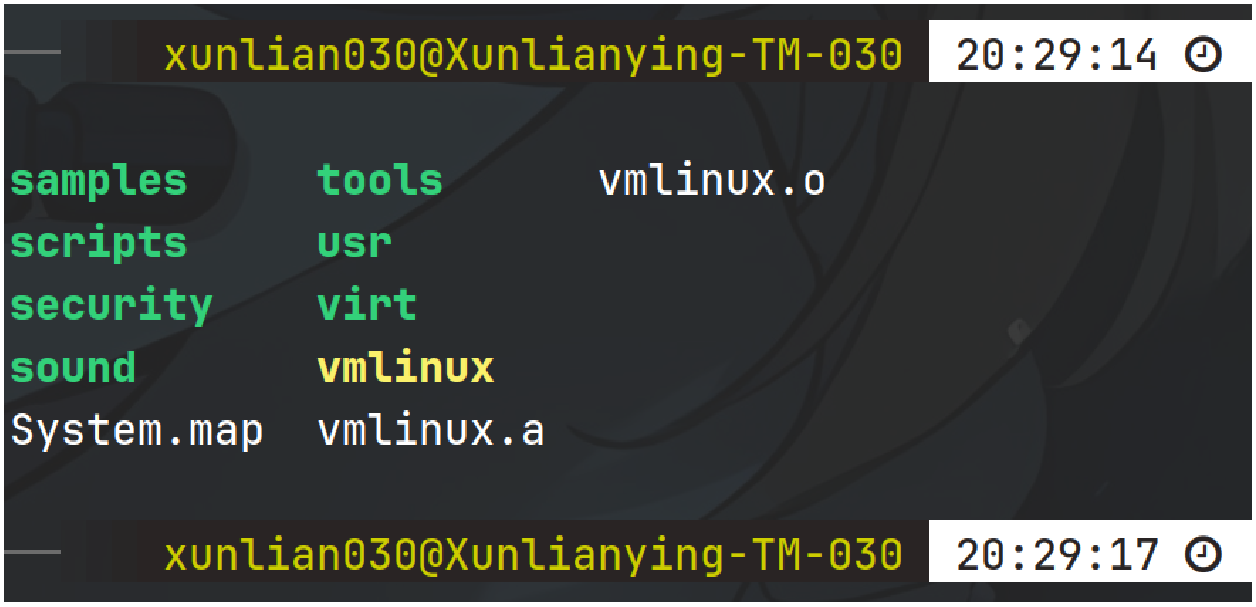
Rust for Linux

编译Linux内核

作业1结果



```
  xunlian030@Xunlianying-TM-030  20:29:14

samples      tools      vmlinux.o
scripts      usr
security     virt
sound        vmlinux
System.map   vmlinux.a

  xunlian030@Xunlianying-TM-030  20:29:17
```

网卡模块

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:

编译成内核模块，是在哪个文件中以哪条语句定义的?

