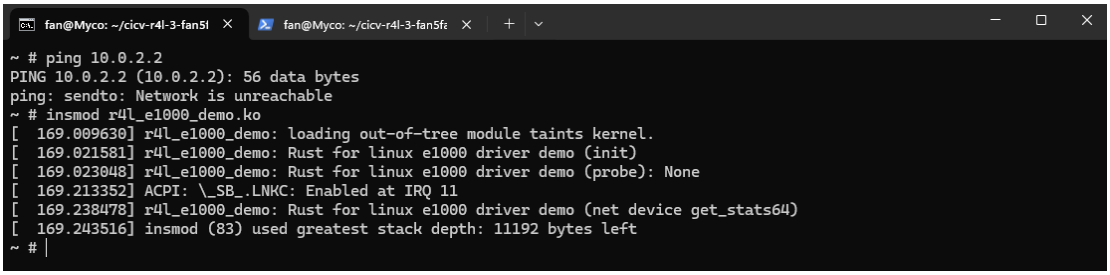


4、运行 src_e1000/build_image.sh

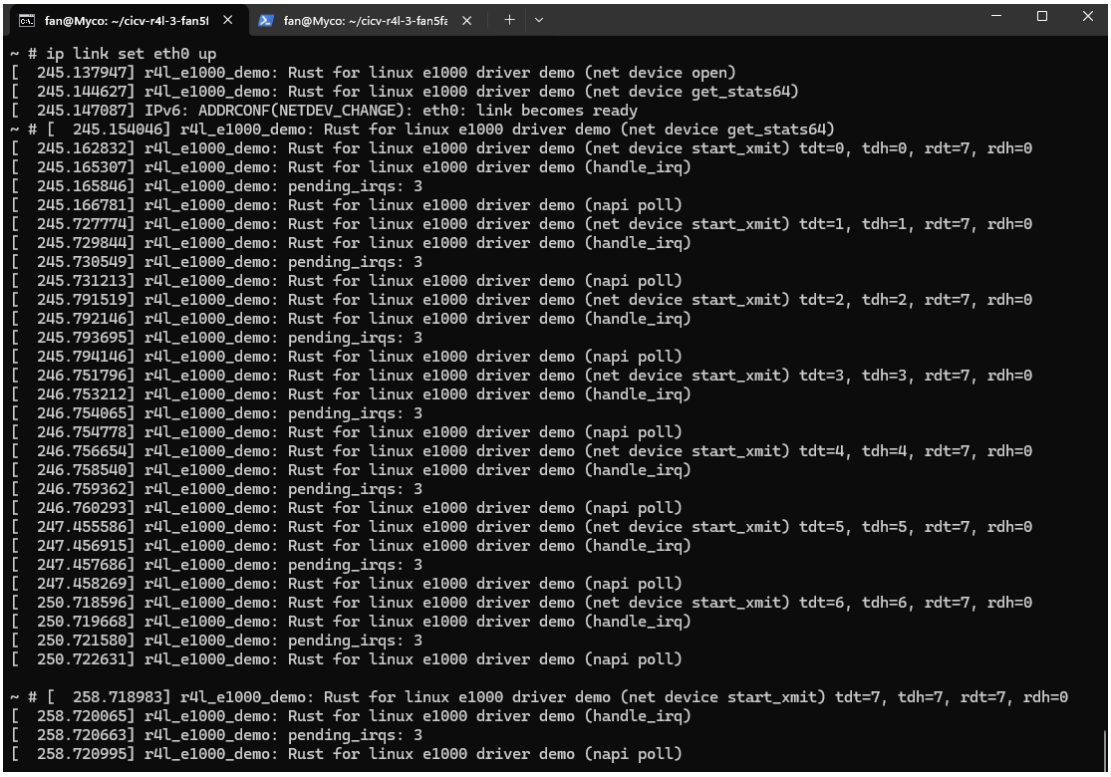
5、进入 qemu，并测试网络

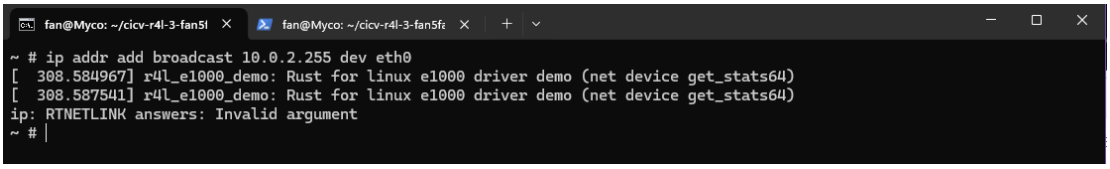


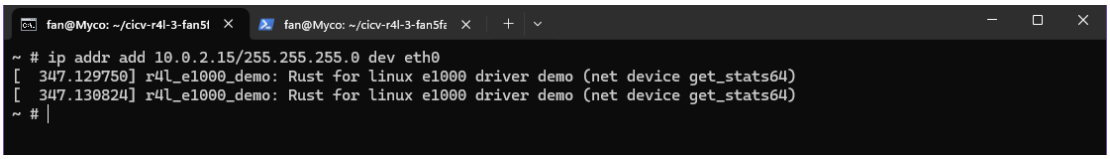
6、加载 r4l_e1000_demo.ko

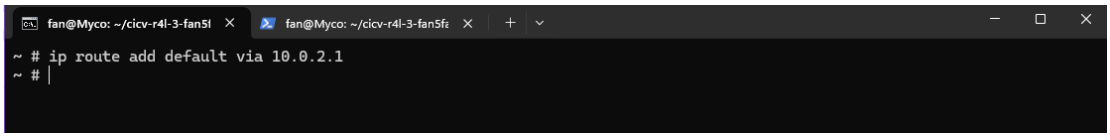

fan@Myco: ~/cicv-r4l-3-fan5l
fan@Myco: ~/cicv-r4l-3-fan5fa
~ # ping 10.0.2.2
PING 10.0.2.2 (10.0.2.2): 56 data bytes
ping: sendto: Network is unreachable
~ # insmod r4l_e1000_demo.ko
[169.009630] r4l_e1000_demo: loading out-of-tree module taints kernel.
[169.021581] r4l_e1000_demo: Rust for linux e1000 driver demo (init)
[169.023048] r4l_e1000_demo: Rust for linux e1000 driver demo (probe): None
[169.213352] ACPI: _SB_.LNKC: Enabled at IRQ 11
[169.238478] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
[169.243516] insmod (83) used greatest stack depth: 11192 bytes left
~ #

