

# 作业1

## 作业1 编译Linux内核



```
ywf@DESKTOP-L1MNRL0: ~/dev-place/rust/cicv-r4l-3-yiwufen/linux ×
nent add rustfmt
rustup component add clippy
info: component 'rustfmt' for target 'x86_64-unknown-linux-gnu' is up to date
info: component 'clippy' for target 'x86_64-unknown-linux-gnu' is up to date
(base) ywf@DESKTOP-L1MNRL0:~/dev-place/rust/cicv-r4l-3-yiwufen/linux$ make LLVM=1
Rust available
Rust is available!
(base) ywf@DESKTOP-L1MNRL0:~/dev-place/rust/cicv-r4l-3-yiwufen/linux$ make x86_64_
defconfig
HOSTCC scripts/basic/fixdep
HOSTCC scripts/kconfig/conf.o
HOSTCC scripts/kconfig/confdata.o
HOSTCC scripts/kconfig/expr.o
LEX scripts/kconfig/lexer.lex.c
YACC scripts/kconfig/parser.tab.[ch]
HOSTCC scripts/kconfig/lexer.lex.o
HOSTCC scripts/kconfig/menu.o
HOSTCC scripts/kconfig/parser.tab.o
HOSTCC scripts/kconfig/preprocess.o
HOSTCC scripts/kconfig/symbol.o
HOSTCC scripts/kconfig/util.o
HOSTLD scripts/kconfig/conf
#
# configuration written to .config
#
```



```
ywf@DESKTOP-L1MNRLO: ~/dev-place/rust/cicv-r4l-3-yiwufen/linux ×
#
# end of Rust hacking
# end of Kernel hacking
(base) ywf@DESKTOP-L1MNRLO:~/dev-place/rust/cicv-r4l-3-yiwufen/linux$ make LLVM=1
menuconfig
HOSTCC scripts/basic/fixdep
HOSTCC scripts/kconfig/confdata.o
HOSTCC scripts/kconfig/expr.o
HOSTCC scripts/kconfig/lexer.lex.o
HOSTCC scripts/kconfig/menu.o
HOSTCC scripts/kconfig/parser.tab.o
HOSTCC scripts/kconfig/preprocess.o
HOSTCC scripts/kconfig/symbol.o
HOSTCC scripts/kconfig/util.o
UPD scripts/kconfig/mconf-cfg
HOSTCC scripts/kconfig/mconf.o
HOSTCC scripts/kconfig/lxdialog/checklist.o
HOSTCC scripts/kconfig/lxdialog/inputbox.o
HOSTCC scripts/kconfig/lxdialog/menubox.o
HOSTCC scripts/kconfig/lxdialog/textbox.o
HOSTCC scripts/kconfig/lxdialog/util.o
HOSTCC scripts/kconfig/lxdialog/yesno.o
HOSTLD scripts/kconfig/mconf
configuration written to .config

*** End of the configuration.
*** Execute 'make' to start the build or try 'make help'.
```



```
(base) ywf@DESKTOP-L1MNRLO:~/dev-place/rust/cicv-r4l-3-yiwufen/linux$ make LLVM=1
-j$(nproc)
SYNC      include/config/auto.conf.cmd
HOSTCC    scripts/kconfig/conf.o
HOSTLD    scripts/kconfig/conf
SYSHDR    arch/x86/include/generated/uapi/asm/unistd_32.h
SYSHDR    arch/x86/include/generated/uapi/asm/unistd_64.h
SYSHDR    arch/x86/include/generated/uapi/asm/unistd_x32.h
SYSTBL    arch/x86/include/generated/asm/syscalls_32.h
SYSHDR    arch/x86/include/generated/asm/unistd_32_ia32.h
