

ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

MÔN: HỆ ĐIỀU HÀNH

Lập trình Linux Kernel Module

Học viên:

Đoàn Minh Hiếu – 1612198

Giảng viên:

Trần Trung Dũng



September 29, 2018

1 Kernel Module là gì

Kernel module là một file chứa code dùng để mở rộng chức năng của kernel. Nó có thể được load hoặc unload. Khi Kernel modules được load, nó giúp kernel đang chạy của hệ điều hành hỗ trợ các thiết bị phần cứng mới cũng như các file hệ thống.

Nếu không có kernel modules, chúng ta phải xây dựng monolithic kernel và thêm chức năng mới trực tiếp vào kernel. Điều này làm cho kernel trở nên rất lớn và đòi hỏi phải khởi động lại khi chúng ta muốn một chức năng mới từ hệ điều hành.

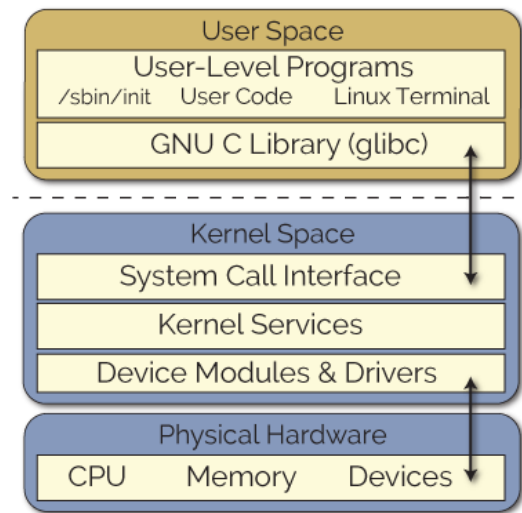


Figure 1: Linux userspace and kernel space

2 Thêm module vào Kernel

Chúng ta cùng tìm hiểu việc thêm một module đơn giản vào Kernel đang chạy thông qua một ví dụ helloworld.

2.1 Tạo module

Vì module bản chất là một file chứa code nên ta phải viết code để tạo ra module. Ngôn ngữ được sử dụng là C với các thư viện trong linux header.

- Cài đặt linux headers: gõ lệnh sau trong terminal

```
$ sudo apt-get install linux-headers-$(uname -r)
```

- Để tham khảo các thư viện hỗ trợ trong linux headers, ta vào link sau để đọc: (nhớ chọn đúng phiên bản)

<https://elixir.bootlin.com/linux/v4.15/source/include/linux>

Sử dụng ngôn ngữ C để viết một module đơn giản:

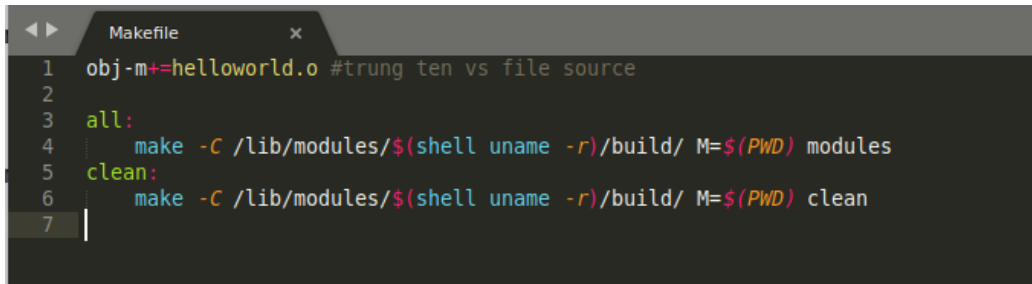
```
1 #include <linux/module.h> //Needed by all module
2 #include <linux/init.h> //Need for the marcos
3 #include <linux/kernel.h> //Needed for KERN_INFO
4 static int hello_init(void){
5     printk(KERN_ALERT "Hello world"); //print message into /var/log
6     return 0;
7 }
8 static void hello_exit(void){
9     printk(KERN_ALERT "Goodbye");
10 }
11 module_init(hello_init); //marco inform where module start
12 module_exit(hello_exit); //marco inform where module exit
```

Đặt tên file là *helloworld.c*, ta thử chèn vào kernel, nếu thành công, message "Helloworld" sẽ xuất ra trong file */var/log/message*. Sau đó remove module, nếu thành công, message "Goodbye" được xuất ra.

