Stage3 作业提交文档

姓名：田睿枭
电话：15869609867
完成情况：完成所有题目

这次是重新提交，因为最终完成了第四题

情况简述：

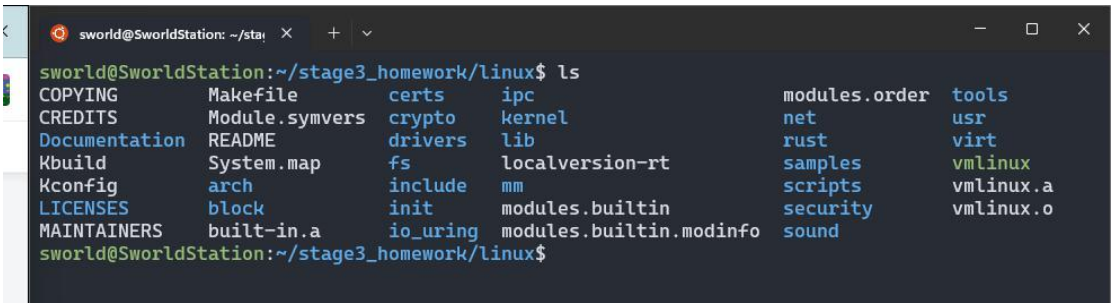
第三阶段的前几题都比我想象中要顺利，在遇到问题后进行搜索也都比较快地解决了问题，然而到最后一题，先是 Rust-Analyzer 的配置让我卡了很久（原因也是我没认真看 README 里的链接），在打上 Patch 后终于解决了，在阅读了带注释的 C 版本源代码后，终于完成了。

在这个过程中，真是深感自己知识不足，因为只是相关专业(信息管理与信息系统)，对计算机底层的一些知识都是靠自学得来的，有的部分掌握尚有欠缺，这段时间也比较忙，没能及时补上，不过此次训练营还是给我带来了挺多收获的，也参与了大家的讨论，我觉得是挺不错的。

作业详情

Exercise 1

环境仍然使用 Stage2 的环境，使用仓库 https://www.github.com/zxgsn/stage3_homework 准备好的代码仓库，按照步骤进行构建，成功生成了所需文件，也可以看到 vmlinux 文件