SYSHDR    arch/x86/include/generated/asm/unistd_64_x32.h
SYSTBL    arch/x86/include/generated/asm/syscalls_64.h
WRAP      arch/x86/include/generated/uapi/asm/bpf_perf_event.h
WRAP      arch/x86/include/generated/uapi/asm/errno.h
WRAP      arch/x86/include/generated/uapi/asm/fcntl.h
WRAP      arch/x86/include/generated/uapi/asm/ioctl.h
WRAP      arch/x86/include/generated/uapi/asm/ioctls.h
WRAP      arch/x86/include/generated/uapi/asm/ipcbuf.h
WRAP      arch/x86/include/generated/uapi/asm/param.h
WRAP      arch/x86/include/generated/uapi/asm/poll.h
WRAP      arch/x86/include/generated/uapi/asm/resource.h
WRAP      arch/x86/include/generated/uapi/asm/socket.h
WRAP      arch/x86/include/generated/uapi/asm/sockios.h
WRAP      arch/x86/include/generated/uapi/asm/termbits.h
```



```
ywf@DESKTOP-L1MNRLO: ~/dev-place/rust/cicv-r4l-3-yiwufen/linux
OBJCOPY   arch/x86/boot/compressed/vmlinux.bin
RELOCS    arch/x86/boot/compressed/vmlinux.relocs
HOSTCC    arch/x86/boot/compressed/mkpiggy
CC        arch/x86/boot/compressed/cpuflags.o
CC        arch/x86/boot/compressed/early_serial_console.o
CC        arch/x86/boot/compressed/kaslr.o
CC        arch/x86/boot/compressed/ident_map_64.o
CC        arch/x86/boot/compressed/idt_64.o
AS        arch/x86/boot/compressed/idt_handlers_64.o
AS        arch/x86/boot/compressed/mem_encrypt.o
CC        arch/x86/boot/compressed/pgtable_64.o
CC        arch/x86/boot/compressed/acpi.o
AS        arch/x86/boot/compressed/efi_thunk_64.o
CC        arch/x86/boot/compressed/efi.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
AS        arch/x86/boot/header.o
LD        arch/x86/boot/setup.elf
OBJCOPY   arch/x86/boot/setup.bin
BUILD     arch/x86/boot/bzImage
Kernel: arch/x86/boot/bzImage is ready (#1)
```



```
(base) ywf@DESKTOP-L1MNRL0:~/dev-place/rust/cicv-r4l-3-yiwufen/linux$ ls
COPYING      README      fs          modules.builtin.modinfo  usr
CREDITS      README.md   include     modules.order            virt
Documentation System.map  init        net                      vmlinux
Kbuild       arch        io_uring    rust                     vmlinux.a
Kconfig      block       ipc         samples                  vmlinux.o
LICENSES     built-in.a kernel      scripts
MAINTAINERS  certs       lib         security
Makefile     crypto      mm          sound
Module.symvers drivers     modules.builtin  tools
(base) ywf@DESKTOP-L1MNRL0:~/dev-place/rust/cicv-r4l-3-yiwufen/linux$
```