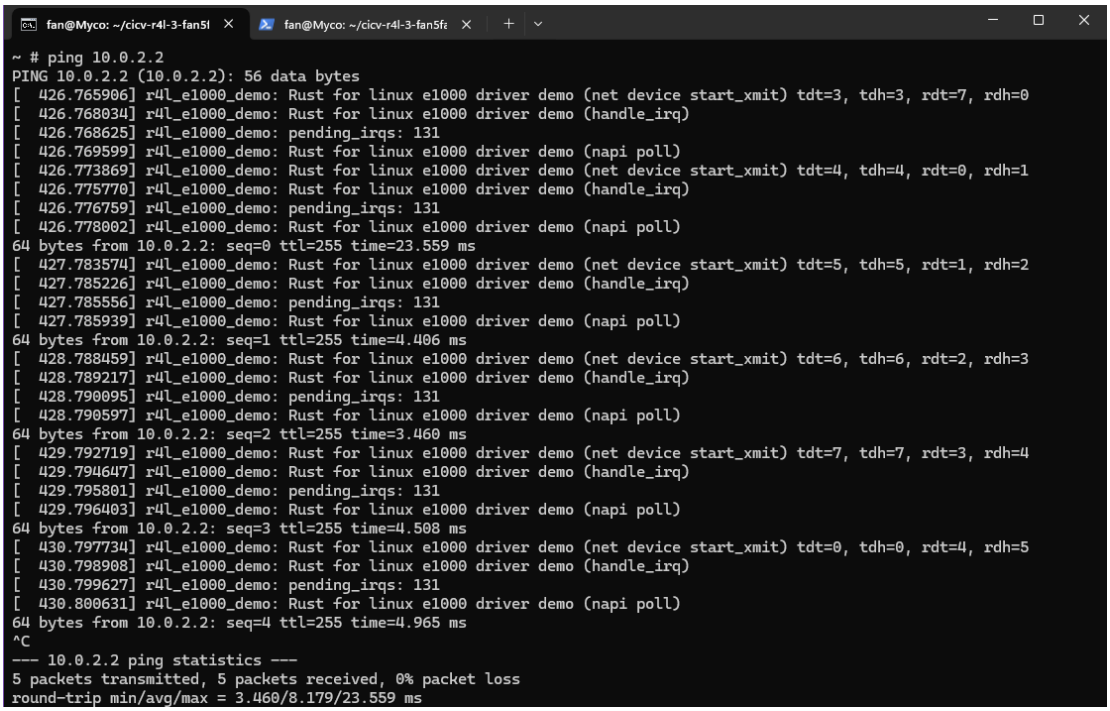
7、添加网络设置


fan@Myco: ~/cicv-r4l-3-fan5l
fan@Myco: ~/cicv-r4l-3-fan5fa
~ # ip link set eth0 up
[245.137947] r4l_e1000_demo: Rust for linux e1000 driver demo (net device open)
[245.144627] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
[245.147087] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
~ # [245.154046] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
[245.162832] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=0, tdh=0, rdt=7, rdh=0
[245.165307] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[245.165846] r4l_e1000_demo: pending_irqs: 3
[245.166781] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[245.727774] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=1, tdh=1, rdt=7, rdh=0
[245.729844] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[245.730549] r4l_e1000_demo: pending_irqs: 3
[245.731213] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[245.791519] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=2, tdh=2, rdt=7, rdh=0
[245.792146] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[245.793695] r4l_e1000_demo: pending_irqs: 3
[245.794146] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[246.751796] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=3, tdh=3, rdt=7, rdh=0
[246.753212] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[246.754065] r4l_e1000_demo: pending_irqs: 3
[246.754778] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[246.756654] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=4, tdh=4, rdt=7, rdh=0
[246.758540] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[246.759362] r4l_e1000_demo: pending_irqs: 3
[246.760293] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[247.455586] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=5, tdh=5, rdt=7, rdh=0
[247.456915] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[247.457686] r4l_e1000_demo: pending_irqs: 3
[247.458269] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[250.718596] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=6, tdh=6, rdt=7, rdh=0
[250.719668] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[250.721580] r4l_e1000_demo: pending_irqs: 3
[250.722631] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
~ # [258.718983] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=7, tdh=7, rdt=7, rdh=0
[258.720065] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[258.720663] r4l_e1000_demo: pending_irqs: 3
[258.720995] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)

