****

**BÁO CÁO**

***ĐỒ ÁN 02***

***KERNEL MODULE***

***&***

***HOOK SYSTEM CALL***

Nguyễn Khánh Hoàng

Nguyễn Hữu Vinh

1. **Thành viên của nhóm**

|  |  |  |  |
| --- | --- | --- | --- |
| **STT** | **HỌ & TÊN** | **MSSV** | **PHÂN CÔNG** |
| 1 | Nguyễn Khánh Hoàng | 1712457 | Phần 1 |
| 2 | Nguyễn Hữu Vinh | 1712206 | Phần 2 |

1. **Nội dung đồ án**
   1. **Phần 1**
      1. **Mô tả yêu cầu**

Mục tiêu hiểu về Linux kernel module và hệ thống quản lý file và device  
trong linux, giao tiếp giữa tiến trình ở user space và kernel space.

* + - Viết một module dùng để tạo ra số ngẫu nhiên.
    - Module này sẽ tạo một character device để cho phép các tiến trình ở  
      user space có thể open và read các số ngẫu nhiên
    1. **Mức độ hoàn thành :** 100%
    2. **Tổ chức :**
       - ***chardevdrv.c***

Chứa mã nguồn để biên dịch và tạo ra một kernel module. Module này tạo ra một character device có chức năng sinh các số ngẫu nhiên và cho phép các tiến trình ở user space đọc các số ngẫu nhiên đó.

* *static int dev\_open(struct inode\* i, struct file \*f);*

Hàm thực hiện thao tác mở file thiết bị

* *static int dev\_close(struct inode \*i, struct file \*f);*

Hàm thực hiện thao tác đóng file thiết bị

* *static ssize\_t dev\_read(struct file \*f, char \_\_user \*buf, size\_t len, loff\_t \*off);*

Hàm thực hiện thao tác đọc dữ liệu từ file thiết bị

* *static ssize\_t dev\_write(struct file \*f, const char \_\_user \*buf, size\_t len, loff\_t \*off);*

Hàm thực hiện thao tác ghi vào file thiết bị

* + - * *static struct file\_operations randnum\_fops =*

*{*

*.owner = THIS\_MODULE,*

*.open = dev\_open,*

*.release = dev\_close,*

*.read = dev\_read,*

*.write = dev\_write*

*};*

Cấu trúc *file operations*, trong đó các thuộc tính sẽ nhận giá trị là tên các hàm sẽ được thực hiện tương ứng với các thao tác  
*mở*, *đóng*, *đọc*, *ghi*.

