实验一 openEuler内核编译与替换

实验背景

- openEuler 是华为推动的一款开源操作系统。当前 openEuler 内核源于 Linux，支持鲲鹏及其它多种处理器，能够充分释放计算芯片的潜能，是由全球开源贡献者构建的高效、稳定、安全的开源操作系统，适用于数据库、大数据、云计算、人工智能等应用场景。同时， openEuler是一个面向全球的操作系统开源社区，通过社区合作，打造创新平台，构建支持多处理器架构、统一和开放的操作系统，推动软硬件应用生态繁荣发展。 更多信息可以参考其官方网站。

实验目的

1. 熟悉 Linux 的运行环境；
2. 掌握 Linux 内核编译的过程；
3. 了解 openEuler 内核，能够编译替换不同版本的内核，为后续自行修改内核代码并编译替换铺垫。

实验要求

- 在 Linux 系统上，下载 openEuler 内核进行编译与替换。

实验环境

- 平台：Vmware Workstation 17 Pro
- 系统：Ubuntu 20.04.6 LTS

实验过程

0. 安装和配置 Ubuntu 20.0406 系统
    - 安装 VScode（推荐），配置好 make、 gcc 等编译工具
    - 用命令 `uname -r` 查看原始内核版本

```
yyyj@ubuntu:~$ make -v
GNU Make 4.2.1
为 x86_64-pc-linux-gnu 编译
Copyright (C) 1988-2016 Free Software Foundation, Inc.
许可证：GPLv3+：GNU 通用公共许可证第 3 版或更新版本<http://gnu.org/licenses/gpl.
html>。
本软件是自由软件：您可以自由修改和重新发布它。
在法律允许的范围内没有其他保证。
yyyj@ubuntu:~$ gcc --version
gcc (Ubuntu 9.4.0-1ubuntu1~20.04.2) 9.4.0
Copyright (C) 2019 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.

yyyj@ubuntu:~$ uname -r
5.15.0-134-generic
yyyj@ubuntu:~$
```

1. 下载 openEuler 内核源码:
   - 到 代码仓库 下载 openEuler 源码并解压
   - 查看仓库代码内核版本

```
yyyj@ubuntu:~/oslab/kernel-4.19.90-2403.3.0$
```

2. 清理源代码树:
   - 进入解压好的源码文件夹清理过去内核编译产生的文件，第一次编译时可不执行此命令。

```
make mrproper
```

3. 生成内核配置文件:
   - 先将系统原配置文件复制到代码仓库文件夹下，原配置文件在/boot 目录下，利用 `uname -r` 获取当前系统的内核版本。将配置在当前目录下保存为.config 文件

```
cp -v /boot/config-$(uname -r) ./.config
```

   - 编译 Linux 内核需要安装 ncurses， Ubuntu 下对应包为 libncurses5-dev：

```
sudo apt install libncurses5-dev
```

