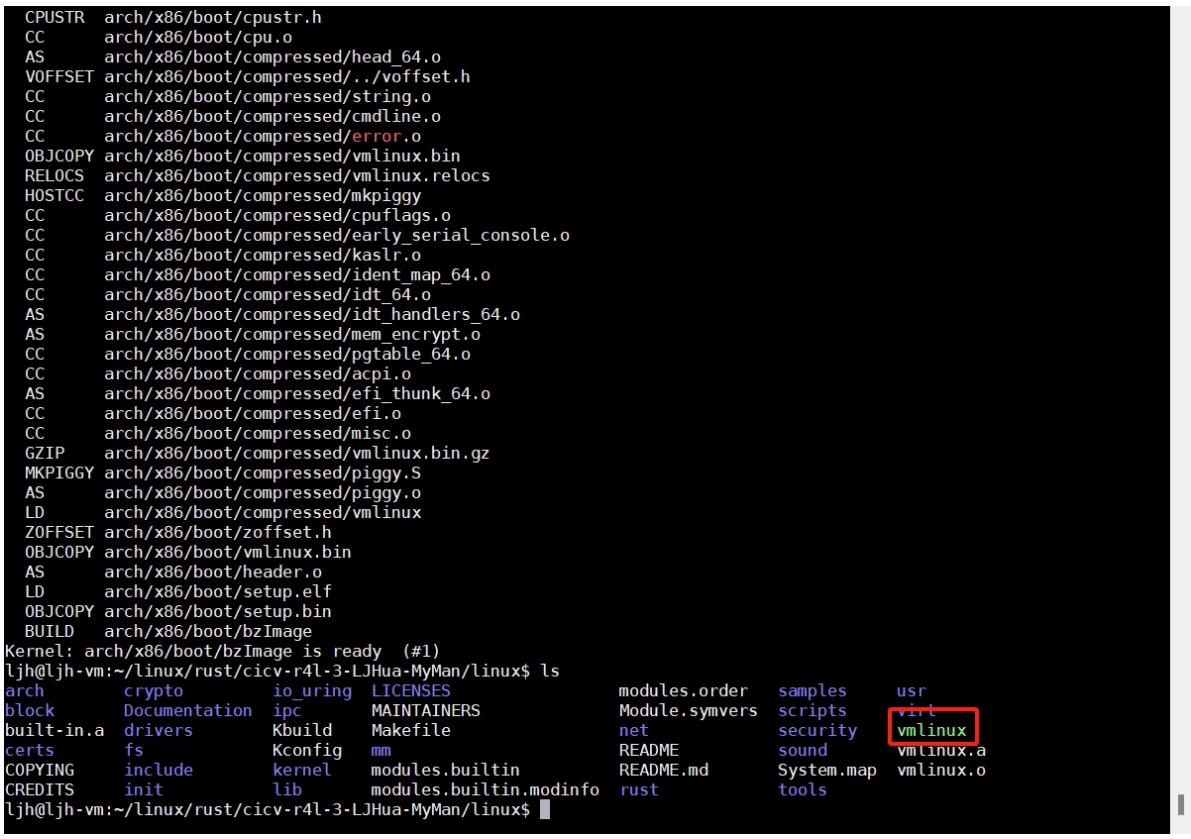
作业1：编译Linux内核

作业说明：

进入Linux文件夹，使用如下命令编译：

```
make x86_64_defconfig
make LLVM=1 menuconfig
#set the following config to yes
General setup
        ---> [*] Rust support
make LLVM=1 -j$(nproc)
```

编译成功后会在Linux文件夹下得到一个vmlinux文件，如下图所示：



作业2：对Linux内核进行一些配置

Q: 在该文件夹中调用make LLVM=1，该文件夹内的代码将编译成一个内核模块。请结合你学到的知识，回答以下两个问题：

1、编译成内核模块，是在哪个文件中以哪条语句定义的?

答：Linux内核使用的是`Kconfig/Kbuild`来配置和编译内核，以`obj-m += 模块名.o`语句定义。

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?