2.2 Biên dịch module

Ta sử dụng terminal để biên dịch file *helloworld.c* và tạo ra module *helloworld.ko*.

Để quá trình biên dịch dễ dàng, ta viết 1 file *Makefile* như sau:



```

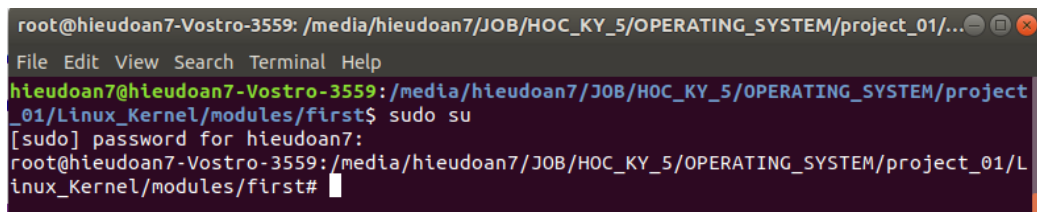
1 obj-m+=helloworld.o #trung ten vs file source
2
3 all:
4     make -C /lib/modules/$(shell uname -r)/build/ M=$(PWD) modules
5 clean:
6     make -C /lib/modules/$(shell uname -r)/build/ M=$(PWD) clean
7

```

Figure 2: Makefile

Makefile là file hỗ trợ lệnh terminal, khi ta gõ lệnh make thì Makefile sẽ mặc định vào tag all:, khi gõ lệnh make clean sẽ vào tag clean: và thực hiện command đó.

- Mở terminal tại thư mục chứa file *helloworld.c*, chúng ta ở chế độ root

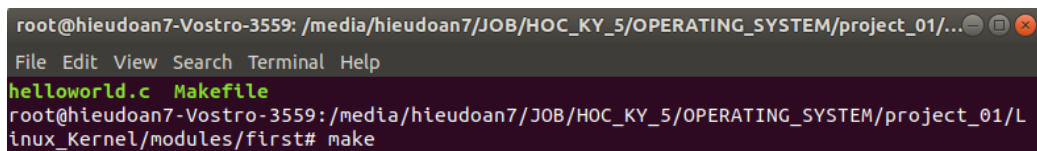


```

root@hieudoan7-Vostro-3559: /media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/project_01/...
File Edit View Search Terminal Help
hieudoan7@hieudoan7-Vostro-3559: /media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/project_01/Linux_Kernel/modules/first$ sudo su
[sudo] password for hieudoan7:
root@hieudoan7-Vostro-3559: /media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/project_01/Linux_Kernel/modules/first#

```

- Gõ lệnh make để biên dịch

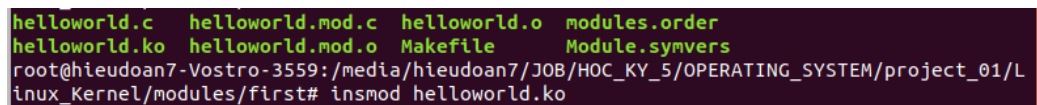


```

root@hieudoan7-Vostro-3559: /media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/project_01/...
File Edit View Search Terminal Help
hieudoan7@hieudoan7-Vostro-3559: /media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/project_01/Linux_Kernel/modules/first$ make

```

- Lúc này trong thư mục sẽ tạo ra rất nhiều file, trong đó có file module: *helloworld.ko*
- Insert module *helloworld.ko* vào Kernel



```

hieudoan7@hieudoan7-Vostro-3559: /media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/project_01/Linux_Kernel/modules/first$ insmod helloworld.ko