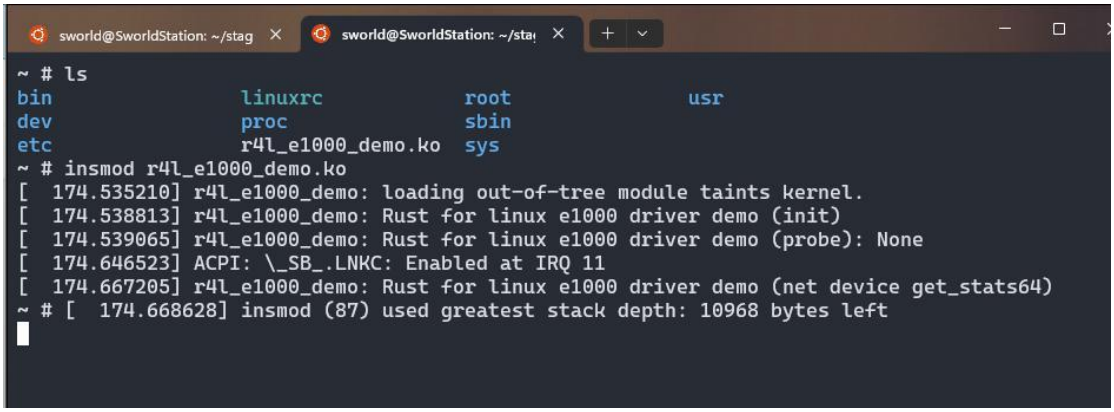
Exercise 2

使用脚本启动，发现启动失败

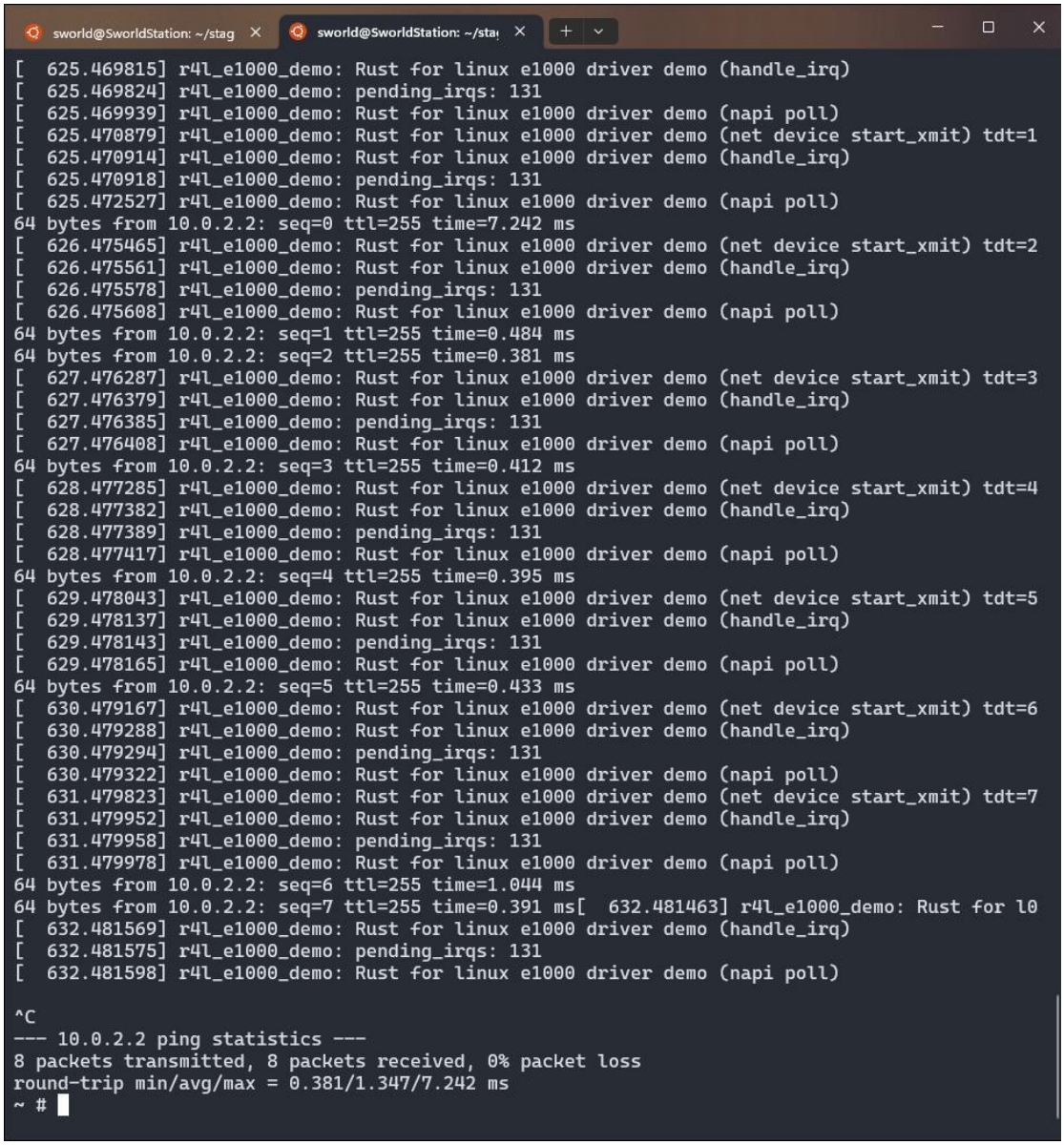
```
sworld@SworldStation:~/stage3_homework/src_e1000$ chmod 777 ./build_image.sh
sworld@SworldStation:~/stage3_homework/src_e1000$ ./build_image.sh
make -C ../linux M=$PWD
make[1]: Entering directory '/home/sworld/stage3_homework/linux'
  RUSTC [M] /home/sworld/stage3_homework/src_e1000/r4l_e1000_demo.o
  MODPOST /home/sworld/stage3_homework/src_e1000/Module.symvers
  CC [M]  /home/sworld/stage3_homework/src_e1000/r4l_e1000_demo.mod.o
  LD [M]  /home/sworld/stage3_homework/src_e1000/r4l_e1000_demo.ko
make[1]: Leaving directory '/home/sworld/stage3_homework/linux'

5313 blocks
qemu-system-x86_64: -netdev user,id=eth0: network backend 'user' is not compiled into this b
inary
sworld@SworldStation:~/stage3_homework/src_e1000$
```

