实验一 openEuler 操作系统 编译内核实验

1.实验介绍

本实验通过安装openEuler操作系统、编译安装openEuler操作系统新内核以及简单的内核模块编程任务操作带领大家了解操作系统及内核编程。

2.实验目的

学习掌握如何在树莓派上安装操作系统。

学习掌握如何编译操作系统内核。

了解内核模块编程。

3.实验任务

3.1 openEuler操作系统安装

步骤1 下载openEuler 22.03 LTS SP3树莓派版本

1.登录openEuler Repo网站。

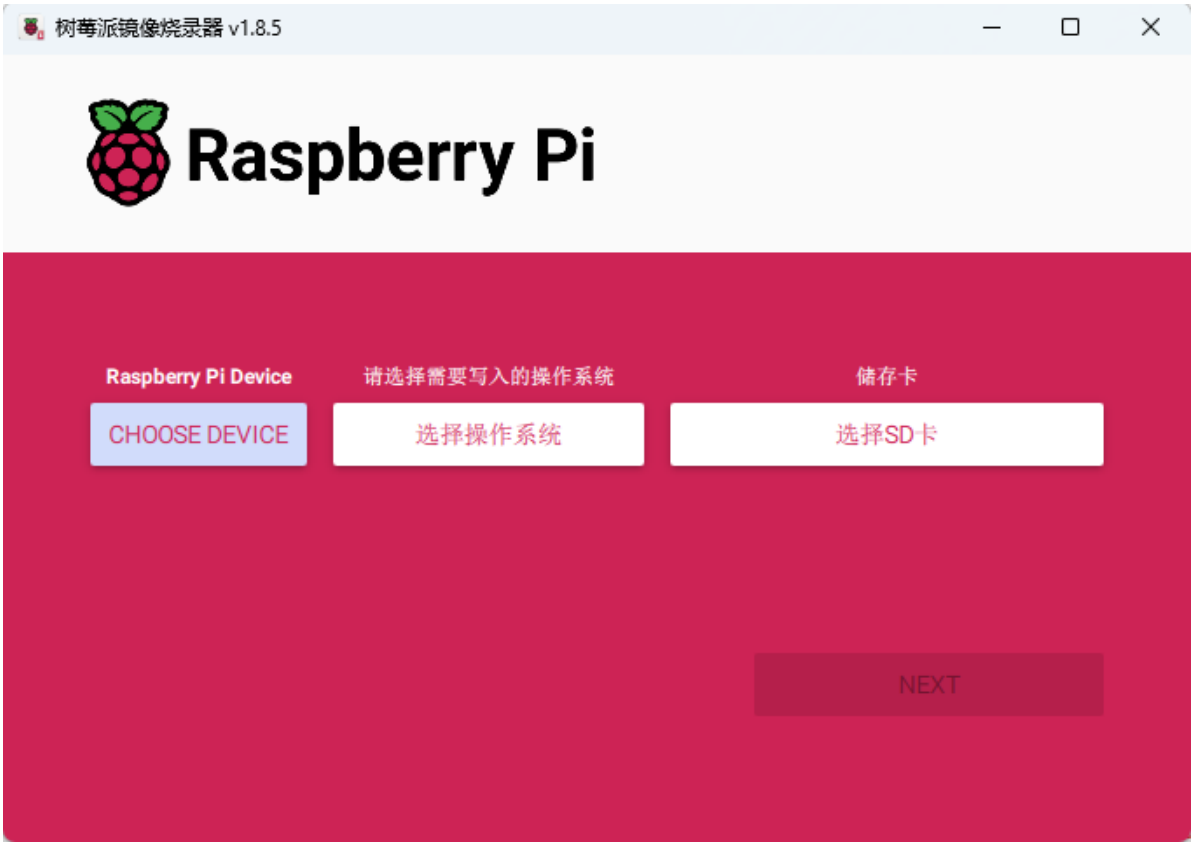
2.单击选择 openEuler 22.03 LTS SP3 版本。

3.单击“raspi_img”，进入树莓派镜像的下载列表。

4.单击“openEuler-22.03-LTS-SP3-raspi-aarch64.img.xz”，将 openEuler 发布的树莓派镜像下载到本地。

步骤2 烧录系统

下载Raspberry Pi Imager，将SD卡插入电脑，进行烧录。



步骤3 连接网络

替换完固件包后，系统可以正常启动，需要使用ssh连接树莓派。

方法一：连接wifi

1.手机打开个人热点。

2.在树莓派上接上键盘，输入以下命令：

（由于此时没有连接ssh，因此看不见树莓派的命令行界面，因此可以使用Hdmi转Micro Hdmi转接头，连接显示器）

```
root                #输入用户名
openeuler           #输入密码
nmcli dev wifi connect 你的WIFI名 password 你的WIFI密码              #连接wifi
```