## 作业2

编译驱动进内核



```
ywf@DESKTOP-L1MNRL0: ~/dev-place/rust/cicv-r4l-3-yiwufen/src_e1000 × + ▾ - □ ×
Kbuild      README.md      dump.dat      r4l_e1000_demo.rs
LICENSE     build_image.sh  e1000_ops.rs ring_buf.rs
Makefile    consts.rs      hw_defs.rs
(base) ywf@DESKTOP-L1MNRL0:~/dev-place/rust/cicv-r4l-3-yiwufen/src_e1000$ make LLVM=1
make -C ../linux M=$PWD
make[1]: Entering directory '/home/ywf/dev-place/rust/cicv-r4l-3-yiwufen/linux'
  RUSTC [M] /home/ywf/dev-place/rust/cicv-r4l-3-yiwufen/src_e1000/r4l_e1000_demo.o
  MODPOST /home/ywf/dev-place/rust/cicv-r4l-3-yiwufen/src_e1000/Module.symvers
  CC [M] /home/ywf/dev-place/rust/cicv-r4l-3-yiwufen/src_e1000/r4l_e1000_demo.mod.o
  LD [M] /home/ywf/dev-place/rust/cicv-r4l-3-yiwufen/src_e1000/r4l_e1000_demo.ko
make[1]: Leaving directory '/home/ywf/dev-place/rust/cicv-r4l-3-yiwufen/linux'
(base) ywf@DESKTOP-L1MNRL0:~/dev-place/rust/cicv-r4l-3-yiwufen/src_e1000$ ls
Kbuild      consts.rs      r4l_e1000_demo.mod
LICENSE     dump.dat      r4l_e1000_demo.mod.c
Makefile    e1000_ops.rs  r4l_e1000_demo.mod.o
Module.symvers hw_defs.rs    r4l_e1000_demo.o
README.md   modules.order r4l_e1000_demo.rs
build_image.sh r4l_e1000_demo.ko ring_buf.rs
(base) ywf@DESKTOP-L1MNRL0:~/dev-place/rust/cicv-r4l-3-yiwufen/src_e1000$ cat Module.symvers
(base) ywf@DESKTOP-L1MNRL0:~/dev-place/rust/cicv-r4l-3-yiwufen/src_e1000$
```

Q1

有kbuild 文件中的 obj-m := r4l\_e1000\_demo.o 定义的

Q2

通过Makefile文件，这个文件执行命令让其跳转到了linux目录下进行编译。

允许build\_image.sh

进入linux



```
ywf@DESKTOP-L1MNRLO: ~/dev-place/rust/cicv-r4l-  
[ 2.072764] clocksource: tsc: mask: 0xffffffffffffffff ms  
[ 2.076700] clocksource: Switched to clocksource tsc  
[ 2.461250] input: ImExPS/2 Generic Explorer Mouse as /d3  
[ 2.498704] e1000: eth0 NIC Link is Up 1000 Mbps Full DuX  
[ 2.501484] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link bey  
[ 2.513881] IP-Config: Complete:  
[ 2.514019]     device=eth0, hwaddr=52:54:00:12:34:56, 1  
[ 2.514539]     host=10.0.2.15, domain=, nis-domain=(no)  
[ 2.515145]     bootserver=255.255.255.255, rootserver==  
[ 2.519325] cfg80211: Loading compiled-in X.509 certifice  
[ 2.571915] modprobe (66) used greatest stack depth: 142t  
[ 2.584850] cfg80211: Loaded X.509 cert 'sforshee: 00b28'  
[ 2.586531] platform regulatory.0: Direct firmware load 2  
[ 2.587056] cfg80211: failed to load regulatory.db  
[ 2.588340] ALSA device list:  
[ 2.588585]     No soundcards found.  
[ 2.634216] Freeing unused kernel image (initmem) memoryK  
[ 2.634923] Write protecting the kernel read-only data: k  
[ 2.638009] Freeing unused kernel image (text/rodata gapK  
[ 2.638984] Freeing unused kernel image (rodata/data gapK  
[ 2.801801] x86/mm: Checked W+X mappings: passed, no W+X.  
[ 2.802357] Run sbin/init as init process  
[ 2.840349] mount (71) used greatest stack depth: 14160 t  
[ 2.968313] mdev (73) used greatest stack depth: 13960 bt  
  
Please press Enter to activate this console.  
~ # █
```

安装内核模块



```
~ # insmod r4l_e1000_demo.ko  
[ 209.633129] r4l_e1000_demo: loading out-of-tree module t.  
[ 209.639984] r4l_e1000_demo: Rust for linux e1000 driver )  
[ 209.641654] insmod (80) used greatest stack depth: 13072t  
~ # █
```

查看模块



```
~ # lsmod | grep r4l_e1000_demo
r4l_e1000_demo 40960 0 - Live 0xfffffffffc022b000 (0)
~ #
```