发现是 Qemu 问题，由于我使用的是 Stage2 留下来的自己编译的 Qemu，因此去除了该环境变量，使用 apt-get 重新下载 Qemu，运行成功

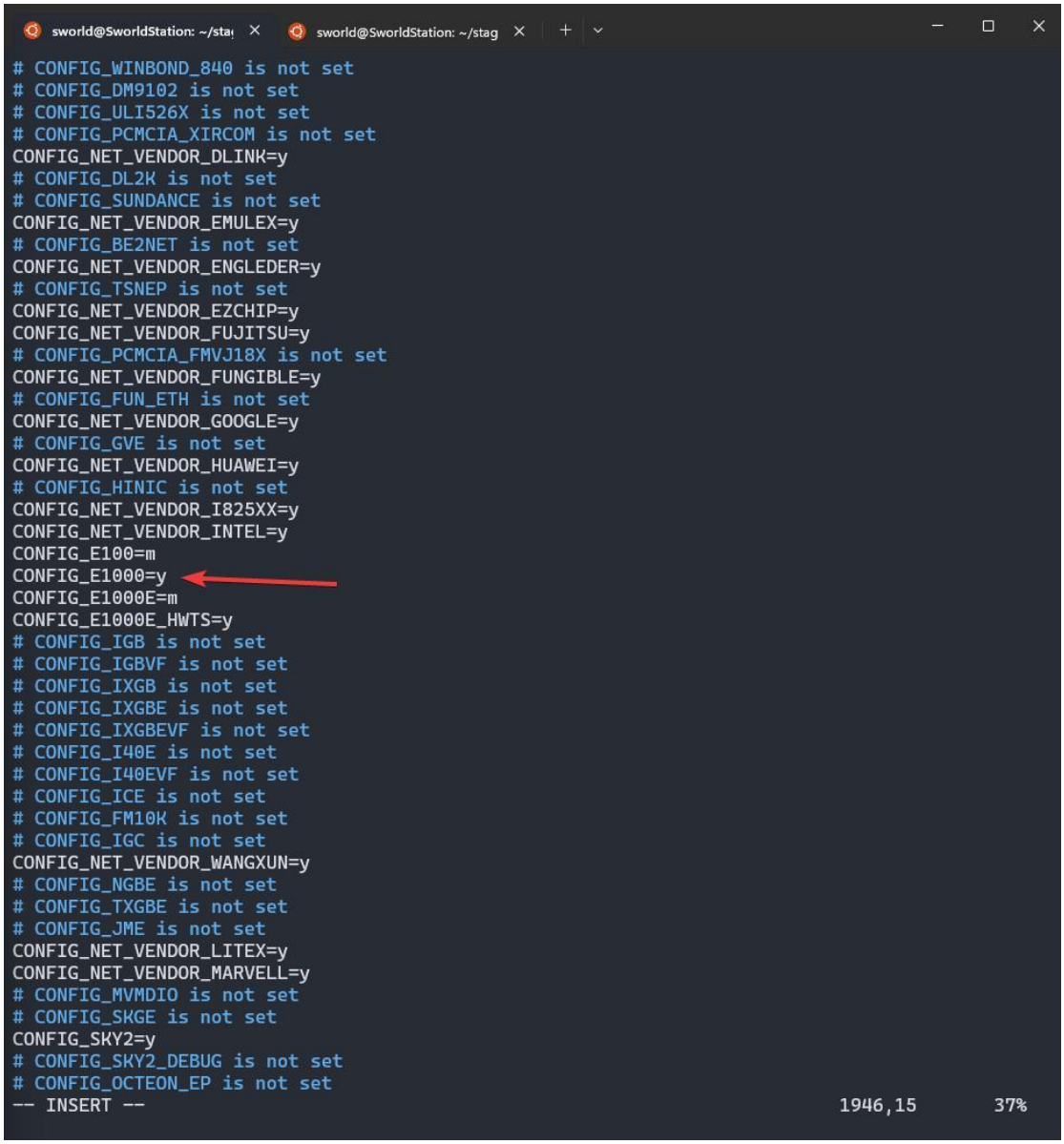
然后启用 Rust 编写的网卡驱动模块



执行后续指令，可以使用 ping 命令



修改 .config 文件，启用原版驱动，并且重新编译

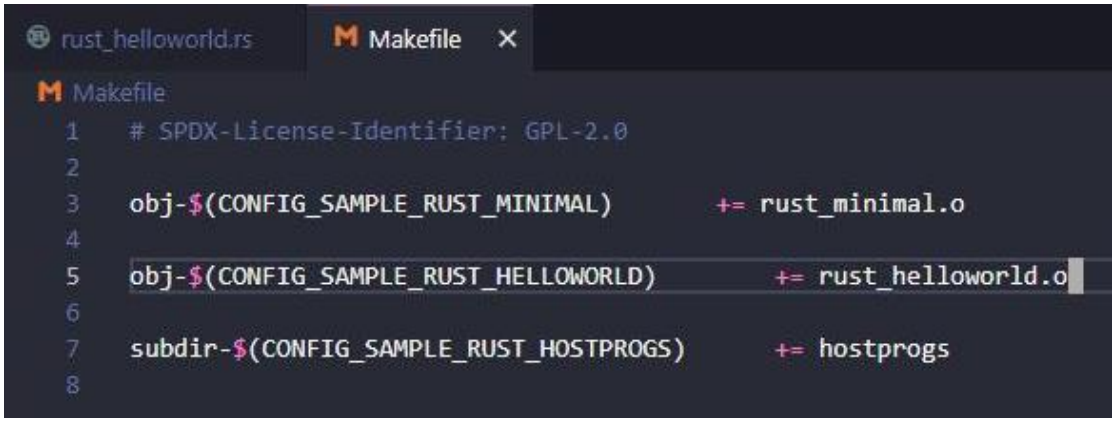


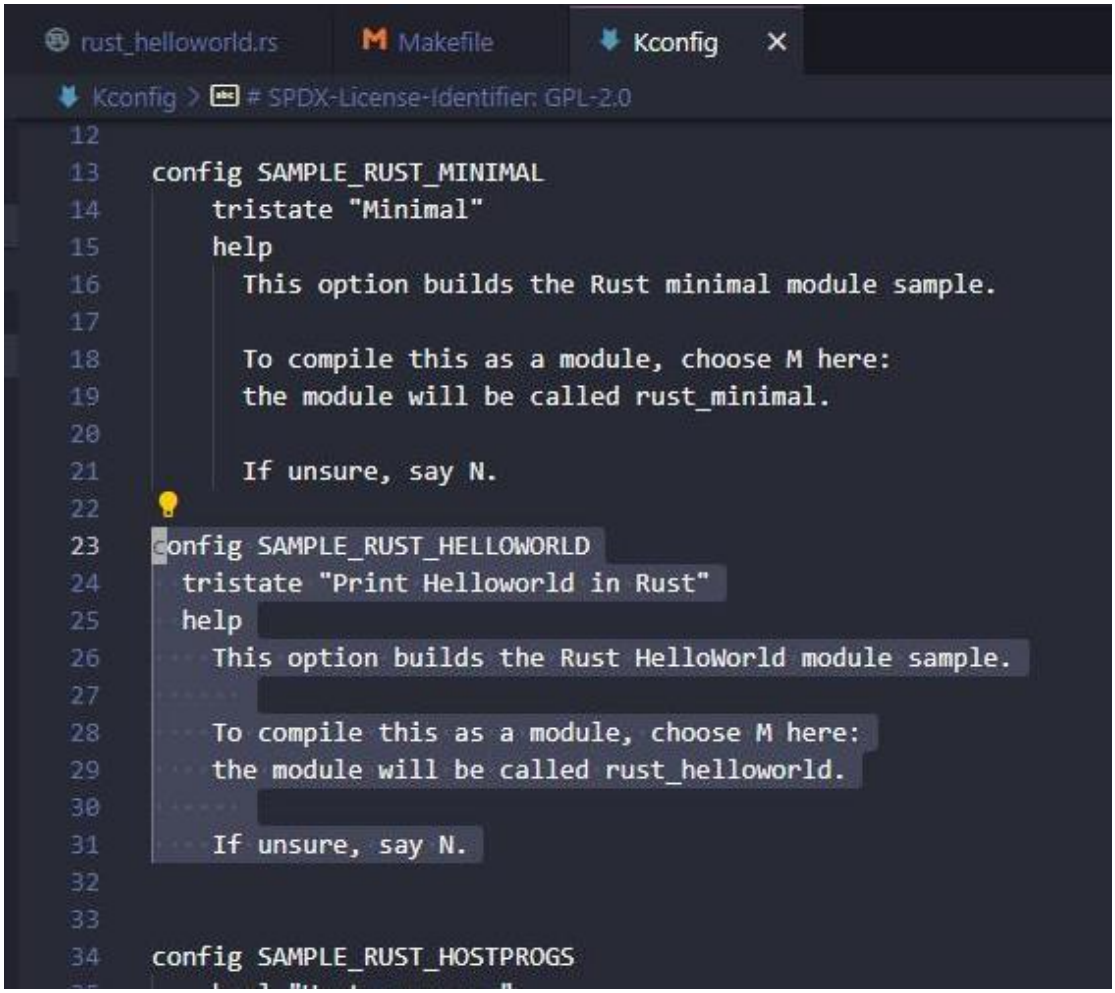
重新运行指令，成功

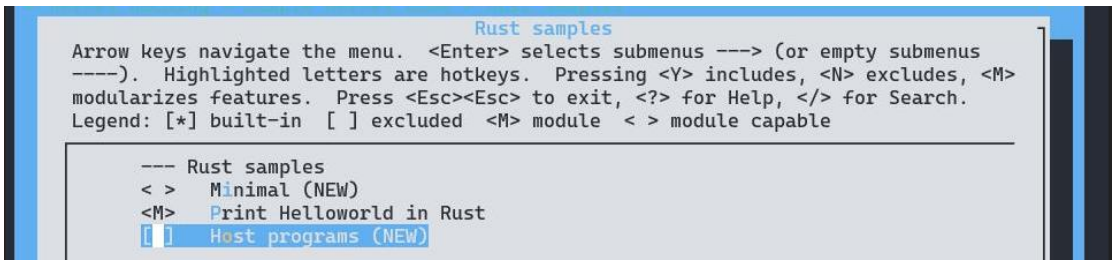

sworld@SworldStation: ~/stag
sworld@SworldStation: ~/sta
[1.936689] modprobe (68) used greatest stack depth: 14216 bytes left
[1.942931] cfg80211: Loaded X.509 cert 'sforshee: 00b28ddf47aef9cea7'
[1.943684] platform regulatory.0: Direct firmware load for regulatory.db failed with err2
[1.943800] cfg80211: failed to load regulatory.db
[1.945011] ALSA device list:
[1.945138] No soundcards found.
[1.977970] Freeing unused kernel image (initmem) memory: 1288K
[1.979563] Write protecting the kernel read-only data: 22528k
[1.981117] Freeing unused kernel image (text/rodata gap) memory: 2032K
[1.982144] Freeing unused kernel image (rodata/data gap) memory: 1596K
[2.103292] x86/mm: Checked W+X mappings: passed, no W+X pages found.
[2.104029] Run sbin/init as init process