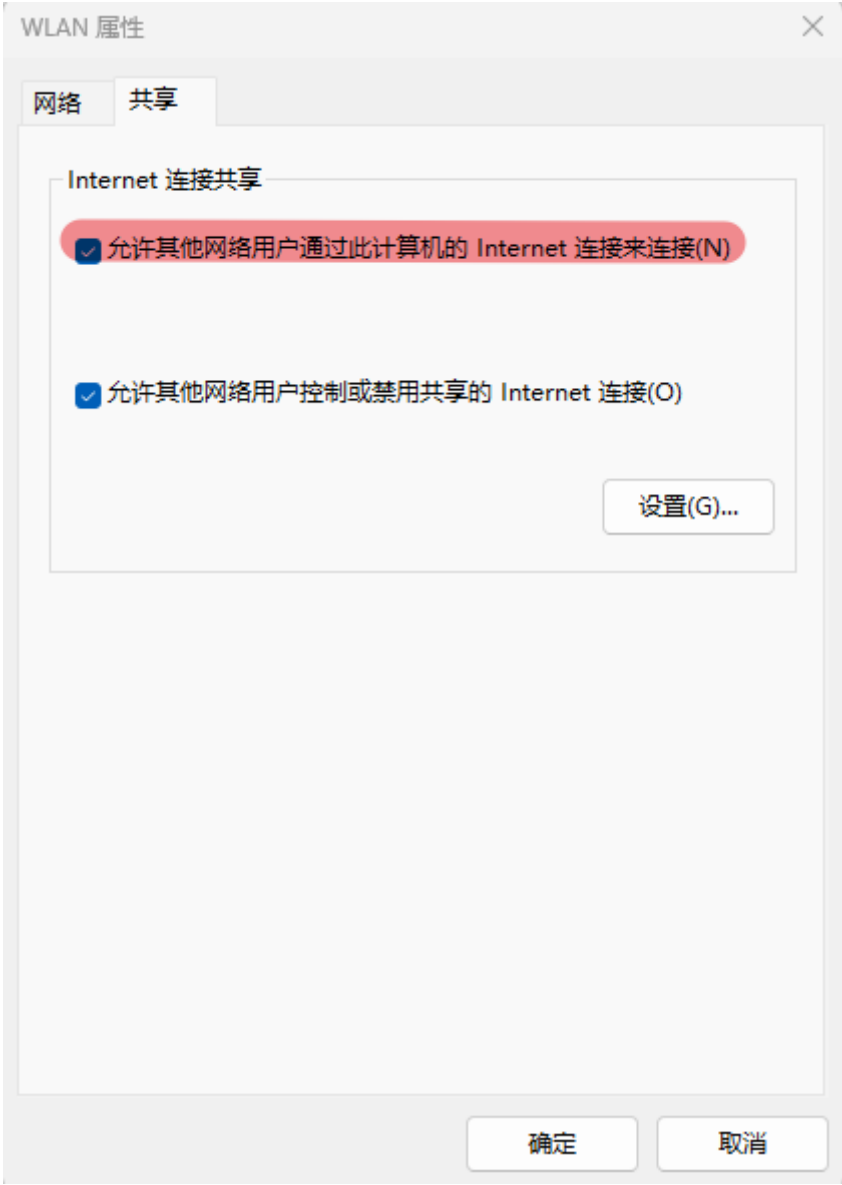
3.连接成功后，若手机上能看见树莓派的ip地址，则说明连接成功。如果有连接显示屏，也可以在树莓派上输入ifconfig，查看ip地址。

4.在cmd中使用ssh或者采用Xshell等工具连接树莓派，用户名为root，密码为openeuler。

方法二：连接网线

1.打开控制面板找到网络和Internet选项，点击高级网络设置，在wifi的更多适配器选项中点击编辑，在共享选项卡上选中“允许其他网络用户通过此计算机的Internet连接来连接”选项。


设置
网络和 Internet › 高级网络设置
网络适配器
以太网
未连接 | Realtek PCIe GbE Family Controller
禁用
蓝牙网络连接
Bluetooth Device (Personal Area Network)
禁用
WLAN
XMUNET+ | RZ616 Wi-Fi 6E 160MHz
禁用
媒体状态: 已启用
发送的字节数: 154,548
接收的字节数: 285,179
链接速度: 400 (Mbps)
持续时间: 00:00:16
重命名此适配器
重命名
查看其他属性
更多适配器选项
编辑
更多设置
高级共享设置
更改网络发现和共享设置
数据使用量