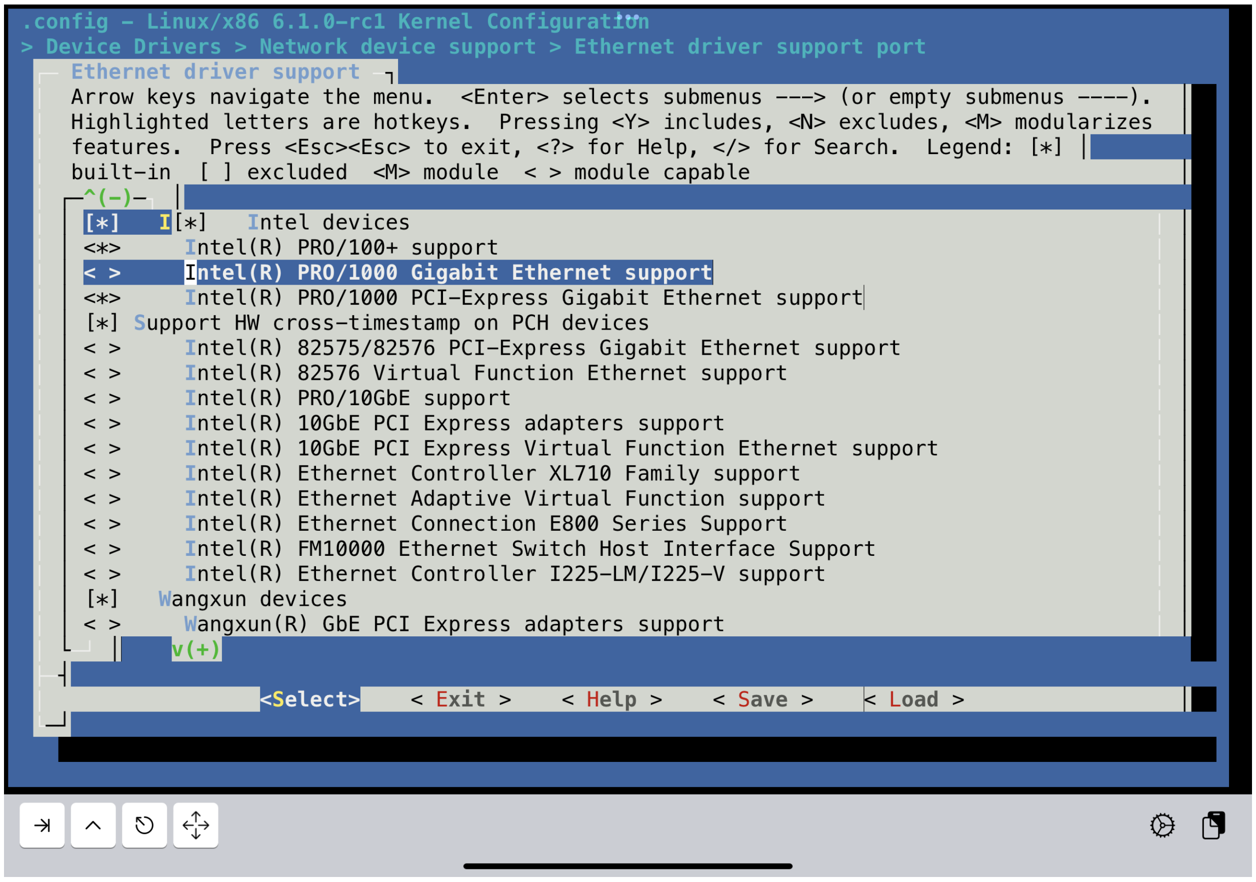
答: 该目录下, `Kbuild` 文件中的

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: 通过 `insmod` 加载到内核中

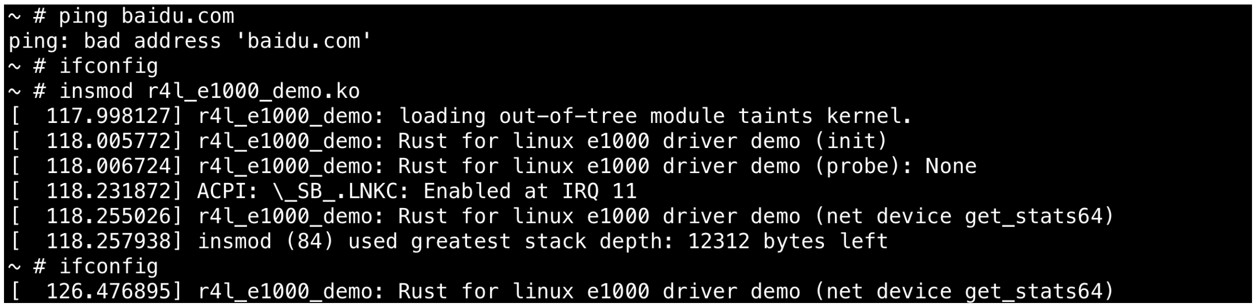
完成步骤:

关闭 e1000 网卡的默认 C 语言驱动 在重新编译内核
配置路径 Device Drivers > Network device support > Ethernet driver support > Intel devices, Intel(R) PRO/1000 Gigabit Ethernet support