查看网络设备，并执行ping操作



```
~ # 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: fe80::5054:ff:fe12:3456/64  Scope:Link
          inet6 addr: fec0::5054:ff:fe12:3456/64  Scope:Site
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:3  errors:0  dropped:0  overruns:0  frame:0
          TX packets:8  errors:0  dropped:0  overruns:0  carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:330 (330.0 B)  TX bytes:672 (672.0 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 10.0.2.15
PING 10.0.2.15 (10.0.2.15): 56 data bytes
64 bytes from 10.0.2.15: seq=0 ttl=64 time=7.267 ms
64 bytes from 10.0.2.15: seq=1 ttl=64 time=1.194 ms
64 bytes from 10.0.2.15: seq=2 ttl=64 time=1.040 ms
64 bytes from 10.0.2.15: seq=3 ttl=64 time=0.361 ms
64 bytes from 10.0.2.15: seq=4 ttl=64 time=0.999 ms
```

禁用默认的C版本的e1000网卡驱动





```
.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 submenus ----). Highlighted letters are hotkeys. Pressing
<Y> includes, <N> excludes, <M> modularizes features. Press
<Esc><Esc> to exit, <?> for Help, </> for Search. Legend: [*]

^(-)
< >   Huawei Intelligent PCIE Network Interface Card
[*]   Intel (82586/82593/82596) devices
[*]   Intel devices
<*>   Intel(R) PRO/100+ support
<Y>   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 supp
< >   Intel(R) 82576 Virtual Function Ethernet support
< >   Intel(R) PRO/10GbE support
< >   Intel(R) 10GbE PCI Express adapters support
< >   Intel(R) 10GbE PCI Express Virtual Function Ethernet s
< >   Intel(R) Ethernet Controller XL710 Family support
v(+)

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

重新编译 `make LLVM=1 -j$(nproc)`





```
(base) ywf@DESKTOP-L1MNRLO:~/dev-place/rust/cicv-r4l-3-yiwufen/linux$ make LL
VM=1 -j$(nproc)
SYNC    include/config/auto.conf.cmd
HOSTCC  scripts/kconfig/conf.o
HOSTLD  scripts/kconfig/conf
DESCEND objtool
CALL    scripts/checksyscalls.sh
RUSTC L rust/bindings.o
EXPORTS rust/exports_bindings_generated.h
RUSTC L rust/kernel.o
EXPORTS rust/exports_kernel_generated.h
CC      rust/exports.o
AR      drivers/net/ethernet/intel/built-in.a
AR      drivers/net/ethernet/built-in.a
AR      drivers/net/built-in.a
AR      rust/built-in.a
AR      drivers/built-in.a
AR      built-in.a
AR      vmlinux.a
LD      vmlinux.o
OBJCOPY modules.builtin.modinfo
GEN      modules.builtin
MODPOST Module.symvers
UPD      include/generated/utsversion.h
CC      init/version-timestamp.o
LD      .tmp_vmlinux.kallsyms1
NM      .tmp_vmlinux.kallsyms1.syms
```

重新插入模块



```
~ # insmod r4l_e1000_demo.ko
[ 51.610908] r4l_e1000_demo: loading out-of-tree module taints kernel.
[ 51.617593] r4l_e1000_demo: Rust for linux e1000 driver demo (init)
[ 51.618126] r4l_e1000_demo: Rust for linux e1000 driver demo (probe): None
[ 51.800266] ACPI: \_SB_.LNKC: Enabled at IRQ 11
[ 51.826182] r4l_e1000_demo: Rust for linux e1000 driver demo (net device )
[ 51.830664] insmod (81) used greatest stack depth: 11192 bytes left
~ #
```

`ip link set eth0 ip` 启动eth0 网络接口