[2.124319] mount (73) used greatest stack depth: 14112 bytes left
[2.177934] mdev (75) used greatest stack depth: 13840 bytes left
Please press Enter to activate this console.
~ # ifconfig
eth0 Link encap:Ethernet HWaddr 52:54:00:12:34:56
inet addr:10.0.2.15 Bcast:10.0.2.255 Mask:255.255.255.0
inet6 addr: fec0::5054:ff:fe12:3456/64 Scope:Site
inet6 addr: fe80::5054:ff:fe12:3456/64 Scope:Link
UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1
RX packets:1 errors:0 dropped:0 overruns:0 frame:0
TX packets:7 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:110 (110.0 B) TX bytes:602 (602.0 B)
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:65536 Metric:1
RX packets:0 errors:0 dropped:0 overruns:0 frame:0
TX packets:0 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:1000
RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)
~ # ping 10.0.2.2
PING 10.0.2.2 (10.0.2.2): 56 data bytes
64 bytes from 10.0.2.2: seq=0 ttl=255 time=4.750 ms
64 bytes from 10.0.2.2: seq=1 ttl=255 time=0.392 ms
64 bytes from 10.0.2.2: seq=2 ttl=255 time=0.334 ms
64 bytes from 10.0.2.2: seq=3 ttl=255 time=0.349 ms
64 bytes from 10.0.2.2: seq=4 ttl=255 time=0.293 ms
^C
--- 10.0.2.2 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.293/1.223/4.750 ms
~ #

Exercise 3

尝试构建 in-tree 模块，首先修改 Makefile 和 Kconfig，并且使用 menuconfig 进行配置







编译，获得 ko 文件，将其复制到对应目录

```
rpmsg/ rust/
sworld@SworldStation:~/stage3_homework/linux$ ls samples/rust/
Kconfig     hostprogs          rust_helloworld.mod    rust_helloworld.o
Makefile    modules.order      rust_helloworld.mod.c  rust_helloworld.rs
built-in.a  rust_helloworld.ko rust_helloworld.mod.o  rust_minimal.rs
```

使用 insmod 加载，可以看到正确的输出

```
~ # ls
bin                 proc                rust_minimal.ko
dev                 r4l_e1000_demo.ko   sbin
etc                 root                sys
linuxrc             rust_helloworld.ko  usr
~ # insmod rust_minimal.ko
[  119.723075] rust_minimal: Rust minimal sample (init)
[  119.723103] rust_minimal: Am I built-in? false
~ # rmmod rust_minimal.ko
[  124.683753] rust_minimal: My numbers are [72, 108, 200]
[  124.683836] rust_minimal: Rust minimal sample (exit)
~ #
~ #
