实验 1：openEuler 操作系统 编译内核实验

姓名：李梓涵　　学号：34520212201574

一、 实验目的

1. 学习掌握如何在树莓派上安装操作系统。
2. 学习掌握如何编译操作系统内核。
3. 了解内核模块编程

二、 实验过程

1. openEuler 操作系统安装

   烧录系统后，使用网线连接电脑，在 cmd 中用 ssh 连接。

   注意：烧录完系统后，再次打开系统盘，在根目录下新建一 ssh 文件，这样才能正常进行 ssh 连接。

```
C:\Users\123>ssh root@192.168.137.192
root@192.168.137.192's password:
Permission denied, please try again.
root@192.168.137.192's password:
Permission denied, please try again.
root@192.168.137.192's password:
Last failed login: Wed May  1 11:40:54 CST 2024 from 192.168.137.1 on ssh:notty
There were 2 failed login attempts since the last successful login.


Welcome to 5.10.0-182.0.0.19.oe2203sp3.raspi.aarch64

System information as of time:  Wed May  1 11:41:28 AM CST 2024

System load:    0.00
Processes:      130
Memory used:    .9%
Swap used:      0%
Usage On:       4%
IP address:     192.168.137.192
Users online:   1


[root@openEuler ~]# nmcli dev wifi connect Redmi_8D6A password tangjiawei
Device 'wlan0' successfully activated with 'b67fc266-6787-4b93-89d6-59b332874cf4'.
[root@openEuler ~]# exit
logout
Connection to 192.168.137.192 closed.

C:\Users\123>
```

连接成功界面，注意用户名为root，初始时密码为openeuler

2. openEuler 内核编译与安装

   下载并解压内核源码后，按照实验手册进行编译，等待大约三个半小时后编译成功

```
  LD [M]  sound/usb/6fire/snd-usb-6fire.ko
  LD [M]  sound/usb/caiaq/snd-usb-caiaq.ko
  LD [M]  sound/usb/hiface/snd-usb-hiface.ko
  LD [M]  sound/usb/line6/snd-usb-line6.ko
  LD [M]  sound/usb/line6/snd-usb-toneport.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
[root@openEuler kernel]# mkdir ../output
[root@openEuler kernel]# make INSTALL_MOD_PATH=../output/modules_install
  CALL    scripts/checksyscalls.sh
  CALL    scripts/atomic/check-atomics.sh
  CHK     include/generated/compile.h
[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
[root@openEuler kernel]#
```

此为旧内核版本

重启后得到新内核版本

3. 任务 1

（1） MODULE_LICENSE：声明此模块的许可证，代码中指定为 GPL。缺少此声明会给出内核被污染的警告。

注释掉该行代码后的编译运行过程如下：

```
bash-5.1# make
make -C /root/kernel M=/root/exp0 modules
make[1]: Entering directory '/root/kernel'
  CC [M]  /root/exp0/hello_world.o
  MODPOST /root/exp0/Module.symvers
WARNING: modpost: missing MODULE_LICENSE() in /root/exp0/hello_world.o
  CC [M]  /root/exp0/hello_world.mod.o
  LD [M]  /root/exp0/hello_world.ko
make[1]: Leaving directory '/root/kernel'
bash-5.1# insmod hello_world.ko guy="Li" year=2024
bash-5.1# lsmod | grep hello_world
hello_world            16384  0
bash-5.1# rmmod hello_world
bash-5.1# dmesg | tail -n4
[ 3271.667248] Disabling lock debugging due to kernel taint
[ 3271.668097] Init module.
[ 3271.668120] Hello, Li, 2024!
[ 3286.128281] Exit module.
```

