**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**ĐẠI HỌC KHOA HỌC TỰ NHIÊN**

**KHOA CÔNG NGHỆ THÔNG TIN**

**🙢🟋🙠**



**ĐỒ ÁN HỆ ĐIỀU HÀNH**

**LINUX KERNEL**

Mục lục

[**I.** **THÀNH VIÊN** 3](#_Toc24141305)

[**II.** **MỨC ĐỘ ĐÁNH GIÁ** 3](#_Toc24141306)

[**III.** **CHI TIẾT** 4](#_Toc24141307)

[**1.** **Phần 1** 4](#_Toc24141308)

[**a)** **Các phần chính** 4](#_Toc24141309)

[**b)** **Chạy module** 5](#_Toc24141310)

[**2.** **Phần 2** 6](#_Toc24141311)

[**a)** **Các phần chính** 6](#_Toc24141312)

[**b)** **Chạy module** 9](#_Toc24141313)

[**IV.** **TÀI LIỆU THAM KHẢO** 11](#_Toc24141314)

1. **THÀNH VIÊN**

|  |  |
| --- | --- |
| **MSSV** | **Họ và tên** |
| 1712026 | Lê Trần Hữu Đắc |
| 1712214 | Phạm Hoàng Nhật Anh |

1. **MỨC ĐỘ ĐÁNH GIÁ**

|  |  |
| --- | --- |
| **Yêu cầu** | **Mức độ hoàn thành** |
| Phần 1: Viết một module tạo một character device để cho phép các tiến trình ở user space có thể open và read các số ngẫu nhiên được sinh ra ở module này. | 100% |
| Phần 2: Chương trình hook vào một system call:   * syscall open => ghi vào dmesg tên tiến trình mở file và tên file được mở. * syscall write => ghi vào dmesg tên tiến trình, tên file bị ghi và số byte được ghi. | 100% |

1. **CHI TIẾT**
2. **Phần 1**

### **Các phần chính**

* Driver thực hiện đăng ký số hiệu và tên module.

static dev\_t dev\_num;

* Trong hàm tạo:

int ret;

ret = alloc\_chrdev\_region(& dev\_num, 0, 1, "RandomNumber");

Major = MAJOR(dev\_num);

if(ret < 0) {

printk(KERN\_INFO "RandomNumber: Registration failed\n");

}

* Trong hàm hủy:

unregister\_chrdev\_region(dev\_num, 1);

* Tạo file thiết bị tự động.

static struct class \* dev\_class;

static struct device \* dev;

* Trong hàm tạo:

dev\_class = class\_create(THIS\_MODULE, "Class\_RandomNumber");

if (dev\_class == NULL) {

printk(KERN\_INFO "RandomNumber: Failed to create a device class\n");

unregister\_chrdev\_region(dev\_num, 1);;

return 0;

}

dev\_class->devnode = RandomNumber\_devnode; //permission

dev = device\_create(dev\_class, NULL, dev\_num, NULL, "RandomNumber");

if (IS\_ERR(dev)){

printk(KERN\_INFO "RandomNumber: Failed to create a device\n");

class\_destroy(dev\_class);

unregister\_chrdev\_region(dev\_num, 1);

return 0;

}

* Trong hàm hủy:

device\_destroy(dev\_class, dev\_num);

class\_destroy(dev\_class);

* Thao tác với file thiết bị.

static struct cdev c\_dev;

static struct file\_operations fops = {

.owner = THIS\_MODULE,

.open = randomNumber\_open,

.read = randomNumber\_read,

.release = randomNumber\_close,

};

* Trong hàm tạo:

cdev\_init(& c\_dev, & fops);

if (cdev\_add(&c\_dev, dev\_num, 1) == -1){

device\_destroy(dev\_class, dev\_num);

class\_destroy(dev\_class);

unregister\_chrdev\_region(dev\_num, 1);

return 0;

}

* Hàm read (sinh số ngẫu nhiên).

static ssize\_t randomNumber\_read(struct file \*filp, char \*buffer, size\_t length, loff\_t \*offset)

{

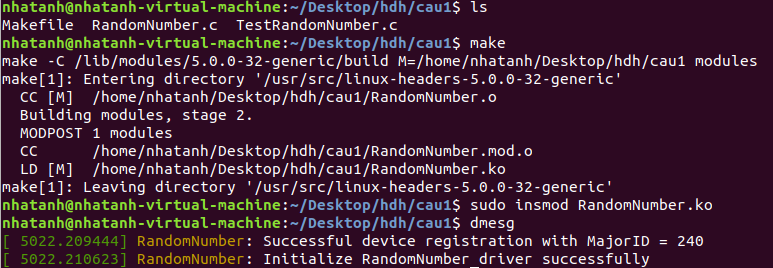
printk(KERN\_INFO "RandomNumber: read()\n");

return get\_random\_int();

}

### **Chạy module**

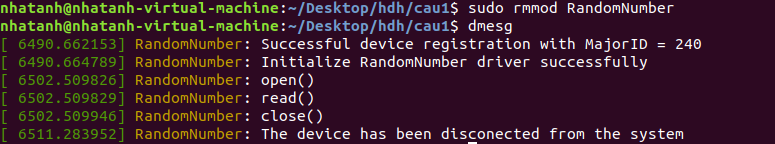
* Dùng lệnh make để tạo file (.ko) và sử dụng lệnh insmod để nạp vào nhân hệ thống.