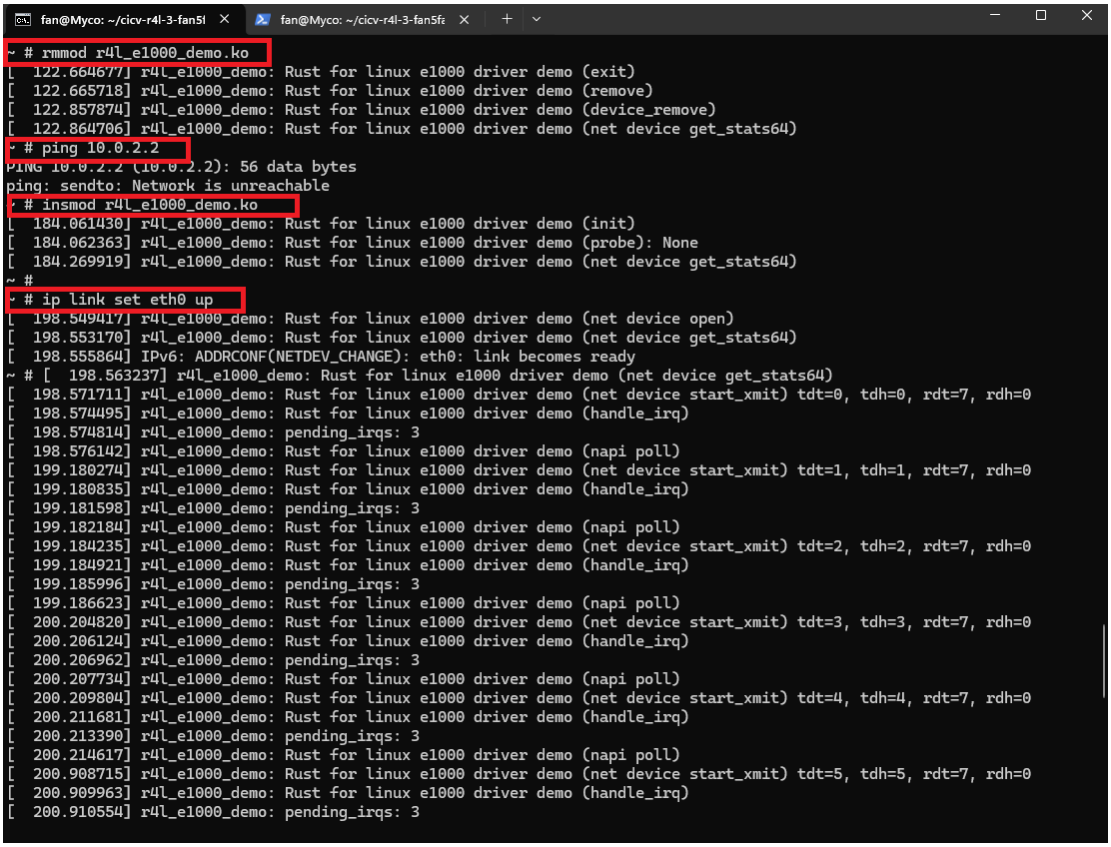

fan@Myco: ~/cicv-r4l-3-fan5l
fan@Myco: ~/cicv-r4l-3-fan5fa
~ # ip addr add broadcast 10.0.2.255 dev eth0
[308.584967] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
[308.587541] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
ip: RTNETLINK answers: Invalid argument
~ #


fan@Myco: ~/cicv-r4l-3-fan5l
fan@Myco: ~/cicv-r4l-3-fan5fa
~ # ip addr add 10.0.2.15/255.255.255.0 dev eth0
[347.129750] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
[347.130824] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
~ #