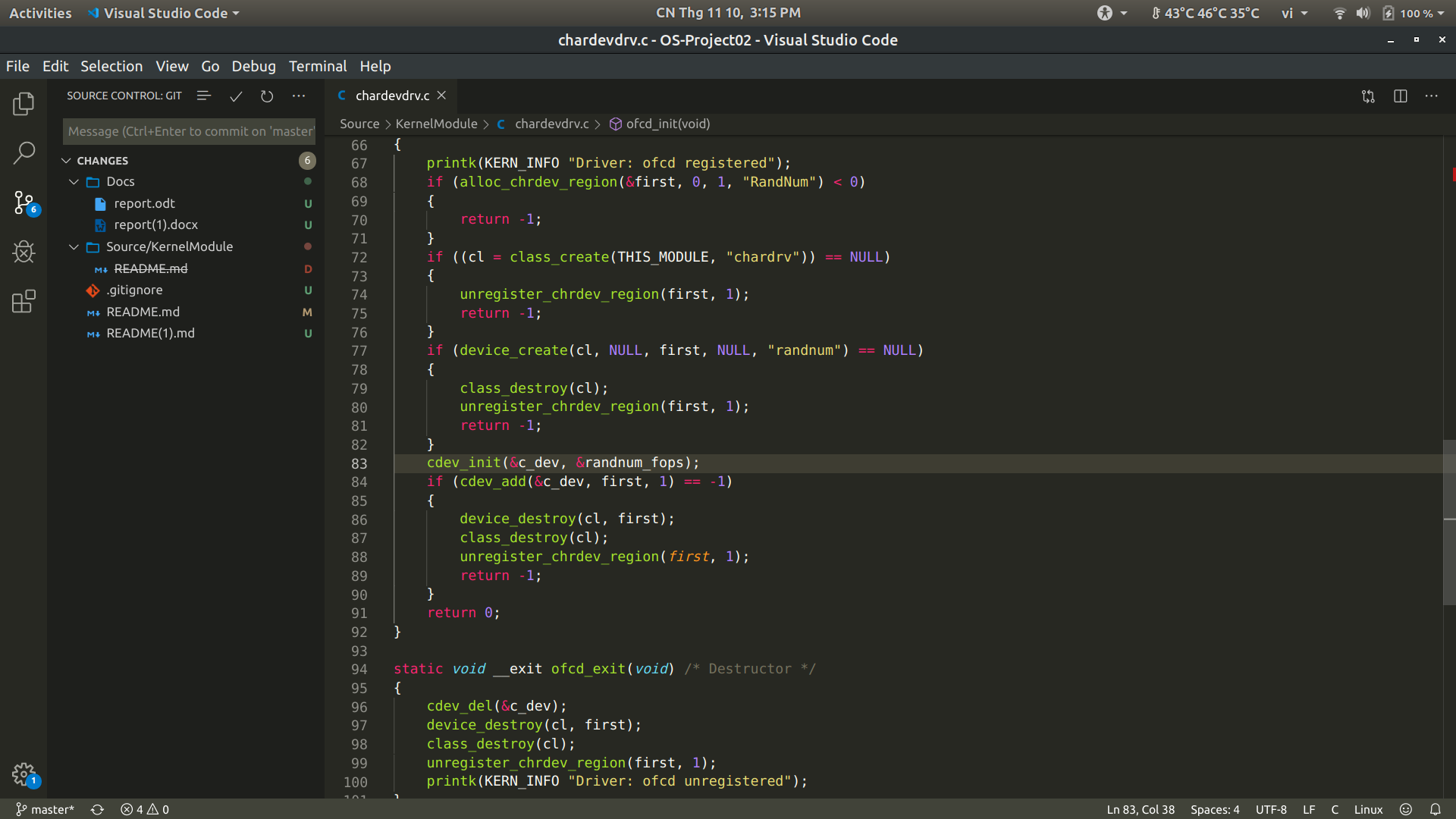
* + - *static int \_\_init ofcd\_init(void); /\* Constructor \*/*

Hàm thực hiện công việc khởi tạo, đăng ký module driver.

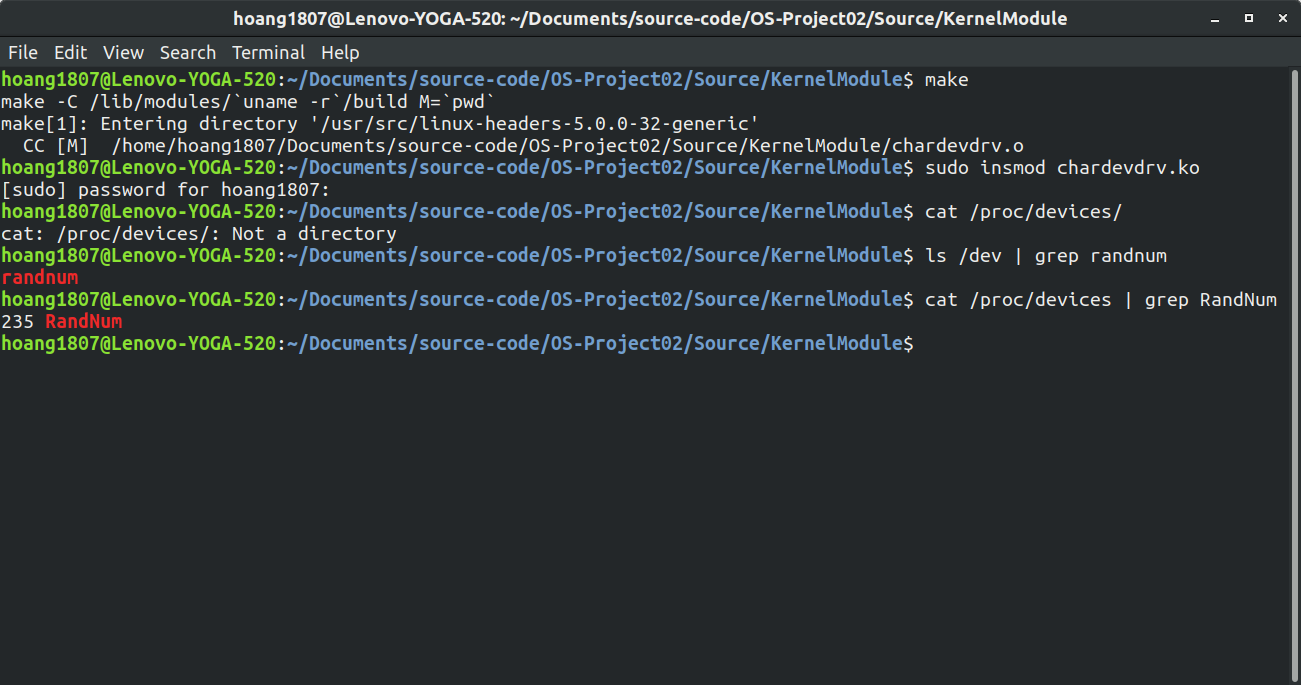
* + - *static void \_\_exit ofcd\_exit(void); /\* Destructor \*/*