* Tạo file Test.exe và tương tác với driver.



* Dùng rmmod để gỡ bỏ driver.



1. **Phần 2**

### **Các phần chính**

* Xác định tham số của syscall cần hook.
* Syscall open:

asmlinkage int (\*original\_call) (const char \_\_user\*, int, mode\_t);

* Syscall write:

asmlinkage size\_t (\*original\_call) (unsigned int, const char \*, size\_t);

* Viết hàm thực thi syscall với đúng tham số của syscall gốc:
* Syscall open:

asmlinkage int our\_sys\_open(const char \_\_user\* filename, int flags, mode\_t mode)

{

printk(KERN\_INFO "\n");

printk(KERN\_INFO "%s opened %s\n", current->comm, filename);

return original\_call(filename, flags, mode);

}

* Syscall write:

asmlinkage size\_t our\_sys\_write(unsigned int fd, const char \*buf, size\_t nbytes)

{

size\_t wrotebytes = original\_call(fd, buf, nbytes);

char fileName[256];

int fileDesc = fd;

struct files\_struct \*files = current->files;

char \*tmp;

char \*pathname;

struct file \*file;

struct path \*path;

spin\_lock(&files->file\_lock);

file = fcheck\_files(files, fileDesc);

if (!file) {

spin\_unlock(&files->file\_lock);

return -ENOENT;

}

path = &file->f\_path;

path\_get(path);

spin\_unlock(&files->file\_lock);

tmp = (char \*)\_\_get\_free\_page(GFP\_KERNEL);

if (!tmp) {

path\_put(path);

return -ENOMEM;

}

pathname = d\_path(path, tmp, PAGE\_SIZE);

path\_put(path);

if (IS\_ERR(pathname)) {

free\_page((unsigned long)tmp);

return PTR\_ERR(pathname);

}

strcpy(fileName, pathname);

free\_page((unsigned long)tmp);

printk(KERN\_INFO "\n");

printk(KERN\_INFO "%s opened %s wrote %zu byte(s)\n", current->comm, fileName, wrotebytes);

return wrotebytes;

}

* Viết hàm init và exit cho custom syscall:
* Syscall open:

static int \_\_init entry\_point(void)

{

printk(KERN\_INFO "Hookopen loaded successfully..\n");

// sys\_call\_table address in System.map

system\_call\_table\_addr = (void\*)kallsyms\_lookup\_name("sys\_call\_table");

original\_call = system\_call\_table\_addr[\_\_NR\_open];

make\_rw((unsigned long)system\_call\_table\_addr);

system\_call\_table\_addr[\_\_NR\_open] = our\_sys\_open;

return 0;

}

static void \_\_exit exit\_point(void)

{

printk(KERN\_INFO "Unloaded hookopen successfully\n");

// Restore the original call

system\_call\_table\_addr[\_\_NR\_open] = original\_call;

make\_ro((unsigned long)system\_call\_table\_addr);

}

* Syscall write:

static int \_\_init entry\_point(void)

{

printk(KERN\_INFO "Hookwrite loaded successfully\n");

// sys\_call\_table address in System.map

system\_call\_table\_addr = (void\*)kallsyms\_lookup\_name("sys\_call\_table");

original\_call = system\_call\_table\_addr[\_\_NR\_write];

make\_rw((unsigned long)system\_call\_table\_addr);

system\_call\_table\_addr[\_\_NR\_write] = our\_sys\_write;

return 0;

}

static void \_\_exit exit\_point(void)

{

printk(KERN\_INFO "Unloaded hookwrite successfully\n");

// Restore the original call

system\_call\_table\_addr[\_\_NR\_write] = original\_call;

make\_ro((unsigned long)system\_call\_table\_addr);

}

### **Chạy module**

* Phiên bản header linux sử dụng sẽ là linux-3.16.0-23-generic.
* Dùng lệnh make để tạo file (.ko) và sử dụng lệnh insmod để nạp vào nhân hệ thống.

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

* Kết quả chạy

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer

Description automatically generated**

****

1. **TÀI LIỆU THAM KHẢO**

Các file pdf hướng dẫn

<https://sites.google.com/site/embedded247/ddcourse/kernelmoduleprogramming>

<https://vimentor.com/vi/lesson/device-file>