   - 使用以下命令对配置进行需要的更改,根据提示需先安装相应的依赖，不同的包在不同 Linux 发行版下名称不同。

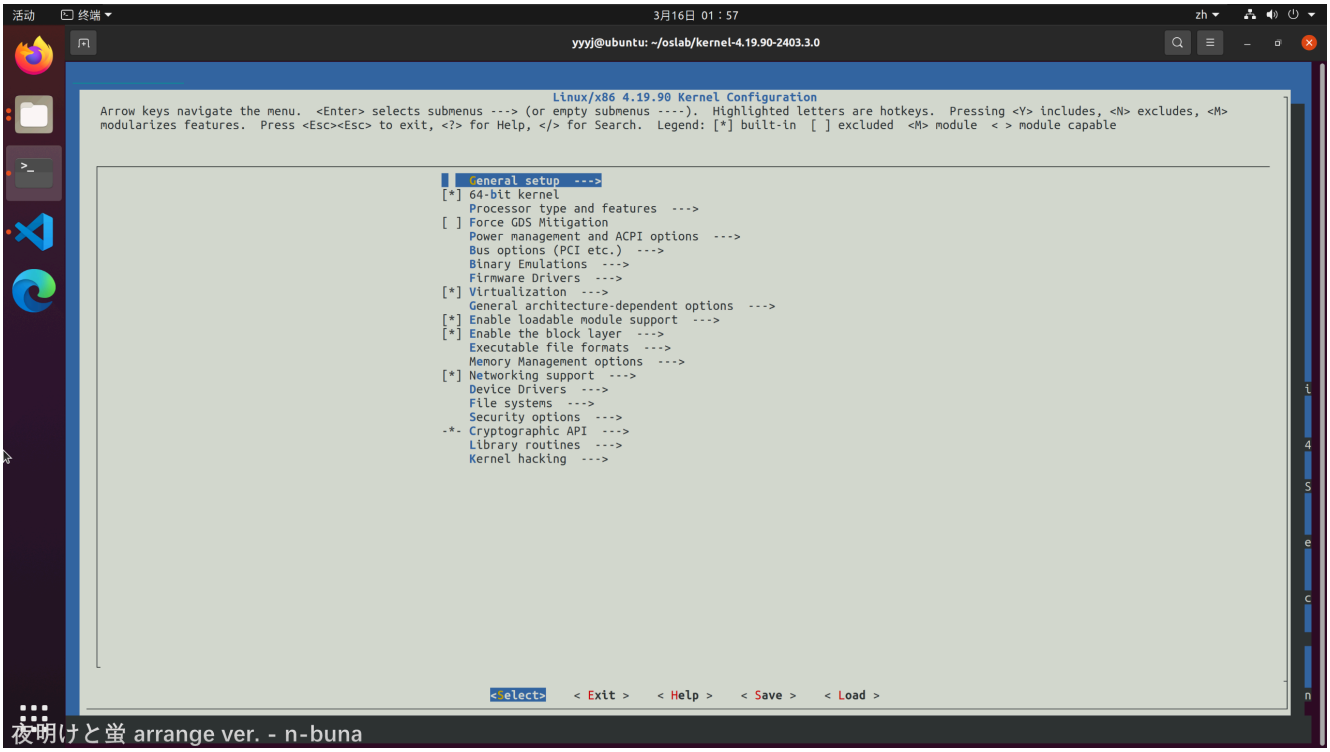
```
make menuconfig
```

     可以直接 Load 原始.config 文件，也可以自行进行配置或者使用默认配置， Save-Exit。
   - 这一步遇到报错：

```
yyyj@ubuntu:~/oslab/kernel-4.19.90-2403.3.0$ make menuconfig
  HOSTCC  scripts/basic/fixdep
  UPD     scripts/kconfig/.mconf-cfg
  HOSTCC  scripts/kconfig/mconf.o
  YACC    scripts/kconfig/zconf.tab.c
/bin/sh: 1: bison: not found
make[1]: *** [scripts/Makefile.lib:196: scripts/kconfig/zconf.tab.c] 错误 127
make: *** [Makefile:534: menuconfig] 错误 2
yyyj@ubuntu:~/oslab/kernel-4.19.90-2403.3.0$
```

分析得缺少依赖bison和flex，安装后问题解决

```
sudo apt install bison flex
```



4. 内核编译与安装

- 首先安装执行编译所需的组件， 包括 libelf-dev,openssl,libssl-dev,bc

```
sudo apt install libelf-dev openssl libssl-dev bc
```

- 开始编译内核，使用 `make -j8` 命令，其中-j8表示使用8个线程进行编译。
  - 编译过程碰到关于 `canonical-certs.pem` 的报错，将生成的配置文件.config的 `CONFIG_SYSTEM_REVOCATION_KEYS="debian/canonical-revoked-certs.pem"` 注释掉。
  - 进行下一步时发现安装模块失败

```
yyyj@ubuntu:~/oslab/kernel-4.19.90-2403.3.0$ sudo make modules_install
cp: 无法获取'./modules.builtin' 的文件状态(stat): 没有那个文件或目录
make: *** [Makefile:1267: _modinst_] 错误 1
yyyj@ubuntu:~/oslab/kernel-4.19.90-2403.3.0$
```