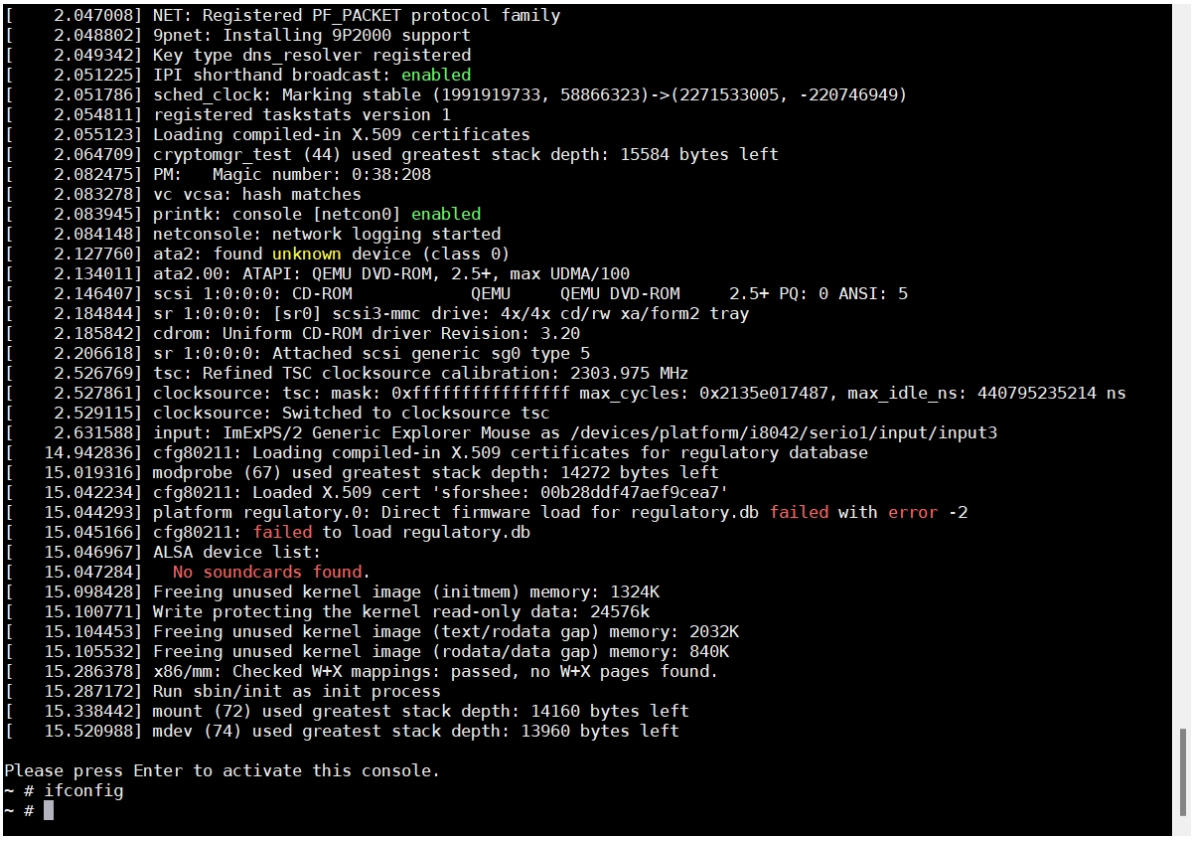
答：在编译时，使用了内核源码树中的`Kbuild`系统。 `Makefile`中会引用内核源码路径，使用如下命令进行编译：

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

KDIR ?= ../linux

default:
	$(MAKE) -C $(KDIR) M=$$PWD
```

这条命令告诉`make`在指定的内核源码目录下进行编译，并将当前目录作为模块源代码目录。这样，尽管模块位于独立文件夹，仍能利用内核的构建系统进行编译。

具体步骤及结果:

1. 重新编译Linux内核，并禁用默认的C版本`e1000`网卡驱动
2. 启动脚本，打开`qemu`并安装内核模块，在未安装e1000驱动前，使用`ifconfig`命令结果为空



3. 安装`e1000`驱动模块，并执行`ip link set eth0 up`连接对应网卡

```
[   15.338442] mount (72) used greatest stack depth: 14160 bytes left
[   15.520988] mdev (74) used greatest stack depth: 13960 bytes left