```

~ # ip link set eth0 up
[ 132.272775] r4l_e1000_demo: Rust for linux e1000 driver demo (net device )
[ 132.276129] r4l_e1000_demo: Rust for linux e1000 driver demo (net device )
[ 132.277905] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[ 132.281345] r4l_e1000_demo: Rust for linux e1000 driver demo (net device )
~ # [ 132.289867] r4l_e1000_demo: Rust for linux e1000 driver demo (net dev0
[ 132.291039] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[ 132.291267] r4l_e1000_demo: pending_irqs: 3
[ 132.292217] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[ 132.598149] r4l_e1000_demo: Rust for linux e1000 driver demo (net device 0
[ 132.600906] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[ 132.602120] r4l_e1000_demo: pending_irqs: 3
[ 132.603000] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[ 132.676905] r4l_e1000_demo: Rust for linux e1000 driver demo (net device 0
[ 132.677753] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[ 132.678615] r4l_e1000_demo: pending_irqs: 3
[ 132.678962] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[ 133.737343] r4l_e1000_demo: Rust for linux e1000 driver demo (net device 0
[ 133.738039] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[ 133.738381] r4l_e1000_demo: pending_irqs: 3
[ 133.738631] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[ 133.739185] r4l_e1000_demo: Rust for linux e1000 driver demo (net device 0

```

配置网络



```

~ # ip addr add broadcast 10.0.2.255 dev eth0
[ 1619.989917] r4l_e1000_demo: Rust for linux e1000 driver demo (net device )
[ 1619.992689] r4l_e1000_demo: Rust for linux e1000 driver demo (net device )
ip: RTNETLINK answers: Invalid argument
~ # ifconfig
[ 1629.763454] r4l_e1000_demo: Rust for linux e1000 driver demo (net device )
eth0      Link encap:Ethernet  HWaddr 52:54:00:12:34:56
          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 addr add 10.0.2.15/255.255.255.0 dev eth0
[ 1652.514474] r4l_e1000_demo: Rust for linux e1000 driver demo (net device )
[ 1652.514825] r4l_e1000_demo: Rust for linux e1000 driver demo (net device )
~ # ip route add default via 10.0.2.1
~ # if
ifconfig  ifdown      ifenslave  ifplugd   ifup
~ # ifconfig
[ 1669.691714] r4l_e1000_demo: Rust for linux e1000 driver demo (net device )
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

```



```
ping 10.0.2.2
```

```

~ # ping 10.0.2.2
PING 10.0.2.2 (10.0.2.2): 56 data bytes
[ 1740.813237] r4l_e1000_demo: Rust for linux e1000 driver demo (net device 0
[ 1740.814114] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[ 1740.814944] r4l_e1000_demo: pending_irqs: 131
[ 1740.815384] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[ 1740.817983] r4l_e1000_demo: Rust for linux e1000 driver demo (net device 1
[ 1740.818490] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[ 1740.819161] r4l_e1000_demo: pending_irqs: 131
[ 1740.820150] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=0 ttl=255 time=14.819 ms
[ 1741.825004] r4l_e1000_demo: Rust for linux e1000 driver demo (net device 2
[ 1741.825754] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[ 1741.826227] r4l_e1000_demo: pending_irqs: 131
[ 1741.826988] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=1 ttl=255 time=3.469 ms
[ 1742.830100] r4l_e1000_demo: Rust for linux e1000 driver demo (net device 3
[ 1742.833401] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[ 1742.835248] r4l_e1000_demo: pending_irqs: 131
[ 1742.836060] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=2 ttl=255 time=6.851 ms
[ 1743.839269] r4l_e1000_demo: Rust for linux e1000 driver demo (net device 4
[ 1743.841730] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[ 1743.842664] r4l_e1000_demo: pending_irqs: 131
[ 1743.842938] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)

```

## 作业3：

创建并编辑 `rust_helloworld.rs`