首先在编译时给出缺少 MODULE_LICENSE 声明的警告，然后在输出文件中给出由于内核污染导致的 Disabling lock debugging 日志。

https://blog.csdn.net/kwame211/article/details/77531748

（2） module_param (name, type, perm)：传递命令行参数，形参依次指定了变量名、变量类型和访问参数的权限，其中 0644 规定了所有者可读可写，其他人只读

https://blog.csdn.net/wangxu696200/article/details/123554397

（3） MODULE_PARM_DESC：描述驱动模块的参数信息。在,ko 文件中记录，提供给用户参考。可以使用 modinfo 指令查看。

```
问题  输出  终端  端口  调试控制台

bash-5.1# modinfo hello_world.ko
filename:       /root/exp0/hello_world.ko
srcversion:     6716CD14BB95D7EB2776D02
depends:
name:           hello_world
vermagic:       5.10.0-v8 SMP preempt mod_unload modversions aarch64
parm:           guy:char* param
 (charp)
parm:           year:int param
 (int)
bash-5.1#
```

https://blog.csdn.net/wangxu696200/article/details/123555378

（4） module_init(x)：驱动程序初始化的入口。在内核启动时或模块插入时被调用，执行

函数 x，每个模块只能有一个。
https://manpages.org/module_init/9

（5） module_exit(x)：驱动程序退出的出口。当驱动程序是模块时，使用命令 rmmod 时调用该函数，进而调用函数 x，同时使用 cleanup_module 包装驱动程序清理代码。如果驱动程序静态编译到内核中，则不起作用。每个模块只能有一个。
https://manpages.org/module_exit/9

（6） __init：宏定义，告知编译器将变量或函数放在特殊区域。标记函数为初始化函数，仅在模块初始化时使用，模块装载后就卸载函数，释放内存。

（7） __exit：宏定义，告知编译器，将函数放在".exit.text"这个区域中。仅对模块有用，模块稳定时不会使用，当模块支持无效时，将__exit 定义的部分丢掉。
https://blog.csdn.net/qingkongyeyue/article/details/72935439

4. 任务 2

（1） 仿照任务 1 中的.c 文件编写 hello_magic_student.c 文件
首先引入头文件、声明模块许可证，然后进行变量的定义、接收命令行参数、变量声明，之后定义函数，最后规定模块初始化的入口和模块的出口。

```
/* hello_magic_student form Krenel! */
#include <linux/module.h>
MODULE_LICENSE("GPL");
static int id = 0;
module_param(id, int, 0644);
MODULE_PARM_DESC(id, "int param\n");
static char* name = "Kernel";
module_param(name, charp, 0644);
MODULE_PARM_DESC(name, "char* param\n");
static int age = 21;
module_param(age, int, 0644);
MODULE_PARM_DESC(age, "int param\n");
void hello_student(int id, char* name, int age){
    printk(KERN_ALERT "My name is %s, student id is %d, I am %d years old.\n",
name, id, age);
}
void hello_magic_student(int id, int age){
    printk(KERN_ALERT "My magic number is %d.\n", id + age);
}

int __init hello_init(void)
{
printk(KERN_ALERT "Init module.\n");
hello_student(id, name, age);
hello_magic_student(id, age);
return 0;
}
void __exit hello_exit(void)
```

```
{
printk(KERN_ALERT "Exit module.\n");
}
module_init(hello_init);
module_exit(hello_exit);
```

（2） 仿照任务 1 中的 Makefile 编写 Makefile 文件

```
# Build module hello_world
ifneq ($(KERNELRELEASE),)
    obj-m := hello_magic_student.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
```

（3） 运行结果如下：

```
bash-5.1# insmod hello_magic_student.ko id=1 name="Li" age=25
bash-5.1# lsmod | grep hello_magic_student
hello_magic_student    16384  0
bash-5.1# rmmod hello_magic_student
bash-5.1# dmesg | tail -n4
[ 2200.993774] Init module.
[ 2200.993820] My name is Li, student id is 1, I am 25 years old.
[ 2200.993840] My magic number is 26.
[ 2215.001403] Exit module.
```

三、 实验小结

1. 本次实验主要花费时间的部分在于配置树莓派系统并编译内核部分。在使用网线连接树莓派前，要先在其根目录下新建 ssh 文件，以确保能正确进行 ssh 连接。
2. 在内核编程部分，其与普通的 C 语言程序有较大的不同，需要用一些宏定义声明模块许可等部分；对于参数的定义和接收也有所不同，使用 printk 函数进行输出，module_param 函数接收命令行参数。
3. 在 Makefile 文件中，首先确认是否是在内核构建环境中运行，如果不是，就设置一些变量并调用内核的 make 进行构建。