查看编译日志，发现报错

```
  AS [M]  arch/x86/crypto/twofish-avx-x86_64-asm_64.o
  AR      arch/x86/kernel/acpi/built-in.a
make[1]: *** 没有规则可制作目标“debian/canonical-certs.pem”，由“certs/x509_certificate_list” 需求。 停止。
make: *** [Makefile:1057: certs] 错误 2
make: *** 正在等待未完成的任务....
  CC [M]  arch/x86/crypto/twofish_avx_glue.o
  CC      arch/x86/kernel/apic/apic_common.o
```

- 将.config文件中的 `2CONFIG SYSTEM TRUSTED KEYS-"debian/canonical-certs.pem"` 注释掉，重新编译。

```
  LD [M]  sound/soc/intel/atom/sst/snd-intel-sst-acpi.ko
  LD [M]  sound/soc/intel/atom/sst/snd-intel-sst-pci.ko
  LD [M]  sound/soc/intel/atom/sst/snd-intel-sst-core.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-byt-cht-da7213.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-byt-cht-es8316.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-bytcr-rt5640.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-bytcr-rt5651.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-cht-bsw-max98090_ti.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-cht-bsw-nau8824.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-cht-bsw-rt5645.ko
  LD [M]  sound/soc/intel/boards/snd-soc-sst-cht-bsw-rt5672.ko
  LD [M]  sound/soc/intel/common/snd-soc-acpi-intel-match.ko
  LD [M]  sound/soc/snd-soc-acpi.ko
  LD [M]  sound/soc/snd-soc-core.ko
  LD [M]  sound/soc/xtensa/snd-soc-xtfpga-i2s.ko
  LD [M]  sound/soundcore.ko
  LD [M]  sound/synth/snd-util-mem.ko
  LD [M]  sound/synth/emux/snd-emux-synth.ko
  LD [M]  sound/usb/6fire/snd-usb-6fire.ko
  LD [M]  sound/usb/bcd2000/snd-bcd2000.ko
  LD [M]  sound/usb/caiaq/snd-usb-caiaq.ko
  LD [M]  sound/usb/line6/snd-usb-line6.ko
  LD [M]  sound/usb/hiface/snd-usb-hiface.ko
  LD [M]  sound/usb/line6/snd-usb-pod.ko
  LD [M]  sound/usb/line6/snd-usb-podhd.ko
  LD [M]  sound/usb/line6/snd-usb-toneport.ko
  LD [M]  sound/usb/line6/snd-usb-variax.ko
  LD [M]  sound/usb/misc/snd-ua101.ko
  LD [M]  sound/usb/snd-usb-audio.ko
  LD [M]  sound/usb/snd-usbmidi-lib.ko
  LD [M]  sound/usb/usx2y/snd-usb-us122l.ko
  LD [M]  sound/usb/usx2y/snd-usb-usx2y.ko
  LD [M]  sound/x86/snd-hdmi-lpe-audio.ko
  LD [M]  sound/xen/snd_xen_front.ko
yyyj@ubuntu:~/oslab/kernel-4.19.90-2403.3.0$
```

