作业1

1、执行`make x86_64_defconfig`，如下所示：

```
leotest@leopc:~/cicv-r4l-foolfor/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
#
```

2、执行`make LLVM=1 menuconfig`，如下所示：

```
leotest@leopc:~/cicv-r4l-foolfor/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


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

3、执行`make LLVM=1 -j$(nproc)`，如下所示：

```
leotest@leopc:~/cicv-r4l-foolfor/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
  HOSTCC  arch/x86/tools/relocs_32.o
  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
```

4、最终在Linux文件夹下，得到vmlinux文件，如下所示：

```
leotest@leopc:~/cicv-r4l-foolfor/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
```

作业2

提问

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：是由`Kbuild`文件中的`obj-m := r4l_e1000_demo.o`语句来定义。

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`build_image.sh`和`Makefile`文件中，指定了Linux内核代码的路径，模块编译时，会与指定内核代码产生联系。

（1）模块的头文件`build_image.sh`中，指定了Linux内核代码路径，如下所示：

```
#!/bin/sh
busybox_folder="../busybox-1.36.1"
kernel_image="../linux/arch/x86/boot/bzImage"
work_dir=$PWD
rootfs="rootfs"
rootfs_img=$PWD"/rootfs_img"
```

（2）`Makefile`文件中，也指定了内核代码的所在路径：

```
root@foolfor:/home/leotest/CICV/cicv-r4l-foolfor/src_e1000# cat Makefile
# SPDX-License-Identifier: GPL-2.0

KDIR ?= ../linux

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

实验

1、执行`make LLVM=1`编译内核模块，如下所示：

```
root@foolfor:/home/leotest/CICV/cicv-r4l-foolfor/src_e1000# make LLVM=1
make -C ../linux M=$PWD
make[1]: 进入目录 “/home/leotest/CICV/cicv-r4l-foolfor/linux”
  RUSTC [M] /home/leotest/CICV/cicv-r4l-foolfor/src_e1000/r4l_e1000_demo.o
  MODPOST /home/leotest/CICV/cicv-r4l-foolfor/src_e1000/Module.symvers
  CC [M]  /home/leotest/CICV/cicv-r4l-foolfor/src_e1000/r4l_e1000_demo.mod.o
  LD [M]  /home/leotest/CICV/cicv-r4l-foolfor/src_e1000/r4l_e1000_demo.ko
make[1]: 离开目录 “/home/leotest/CICV/cicv-r4l-foolfor/linux”
root@foolfor:/home/leotest/CICV/cicv-r4l-foolfor/src_e1000#
```

2、执行`build_image.sh`脚本后，如下所示，进入到Linux子系统下

```
[    2.725312] Freeing unused kernel image (initmem) memory: 1328K
[    2.729943] Write protecting the kernel read-only data: 24576k
[    2.734709] Freeing unused kernel image (text/rodata gap) memory: 2032K
[    2.736004] Freeing unused kernel image (rodata/data gap) memory: 824K
[    2.918320] x86/mm: Checked W+X mappings: passed, no W+X pages found.
[    2.919146] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[    2.922075] Run sbin/init as init process
[    2.987814] mount (71) used greatest stack depth: 14160 bytes left
[    3.205193] mdev (73) used greatest stack depth: 13920 bytes left

Please press Enter to activate this console.
~ #
~ # echo "foolfor"
foolfor
```

3、在当前系统下，执行`ifconfig`和`ping`命令如下所示：

```
~ # echo "foolfor"
foolfor
~ #
~ # 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:2 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:220 (220.0 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=13.307 ms
64 bytes from 10.0.2.15: seq=1 ttl=64 time=1.103 ms
64 bytes from 10.0.2.15: seq=2 ttl=64 time=0.836 ms
```

4、在Linux目录下，执行`make menuconfig`，按照要求，找到E1000驱动，并禁用驱动加载，如下所示：

```
< >     Fujitsu FMV-J18x PCMCIA support
[*]   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
```

5、在Linux目录下，执行`make LLVM=1 -j$(nproc)`重新编译Linux内核。

```
  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  (#2)
root@foolfor:/home/leotest/CICV/cicv-r4l-foolfor/linux# ls
arch   built-in.a  COPYING  crypto        drivers  include  io_uring  Kbuild   kernel  LICENSES     Makefile  modules.builtin         modules.order  net     README.md  samples  security  System.map  usr   vmlinux    vmlinux.o
block  certs       CREDITS  Documentation  fs       init     ipc       Kconfig  lib     MAINTAINERS  mm        modules.builtin.modinfo  Module.symvers  README  rust       scripts  sound     tools       virt  vmlinux.a
root@foolfor:/home/leotest/CICV/cicv-r4l-foolfor/linux#
```