```

- Kiểm tra thông tin module vừa insert

```
root@hieudoan7-Vostro-3559:/media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/project_01/L
inux_Kernel/modules/first# modinfo helloworld.ko
filename:      /media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/project_01/L
inux_Kernel
/modules/first/helloworld.ko
srcversion:    03B0722FBBD40126B8F7792
depends:
retpoline:    Y
name:         helloworld
vermagic:     4.15.0-34-generic SMP mod_unload
root@hieudoan7-Vostro-3559:/media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/project_01/L
```

- Kiểm tra message

```
root@hieudoan7-Vostro-3559:/media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/project_01/L
inux_Kernel/modules/first# dmesg
[50540.790255] Hello world
```

- Remove module *helloworld.ko*

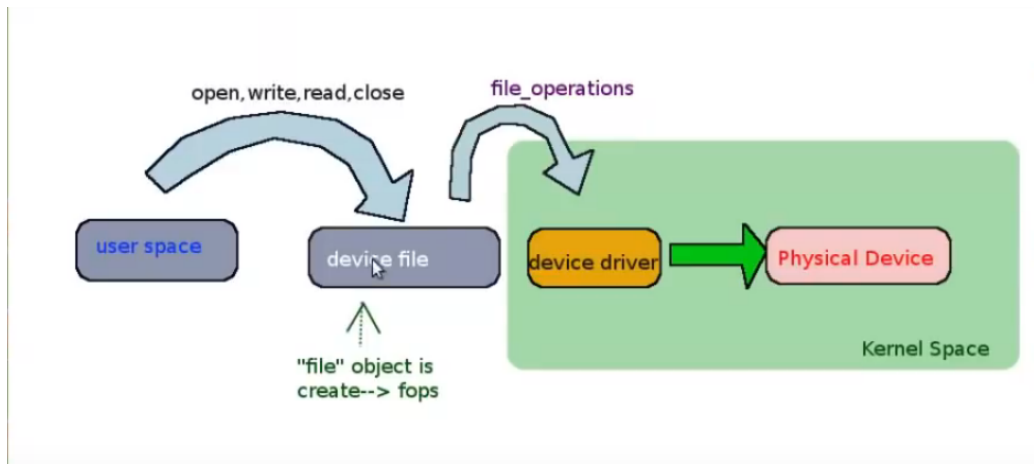
```
root@hieudoan7-Vostro-3559:/media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/project_01/L
inux_Kernel/modules/first# rmmod helloworld.ko
```

- Kiểm tra message

```
[51560.339305] Goodbye
```

3 Device Driver

Device Driver là một lớp các module có chức năng điều khiển các thiết bị phần cứng như ổ đĩa, usb, chuột. Trong Unix, mỗi thiết bị phần cứng được đại diện bởi một file nằm ở */dev* tên là **Device File**, cung cấp phương tiện để giao tiếp với phần cứng.



Các **Device File** trong máy tính:

```

root@hieudoan7-Vostro-3559: /dev
File Edit View Search Terminal Help
root@hieudoan7-Vostro-3559:/home/hieudoan7# cd /dev
root@hieudoan7-Vostro-3559:/dev# ls
autofs          loop13          sda4            tty38            ttyS24
block           loop14          sda5            tty39            ttyS25
bsg             loop15          sda6            tty4             ttyS26
btrfs-control  loop16          sda7            tty40            ttyS27
bus             loop17          sda8            tty41            ttyS28
cdrom           loop18          sda9            tty42            ttyS29
cdrw            loop19          sg0             tty43            ttyS3
char            loop2           sg1             tty44            ttyS30
console         loop20          shm             tty45            ttyS31
core            loop21          snapshot        tty46            ttyS4
cpu             loop22          snd             tty47            ttyS5
cpu_dma_latency loop23          sr0             tty48            ttyS6
cuse            loop24          stderr          tty49            ttyS7
disk            loop25          stdin           tty5             ttyS8
dri             loop26          stdout          tty50            ttyS9
drm_dp_aux0     loop27          tpm0            tty51            uhid
drm_dp_aux1     loop28          tpmrm0          tty52            uinput
dvd             loop29          tty             tty53            urandom
dvdrw           loop3           tty0            tty54            userio
ecryptfs        loop4           tty1            tty55            v4l
fb0             loop5           tty10           tty56            v4l-touch0
fd              loop6           tty11           tty57            vboxdrv
full            loop7           tty12           tty58            vboxdrv
fuse            loop8           tty13           tty59            vboxnetctl
gpiochip0       loop9           tty14           tty6             vboxusb