- 编译完成后安装模块和内核

```
sudo make modules_install
sudo make install
```

```
  INSTALL sound/usb/line6/snd-usb-podhd.ko
  INSTALL sound/usb/line6/snd-usb-toneport.ko
  INSTALL sound/usb/line6/snd-usb-variax.ko
  INSTALL sound/usb/misc/snd-ua101.ko
  INSTALL sound/usb/snd-usb-audio.ko
  INSTALL sound/usb/snd-usbmidi-lib.ko
  INSTALL sound/usb/usx2y/snd-usb-us122l.ko
  INSTALL sound/usb/usx2y/snd-usb-usx2y.ko
  INSTALL sound/x86/snd-hdmi-lpe-audio.ko
  INSTALL sound/xen/snd_xen_front.ko
  DEPMOD  4.19.90
yyyj@ubuntu:~/oslab/kernel-4.19.90-2403.3.0$ sudo make install
sh ./arch/x86/boot/install.sh 4.19.90 arch/x86/boot/bzImage \
        System.map "/boot"
run-parts: executing /etc/kernel/postinst.d/initramfs-tools 4.19.90 /boot/vmlinuz-4.19.90
update-initramfs: Generating /boot/initrd.img-4.19.90
run-parts: executing /etc/kernel/postinst.d/unattended-upgrades 4.19.90 /boot/vmlinuz-4.19.90
run-parts: executing /etc/kernel/postinst.d/update-notifier 4.19.90 /boot/vmlinuz-4.19.90
run-parts: executing /etc/kernel/postinst.d/xx-update-initrd-links 4.19.90 /boot/vmlinuz-4.19.90
I: /boot/initrd.img.old is now a symlink to initrd.img-5.15.0-134-generic
I: /boot/initrd.img is now a symlink to initrd.img-4.19.90
run-parts: executing /etc/kernel/postinst.d/zz-update-grub 4.19.90 /boot/vmlinuz-4.19.90
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
正在生成 grub 配置文件 ...
找到 Linux 镜像: /boot/vmlinuz-5.15.0-134-generic
找到 initrd 镜像: /boot/initrd.img-5.15.0-134-generic
找到 Linux 镜像: /boot/vmlinuz-5.15.0-67-generic
找到 initrd 镜像: /boot/initrd.img-5.15.0-67-generic
找到 Linux 镜像: /boot/vmlinuz-4.19.90
找到 initrd 镜像: /boot/initrd.img-4.19.90
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
完成
yyyj@ubuntu:~/oslab/kernel-4.19.90-2403.3.0$
```

- 完成安装，在 /boot 下看到新安装的内核

```
yyyj@ubuntu:~/oslab/kernel-4.19.90-2403.3.0$ cd /boot
yyyj@ubuntu:/boot$ ll
总用量 782012
drwxr-xr-x  4 root root      4096 3月  16 17:51 ./
drwxr-xr-x 20 root root      4096 3月  16 15:10 ../
-rw-r--r--  1 root root    217177 3月  16 17:50 config-4.19.90
-rw-r--r--  1 root root    262530 2月  17 21:07 config-5.15.0-134-generic
-rw-r--r--  1 root root    262250 2月  22  2023 config-5.15.0-67-generic
drwx------  2 root root      4096 1月   1  1970 efi/
drwxr-xr-x  4 root root      4096 3月  16 17:51 grub/
lrwxrwxrwx  1 root root        18 3月  16 17:51 initrd.img -> initrd.img-4.19.90
-rw-r--r--  1 root root 583556100 3月  16 17:51 initrd.img-4.19.90
-rw-r--r--  1 root root  89013576 3月  16 15:42 initrd.img-5.15.0-134-generic
-rw-r--r--  1 root root  76551700 3月  16 15:41 initrd.img-5.15.0-67-generic
lrwxrwxrwx  1 root root        29 3月  16 17:51 initrd.img.old -> initrd.img-5.15.0-134-generic
-rw-r--r--  1 root root    182704 8月  18  2020 memtest86+.bin
-rw-r--r--  1 root root    184380 8月  18  2020 memtest86+.elf
-rw-r--r--  1 root root    184884 8月  18  2020 memtest86+_multiboot.bin
-rw-r--r--  1 root root   4632930 3月  16 17:50 System.map-4.19.90
-rw-------  1 root root   6261924 2月  17 21:07 System.map-5.15.0-134-generic
-rw-------  1 root root   6223039 2月  22  2023 System.map-5.15.0-67-generic
lrwxrwxrwx  1 root root        15 3月  16 17:50 vmlinuz -> vmlinuz-4.19.90
-rw-r--r--  1 root root  10156416 3月  16 17:50 vmlinuz-4.19.90
-rw-------  1 root root  11572936 2月  17 21:11 vmlinuz-5.15.0-134-generic
-rw-r--r--  1 root root  11458952 3月  16  2023 vmlinuz-5.15.0-67-generic
lrwxrwxrwx  1 root root        26 3月  16 15:12 vmlinuz.old -> vmlinuz-5.15.0-134-generic
```