Hàm thực hiện công việc hủy đăng ký, gỡ bỏ module driver.

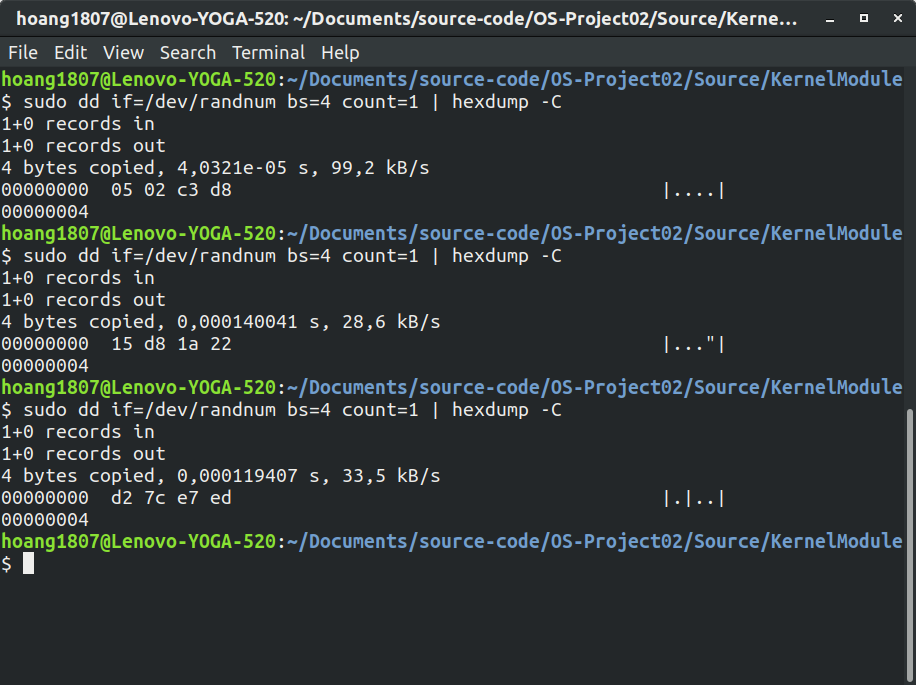
* ***Makefile :*** File qui định cách thực thi lệnh make
  + 1. **Xử lí :**
       - * Biên dịch file mã nguồn chardevdrv.c để tạo ra file module chardevdrv.ko
         * Lắp đặt module vào kernel
         * Kiểm tra chức năng của character device bằng cách đọc số ngẫu nhiên từ file thiết bị ở trong /dev
         * Tháo module ra khỏi kernel.
    2. **Ảnh minh họa :**



