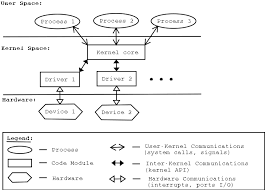
1. **Cấu trúc cài đặt**

* Thực hiện tạo lập một kernel module
* Sau đó nạp module vào kernel
* User thực hiện cái giao tiếp với module thông qua các phương thức open(), read(), write(), release()



1. **Cách thức cài đặt một kernel module:**

* Đầu tiên chúng ta tạo một kernel module với các phương thúc open(), read(), write(), release().
  + Trong trường hợp này chúng ta chỉ sử dụng phương read() để lấy 1 số ngẫu nhiên từ kernel
  + Phương thức open(), write(), release() chúng ta chỉ cần thực hiện dòng lệnh in màn hình tên phương thức thực thi.
  + Phương thức read():

static ssize\_t dev\_read(struct file \*filep, char \*buffer, size\_t len, loff\_t \*offset){

int error\_count = 0;

int number;

get\_random\_bytes(&number,sizeof(number));

// copy\_to\_user has the format ( \* to, \*from, size) and returns 0 on success

error\_count = copy\_to\_user(buffer, &number, 1);

if (error\_count==0){ // if true then have success

printk(KERN\_INFO "randomnumber: read()");

return 0;

}

else {

printk(KERN\_INFO "randomnumber: read() failed");

return -EFAULT;

}

}

* Tạo file Kbuild và file Makefile để khởi tạo ra kernel object
  + Kbuild

EXTRA\_CFLAGS = -Wall

obj-m = random.o

* + - Biến obj-m chỉ ra rằng: object file sẽ được biên dịch theo kiểu kernel module.
    - Cờ -Wall cho phép trình biên dịch hiển thị tất cả các bản tin cảnh báo trong quá  
      trình biên dịch.
  + Makefile

KDIR = /lib/modules/`uname -r`/build

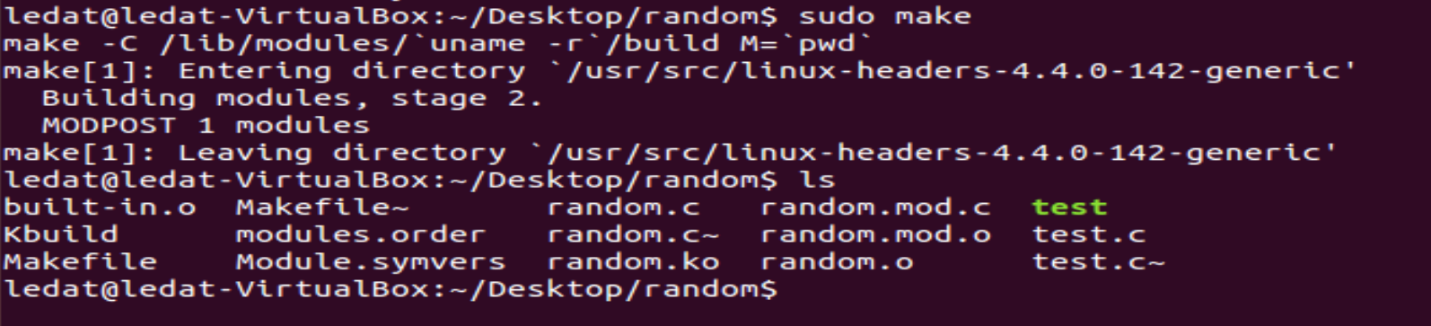
all:

make -C $(KDIR) M=`pwd`

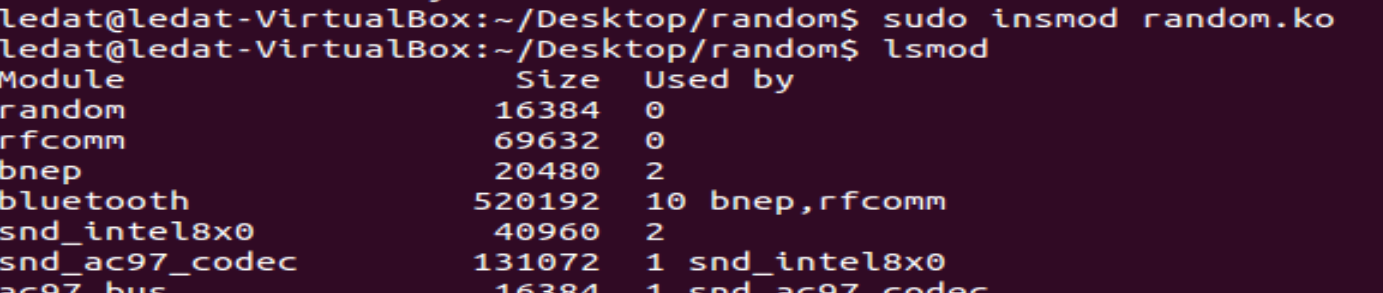
clean:

make -C $(KDIR) M=`pwd` clean

* + - Thẻ all chứa câu lệnh để biên dịch các module trong thư mục hiện tại.
    - Thẻ clean chứa lệnh xóa tất cả các object file có trong thư mục hiện tại
* Biên dịch tạo file kernel object



* Nạp module vào kernel, kiểm tra kernel đã nạp chưa bằng lệnh lsmod, ta thấy module random đã có trong danh sách => nạp thành công



* Viết chương trình giao tiếp với module test.c

static int receive; ///< The receive number from the LKM

int main(){

int ret, fd;

printf("Starting device test code example...\n");

fd = open("/dev/randomnumber", O\_RDWR); // Open the device with read/write access

if (fd < 0){

perror("Failed to open the device...");

return errno;

}

printf("Reading from the device...\n");

ret = read(fd, &receive, 1); // Read the response from the LKM

if (ret < 0){

perror("Failed to read the message from the device.");

return errno;

}

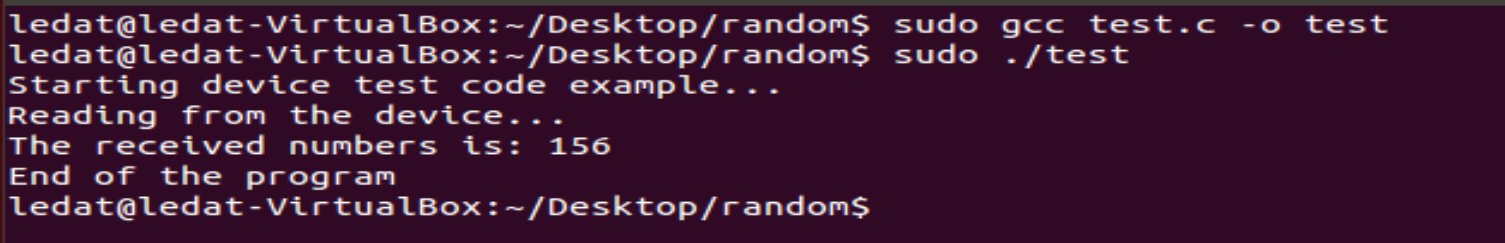
printf("The received numbers is: %d\n",receive);

printf("End of the program\n");

return 0;

}

* Thực hiện kiểm tra module, ta thấy module trả về kết quả ngẫu nhiên



Đây là toàn bộ quá trình thực hiện để xây dựng một character device driver.

Reference: <http://derekmolloy.ie/writing-a-linux-kernel-module-part-2-a-character-device/>