```

3.1 Có 2 loại

- Character Device

- Đọc và ghi theo từng Character
- Hoạt động ở chế độ blocking, đồng bộ với các thao tác
- Phổ biến hầu hết các thiết bị Device Driver
- Ví dụ: serial ports, parallel port, card âm thanh.

- Block Device

- Đọc và ghi theo từng Block
- Hoạt động sâu trong CPU, mất nhiều thời gian để hoàn thành, không đồng bộ với các thao tác
- Ví dụ: hard disks, USB camera, Disk-On-Key

Ứng với 2 loại Device Driver, có 2 loại Device File tương ứng là Character Device File và Block Device File.

3.2 Major and Minor Numbers

```
root@hieudoan7-Vostro-3559:/dev# ls -l tty0
crw--w---- 1 root tty 4, 0 Thg 9 24 19:58 tty0
root@hieudoan7-Vostro-3559:/dev#
```

Với Device File tty0 như trên ta biết được:

- crw Cho biết là Device File này là loại Character
- 4 Major number
- 0 Minor number

- Major number: Mỗi Device Driver có 1 major number, con số này trong Device File cho biết Device Driver nào nó đang quản lý

- Minor number: Để phân biệt các Device File quản lý cùng 1 Device Driver

4 Character Device File

Character Device File là Device File quản lý Character Device Driver.

4.1 File Operations

```
// Tell the kernel which functions to call when user operates on our device file
struct file_operations fops={
    .owner = THIS_MODULE,    //prevent unloading of this module when operations are in use
    .open = device_open,     //points to the method to call when opening the device
    .release = device_close,  //points to the method to call when closing the device
    .write = device_write,    //points to the method to call when writing to the device
    .read = device_read       //points to the method to call when reading from the device
};
```

4.2 Register A Device

Như chúng ta đã nói, Character device được tiếp cận thông qua device files, thường nằm trong /dev. Major number cho biết driver nào quản lý device file nào. Minor number được sử dụng chỉ để phân biệt thiết bị nào đang sử dụng trong trường hợp mà driver quản lý nhiều device.

Thêm 1 driver vào hệ thống có nghĩa là đăng kí nó với kernel. Chúng ta có thể làm điều đó bằng cách sử dụng hàm `register_chrdev` trong thư viện `linux/fs.h`

```
1 int register_chrdev(unsigned int major, const char *name, struct
    file_operations *fops);
```

4.3 Unregister A Device

```
1 void unregister_chrdev_region ( dev_t from, unsigned count);
```

Hàm này sẽ unregister một range gồm count device numbers, bắt đầu từ from.

5 Bài tập

Đề bài: Viết một Module tạo ra một Character Device cho phép các tiến trình ở userspace có thể open và read các số ngẫu nhiên.

5.1 Tạo module sinh số ngẫu nhiên và chèn vào hệ thống

Sử dụng hàm `get_random_bytes` trong `<linux/random.h>` để sinh ra số ngẫu nhiên. Sau đó ta `copy_to_user` để gửi kết quả cho user.

```
1 static ssize_t device_read(struct file *filp, char *buffer, size_t
   length, loff_t *offset)
2 {
3     int randomNumber;
4     get_random_bytes(&randomNumber, sizeof(randomNumber));
5     int ret = copy_to_user(buffer, &randomNumber, BUFFER_LENGTH);
6     if (ret == 0) {
7         printk(KERN_ALERT "Sent %ld character to the user\n", sizeof(
           randomNumber));
8         return sizeof(randomNumber);
9     } else {
10        printk(KERN_ALERT "Failed to send!\n");
11        return -EFAULT;
12    }
13 }
14 }
```