Please press Enter to activate this console.
~ # ifconfig
~ # insmod r4l_e1000_demo.ko
[   69.717211] r4l_e1000_demo: loading out-of-tree module taints kernel.
[   69.733831] r4l_e1000_demo: Rust for linux e1000 driver demo (init)
[   69.734565] r4l_e1000_demo: Rust for linux e1000 driver demo (probe): None
[   69.942554] ACPI: \_SB_.LNKC: Enabled at IRQ 11
[   69.966541] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
[   69.969401] insmod (81) used greatest stack depth: 11192 bytes left
~ # ip link set eth0 up
[   81.540231] r4l_e1000_demo: Rust for linux e1000 driver demo (net device open)
[   81.544011] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
[   81.545650] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
~ # [   81.551122] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
[   81.558378] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=0, tdh=0, rdt=7, rdh=0
[   81.559567] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[   81.559929] r4l_e1000_demo: pending_irqs: 3
[   81.560961] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[   81.565209] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=1, tdh=1, rdt=7, rdh=0
[   81.565757] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[   81.566264] r4l_e1000_demo: pending_irqs: 3
[   81.566863] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[   81.910881] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=2, tdh=2, rdt=7, rdh=0
[   81.911245] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[   81.911769] r4l_e1000_demo: pending_irqs: 3
[   81.912198] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[   82.591781] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=3, tdh=3, rdt=7, rdh=0
[   82.592427] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[   82.592978] r4l_e1000_demo: pending_irqs: 3
[   82.593508] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[   82.594867] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=4, tdh=4, rdt=7, rdh=0
[   82.595553] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[   82.596048] r4l_e1000_demo: pending_irqs: 3
[   82.596472] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[   83.487328] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=5, tdh=5, rdt=7, rdh=0
[   83.487747] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[   83.488092] r4l_e1000_demo: pending_irqs: 3
[   83.488253] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
```

4. 执行`ip addr add 10.0.2.15/255.255.255.0 brd + dev eth0`为网卡配置`ip`地址和广播地址；为网卡添加一条新的路由`ip route add default via 10.0.2.1`，之后`ping 10.0.2.2`结果如下：

```
[  139.424709] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)

~ # ip addr add 10.0.2.15/255.255.255.0 brd + dev eth0
[  154.183427] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
[  154.185365] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
~ # 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
[  171.546070] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=2, tdh=2, rdt=7, rdh=0
[  171.546882] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  171.547153] r4l_e1000_demo: pending_irqs: 131
[  171.547652] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[  171.549511] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=3, tdh=3, rdt=0, rdh=1
[  171.549961] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  171.550200] r4l_e1000_demo: pending_irqs: 131
[  171.551157] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=0 ttl=255 time=12.575 ms
[  172.555781] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=4, tdh=4, rdt=1, rdh=2
[  172.556745] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  172.557438] r4l_e1000_demo: pending_irqs: 131
[  172.558231] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=1 ttl=255 time=3.575 ms
[  173.559990] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=5, tdh=5, rdt=2, rdh=3
[  173.560446] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  173.560817] r4l_e1000_demo: pending_irqs: 131
[  173.560954] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=2 ttl=255 time=1.523 ms
[  174.563084] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=6, tdh=6, rdt=3, rdh=4
[  174.563584] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  174.564270] r4l_e1000_demo: pending_irqs: 131
[  174.565022] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=3 ttl=255 time=2.740 ms
[  175.567111] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=7, tdh=7, rdt=4, rdh=5
[  175.567761] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  175.568177] r4l_e1000_demo: pending_irqs: 131
[  175.568372] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=4 ttl=255 time=2.096 ms

[  176.570887] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=0, tdh=0, rdt=5, rdh=6
[  176.571399] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  176.571849] r4l_e1000_demo: pending_irqs: 131
```

5. 执行`ifconfig`验证

```
[  174.563584] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  174.564270] r4l_e1000_demo: pending_irqs: 131
[  174.565022] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=3 ttl=255 time=2.740 ms
[  175.567111] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=7, tdh=7, rdt=4, rdh=5
[  175.567761] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  175.568177] r4l_e1000_demo: pending_irqs: 131
[  175.568372] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=4 ttl=255 time=2.096 ms