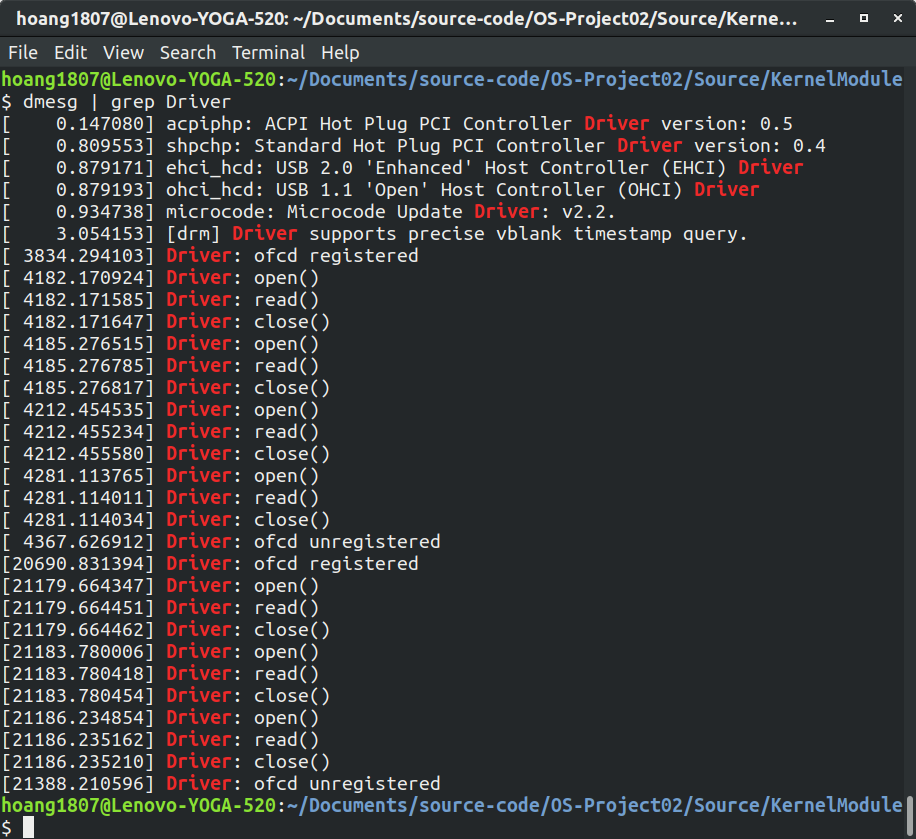
File mã nguồn chardevdrv.c

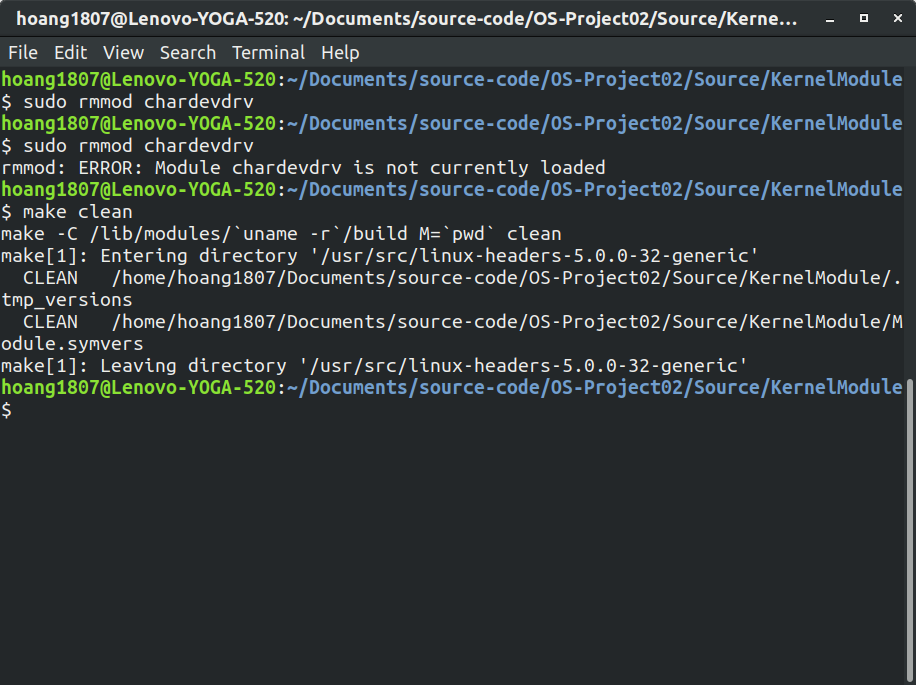


Thực hiện biên dịch và tạo ra file module chardevdrv.ko rồi lắp đặt vào trong kernel



Thử chức năng của character device bằng cách đọc dữ liệu từ file thiết bị và  
hiển thị kết quả đọc được dưới dạng hệ hexa

Dùng lệnh dmesg để xem quá trình hoạt động của việc cài đặt module driver



Gỡ bỏ module ra khỏi kernel và dọn dẹp các file tạm trong thư mục

* 1. **Phần 2**
     1. **Mô tả yêu cầu**

Viết chương trình hook vào một system call.