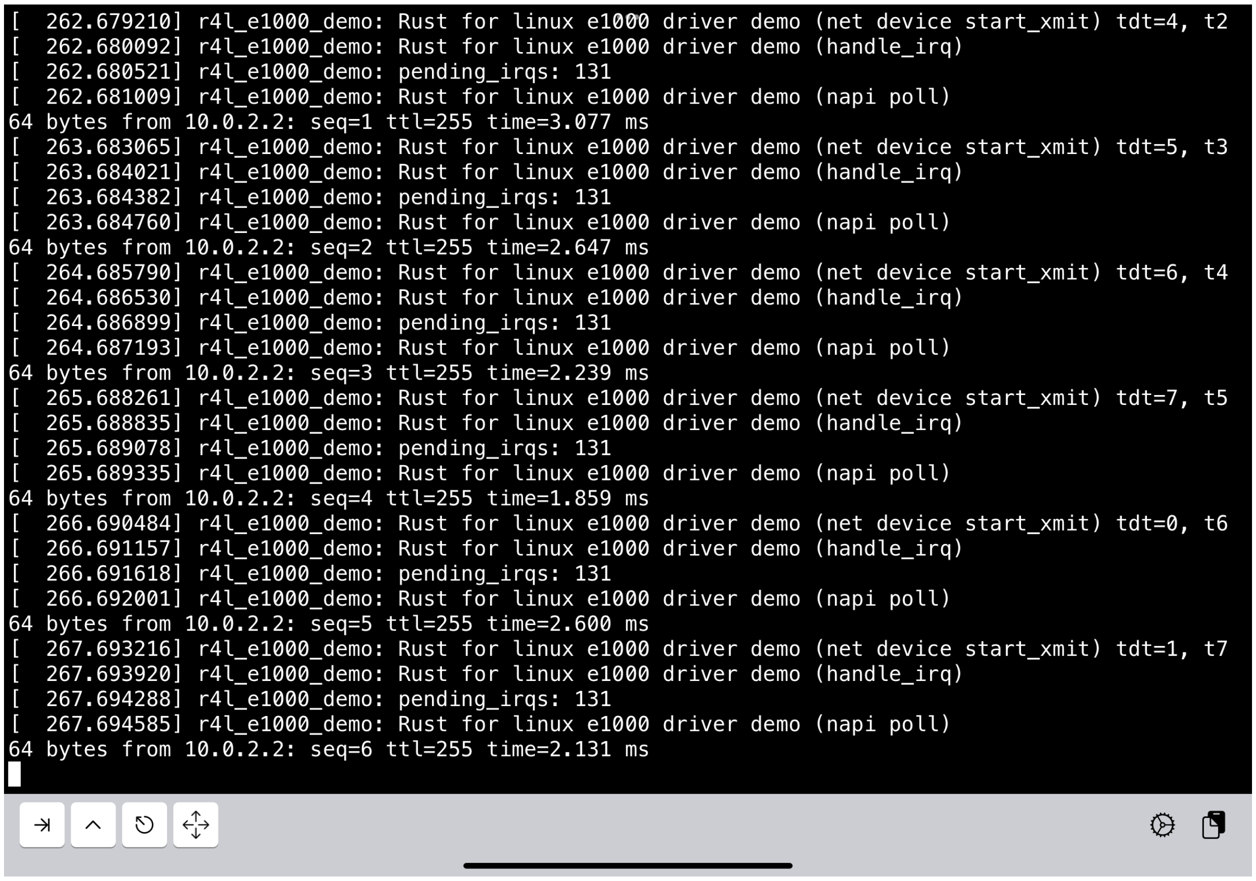


```
.config - Linux/x86 6.1.0-rc1 Kernel Configuration
> Device Drivers > Network device support > Ethernet driver support port
  Ethernet driver support
  Arrow keys navigate the menu.  <Enter> selects submenus ---> (or empty submenus ----).
  Highlighted letters are hotkeys.  Pressing <Y> includes, <N> excludes, <M> modularizes
  features.  Press <Esc><Esc> to exit, <?> for Help, </> for Search.  Legend: [*]
  built-in  [ ] excluded  <M> module  < > module capable
   ^(-)
    [*]    I[*]   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
    < >       Intel(R) 10GbE PCI Express Virtual Function Ethernet support
    < >       Intel(R) Ethernet Controller XL710 Family support
    < >       Intel(R) Ethernet Adaptive Virtual Function support
    < >       Intel(R) Ethernet Connection E800 Series Support
    < >       Intel(R) FM10000 Ethernet Switch Host Interface Support
    < >       Intel(R) Ethernet Controller I225-LM/I225-V support
    [*]     Wangxun devices
    < >       Wangxun(R) GbE PCI Express adapters support
          v(+)

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

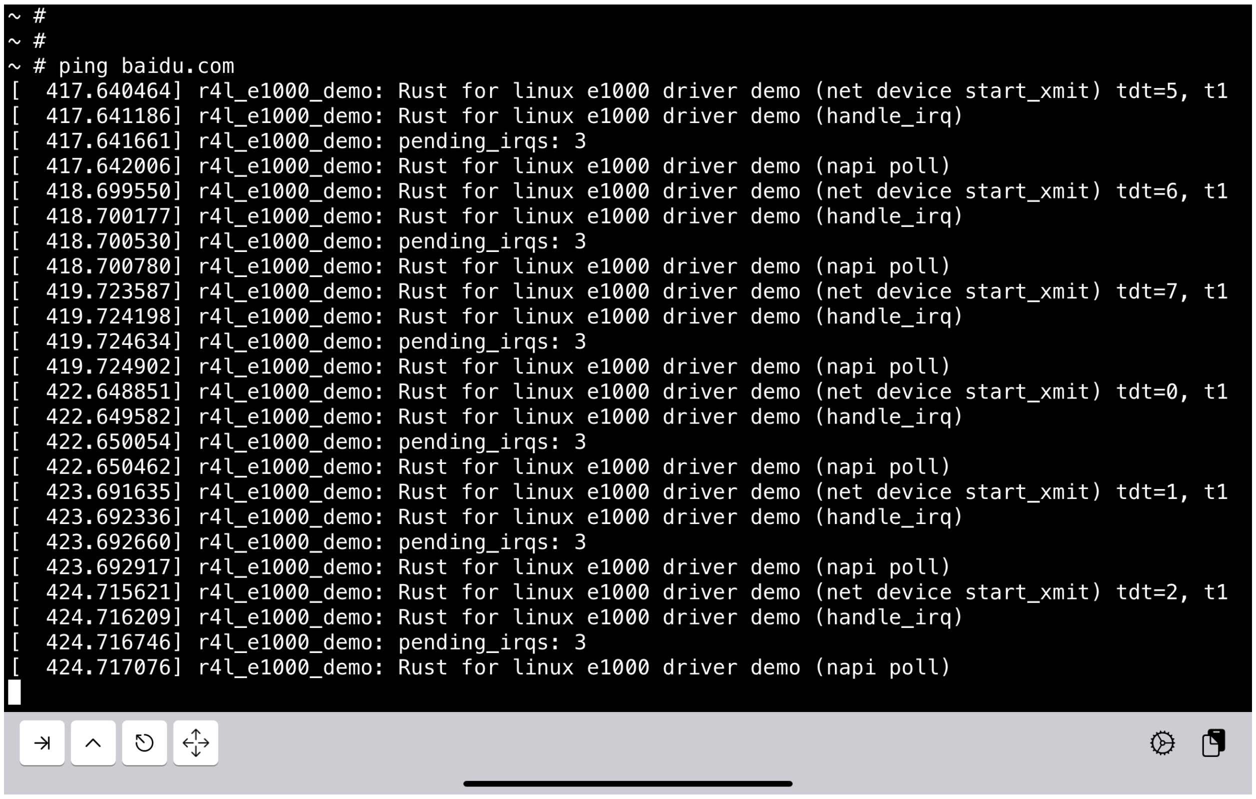
```
insmod r4l_e1000_demo.ko # 加载模块到内核中

ip link set eth0 up # 启用名为 eth0 的网络接口
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
```



```
~ # ping baidu.com
ping: bad address 'baidu.com'
~ # ifconfig
~ # insmod r4l_e1000_demo.ko
[  117.998127] r4l_e1000_demo: loading out-of-tree module taints kernel.
[  118.005772] r4l_e1000_demo: Rust for linux e1000 driver demo (init)
[  118.006724] r4l_e1000_demo: Rust for linux e1000 driver demo (probe): None
[  118.231872] ACPI: \_SB_.LNKC: Enabled at IRQ 11
[  118.255026] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
[  118.257938] insmod (84) used greatest stack depth: 12312 bytes left
~ # ifconfig
[  126.476895] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
```