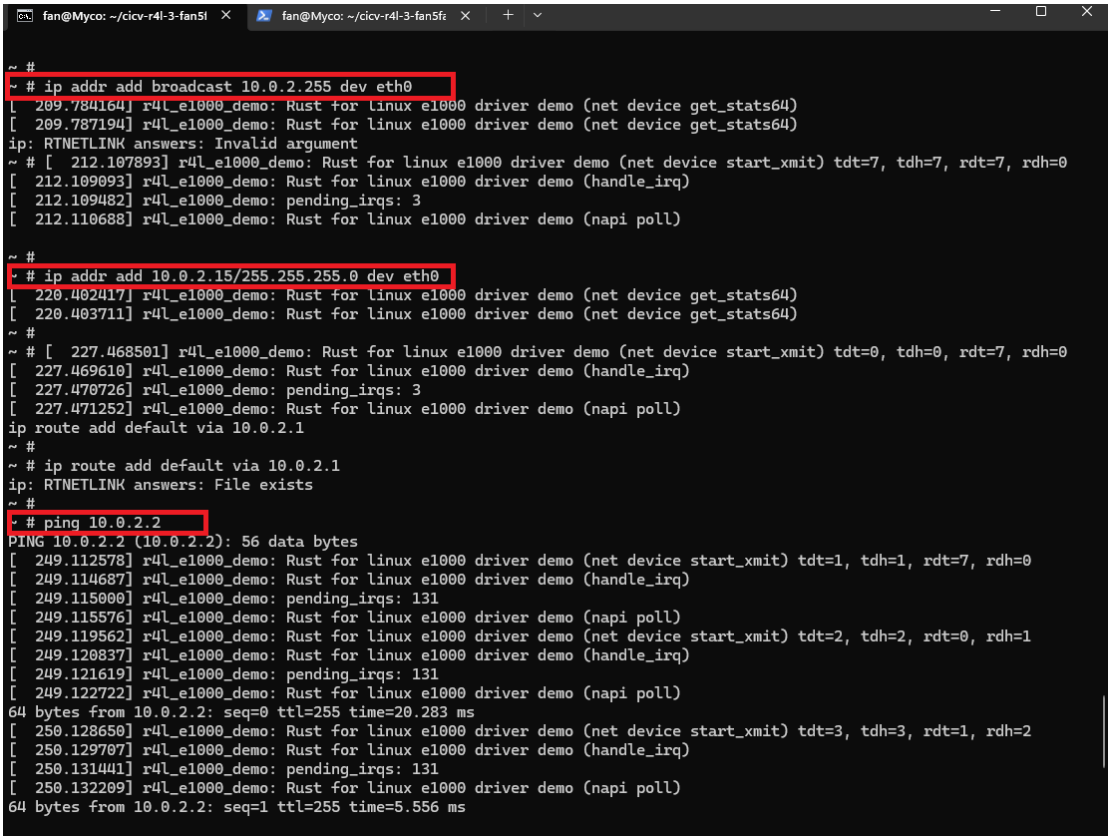

fan@Myco: ~/cicv-r4l-3-fan5l
fan@Myco: ~/cicv-r4l-3-fan5fa
~ # ip route add default via 10.0.2.1
~ #