* + - * + System call ‘Open’ : Ghi vào dmesg tên tiến trình mở file và tên file được mở.
        + System call ‘Write’ : Ghi vào dmesg tên tiến trình ghi file, tên file bị ghi và số byte được ghi
    1. **Mức độ hoàn thành:**
* Hook system call Open: hoàn thành
  + Ghi vào dmesg tiến trình mở file, tên file được mở
* Hook system call Write: hoàn thành
  + Ghi vào dmesg tiến trình ghi file, tên file bị ghi, số byte được ghi
    1. **Tổ chức:**
* ***hookSyscall.c***

Chứa mã nguồn chương trình hook system call Open & Write. Bao gồm các hàm:

* **asmlinkage int (\*original\_open) (const char \*pathname, int flags);**
* **asmlinkage ssize\_t (\*original\_write)(unsigned int fd, const void \*buf,  
  size\_t count);**

Đóng vai trò các hàm tạm thời, dùng để lưu lại địa chỉ mặc định trong system call table của hàm Open và Write

* **asmlinkage int new\_open(const char \*pathname, int flags)**

Hàm thay thế system call Open

* **asmlinkage ssize\_t new\_write(unsigned int fd, const void \*buf, size\_t count)**

Hàm thay thế system call Write

* **int make\_rw(unsigned long address)**

Tắt thuộc tính read-only của system call table

* **int make\_ro(unsigned long address)**

Bật thuộc tính read-only của system call table

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