~ # insmod rust_helloworld.ko
~ # rmmod rust_helloworld.ko
~ # [  130.172696] rust_helloworld: Hello World from Rust module
[  332.850941] kworker/dying (29) used greatest stack depth: 13664 bytes left
```

Exercise 4

首先清理环境

```
sworld@SworldStation:~/stage3_homework/linux$ make mrproper
  CLEAN   arch/x86/boot/compressed
  CLEAN   arch/x86/boot
  CLEAN   arch/x86/entry/vdso
  CLEAN   arch/x86/kernel/cpu
  CLEAN   arch/x86/kernel
  CLEAN   arch/x86/realmode/rm
  CLEAN   arch/x86/tools
  CLEAN   arch/x86/lib
  CLEAN   certs
  CLEAN   drivers/firmware/efi/libstub
  CLEAN   drivers/scsi
  CLEAN   drivers/tty/vt
  CLEAN   init
  CLEAN   lib
  CLEAN   net/wireless
  CLEAN   rust
  CLEAN   security/selinux
  CLEAN   usr
  CLEAN   .
  CLEAN   modules.builtin modules.builtin.modinfo .vmlinux.export.c
  CLEAN   scripts/basic
  CLEAN   scripts/kconfig
  CLEAN   scripts/mod
  CLEAN   scripts/selinux/genheaders
  CLEAN   scripts/selinux/mdp
  CLEAN   scripts
  CLEAN   include/config include/generated arch/x86/include/generated .config .version Module
.symvers rust/target.json rust/libmacros.so
```