[  176.570887] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=0, tdh=0, rdt=5, rdh=6
[  176.571399] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  176.571849] r4l_e1000_demo: pending_irqs: 131
[  176.572350] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=5 ttl=255 time=2.384 ms
[  177.574077] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=1, tdh=1, rdt=6, rdh=7
[  177.574712] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  177.575011] r4l_e1000_demo: pending_irqs: 131
[  177.575258] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=6 ttl=255 time=2.014 ms
^C
--- 10.0.2.2 ping statistics ---
7 packets transmitted, 7 packets received, 0% packet loss
round-trip min/avg/max = 1.523/3.843/12.575 ms
~ # [  197.790467] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=2, tdh=2, rdt=7, rdh0
[  197.791478] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  197.791884] r4l_e1000_demo: pending_irqs: 3
[  197.792084] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)

~ # ifconfig
[  226.748924] 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:10.0.2.255  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)

~ #
```

作业 3：使用 rust 编写一个简单的内核模块并运行

1. 进入Linux目录下的samples/rust文件夹，添加一个`rust_helloworld.rs`文件，并添加以下内容

```
// 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\n");
      Ok(RustHelloWorld {})
  }
}
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
"rust_helloworld.rs" 22L, 491B                                        22,0-1        All
```

2. 在`linux/sample/rust/Kconfig`添加配置

```
        help
          This option builds the Rust netfilter module sample.

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

          If unsure, say N.

config SAMPLE_RUST_ECHO_SERVER
        tristate "Echo server module"
        help
          This option builds the Rust echo server  module sample.

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

          If unsure, say N.

config SAMPLE_RUST_HOSTPROGS
        bool "Host programs"
        help
          This option builds the Rust host program samples.

          If unsure, say N.

config SAMPLE_RUST_SELFTESTS
        tristate "Self tests"
        help
          This option builds the self test cases for Rust.

          If unsure, say N.

config SAMPLE_RUST_HELLOWORLD
        tristate "HelloWorld"
        help
          This option builds the Rust HelloWorld module sample.

          If unsure, say N.

endif # SAMPLES_RUST
-- INSERT --                                                    172,1         Bot
```

3. 在`linux/sample/rust/Makefile`添加配置

```
# 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_SAMPLE_RUST_HELLOWORLD)            += rust_helloworld.o

subdir-$(CONFIG_SAMPLE_RUST_HOSTPROGS)          += hostprogs
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
~
-- INSERT --                                                    19,1          All
```

4. 重新编译内核

```
ljh@ljh-vm:~/linux/rust/cicv-r4l-3-LJHua-MyMan/linux$ make LLVM=1 -j8
  SYNC    include/config/auto.conf.cmd
  DESCEND objtool
  CALL    scripts/checksyscalls.sh
  AR      samples/vfio-mdev/built-in.a
  AR      samples/rust/built-in.a
  RUSTC [M] samples/rust/rust_helloworld.o
  AR      samples/built-in.a
  AR      built-in.a
  AR      vmlinux.a
  LD      vmlinux.o
  OBJCOPY modules.builtin.modinfo
  GEN     modules.builtin
  MODPOST Module.symvers
  CC [M]  samples/rust/rust_helloworld.mod.o
  UPD     include/generated/utsversion.h
  CC      init/version-timestamp.o
  LD      .tmp_vmlinux.kallsyms1
  LD [M]  samples/rust/rust_helloworld.ko
  NM      .tmp_vmlinux.kallsyms1.syms
  KSYMS   .tmp_vmlinux.kallsyms1.S
  AS      .tmp_vmlinux.kallsyms1.S
  LD      .tmp_vmlinux.kallsyms2
  NM      .tmp_vmlinux.kallsyms2.syms
  KSYMS   .tmp_vmlinux.kallsyms2.S
  AS      .tmp_vmlinux.kallsyms2.S
  LD      vmlinux
  NM      System.map
  SORTTAB vmlinux
  CC      arch/x86/boot/version.o
  VOFFSET arch/x86/boot/compressed/../voffset.h
  OBJCOPY arch/x86/boot/compressed/vmlinux.bin
  RELOCS  arch/x86/boot/compressed/vmlinux.relocs
  CC      arch/x86/boot/compressed/kaslr.o
  GZIP    arch/x86/boot/compressed/vmlinux.bin.gz
  CC      arch/x86/boot/compressed/misc.o
  MKPIGGY arch/x86/boot/compressed/piggy.S
  AS      arch/x86/boot/compressed/piggy.o
  LD      arch/x86/boot/compressed/vmlinux
  ZOFFSET arch/x86/boot/zoffset.h
  OBJCOPY arch/x86/boot/vmlinux.bin
```

5. 将在`samples/rust`下看到一份`rust_helloworld.ko`的文件，将该文件复制到仓库中`src_e1000/rootfs`目录下，然后重新跑`build_image.sh`