2.使用网线连接电脑和树莓派。

3.在电脑命令行窗口输入arp –a查看新增加的IP地址即为raspberry pi的ip地址，说明树莓派以太网连接成功。

注意：ip地址一般为一般为192.168.137.x。若首次连接时能看见ip地址，一段时间后重启树莓派看不见ip地址，请将1中的“允许其他网络用户通过此计算机的Internet连接来连接”取消，点击确定后再重新勾选。

```
C:\environment\git>arp -a

接口: 10.30.67.241 --- 0x3
  Internet 地址         物理地址              类型
  10.30.64.1            40-fe-95-fe-80-01     动态
  10.30.95.255          ff-ff-ff-ff-ff-ff     静态
  224.0.0.22            01-00-5e-00-00-16     静态
  224.0.0.251           01-00-5e-00-00-fb     静态
  224.0.0.252           01-00-5e-00-00-fc     静态
  225.4.172.59          01-00-5e-04-ac-3b     静态
  236.70.63.36          01-00-5e-46-3f-24     静态
  237.196.75.49         01-00-5e-44-4b-31     静态
  239.255.255.250       01-00-5e-7f-ff-fa     静态
  255.255.255.255       ff-ff-ff-ff-ff-ff     静态

接口: 192.168.137.1 --- 0xe
  Internet 地址         物理地址              类型
  192.168.137.229       dc-a6-32-52-cd-f6     静态
  192.168.137.255       ff-ff-ff-ff-ff-ff     静态
  224.0.0.22            01-00-5e-00-00-16     静态
  224.0.0.251           01-00-5e-00-00-fb     静态
  224.0.0.252           01-00-5e-00-00-fc     静态
  239.255.255.250       01-00-5e-7f-ff-fa     静态
  255.255.255.255       ff-ff-ff-ff-ff-ff     静态

C:\environment\git>
```

4.在cmd中使用ssh或者采用Xshell等工具连接树莓派，用户名为root，密码为openeuler。

步骤4 安装所需组件

1.安装yum

```
dnf install yum
```

2.安装unzip，tar，lrzsz，gcc，make，flex，bison，openssl-devel，perl

```
yum -y install unzip
yum -y install tar
yum -y install lrzsz
yum -y install gcc
yum -y install make
yum -y install flex
yum -y install bison
yum -y install openssl-devel
yum -y install perl
```

3.2 openEuler内核编译与安装

步骤1 备份boot目录以防后续步骤更新内核失败

```
tar czvf boot_origin.tgz /boot/
sz boot_origin.tgz    # 将备份文件发送到本地，该命令使用Xshell会比较方便
```

步骤2 获取内核源码并解压

1.在以下网址下载openEuler-22.03-LTS-SP2的内核：

https://gitee.com/openeuler/raspberrypi-kernel/tree/openEuler-22.03-LTS-SP2/

2.输入rz或采用Xftp等工具，将源码压缩包传入树莓派。

3.解压内核源码

```
unzip raspberrypi-kernel-openEuler-22.03-LTS-SP2.zip
mkdir -p /root/kernel && mv /root/raspberrypi-kernel-openEuler-22.03-LTS-SP2/*
/root/kernel && rm -rf /root/raspberrypi-kernel-openEuler-22.03-LTS-SP2 #重命名
```

步骤3 编译内核

1.加载默认配置

```
cd kernel
make bcm2711_defconfig
```

```
[root@openEuler kernel]# make bcm2711_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/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.编译内核（后续步骤强烈建议课后完成！）

耗时长，在树莓派中需要4个小时左右。

