1 编译内核

首先进入内核所在目录，创建新的git分支

```
git branch lzzz
git checkout lzzz
```

然后修改Makefile，添加自己的名字后缀，提交更改

```
vim Makefile
add "lzzz" after EXTRAVERSION
git commit -asm "change EXTRAVERSION"
```

配置内核

```
cp /boot/config-4.15.0-36-generic .config
make menuconfig //选择默认配置 Save Exit
```

编译内核 -- make

```
[error 1]
./scripts/ubuntu-retpoline-extract-one: No such file or directory
[solution]
cp debian/scripts/retpoline-extract-one scripts/ubuntu-retpoline-extract-one

[error 2]
Unable to find 'openssl/opensslv.h'
[solution]
sudo apt install libssl-dev
```

安装内核

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

重启加载，查看内核版本

```
sudo reboot
uname -a // 4.15.18lzzz+
```

2 完成hello-5.c

```
make
sudo insmod ./hello-5.ko
lsmod # 有hello_5的信息说明加载成功
```

```
# Makefile
obj-m += hello-5.o
all:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) modules
clean:
    make -C /lib/modules/$(shell uname -r)/build M=$(PWD) clean
```

```
/*
 *  hello-5.c - Demonstrates command line argument passing to a module.
 */
#include <linux/module.h>
#include <linux/moduleparam.h>
#include <linux/kernel.h>
#include <linux/init.h>
#include <linux/stat.h>

MODULE_LICENSE("GPL");
MODULE_AUTHOR("Lzzz");

static short int myshort = 1;
static int myint = 420;
static long int mylong = 9999;
static char *mystring = "hahaha";
static int myintArray[2] = { -1, -1 };
static int arr_argc = 0;

/*
 * module_param(foo, int, 0000)
 * The first param is the parameters name
 * The second param is it's data type
 * The final argument is the permissions bits,
 * for exposing parameters in sysfs (if non-zero) at a later stage.
 */

module_param(myshort, short, S_IRUSR | S_IWUSR | S_IRGRP | S_IWGRP);
MODULE_PARM_DESC(myshort, "A short integer");
module_param(myint, int, S_IRUSR | S_IWUSR | S_IRGRP | S_IROTH);
MODULE_PARM_DESC(myint, "An integer");
module_param(mylong, long, S_IRUSR);
MODULE_PARM_DESC(mylong, "A long integer");
module_param(mystring, charp, 0000);
MODULE_PARM_DESC(mystring, "A character string");

/*
 * module_param_array(name, type, num, perm);
 * The first param is the parameter's (in this case the array's) name
 * The second param is the data type of the elements of the array
 * The third argument is a pointer to the variable that will store the number
 * of elements of the array initialized by the user at module loading time
 * The fourth argument is the permission bits
 */
module_param_array(myintArray, int, &arr_argc, 0000);
```

```
MODULE_PARM_DESC(myintArray, "An array of integers");

static int __init hello_5_init(void)
{
    int i;
    printk(KERN_INFO "Hello, world 5\n=============\n");
    printk(KERN_INFO "myshort is a short integer: %hd\n", myshort);
    printk(KERN_INFO "myint is an integer: %d\n", myint);
    printk(KERN_INFO "mylong is a long integer: %ld\n", mylong);
    printk(KERN_INFO "mystring is a string: %s\n", mystring);
    for (i = 0; i < (sizeof myintArray / sizeof (int)); i++)
    {
        printk(KERN_INFO "myintArray[%d] = %d\n", i, myintArray[i]);
    }
    printk(KERN_INFO "got %d arguments for myintArray.\n", arr_argc);
    return 0;
}

static void __exit hello_5_exit(void)
{
    printk(KERN_INFO "Goodbye, world 5\n");
}

module_init(hello_5_init);
module_exit(hello_5_exit);
```

3 Linux Kernel Coding Style

(1) 布局规范

合理的代码布局可以使程序更清晰，视觉上更舒适。

缩进：

使用8字符的tab，不用空格缩进，"case"语句与"switch"对齐不缩进

大括号：

一般情况左括号不换行，右括号单独成行。

函数语句左括号不换行，do-while语句和if-else-if语句右括号不单独成行。

只有一行内容时，大括号可以省略，而if-else语句只要有一个分支超过一行，所有分支都加大括号。

空格：

```
if switch case for do while // 后加空格
=  +  -  <  >  *  /  %  |  &  ^  <=  >=  ==  !=  ?  : // 左右加空格
&  *  +  -  ~  !  sizeof  typeof  alignof  __attribute__  defined // 单目运算符后不加空格
++ -- . -> // 不加空格
// 行末不留空格
```