```
[  262.679210] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=4, t2
[  262.680092] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  262.680521] r4l_e1000_demo: pending_irqs: 131
[  262.681009] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=1 ttl=255 time=3.077 ms
[  263.683065] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=5, t3
[  263.684021] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  263.684382] r4l_e1000_demo: pending_irqs: 131
[  263.684760] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=2 ttl=255 time=2.647 ms
[  264.685790] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=6, t4
[  264.686530] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  264.686899] r4l_e1000_demo: pending_irqs: 131
[  264.687193] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=3 ttl=255 time=2.239 ms
[  265.688261] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=7, t5
[  265.688835] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  265.689078] r4l_e1000_demo: pending_irqs: 131
[  265.689335] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=4 ttl=255 time=1.859 ms
[  266.690484] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=0, t6
[  266.691157] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  266.691618] r4l_e1000_demo: pending_irqs: 131
[  266.692001] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=5 ttl=255 time=2.600 ms
[  267.693216] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=1, t7
[  267.693920] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  267.694288] r4l_e1000_demo: pending_irqs: 131
[  267.694585] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=6 ttl=255 time=2.131 ms
```



```
~ #
~ #
~ # ping baidu.com
[  417.640464] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=5, t1
[  417.641186] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  417.641661] r4l_e1000_demo: pending_irqs: 3
[  417.642006] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[  418.699550] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=6, t1
[  418.700177] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  418.700530] r4l_e1000_demo: pending_irqs: 3
[  418.700780] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[  419.723587] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=7, t1
[  419.724198] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  419.724634] r4l_e1000_demo: pending_irqs: 3
[  419.724902] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[  422.648851] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=0, t1
[  422.649582] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  422.650054] r4l_e1000_demo: pending_irqs: 3
[  422.650462] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[  423.691635] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=1, t1
[  423.692336] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  423.692660] r4l_e1000_demo: pending_irqs: 3
[  423.692917] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[  424.715621] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=2, t1
[  424.716209] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  424.716746] r4l_e1000_demo: pending_irqs: 3
[  424.717076] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
```

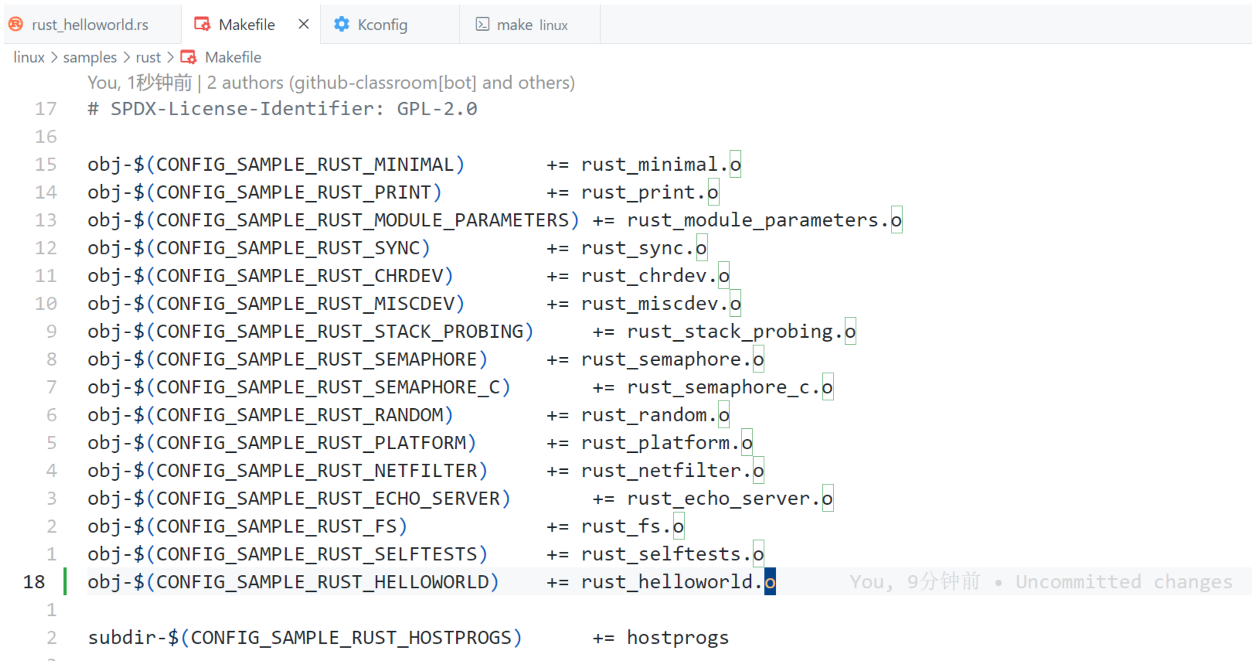
作业三

makefile

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

Kconfig

```
config SAMPLE_RUST_HELLOWORLD
    tristate "Helloworld"
    help
        The option builds the Rust Helloworld module.