```
make ARCH=arm64 -j4
```

```
[root@openEuler kernel]# make ARCH=arm64 -j4
  WRAP    arch/arm64/include/generated/uapi/asm/kvm_para.h
  WRAP    arch/arm64/include/generated/uapi/asm/errno.h
  WRAP    arch/arm64/include/generated/uapi/asm/ioctl.h
  WRAP    arch/arm64/include/generated/uapi/asm/ioctls.h
  WRAP    arch/arm64/include/generated/uapi/asm/ipcbuf.h
  WRAP    arch/arm64/include/generated/uapi/asm/msgbuf.h
  WRAP    arch/arm64/include/generated/uapi/asm/poll.h
  WRAP    arch/arm64/include/generated/uapi/asm/resource.h
  WRAP    arch/arm64/include/generated/uapi/asm/sembuf.h
  WRAP    arch/arm64/include/generated/uapi/asm/shmbuf.h
  WRAP    arch/arm64/include/generated/uapi/asm/siginfo.h
  WRAP    arch/arm64/include/generated/uapi/asm/socket.h
  WRAP    arch/arm64/include/generated/uapi/asm/sockios.h
  WRAP    arch/arm64/include/generated/uapi/asm/stat.h
  WRAP    arch/arm64/include/generated/uapi/asm/swab.h
  WRAP    arch/arm64/include/generated/uapi/asm/termbits.h
  WRAP    arch/arm64/include/generated/uapi/asm/termios.h
  WRAP    arch/arm64/include/generated/uapi/asm/types.h
  HOSTCC  scripts/dtc/dtc.o
  HOSTCC  scripts/dtc/flattree.o
  WRAP    arch/arm64/include/generated/asm/early_ioremap.h
  WRAP    arch/arm64/include/generated/asm/mcs_spinlock.h
  WRAP    arch/arm64/include/generated/asm/qrwlock.h
  WRAP    arch/arm64/include/generated/asm/set_memory.h
  UPD     include/config/kernel.release
  WRAP    arch/arm64/include/generated/asm/user.h
  WRAP    arch/arm64/include/generated/asm/bugs.h
```

3.创建编译内核模块目录

```
mkdir ../output
```

4.编译内核模块

```
make INSTALL_MOD_PATH=../output/modules_install
```

至此，内核编译完成

步骤4 切换内核

1.查看当前内核版本

```
uname -a
```

```
[root@openEuler kernel]# uname -a
Linux openEuler 5.10.0-182.0.0.19.oe2203sp3.raspi.aarch64 #1 SMP PREEMPT Sat Dec 30 13:16:02 CST 2023 aarch64 aarch64 aarch64 GNU/Linux
```

2.将内核放进引导

```
cp arch/arm64/boot/Image /boot/kernel8.img
```

3.将设备树文件放进引导

```
cp arch/arm64/boot/dts/broadcom/*.dtb /boot/
cp arch/arm64/boot/dts/overlays/*.dtb* /boot/overlays/
cp arch/arm64/boot/dts/overlays/README /boot/overlays/
```

4.重启系统

（如果启动失败，将备份好的boot_origin.tgz解压覆盖到/boot中并重启）

5.查看新的内核版本

```
uname -a
```

```
[root@openEuler /]# uname -a
Linux openEuler 5.10.0-v8 #1 SMP PREEMPT Thu Mar 7 20:56:46 CST 2024 aarch64 aarch64 aarch64 GNU/Linux
```

tips：网络配置

如果是采用wifi连接的方式，成功切换内核后，系统将找不到wlan0，导致无法联网。请将树莓派连接键盘，输入以下命令，随后才能连接上wifi。

```
insmod ~/output/lib/modules/5.10.0-v8/kernel/net/rfkill/rfkill.ko
insmod ~/output/lib/modules/5.10.0-
v8/kernel/drivers/net/wireless/broadcom/brcm80211/brcmutil/brcmutil.ko
insmod ~/output/lib/modules/5.10.0-v8/kernel/net/wireless/cfg80211.ko
insmod ~/output/lib/modules/5.10.0-
v8/kernel/drivers/net/wireless/broadcom/brcm80211/brcmfmac/brcmfmac.ko
```

执行完上述命令后输入ifconfig，将能够查看到wlan0，从而能连接上wifi。

```
ifconfig
```

注意，上述命令每次重启树莓派后都需重新输入。为保证每次启动树莓派，无需重复输入上述命令，可以修改rc.local文件，使得每次启动时自动执行。

```
vim /etc/rc.local
```

在rc.local中增添四行代码。

```
insmod ~/output/lib/modules/5.10.0-v8/kernel/net/rfkill/rfkill.ko
insmod ~/output/lib/modules/5.10.0-
v8/kernel/drivers/net/wireless/broadcom/brcm80211/brcmutil/brcmutil.ko
insmod ~/output/lib/modules/5.10.0-v8/kernel/net/wireless/cfg80211.ko
insmod ~/output/lib/modules/5.10.0-
v8/kernel/drivers/net/wireless/broadcom/brcm80211/brcmfmac/brcmfmac.ko
```

重新启动。

```
reboot
```

3.3 内核编程入门示例 - Hello，world!

接下来我们可以在新内核中完成后续实验，这里先以一个简单的 HelloWorld 为示例，展示内核编程的基本过程。

步骤1 正确编写满足功能的源文件，包括.c源文件和Makefile文件，并放在同一目录下如/root/exp0/。