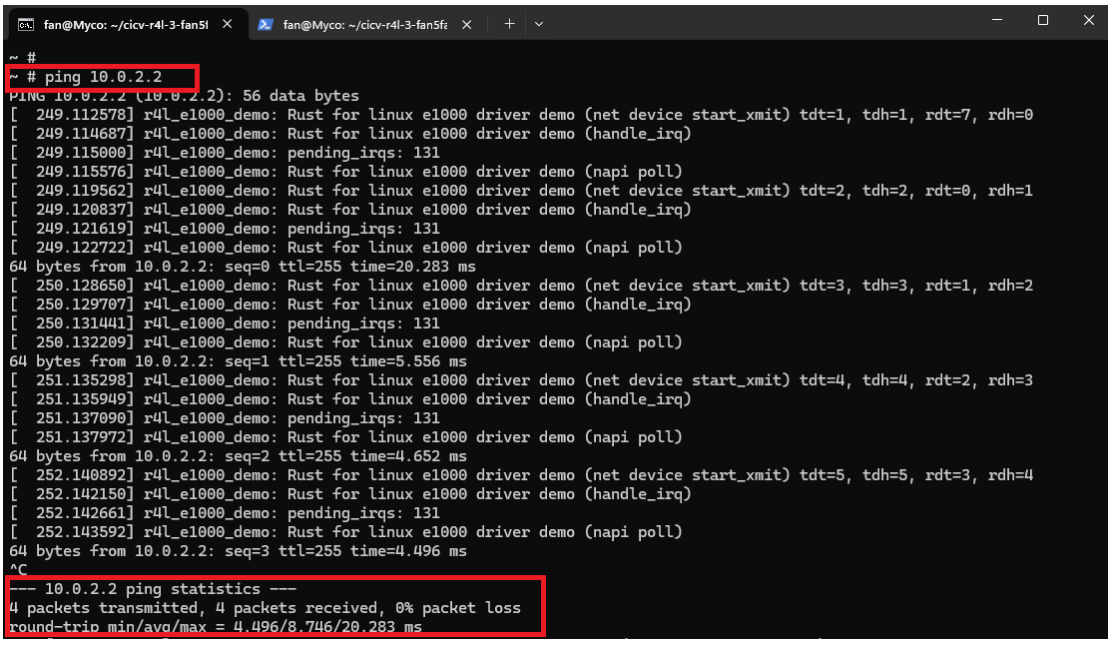


8、卸载 r4l_e1000_demo.ko， 重新加载 r4l_e1000_demo.ko



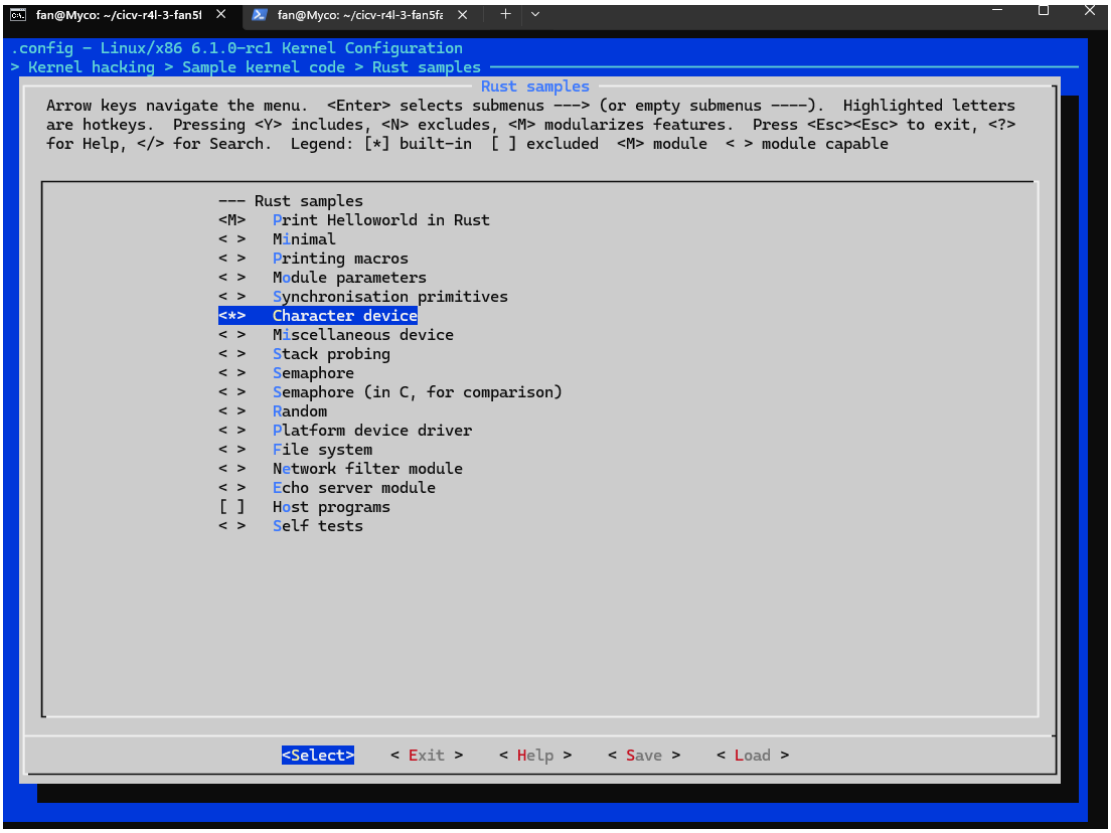
9、设置网络并 ping 通





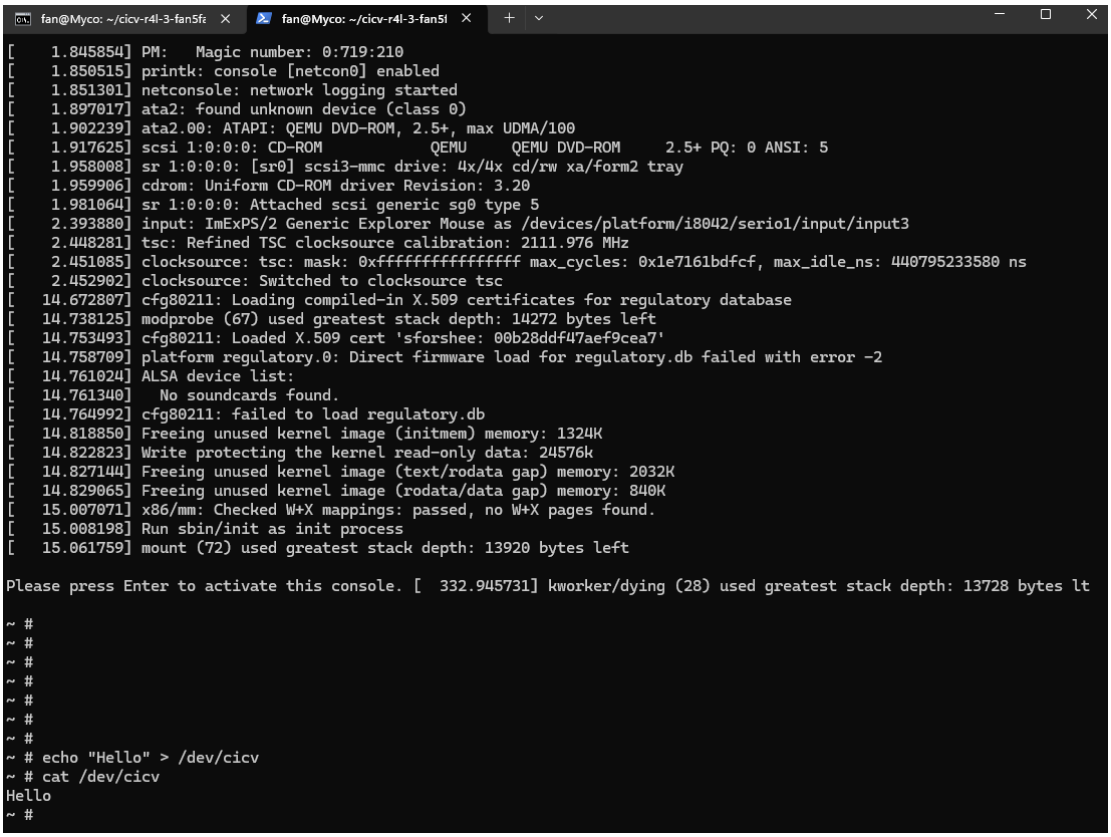
六、作业五

1、make LLVM=1 menuconfig



2、重新编译内核

3、执行操作



4、作业 5 中的字符设备/dev/cicv 是怎么创建的？它的设备号是多少？它是如何与我们写的字符设备驱动关联上的？

echo "mknod /dev/cicv c 248 0" >> etc/init.d/rcS

1、由系统启动时 init.d/rcS 文件的 mknod 指令创建

2、它的记设备号是 248，次设备号是 0

3、kernel::file::Operations trait 是一个函数指针的集合. 这些操作大部分负责实现系统调用, 在驱动注册时，实现了这个 trait 的结构与驱动关联.

```
let mut chrdev_reg = chrdev::Registration::new_pinned(name, 0, module)?;
chrdev_reg.as_mut().register::<RustFile>()?;
```

RustFile 这个结构实现了 kernel::file::Operations。