        To compile this as a module, choose M here:
        the module will be called rust_helloworld.
        If unsure, say N.
```

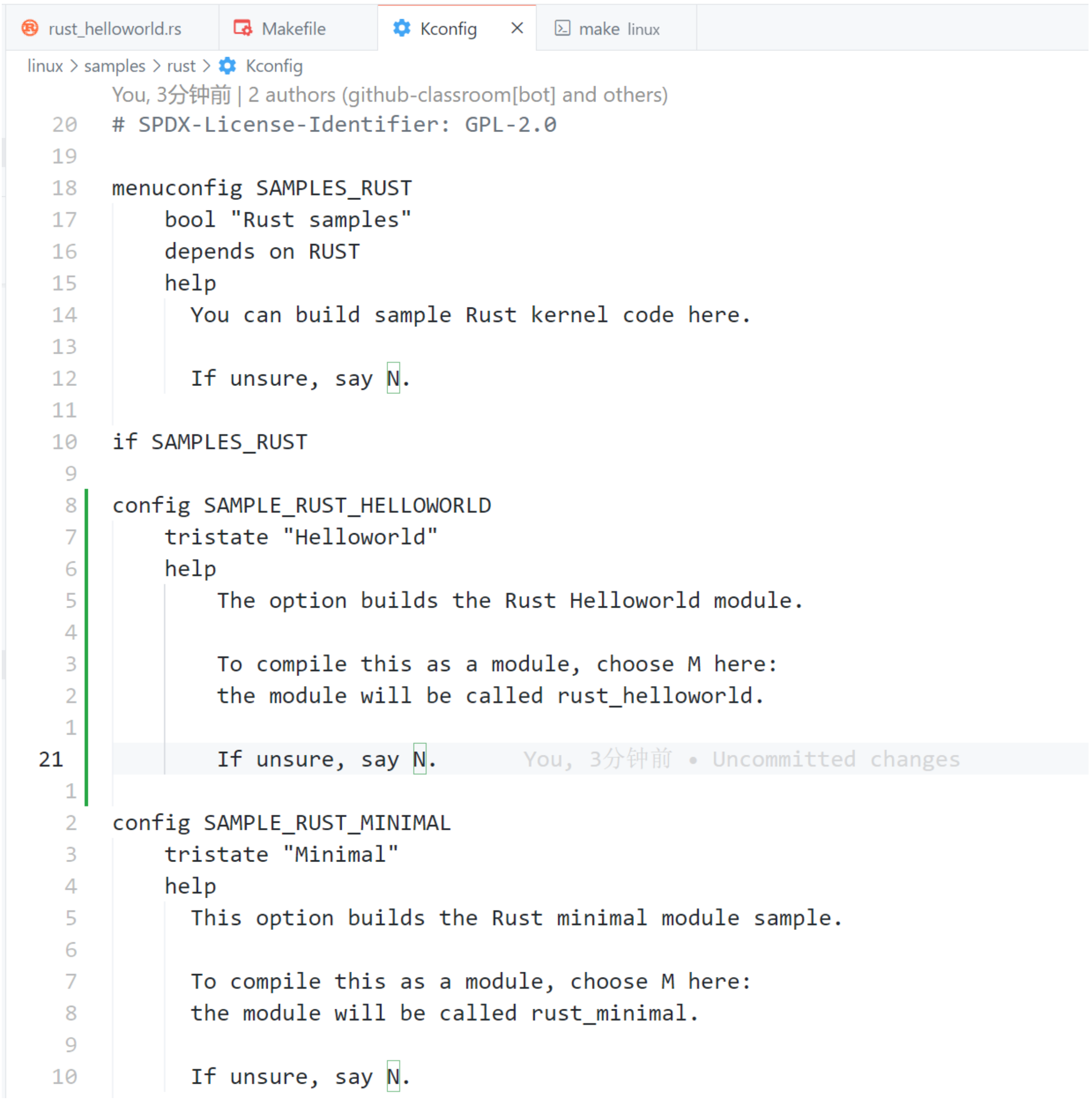


rust_helloworld.rs | Makefile | Kconfig | make linux

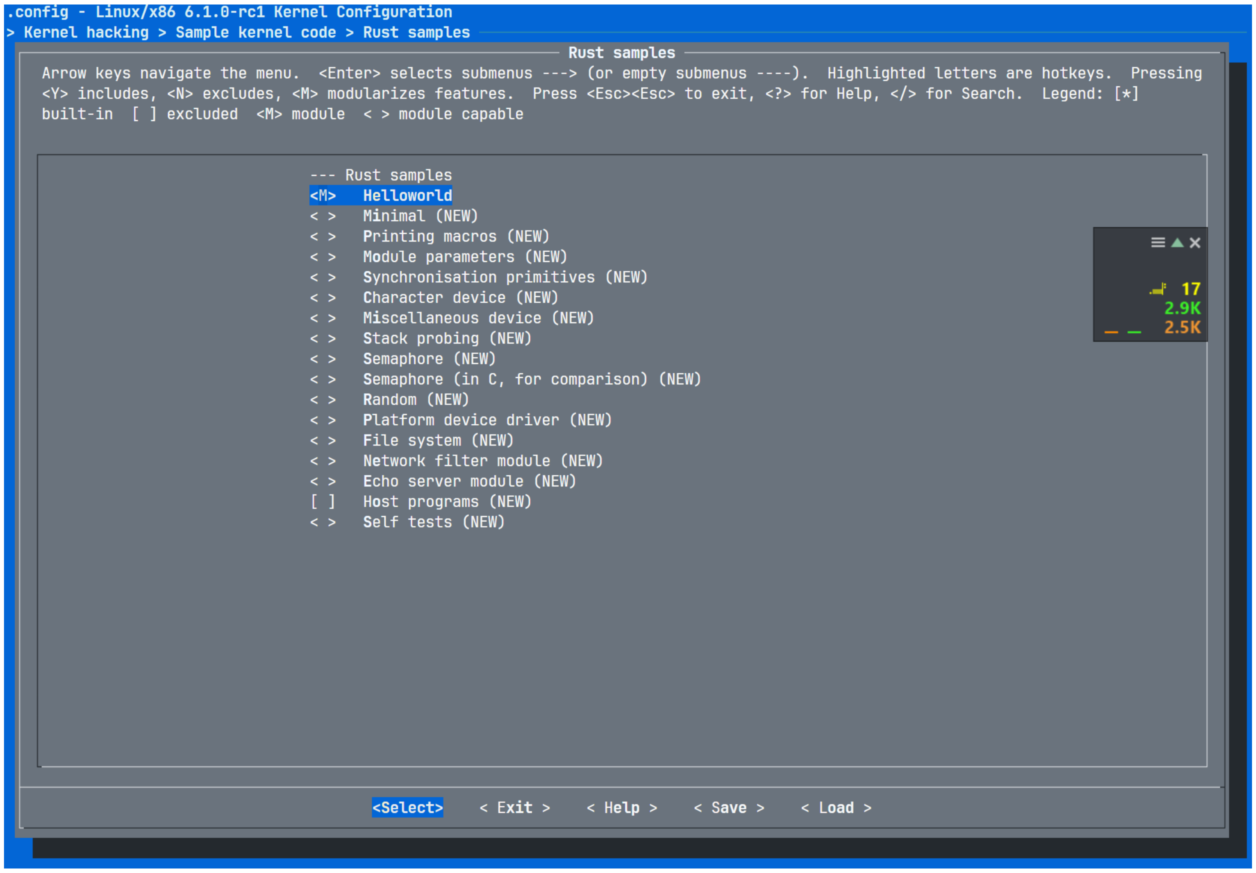
linux > samples > rust > Makefile

You, 1秒钟前 | 2 authors (github-classroom[bot] and others)

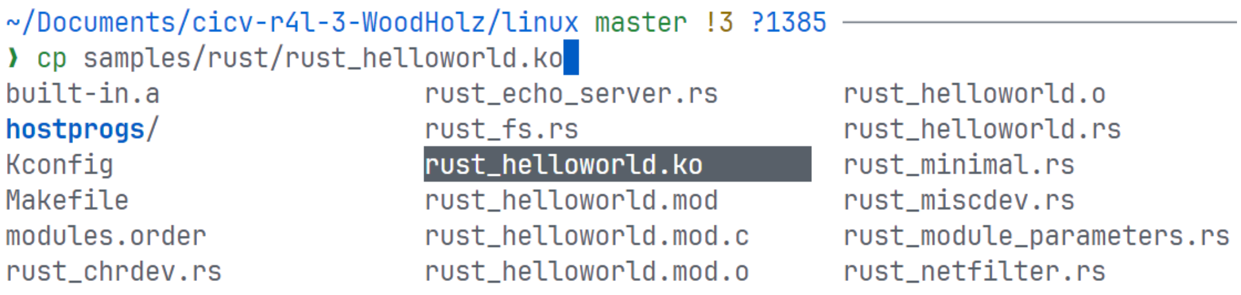
```
# SPDX-License-Identifier: GPL-2.0