Các bước khác để tạo module có thể tham khảo source code sau:

https://github.com/hieudoan190598/My-University/blob/master/Linux%20Kernel%20Module%20Progammig/1612198_module.c

Biên dịch module vừa viết và chèn vào hệ thống

```
root@hieudoan7-Vostro-3559:/media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/project_01/Official# modinfo 1612198_module.ko
filename:          /media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/project_01/Official/1612198_module.ko
srcversion:        D71B59632286C3D8FFCAE8A
depends:
retpoline:         Y
name:              1612198_module
vermagic:          4.15.0-34-generic SMP mod_unload
root@hieudoan7-Vostro-3559:/media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/project_01/Official#
```

5.2 Tạo ra Character Device File

Sau khi insert module thành công, ta gõ command `dmesg` để đọc message, thực hiện lệnh

`mknod /dev/file_device_name c major_number minor_number`
để tạo ra Character File Device trong `/dev`.

```
[17245.638054] 1612198: major number is 239
root@hieudoan7-Vostro-3559:/media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/projec
t_01/Official# mknod /dev/1612198_device c 239 0
root@hieudoan7-Vostro-3559:/media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/projec
t_01/Official# ls -l /dev/1612198_device
crw-r--r-- 1 root root 239, 0 Thg 9 25 00:50 /dev/1612198_device
root@hieudoan7-Vostro-3559:/media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/projec
t_01/Official#
```

Đến đây ta đã tạo xong module và Device File cho hệ thống, bây giờ viết chương trình để sử dụng nó.

5.3 Viết chương trình ở userspace để open và read số ngẫu nhiên

Chương trình `get_random_number.c`

```
1 #include <stdio.h>
2 #include <stdlib.h>
3 #include <fcntl.h>
4
5 #define DEVICE "/dev/1612198_device"
6 int main(){
7     int fd; //file description
8     char read_buf[100];
9     fd = open(DEVICE,O_RDWR); //open for reading and writing
10    if (fd==-1){
11        printf("file %s either does not exist or has been locked by
12        another process\n",DEVICE);
13        exit(-1);
14    }
15    int randNumber=read(fd,read_buf,sizeof(read_buf));
16    printf("The random Number: %d\n",randNumber);
17    close(fd);
18    return 0;
19 }
```

Tiến hành biên dịch và chạy chương trình `get_random_number.c`

```

root@hieudoan7-Vostro-3559:/media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/project_01/Official# ls
1612198_module.c      1612198_module.mod.o  Makefile
1612198_module.ko      1612198_module.o      modules.order
1612198_module.mod.c  get_random_number.c    Module.symvers
root@hieudoan7-Vostro-3559:/media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/project_01/Official# gcc get_random_number.c -o getNumberApp

```

Kết quả thu được như mong muốn.

```

root@hieudoan7-Vostro-3559:/media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/project_01/Official# ./getNumberApp
The random Number: 1085073023
root@hieudoan7-Vostro-3559:/media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/project_01/Official# ./getNumberApp
The random Number: -446963276
root@hieudoan7-Vostro-3559:/media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/project_01/Official# ./getNumberApp
The random Number: 603788129
root@hieudoan7-Vostro-3559:/media/hieudoan7/JOB/HOC_KY_5/OPERATING_SYSTEM/project_01/Official#

```

Như vậy, ta đã có cái nhìn sơ lược về lập trình Linux Kernel Module. Qua đó biết cách viết một module đơn giản và một Character Device để cho phép các tiến trình ở userspace có thể open và read số ngẫu nhiên.

6 Tài liệu tham khảo

1. [Series Linux Kernel Module Programming của SolidusCode Youtube Channel](#)
2. [Writing a Linux Kernel Module — Part 1: Introduction - derekmolloy](#)
3. [Hệ thống thư viện của linux](#)
4. [Viết một driver đơn giản theo cơ chế kernel module - Embedded247](#)
5. [Book: The Linux Kernel Module Programming Guide - Peter Jay Salzman](#)