6、在`src-e1000`中，执行`make LLVM=1`，重新编译模块。

```
root@foolfor:/home/leotest/CICV/cicv-r4l-foolfor/src_e1000# make LLVM=1
make -C ../linux M=$PWD
make[1]: 进入目录“/home/leotest/CICV/cicv-r4l-foolfor/linux”
  RUSTC [M] /home/leotest/CICV/cicv-r4l-foolfor/src_e1000/r4l_e1000_demo.o
  MODPOST /home/leotest/CICV/cicv-r4l-foolfor/src_e1000/Module.symvers
  LD [M]  /home/leotest/CICV/cicv-r4l-foolfor/src_e1000/r4l_e1000_demo.ko
make[1]: 离开目录“/home/leotest/CICV/cicv-r4l-foolfor/linux”
root@foolfor:/home/leotest/CICV/cicv-r4l-foolfor/src_e1000#
```

7、再次执行`./build_image.sh`进入子系统，执行`ifconfig`如下所示，原`E1000`驱动程序没有加载：

```
[   16.567242] mdev (74) used greatest stack depth: 13952 bytes left
[   16.573021] mknod (75) used greatest stack depth: 13920 bytes left

Please press Enter to activate this console.
~ #
~ # ifconfig
~ #
~ # ip a
1: lo: <LOOPBACK> mtu 65536 qdisc noop qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
2: sit0@NONE: <NOARP> mtu 1480 qdisc noop qlen 1000
    link/sit 0.0.0.0 brd 0.0.0.0
[   25.575170] ip (81) used greatest stack depth: 13032 bytes left
~ #
```

8、执行如下命令为网卡添加地址参数

```
insmod r4l_e1000_demo.ko
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
```

```
~ # insmod r4l_e1000_demo.ko
[   44.465619] r4l_e1000_demo: loading out-of-tree module taints kernel.
[   44.475388] r4l_e1000_demo: Rust for linux e1000 driver demo (init)
[   44.476414] r4l_e1000_demo: Rust for linux e1000 driver demo (probe): None
[   44.704148] ACPI: \_SB_.LNKC: Enabled at IRQ 11
[   44.743193] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
[   44.746751] insmod (82) used greatest stack depth: 11144 bytes left
~ #
~ # ip link set eth0 up
[   57.847287] r4l_e1000_demo: Rust for linux e1000 driver demo (net device open)
[   57.856673] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
[   57.858795] IPv6: ADDRCONF(NETDEV_CHANGE): eth0: link becomes ready
[   57.863409] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
~ # [   57.884822] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=0, tdh=0, rdt=7, rdh=0
[   57.887083] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[   57.887761] r4l_e1000_demo: pending_irqs: 3
[   57.889041] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[   58.454118] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=1, tdh=1, rdt=7, rdh=0
[   58.461612] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
```

```
~ # ip addr add broadcast 10.0.2.255 dev eth0
[  101.495759] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
[  101.499158] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
ip: RTNETLINK answers: Invalid argument
~ #
~ # ip addr add 10.0.2.15/255.255.255.0 dev eth0
[  119.296128] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
[  119.297056] r4l_e1000_demo: Rust for linux e1000 driver demo (net device get_stats64)
~ #
~ # [  124.050889] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=1, tdh=1, rdt=7, rdh=0
[  124.058028] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  124.061231] r4l_e1000_demo: pending_irqs: 3
[  124.063737] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)

~ # ip route add default via 10.0.2.1
```

9、执行`ping 10.0.2.2`如下所示：

```
~ # ping 10.0.2.2
PING 10.0.2.2 (10.0.2.2): 56 data bytes
[  158.022325] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=2, tdh=2, rdt=7, rdh=0
[  158.023570] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  158.023822] r4l_e1000_demo: pending_irqs: 131
[  158.024199] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
[  158.026560] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=3, tdh=3, rdt=0, rdh=1
[  158.026885] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  158.027046] r4l_e1000_demo: pending_irqs: 131
[  158.029064] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=0 ttl=255 time=15.864 ms
[  159.042256] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=4, tdh=4, rdt=1, rdh=2
[  159.043566] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  159.044345] r4l_e1000_demo: pending_irqs: 131
[  159.044902] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=1 ttl=255 time=4.368 ms
[  160.047462] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=5, tdh=5, rdt=2, rdh=3
[  160.049724] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  160.050631] r4l_e1000_demo: pending_irqs: 131
[  160.051386] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=2 ttl=255 time=5.275 ms
[  161.054562] r4l_e1000_demo: Rust for linux e1000 driver demo (net device start_xmit) tdt=6, tdh=6, rdt=3, rdh=4
[  161.055864] r4l_e1000_demo: Rust for linux e1000 driver demo (handle_irq)
[  161.056569] r4l_e1000_demo: pending_irqs: 131
[  161.057139] r4l_e1000_demo: Rust for linux e1000 driver demo (napi poll)
64 bytes from 10.0.2.2: seq=3 ttl=255 time=3.851 ms
^C
--- 10.0.2.2 ping statistics ---
4 packets transmitted, 4 packets received, 0% packet loss
```

作业3

作业4

作业5