

## 实验操作记录

### 1. 没有取消 c 版本的网卡驱动，可以直接 ping 通

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS
-i SECS Interval
-A Ping as soon as reply is received
-t TTL Set TTL
-I IFACE/IP Source interface or IP address
-W SEC Seconds to wait for the first response (default 10)
(after all -c CNT packets are sent)
-w SEC Seconds until ping exits (default:infinite)
(can exit earlier with -c CNT)
-q Quiet, only display output at start/finish
-p HEXBYTE Payload pattern

~ # 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=4.220 ms
64 bytes from 10.0.2.2: seq=1 ttl=255 time=0.617 ms
64 bytes from 10.0.2.2: seq=2 ttl=255 time=1.742 ms
64 bytes from 10.0.2.2: seq=3 ttl=255 time=0.994 ms
64 bytes from 10.0.2.2: seq=4 ttl=255 time=1.212 ms
^C
--- 10.0.2.2 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max = 0.617/1.757/4.220 ms
~ # dmesg
[ 0.000000] Linux version 6.1.0-rc1 (y4ng@DESKTOP-Q77DQ86) (Debian clang version 14.0.6, Debian LLD 14.0.6) #1 SM4
[ 0.000000] Command line: root=/dev/ram rdinit=sbin/init ip=10.0.2.15::10.0.2.1:255.255.255.0 console=ttyS0 no_tik
[ 0.000000] x86/fpu: x87 FPU will use FXSAVE
[ 0.000000] signal: max sigframe size: 1440
[ 0.000000] BIOS-provided physical RAM map:

```

## 2. 取消对应驱动模块的编译

```

make
x + v
.config - Linux/x86 6.1.0-rc1 Kernel Configuration
> Device Drivers > Network device support > Ethernet driver support
Ethernet driver support
Arrow keys navigate the menu. <Enter> selects submenus — (or empty submenu —). Highlighted letters
are hotkeys. Pressing <Y> includes, <N> excludes, <M> modularizes features. Press <Esc><Esc> to exit, <?>
for Help, </> for Search. Legend: [*] built-in [ ] excluded <M> module < > module capable

^(-)
[*] Fungible devices
< > Fungible Ethernet device driver
[*] Google Devices
< > Google Virtual NIC (gVNIC) support
[*] Huawei devices
< > Huawei Intelligent PCIE Network Interface Card
[*] Intel (82586/82593/82596) devices
[*] Intel devices
<> Intel(R) PRO/100+ support
< > Intel(R) PRO/1000 Gigabit Ethernet support
<> Intel(R) PRO/1000 PCI-Express Gigabit Ethernet support
[*] Support HW cross-timestamp on PCH devices
< > Intel(R) 82575/82576 PCI-Express Gigabit Ethernet support
< > Intel(R) 82576 Virtual Function Ethernet support
< > Intel(R) PRO/10GbE support
< > Intel(R) 10GbE PCI Express adapters support
v(+)

<Select> < Exit > < Help > < Save > < Load >

```

### 3. 重新编译内核，进入 qemu 后载入 rust 版本驱动模块，配置网卡后可以执行 ping 命令

```
./build_image.sh
~ # ifconfig
[ 396.043461] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
eth0      Link encap:Ethernet  HWaddr 52:54:00:12:34:56
          inet addr:10.0.2.15  Bcast:0.0.0.0   Mask:255.255.255.0
          inet6 addr: fe80::5054:ff:fe12:3456/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  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)

~ # ip route add default via 10.0.2.1
~ # ping 10.0.2.2
PING 10.0.2.2 (10.0.2.2): 56 data bytes
[ 420.667602] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=4, tdh=4, rdt=7, rdh=0
[ 420.668039] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[ 420.668169] r4l_e1000_demo: pending_irqs: 131
[ 420.668324] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[ 420.669829] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=5, tdh=5, rdt=0, rdh=1
[ 420.670297] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[ 420.670672] r4l_e1000_demo: pending_irqs: 131
[ 420.671355] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=0 ttl=255 time=9.010 ms
[ 421.676511] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=6, tdh=6, rdt=1, rdh=2
[ 421.678538] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[ 421.679019] r4l_e1000_demo: pending_irqs: 131
[ 421.679829] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=1 ttl=255 time=5.416 ms
[ 422.682332] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=7, tdh=7, rdt=2, rdh=3
[ 422.684443] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[ 422.685116] r4l_e1000_demo: pending_irqs: 131
[ 422.686063] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=2 ttl=255 time=5.000 ms
[ 423.688296] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=0, tdh=0, rdt=3, rdh=4
[ 423.689112] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[ 423.690152] r4l_e1000_demo: pending_irqs: 131
[ 423.690732] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=3 ttl=255 time=4.818 ms
[ 424.693536] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=1, tdh=1, rdt=4, rdh=5
[ 424.694673] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[ 424.695190] r4l_e1000_demo: pending_irqs: 131
[ 424.696109] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=4 ttl=255 time=4.108 ms
^C
— 10.0.2.2 ping statistics —
5 packets transmitted, 5 packets received, 0% packet loss
```

## 问答题

- 编译成内核模块，是在哪个文件中以哪条语句定义的？
  - 是由 `src_e1000/Makefile` 里的 `$(MAKE) -C $(KDIR) M=$$PWD` 指定的，通过同时指定已完成编译的内核文件夹和当前文件夹，使用当前文件夹下的 `Kbuild` 来指定编译出的模块名以及依赖的文件
- 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？
  - 在编译期通过使用内核代码中的 `module` 过程宏，生成了内核模块被载入时所需的许可证、作者等内容。同时也定义了实现模块的类型 `E1000KernelMod`
  - 通过实现 `Kernel::Module` 向外提供了模块入口函数
  - 通过实现 `Drop` 向外提供了模块出口函数