```
mkdir -p /root/exp0/
touch /root/exp0/hello_world.c
touch /root/exp0/Makefile
cd /root/exp0
```

Hello_world.c文件内容如下

```c
/* Hello from Kernel! */
#include <linux/module.h>
MODULE_LICENSE("GPL");
static char* guy = "Kernel";
module_param(guy, charp, 0644);
MODULE_PARM_DESC(guy, "char* param\n");
static int year = 2021;
module_param(year, int, 0644);
MODULE_PARM_DESC(year, "int param\n");
void hello_world(char* guy, int year)
{
 printk(KERN_ALERT "Hello, %s, %d!\n", guy, year);
}
int __init hello_init(void)
{
 printk(KERN_ALERT "Init module.\n");
 hello_world(guy, year);
 return 0;
}
void __exit hello_exit(void)
{
 printk(KERN_ALERT "Exit module.\n");
}
module_init(hello_init);
module_exit(hello_exit);
```

Makefile文件内容如下

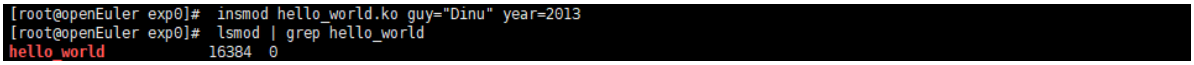
```
# Build module hello_world
ifneq ($(KERNELRELEASE),)
        obj-m := hello_world.o
else
        KERNELDIR ?=/root/kernel
        PWD := $(shell pwd)
default:
        $(MAKE) -C $(KERNELDIR) M=$(PWD) modules
endif
.PHONY:clean
clean:
        -rm *.mod.c *.o *.order *.symvers *.ko
```

步骤2 编译源文件

```
make
```

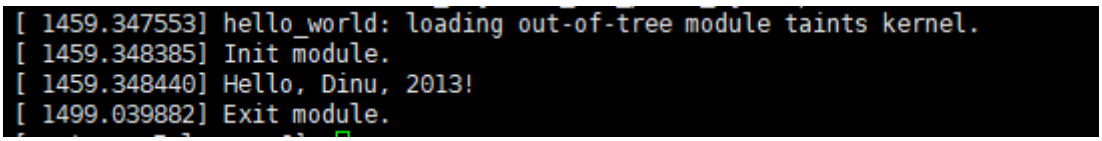
步骤3 加载编译完成的内核模块，并查看加载结果

```
 insmod hello_world.ko guy="Dinu" year=2013
 lsmod | grep hello_world
```



步骤4 卸载内核模块，并查看结果

```
rmmod hello_world
dmesg | tail -n4
```



3.4 任务一

查找相关资料，解释hello_world.c文件中以下代码的含义和作用

[1]MODULE_LICENSE

[2]module_param

[3]MODULE_PARM_DESC

[4]module_init

[5]module_exit

[6]__init

[7]__exit

3.5 任务二

请参考hello_world.c 和 Makefile 文件，编写hello_magic_student.c 和 Makefile，完成以下任务：

1.在 hello_magic_student.c 文件中定义函数 hello_student(...)，该函数包含 3 个参数：id, name, age，分别代 表学号、姓名和年龄，并通过printk输出：” My name is ${name}, student id is ${id}. I am ${age} years old.”

2.在 hello_magic_student.c 文件中定义函数 my_magic_number(...)，该函数包含 2 个参数：id 和 age，分别代表学号和年龄。请你在该函数里将学号的每一位数字相加后再与年龄求和，将求和结果的个位数作为magic_number，并使用printk 输出：”My magic number is ${magic_number}.”。

完成hello_magic_student.c 文件的编写后，参考 hello_world 模块的 Makefile 并适当调整，在加载内核时提供学号、姓名和年龄，通过dmesg命令查看printk的输出。

P.S. 在接下来的内核编程实验中，我们可能会用到需要其他内核函数，同学们可以在 https://manpages.org/ 上查找指定函数的功能和用法

4.提交要求

1.实验报告

2.hello_magic_student.c 源代码文件

3.用于编译hello_magic_student.c 的 Makefile 文件。

其中，实验报告需包含但不限于以下内容：

a)正确编译内核源码，并完成安装，提交新旧内核版本的截图。

b)对实验任务一，提供解答以及对参考资料出处的引用。

c)对实验任务二，提供对hello_magic_student.c 中关键代码的说明，对实验步骤的描述，以及能显示实验正确完成的结果截图。

d)实验心得体会。