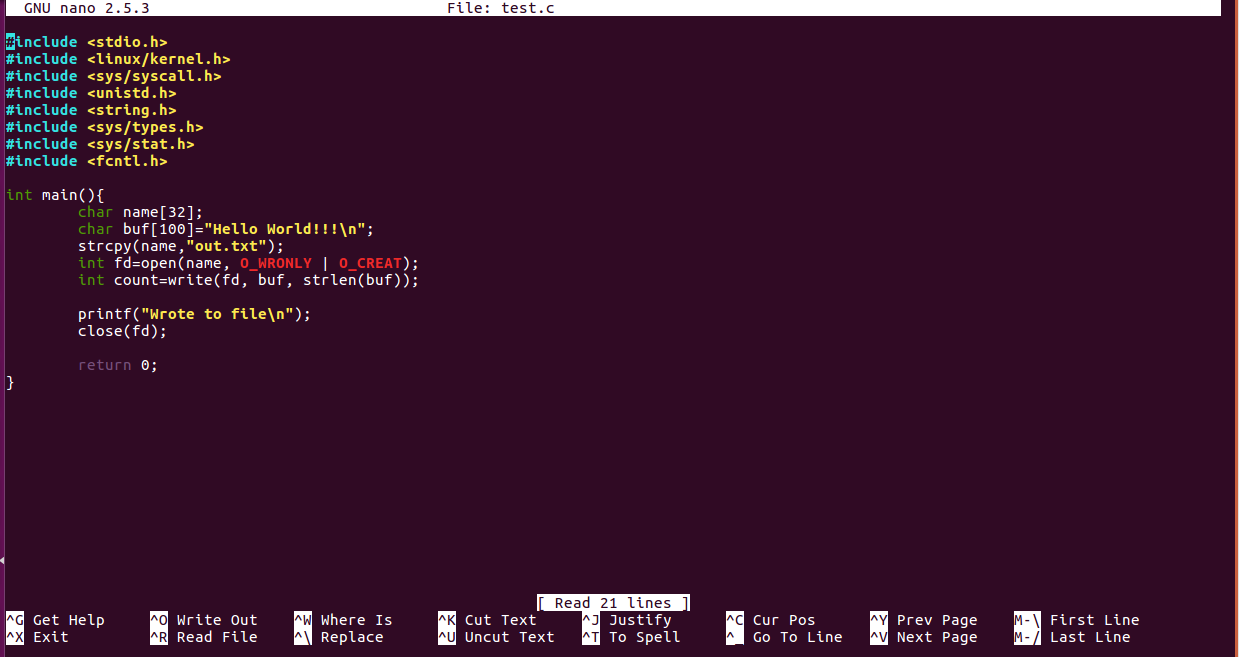
Hàm khởi tạo khi hook syscall

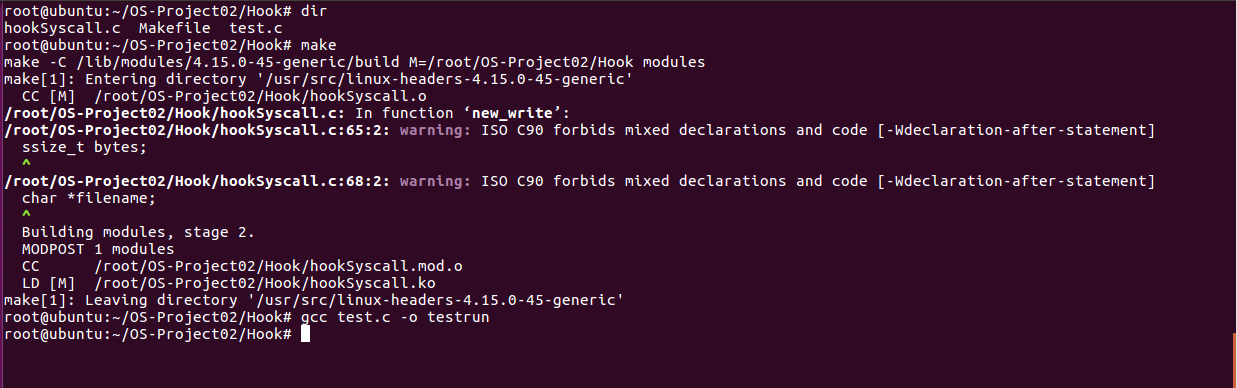
* **static void \_\_exit exit\_point(void)**

Hàm thoát khi gỡ hook syscall

* ***Makefile :*** File qui định để thực thi lệnh make
  + 1. **Xử lí**

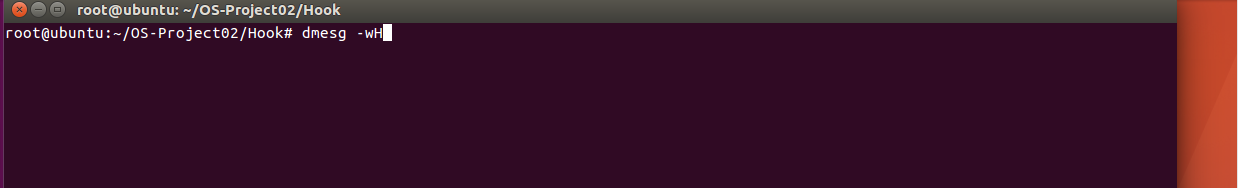
1. Tìm địa chỉ system call table
2. Lưu địa chỉ mặc định của hàm Open và Write vào các hàm tạm thời
3. Tắt thuộc tính read-only của bảng system call table
4. Gán địa chỉ hàm thay thế các syscall vào vị trí hàm Open và Write vào bảng
5. Gọi lại hàm mặc định trong hàm thay thế đồng thời thêm các xử lí in ra dmesg
6. Gán địa chỉ ban đầu của hàm Open và Write vào bảng
7. Bật thuộc tính read-only của bảng system call table
   * 1. **Ảnh minh họa:**

Chương trình test.c:

Tiến hành compile test.c sang testrun và compile chương trình hook

Gắn hook module và chạy testrun để kiểm tra

Dùng lệnh dmesg để kiểm tra kết quả in ra





Gỡ hook module và kết thúc

**Nguồn tham khảo:**

* https://github.com/GreenCandlePrinter/syscall-hook/blob/master/syscall-hook.c
* https://stackoverflow.com/questions/2103315/linux-kernel-system-call-hooking-example
* https://stackoverflow.com/questions/8250078/how-can-i-get-a-filename-from-a-file-descriptor-inside-a-kernel-module