```
[   14.651116] Freeing unused kernel image (text/rodata gap) memory: 2032K
[   14.652909] Freeing unused kernel image (rodata/data gap) memory: 840K
[   14.845650] x86/mm: Checked W+X mappings: passed, no W+X pages found.
[   14.846284] Run sbin/init as init process
[   14.901597] mount (72) used greatest stack depth: 14160 bytes left
[   15.087647] mdev (74) used greatest stack depth: 13960 bytes left

Please press Enter to activate this console.
~ # ls
bin               linuxrc           root              usr
dev               proc              sbin
etc               r4l_e1000_demo.ko  sys
~ # QEMU: Terminated
ljh@ljh-vm:~/linux/rust/cicv-r4l-3-LJHua-MyMan/src_e1000$ cd ../linux/samples/rust/
ljh@ljh-vm:~/linux/rust/cicv-r4l-3-LJHua-MyMan/linux/samples/rust$ ls
built-in.a      rust_echo_server.rs    rust_helloworld.o          rust_platform.rs    rust_stack_probing.rs
hostprogs       rust_fs.rs             rust_helloworld.rs         rust_print.rs       rust_sync.rs
Kconfig         rust_helloworld.ko     rust_minimal.rs            rust_random.rs
Makefile        rust_helloworld.mod    rust_miscdev.rs            rust_selftests.rs
modules.order   rust_helloworld.mod.c  rust_module_parameters.rs  rust_semaphore_c.c
rust_chrdev.rs  rust_helloworld.mod.o  rust_netfilter.rs          rust_semaphore.rs
ljh@ljh-vm:~/linux/rust/cicv-r4l-3-LJHua-MyMan/linux/samples/rust$ cp rust_helloworld.ko ../../../src_e1000/rootfs/
ljh@ljh-vm:~/linux/rust/cicv-r4l-3-LJHua-MyMan/linux/samples/rust$ cd ../../../src_e1000/rootfs/
ljh@ljh-vm:~/linux/rust/cicv-r4l-3-LJHua-MyMan/src_e1000/rootfs$ ls
bin  dev  etc  linuxrc  proc  r4l_e1000_demo.ko  rust_helloworld.ko  sbin  sys  usr
ljh@ljh-vm:~/linux/rust/cicv-r4l-3-LJHua-MyMan/src_e1000/rootfs$ cd ../
ljh@ljh-vm:~/linux/rust/cicv-r4l-3-LJHua-MyMan/src_e1000$ ./build_image.sh
make -C ../linux M=$PWD
make[1]: Entering directory '/home/ljh/linux/rust/cicv-r4l-3-LJHua-MyMan/linux'
make[1]: Leaving directory '/home/ljh/linux/rust/cicv-r4l-3-LJHua-MyMan/linux'

5321 blocks
SeaBIOS (version 1.15.0-1)


iPXE (https://ipxe.org) 00:03.0 CA00 PCI2.10 PnP PMM+07F8B340+07ECB340 CA00



Booting from ROM...
[    0.000000] Linux version 6.1.0-rc1 (ljh@ljh-vm) (Ubuntu clang version 14.0.0-1ubuntu1.1, Ubuntu LLD 14.0.0) #3 S4
```

6. 进入内核，并执行`insmod rust_helloworld.ko`