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       You, 9分钟前 • Uncommitted changes

subdir-$(CONFIG_SAMPLE_RUST_HOSTPROGS)	+= hostprogs
```


rust_helloworld.rs
Makefile
Kconfig
make linux
linux > samples > rust > Kconfig
You, 3分钟前 | 2 authors (github-classroom[bot] and others)
You, 3分钟前 • Uncommitted changes

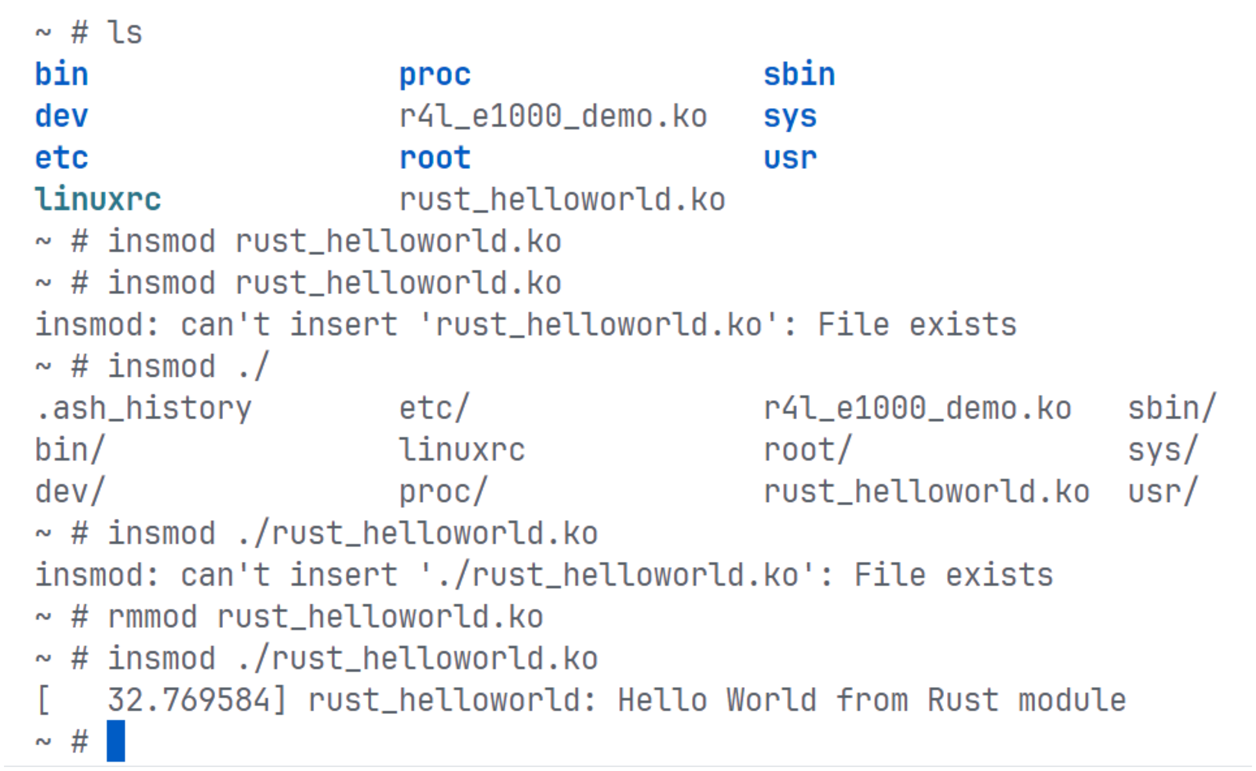
```
# SPDX-License-Identifier: GPL-2.0