```

home > ywf > dev-place > rust > cicv-r4l-3-yiwufen > linux > samples > rust > rust_helloworld.rs > ...
1  // SPDX-License-Identifier: GPT-2.0
2  // ! Rust minimal sample.
3
4  use kernel::prelude::*;
5
6  module! {
7      type: RustHelloWorld,
8      name: "rust_helloworld",
9      author: "whocare",
10     description: "hello world module in rust",
11     license: "GPL",
12 }
13
14 0 implementations
15 struct RustHelloWorld {}
16
17 impl kernel::Module for RustHelloWorld {
18     fn init(_name: &'static Cstr, _module: &'static ThisModule) -> Result<Self> {
19         pr_info!("Hello, world! from Rust module\n");
20         Ok(RustHelloWorld {})
21     }
22 }

```

在Kconfig中添加配置

```

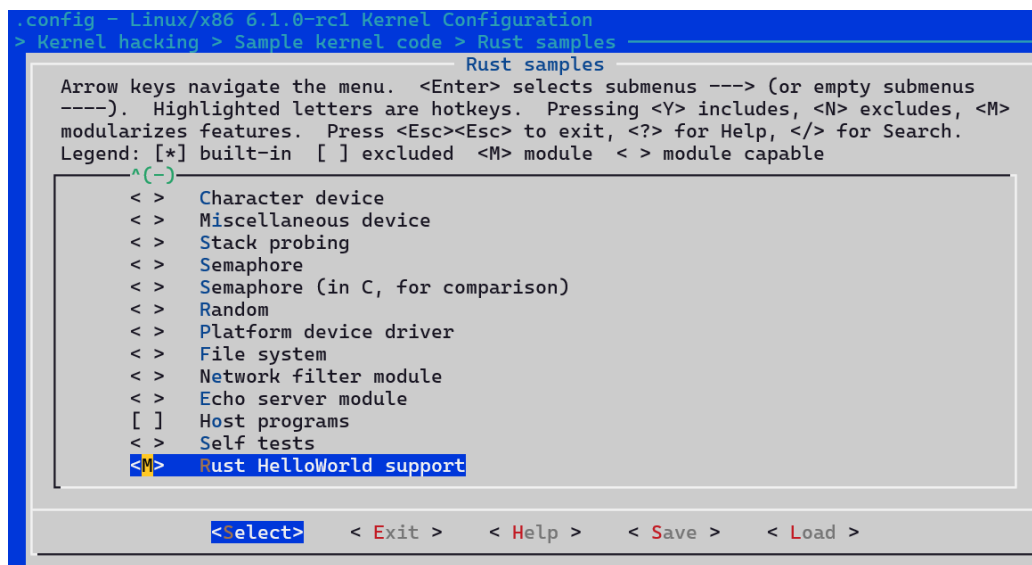
165
166 # new config
167 config RUST_HELLOWORLD
168     tristate "Rust HelloWorld support"
169     help
170         Choose this option to enable the Rust HelloWorld demo.
171         This demo shows how to integrate Rust code into the Linux
172
173     If unsure, say N.
174

```

在Makefile添加编译方式以及配置为模块

```
3
9  # 当 RUST_HELLOWORLD 被配置为模块 (m) 时
0  obj-$(CONFIG_RUST_HELLOWORLD) += rust_helloworld.o
1
```

执行 `make LLVM=1 menuconfig` 添加配置



编译内核 `make LLVM=1 -j$(nproc)`

```
(base) ywf@DESKTOP-L1MNRLO:~/dev-place/rust/cicv-r4l-3-yiwufen/linux$ make LLVM=1 -j$(nproc)
DESCEND objtool
CALL scripts/checksyscalls.sh
RUSTC [M] samples/rust/rust_helloworld.o
MODPOST Module.symvers
CC [M] samples/rust/rust_helloworld.mod.o
LD [M] samples/rust/rust_helloworld.ko
Kernel: arch/x86/boot/bzImage is ready (#3)
```

运行 `src_e1000/build_image.sh`