换行：

一行最多80的字符宽度，如果超过需要换行

注意不能拆开特定的信息，否则会影响grep查找

(2) 注释和文档

注释：

格式符合规范，应该说what而不是how，不要过于详细、复杂

Kconfig文件：

符合Kconfig的规范

Print kernel message：

符合具体文档的要求

(3) 函数规范

函数应该简洁，清楚，只做一件特定的事情

函数的长度应该与复杂度负相关：

对逻辑简单的函数，可以具体入微，较长一点

对逻辑复杂的函数，需要使用helper function，简化函数逻辑

函数的局部变量建议在5-10个

函数间需要空行，如果有EXPORT_SYMBEL，紧接在函数之后

函数的出口要集中，如果使用goto语句，label需要规范

只在适当的时候使用内联函数，一般是函数体不超过三行，或是编译器可以优化的特定情况

函数的返回值要满足要求，如果函数名是行为，0表示成功，如果函数名是预测，1表示成功

(4) 宏、类型定义

为结构体和指针使用typedef是错误的

用宏定义的相关常数应使用ENUM

宏函数需要符合几个规范

学会使用kernel macros

(5) 编写规范

一行只书写一条语句或定义

命名规范：

局部变量名要简短，全局变量/函数名称要清楚表达功能

可以使用goto语句的情况：

```
unconditional statements are easier to understand and follow
nesting is reduced
errors by not updating individual exit points when making modifications are prevented
saves the compiler work to optimize redundant code away ;)
```

Count reference：

对可以被外部访问的数据结构，一定要有reference counter

Allocating memory：

规范地分配内存

不要在文件的开始添加编辑器的config信息

条件编译和内联汇编规范

4 Modify Hello-5.c

用scripts/checkpatch.pl查找hello-5.c中不符合代码规范的地方

- 行末多余空格
- module_param第三个参数建议使用八进制
- 静态变量不应初始化为0
- printk不建议使用
- sizeof后应有括号，没有空格
- 文件结尾应有空行

修改结束后：

```
total: 0 errors, 0 warnings, 71 lines checked
hello-5.c has no obvious style problems and is ready for submission.
```

5 Kernel Modules

- localmodconfig

  ```
  make localmodconfig
  cat .config | grep -i '=m' | wc -l
  ```

  查看.config文件，其中xxx=m配置为module，共110个

- allmodconfig

  ```
  make allmodconfig
  cat .config | grep -i '=m' | wc -l
  ```

  module共6385个

- /boot目录下的config

```
cp /boot/config-4.15.0-36-generic .config
cat .config | grep -i '=m' | wc -l
```

module共4936个

- alldefconfig

```
make alldefconfig
cat .config | grep -i '=m' | wc -l
```

module 0个

6 Kernel Size

内核的构建过程如下：

- 生成内核链接脚本vmlinux.lds
- 内核子目录built-in.o、模块、符号表的生成
- elf格式vmlinux与vmlinux.bin的生成
- 压缩elf格式vmlinux与压缩vmlinux.bin生成
- setup.elf与setup.bin生成
- setup.bin和vmlinux.bin生成最终的内核镜像bzImage
- 最终vmlinuz的生成

其中elf格式的vmlinux和vmlinux.bin由arch/x86/kernel/vmlinux.lds和各级子目录的built-in.o链接生成

vmlinux.bin由vmlinux用objcopy删掉.comment节、符号表、重定位表生成，在路径arch/x86/boot/compressed

压缩的vmlinux路径同上

压缩的vmlinux.bin路径在arch/x86/boot，之后的setup.elf和setup.bin也在这个路径下，最终由setup.bin和vmlinux.bin生成内核镜像bzImage

内核运行前，先由bzImage解压，再装入内存，最终大小与压缩成为bzImage前的vmlinux.bin大小接近

- localmodconfig

```
make localmodconfig
make
cd arch/x86/boot
ls -l vmlinux.bin
# -rw-rw-r-- 1 lzzz lzzz 12796888 Oct 15 14:54 vmlinux.bin
```

压缩前的vmlinux为114M，vmlinux.bin约13M，bzImage大小也约13M.

- allnoconfig

```
make allnoconfig
make
cd arch/x86/boot/compressed
```

vmlinux大小为1.47M.

- allyesconfig

  ```
  make allyesconfig
  ```

  vmlinux大小为800M左右。

- /boot/configxxx

  ```
  cp /boot/configxxx .config
  make
  ```

  vmlinux大小为580M左右

  arch/x86/boot/vmlinux.bin 大小为8M

  arch/x86/boot/bzImage 大小为8M

  arch/x86/boot/vmlinux 大小为10M