根据 [该链接](#) 修改对应的文件(Makefile, Script 等), 然后在 off-tree 网卡驱动目录执行指令, 生成 Rust-Analyzer 所需文件

```
sworld@SworldStation:~/stage3_homework/src_e1000$ make -C ../linux M=$PWD rust-analyzer
make: Entering directory '/home/sworld/stage3_homework/linux'
make: Leaving directory '/home/sworld/stage3_homework/linux'
```

修改 stop 函数，重新构建

```
fn stop(_dev: &net::Device, _data: &NetDevicePrvData) -> Result {
    pr_info!("Rust for linux e1000 driver demo (net device stop)\n");

    _dev.netif_tx_disable();
    _dev.netif_carrier_off();
    _data.napi.disable();
    *_data.tx_ring.lock() = None;
    *_data.rx_ring.lock() = None;

    Ok(())
}
```

执行指令，并且可以 ping

```
--- 10.0.2.2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 0.444/323.426/968.579 ms
~ #
```

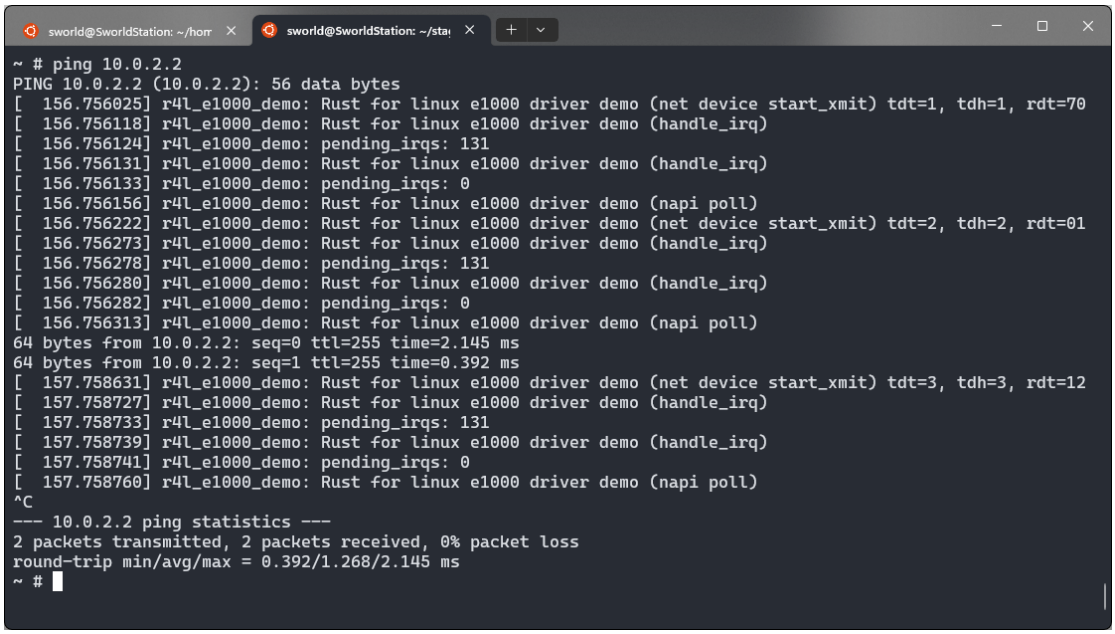
执行 ip link set eth0 down

```
ip link set eth0 down
[   97.918909] r4l_e1000_demo: Rust for linux e1000 driver demo (net device stop)
[   97.919176] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
~ #
```

执行 ip link set eth0 up

```
~ # ip link set eth0 up
[  116.966310] r4l_e1000_demo: Rust for linux e1000 driver demo (net device open)
[  116.966840] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
[  116.967192] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[  116.967299] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
```

执行 ping



可见，up-down-up 执行过程没有报错，在完成后 ping 指令也可以正确运行