5. 更新引导文件
   - 根据/etc/default/grub 目录下的内核文件自动更新启动引导文件

```
sudo update-grub
```

```
yyyj@ubuntu:/boot$ sudo update-grub
[sudo] yyyj 的密码:
Sourcing file `/etc/default/grub'
Sourcing file `/etc/default/grub.d/init-select.cfg'
正在生成 grub 配置文件 ...
找到 Linux 镜像: /boot/vmlinuz-5.15.0-134-generic
找到 initrd 镜像: /boot/initrd.img-5.15.0-134-generic
找到 Linux 镜像: /boot/vmlinuz-5.15.0-67-generic
找到 initrd 镜像: /boot/initrd.img-5.15.0-67-generic
找到 Linux 镜像: /boot/vmlinuz-4.19.90
找到 initrd 镜像: /boot/initrd.img-4.19.90
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
完成
yyyj@ubuntu:/boot$
```

- 修改/boot/grub/grub.cfg 中 `menuentry` 后面的字符串，在安装的 openEuler 版本号后增加姓名学号，自定义启动菜单时的选项名称

```
245         initrd  /boot/initrd.img-5.15.0-67-generic
246     }
247     menuentry 'Ubuntu, Linux 4.19.90 - 马悦钊 - 523031910684' --class ubuntu --class gnu-linux --class gn
248         recordfail
249         load_video
250         gfxmode $linux_gfx_mode
251         insmod gzio
252         if [ x$grub_platform = xxen ]; then insmod xzio; insmod lzopio; fi
253         insmod part_msdos
254         insmod ext2
255         set root='hd0,msdos5'
256         if [ x$feature_platform_search_hint = xy ]; then
257           search --no-floppy --fs-uuid --set=root --hint-bios=hd0,msdos5 --hint-efi=hd0,msdos5 --hint-ba
258         else
259           search --no-floppy --fs-uuid --set=root ecc133e9-f4b0-485b-99b2-3f6a18e1310c
260         fi
261         echo    '载入 Linux 4.19.90 ...'
262         linux   /boot/vmlinuz-4.19.90 root=UUID=ecc133e9-f4b0-485b-99b2-3f6a18e1310c ro find_preseed=/pr
263         echo    '载入初始化内存盘...'
264         initrd  /boot/initrd.img-4.19.90
265     }
266     menuentry 'Ubuntu, with Linux 4.19.90 (recovery mode)' --class ubuntu --class gnu-linux --class gnu
```

- 使用 `reboot` 命令重启系统，按住 `shift` 键进入引导菜单，选择 `Ubuntu 的高级选项`，选择新安装的内核版本启动系统。

- 查看当前内核版本，为4.19.90，说明成功安装 openEuler 内核!

```
yyyj@ubuntu:~$ uname -r
4.19.90
yyyj@ubuntu:~$
```

6. 至此，实验一完成。

实验遇到的问题及解决方法

- 编译内核时遇到 `bison` 和 `flex` 依赖问题，解决方法是安装相应的依赖包。
- 编译内核时遇到 `canonical-certs.pem` 问题，解决方法是注释掉 `.config` 文件中的 `CONFIG_SYSTEM_REVOCATION_KEYS="debian/canonical-revoked-certs.pem"` 。
- 编译内核时遇到 `debian/canonical-certs.pem` 问题，解决方法是注释掉 `.config` 文件中的 `CONFIG SYSTEM TRUSTED KEYS-"debian/canonical-certs.pem"` 。

实验总结

- 通过本次实验，我学会了如何在 Linux 系统上下载 openEuler 内核源码并进行编词替换，了解了 openEuler 内核的编译过程，掌握了内核编译的基本步骤，为后续实验内容如自行修改内核代码并编译替换等操作打下了基础。
- 在实验过程中，我遇到了不少问题，但最后都顺利解决了，这大大锻炼了我上网查找问题解决方案的能力，也提高了我动手能力和解决问题的能力。

参考资料

- openEuler 官方网站
- openEuler 内核源码下载
- OpenEuler内核编译及替换
- 实验手册：实验一 openEuler 内核编译与替换