```
[    1.509452] Loading compiled-in X.509 certificates
[    1.515809] cryptomgr_test (44) used greatest stack depth: 15584 bytes left
[    1.522339] PM:   Magic number: 0:450:612
[    1.522751] tty tty: hash matches
[    1.523959] printk: console [netcon0] enabled
[    1.524253] netconsole: network logging started
[    1.593993] ata2: found unknown device (class 0)
[    1.598769] ata2.00: ATAPI: QEMU DVD-ROM, 2.5+, max UDMA/100
[    1.608180] scsi 1:0:0:0: CD-ROM            QEMU     QEMU DVD-ROM     2.5+ PQ: 0 ANSI: 5
[    1.646863] sr 1:0:0:0: [sr0] scsi3-mmc drive: 4x/4x cd/rw xa/form2 tray
[    1.647464] cdrom: Uniform CD-ROM driver Revision: 3.20
[    1.665531] sr 1:0:0:0: Attached scsi generic sg0 type 5
[    2.099062] input: ImExPS/2 Generic Explorer Mouse as /devices/platform/i8042/serio1/input/input3
[    2.225639] tsc: Refined TSC clocksource calibration: 2303.978 MHz
[    2.226067] clocksource: tsc: mask: 0xffffffffffffffff max_cycles: 0x2135e30d8d2, max_idle_ns: 440795222959 ns
[    2.226438] clocksource: Switched to clocksource tsc
[   14.385777] cfg80211: Loading compiled-in X.509 certificates for regulatory database
[   14.447034] modprobe (67) used greatest stack depth: 14272 bytes left
[   14.460525] cfg80211: Loaded X.509 cert 'sforshee: 00b28ddf47aef9cea7'
[   14.462541] platform regulatory.0: Direct firmware load for regulatory.db failed with error -2
[   14.463235] cfg80211: failed to load regulatory.db
[   14.465192] ALSA device list:
[   14.465464]   No soundcards found.
[   14.514622] Freeing unused kernel image (initmem) memory: 1324K
[   14.515446] Write protecting the kernel read-only data: 24576k
[   14.518766] Freeing unused kernel image (text/rodata gap) memory: 2032K
[   14.519895] Freeing unused kernel image (rodata/data gap) memory: 840K
[   14.700628] x86/mm: Checked W+X mappings: passed, no W+X pages found.
[   14.701498] Run sbin/init as init process
[   14.748189] mount (72) used greatest stack depth: 14160 bytes left
[   14.890514] mdev (74) used greatest stack depth: 13960 bytes left

Please press Enter to activate this console.
~ # ls
bin                proc               sbin
dev                r4l_e1000_demo.ko  sys
etc                root               usr
linuxrc            rust_helloworld.ko
~ # insmod rust_helloworld.ko
[   36.872710] rust_helloworld: Hello World from Rust module
~ #
```

作业5：注册字符设备

Q：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？

答：通过命令`mknod /dev/cicv c 248 0`可以创建一个设备文件，其中参数c表示为字符设备，248为主设备号，0为次设备号，设备驱动通过系统动态分配设备号的方式关联。

1. 修改`Linux/samples/rust/rust_chrdev.rs`文件，具体如下：

```
    fn open(_shared: &(), _file: &file::File) -> Result<Box<Self>> {
        Ok(
            Box::try_new(RustFile {
                inner: &GLOBALMEM_BUF
            })?
        )
    }

    fn write(
        this: &Self,
        _file: &file::File,
        reader: &mut impl kernel::io_buffer::IoBufferReader,
        offset: u64,
    ) -> Result<usize> {
        let offset = offset.try_into()?;
        let mut vec = this.inner.lock();
        let len = core::cmp::min(reader.len(), vec.len().saturating_sub(offset));
        reader.read_slice(&mut vec[offset..][..len])?;
        Ok(len)
    }

    fn read(
        this: &Self,
        _file: &file::File,
        writer: &mut impl kernel::io_buffer::IoBufferWriter,
        offset: u64,
    ) -> Result<usize> {
        let offset = offset.try_into()?;
        let vec = this.inner.lock();
        let len = core::cmp::min(writer.len(), vec.len().saturating_sub(offset));
        writer.write_slice(&vec[offset..][..len])?;
        Ok(len)
    }
}
struct RustChrdev {
    _dev: Pin<Box<chrdev::Registration<2>>>,
}

impl kernel::Module for RustChrdev {
                                                                38,1          61%
```

2. 重新编译内核，并如下更新配置

```
Kernel hacking
  ---> Sample Kernel code
      ---> Rust samples
              ---> <*>Character device (NEW)
```

3. 进入内核进行测试，测试结果如下：