```

[ 14.517933] cfg80211: Loading compiled-in X.509 certificates for regulatory da
[ 14.572352] modprobe (67) used greatest stack depth: 14272 bytes left
[ 14.584742] cfg80211: Loaded X.509 cert 'sforshee: 00b28ddf47aef9cea7'
[ 14.586314] platform regulatory.0: Direct firmware load for regulatory.db fail
[ 14.587056] cfg80211: failed to load regulatory.db
[ 14.588494] ALSA device list:
[ 14.588923]   No soundcards found.
[ 14.634923] Freeing unused kernel image (initmem) memory: 1324K
[ 14.635505] Write protecting the kernel read-only data: 24576k
[ 14.638769] Freeing unused kernel image (text/rodata gap) memory: 2032K
[ 14.639597] Freeing unused kernel image (rodata/data gap) memory: 840K
[ 14.800760] x86/mm: Checked W+X mappings: passed, no W+X pages found.
[ 14.801205] Run sbin/init as init process
[ 14.836126] mount (72) used greatest stack depth: 14160 bytes left
[ 14.962391] 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
[ 19.920902] ls (80) used greatest stack depth: 13920 bytes left
~ # ls
bin                linuxrc            root               usr
dev                proc              sbin
etc                r4l_e1000_demo.ko sys
~ #

```

查看rust\_helloworld.ko的文件

```

(base) ywf@DESKTOP-L1MNRLO:~/dev-place/rust/cicv-r4l-3-yiwufen/linux/sam
ples/rust$ ls
Kconfig            rust_helloworld.mod      rust_platform.rs
Makefile           rust_helloworld.mod.c    rust_print.rs
built-in.a         rust_helloworld.mod.o    rust_random.rs
hostprogs          rust_helloworld.o        rust_selftests.rs
modules.order      rust_helloworld.rs       rust_semaphore.rs
rust_chrdev.rs     rust_minimal.rs          rust_semaphore_c.c
rust_echo_server.rs rust_miscdev.rs           rust_stack_probing.rs
rust_fs.rs         rust_module_parameters.rs rust_sync.rs
rust_helloworld.ko rust_netfilter.rs

```

将该文件复制到仓库中src\_e1000/rootfs目录下，然后重新跑build\_image.sh， 查看rust\_helloworld.ko文件

```

~ # ls
bin                proc              sbin
dev                r4l_e1000_demo.ko sys
etc                root              usr
linuxrc           rust_helloworld.ko

```

insmod命令进行安装该模块rust\_helloworld.ko



```
~ # insmod rust_helloworld.ko  
[ 373.981198] rust_helloworld: Hello World from Rust module
```

## 作业4：为e1000网卡驱动添加remove代码

添加代码

```
fn remove(data: &Self::Data) {  
    pr_info!("Rust for linux e1000 driver demo (remove)\n");  
    data.device_remove();  
}
```

```
impl driver::DeviceRemoval for E1000DrvPrvData {  
    fn device_remove(&self) {  
        pr_info!("Rust for linux e1000 driver demo (device_remo  
  
        // 执行任何必要的清理或关闭操作  
        // 此函数在设备被移除或卸载时调用  
        self._netdev_reg.dev_get().netif_carrier_off();  
        self._netdev_reg.dev_get().netif_stop_queue();  
    }  
}
```

编译代码，配置网卡，并ping通

```
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
```

```

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
[ 3340.321225] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=1, tdh=1, rdt=7
[ 3340.322847] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[ 3340.323431] r4l_e1000_demo: pending_irqs: 131
[ 3340.324219] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[ 3340.326603] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=2, tdh=2, rdt=0
[ 3340.327398] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[ 3340.327610] r4l_e1000_demo: pending_irqs: 131
[ 3340.328665] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=0 ttl=255 time=15.327 ms
[ 3341.332649] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=3, tdh=3, rdt=1
[ 3341.333109] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[ 3341.333493] r4l_e1000_demo: pending_irqs: 131
[ 3341.333805] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=1 ttl=255 time=1.962 ms
[ 3342.335392] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=4, tdh=4, rdt=2
[ 3342.336084] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[ 3342.336495] r4l_e1000_demo: pending_irqs: 131
[ 3342.336793] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=2 ttl=255 time=2.388 ms
^C
--- 10.0.2.2 ping statistics ---
3 packets transmitted, 3 packets received, 0% packet loss
round-trip min/avg/max = 1.962/6.559/15.327 ms

```

rmmod