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 SAMPLE_RUST_HELLOWORLD
	tristate "Helloworld"
	help
		The option builds the Rust Helloworld module.

		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"
	help
	  This option builds the Rust minimal module sample.

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

	  If unsure, say N.
```



```
.config - Linux/x86 6.1.0-rc1 Kernel Configuration
> Kernel hacking > Sample kernel code > Rust samples
                              Rust samples
  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
                              <M>   Helloworld
                              < >   Minimal (NEW)
                              < >   Printing macros (NEW)
                              < >   Module parameters (NEW)
                              < >   Synchronisation primitives (NEW)
                              < >   Character device (NEW)
                              < >   Miscellaneous device (NEW)
                              < >   Stack probing (NEW)
                              < >   Semaphore (NEW)
                              < >   Semaphore (in C, for comparison) (NEW)
                              < >   Random (NEW)
                              < >   Platform device driver (NEW)
                              < >   File system (NEW)
                              < >   Network filter module (NEW)
                              < >   Echo server module (NEW)
                              [ ]   Host programs (NEW)
                              < >   Self tests (NEW)

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



```
~/Documents/cicv-r4l-3-WoodHolz/linux master !3 ?1385
❯ cp samples/rust/rust_helloworld.ko
built-in.a              rust_echo_server.rs      rust_helloworld.o
hostprogs/              rust_fs.rs               rust_helloworld.rs
Kconfig                 rust_helloworld.ko       rust_minimal.rs
Makefile                rust_helloworld.mod      rust_miscdev.rs
modules.order           rust_helloworld.mod.c    rust_module_parameters.rs
rust_chrdev.rs          rust_helloworld.mod.o    rust_netfilter.rs
```



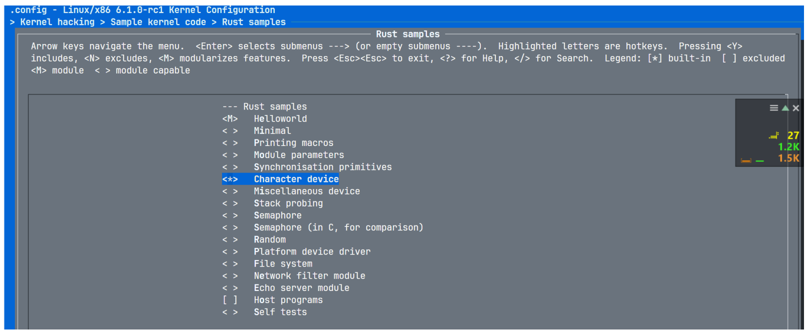
```
~ # ls
bin                    proc                   sbin
dev                    r4l_e1000_demo.ko      sys
etc                    root                   usr
linuxrc                rust_helloworld.ko
~ # insmod rust_helloworld.ko
~ # insmod rust_helloworld.ko
insmod: can't insert 'rust_helloworld.ko': File exists
~ # insmod ./
.ash_history           etc/                   r4l_e1000_demo.ko    sbin/
bin/                   linuxrc                root/                sys/
dev/                   proc/                  rust_helloworld.ko   usr/
~ # insmod ./rust_helloworld.ko
insmod: can't insert './rust_helloworld.ko': File exists
~ # rmmod rust_helloworld.ko
~ # insmod ./rust_helloworld.ko
[   32.769584] rust_helloworld: Hello World from Rust module
~ #
```