```

~ # rmmod r4l_e1000_demo.ko
[ 3452.467045] r4l_e1000_demo: Rust for linux e1000 driver demo (exit)
[ 3452.467603] r4l_e1000_demo: Rust for linux e1000 driver demo (remove)
[ 3452.467750] r4l_e1000_demo: Rust for linux e1000 driver demo (device_remove)
[ 3452.468043] r4l_e1000_demo: Rust for linux e1000 driver demo (device_remove)
[ 3452.470415] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
[ 3452.471103] r4l_e1000_demo: Rust for linux e1000 driver demo (net device stop)
[ 3452.471346] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
[ 3452.479703] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
~ #

```

## 作业5

完善write、read 函数

```

/// 写入函数, 用于将数据写入设备
fn write(_this: &Self, _file: &file::File, _reader: &mut impl kernel::io_buffer::IoBufferReader, _offset: u64) -> Result<usize> {
    let mut buf: Guard<'static, Mutex<[u8; 4096]>, ...> = _this.inner.lock();
    let len: usize = _reader.len();

    // 根据偏移量调整写入位置
    let offset: usize = _offset as usize;
    let len: usize = core::cmp::min(v1: len, v2: GLOBALMEM_SIZE - offset);
    let data: <Result<Vec<u8>, Error> as Try>::Output = _reader.read_all()?;
    buf[offset..offset + len].copy_from_slice(&data[..len]);
    Ok(len)
}

fn read(_this: &Self, file: &file::File, writer: &mut impl kernel::io_buffer::IoBufferWriter, _offset: u64,) -> Result<usize> {
    let buf: Guard<'static, Mutex<[u8; 4096]>, ...> = _this.inner.lock();
    // 计算可从偏移量开始读取的最大字节数
    let max_readable: usize = GLOBALMEM_SIZE - _offset as usize;
    // 获取实际要读取的数据切片
    let data_to_read: &{unknown} as Index<Range<...>>::Output = &buf[_offset as usize.._offset as usize + max_readable];
    writer.write_slice(data_to_read)?;
    Ok(data_to_read.len())
}
} impl Operations for RustFile

```

## 添加配置

Arrow keys navigate the menu. <Enter> selects submenus ---> (or empty submenus ----). 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

```

--- Rust samples
< > Minimal
< > Printing macros
< > Module parameters
< > Synchronisation primitives
[*] Character device
< > Miscellaneous device
< > Stack probing
< > Semaphore
< > Semaphore (in C, for comparison)
< > Random
< > Platform device driver
< > File system
< > Network filter module
< > Echo server module
v(+)

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

```

## 编译内核并进入系统

## 执行测试命令

```

~ # echo "hi boy" > /dev/cicv
~ # cat /dev/cicv
hi boy
~ # █

```

**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？**

▼ 通过字符设备的创建是通过 `chrdev::Registration` 结构体实现的。这个结构体提供了一个简化的接口来注册和注销字符设备。在 `new_pinned` 函数调用时，第二个参数是期望的主设备号。在示例中，这个参数被设置为0，这意味着主设备号是由内核动态分配的。这是推荐的做法，因为它可以避免设备号冲突。

设备与其驱动程序的关联是通过设备的主设备号完成的。当内核模块使用 `chrdev::Registration` 注册一个设备时，代码中提供的设备类型（在这个例子中是 `RustFile`）和设备名（`name` 参数）会被用来处理对应设备文件的IO操作。内核根据设备文件的主设备号调用注册到该号码的驱动程序代码。

在提供的代码片段中，通过调用 `chrdev_reg.as_mut().register:::<RustFile>()?;` 两次，注册了两个相同类型的设备实例，它们将共享相同的主设备号但拥有不同的次设备号。