作业四

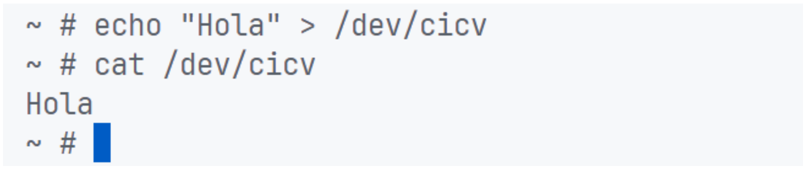
```
~ # insmod r4l_e1000_demo.ko
[   26.427530] r4l_e1000_demo: loading out-of-tree module taints kernel.
[   26.438371] r4l_e1000_demo: Rust for linux e1000 driver demo (init)
[   26.439349] r4l_e1000_demo: Rust for linux e1000 driver demo (probe): None
[   26.704096] ACPI: \_SB_.LNKC: Enabled at IRQ 11
[   26.726322] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
[   26.729596] insmod (80) used greatest stack depth: 11176 bytes left
~ # rmmod r4l_e1000_demo.ko
[   32.243735] r4l_e1000_demo: Rust for linux e1000 driver demo (exit)
[   32.245152] r4l_e1000_demo: Rust for linux e1000 driver demo (remove)
[   32.476049] r4l_e1000_demo: Rust for linux e1000 driver demo (device_remove)
[   32.481251] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
~ # insmod r4l_e1000_demo.ko
[   37.432443] r4l_e1000_demo: Rust for linux e1000 driver demo (init)
[   37.433526] r4l_e1000_demo: Rust for linux e1000 driver demo (probe): None
[   37.756591] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
~ # ifconfig
[   50.001436] 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
[  367.166471] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=1, tdh=1, rdt=7, rdh=0
[  367.168372] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  367.168759] r4l_e1000_demo: pending_irqs: 131
[  367.169256] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[  367.172276] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=2, tdh=2, rdt=0, rdh=1
[  367.173538] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  367.173985] r4l_e1000_demo: pending_irqs: 131
[  367.175456] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=0 ttl=255 time=20.171 ms
[  368.180872] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=3, tdh=3, rdt=1, rdh=2
[  368.184184] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  368.187353] r4l_e1000_demo: pending_irqs: 131
[  368.189020] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=1 ttl=255 time=9.430 ms
[  369.190443] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=4, tdh=4, rdt=2, rdh=3
[  369.192477] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  369.192923] r4l_e1000_demo: pending_irqs: 131
[  369.193361] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=2 ttl=255 time=4.134 ms
[  370.194768] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=5, tdh=5, rdt=3, rdh=4
[  370.197110] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  370.197629] r4l_e1000_demo: pending_irqs: 131
[  370.198284] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=3 ttl=255 time=4.636 ms
^C
--- 10.0.2.2 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
round-trip min/avg/max = 4.134/9.592/20.171 ms
```

作业五



```
.config - Linux/x86 6.1.0-rc1 Kernel Configuration
> Kernel hacking > Sample kernel code > Rust samples
                              Rust samples
  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
                    <M>   Helloworld
                    < >   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
                    [ ]   Host programs
                    < >   Self tests
```



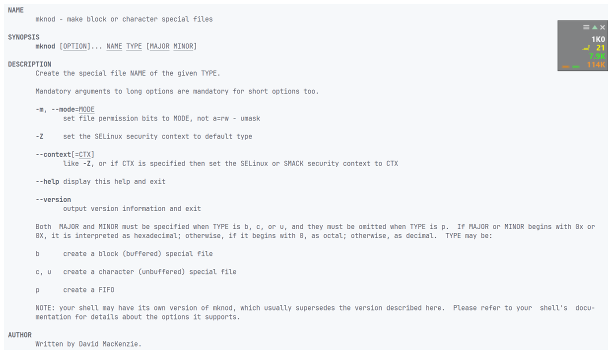
```
~ # echo "Hola" > /dev/cicv
~ # cat /dev/cicv
Hola
~ #
```

▼ 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: 在 `build_image.sh` 中通过以下命令创建

```
mknod /dev/cicv c 248 0
```

参考 man 手册可知

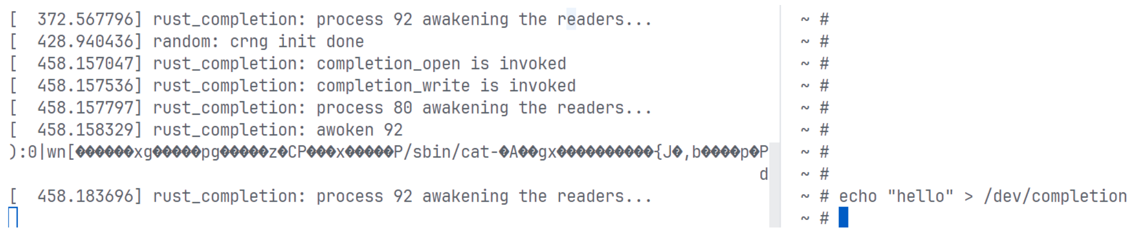


```
NAME
       mknod - make block or character special files

SYNOPSIS
       mknod [OPTION]... NAME TYPE [MAJOR MINOR]

DESCRIPTION
       Create the special file NAME of the given TYPE.

       Mandatory arguments to long options are mandatory for short options too.

       -m, --mode=MODE
              set file permission bits to MODE, not a=rw - umask

       -Z     set the SELinux security context to default type

       --context[=CTX]
              like -Z, or if CTX is specified then set the SELinux or SMACK security context to CTX

       --help display this help and exit

       --version
              output version information and exit

       Both  MAJOR and MINOR must be specified when TYPE is b, c, or u, and they must be omitted when TYPE is p.  If MAJOR or MINOR begins with 0x or
       0X, it is interpreted as hexadecimal; otherwise, if it begins with 0, as octal; otherwise, as decimal.  TYPE may be:

       b      create a block (buffered) special file

       c, u   create a character (unbuffered) special file

       p      create a FIFO

       NOTE: your shell may have its own version of mknod, which usually supersedes the version described here.  Please refer to your  shell's  docu-
       mentation for details about the options it supports.

AUTHOR
       Written by David MacKenzie.
```

设备号是 248

`TYPE` 为 `c` 表面 `/dev/cicv` 是一个字符文件 与字符设备驱动关联

实验



```
[  372.567796] rust_completion: process 92 awakening the readers...
[  428.940436] random: crng init done
[  458.157047] rust_completion: completion_open is invoked
[  458.157536] rust_completion: completion_write is invoked
[  458.157797] rust_completion: process 80 awakening the readers...
[  458.158329] rust_completion: awoken 92
):0|wn[������xg�����pg�����z�CP���x�����P/sbin/cat-�A��gx����������{J�,b����p�P
                                                                              d
[  458.183696] rust_completion: process 92 awakening the readers...
```

```
~ #
~ #
~ #
~ #
~ #
~ #
~ #
~ #
~ # echo "hello" > /dev/completion
~ #
```