BÁO CÁO ĐỒ ÁN 1

Trần Quốc Cường - 1612843

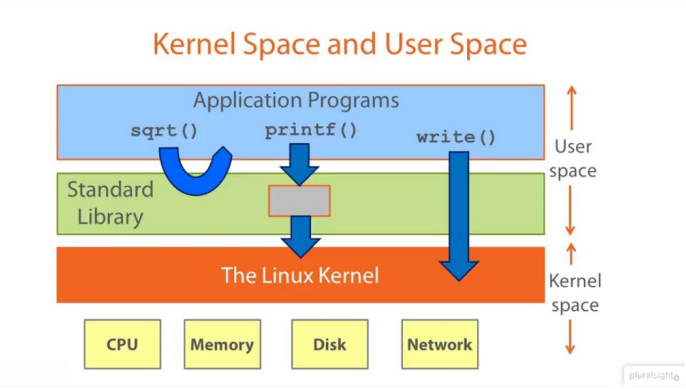
September 2018

1 Kernel Module là gì?

Một Loadable Kernel Module (LKM) là một cơ chế cho phép thêm và sửa code vào nhân Linux trong thời gian thực chạy. LKM vô cùng phù hợp cho driver của các thiết bị, cho phép kernel giao tiếp với phần cứng mà không cần phải biết phần cứng làm việc thế nào.

Nhờ có khả năng chỉ lắp ghép các module cần thiết mà nhân Linux tránh việc trở nên rất lớn khi phải cài sẵn tất cả driver của các thiết bị. Chúng ta cũng hạn chế được việc phải "rebuild" nhân Module mỗi khi phải thêm hay xóa driver mới. Kernel Module được thực thi như là một phần của nhân (không phải trong không gian người dùng).

Kernel module và ứng dụng người dùng chạy trong 2 không gian khác nhau (kernel space và user space). Chúng cũng sử dụng 2 địa chỉ ô nhớ khác nhau. Điều này giúp cho ứng dụng người dùng chạy ổn định hơn và tránh sự giao tranh tài nguyên của các ứng dụng người dùng.



Hình 1: Kernel space và User space

2 Cách viết code và gắn Kernel Module vào hệ thống

Kernel module được viết bằng bằng C và không phải là một ứng dụng do đó code của nó không có hàm main() và có những đặc tính đặc biệt sau:

- Không thực hiện tuần tự: Những kernel module đăng ký nó bằng hàm khởi tạo, chúng xử lý bằng những module đã được chỉ sẵn trong source code.
- Không được dọn rác tự động: Những tài nguyên hay bộ nhớ mà kernel module sử dụng phải được dọn dẹp thủ công hoặc là sẽ không thể sử dụng cho đến khi khởi động lại kernel
- Không có hàm printf(): Kernel Module không thể sử dụng các thư viện của không gian người dùng, có thể dùng hàm printk() để thay thế
- Có thể bị gián đoạn: Kernel Module có thể được sử dụng bởi nhiều ứng dụng cùng một lúc. Chúng ta phải xử lý trường hợp này để tránh gây ra lỗi không mong muốn
- Có quyền thực thi ở mức độ cao: do nằm ở không gian nhân (kernel space) nên kernel module có quyền truy cập cao hơn ứng dụng bình thường
- Không hỗ trợ số thực dấu phẩy động: code của nhân sử dụng traps để chuyển từ số nguyên sang số chấm phẩy động điều nay không được khuyên dùng và nên được thực hiện tại không giao người dùng

Source code của một kernel module gồm nhiều phần (tham khảo *kernel_module.c*)

```
#include <linux/init.h>           // Macros used to mark up functions e.g. __init __exit
#include <linux/module.h>         // Core header for loading LKMs into the kernel
#include <linux/device.h>         // Header to support the kernel Driver Model
#include <linux/kernel.h>         // Contains types, macros, functions for the kernel
#include <linux/fs.h>             // Header for the Linux file system support
#include <linux/uaccess.h>        // Required for the copy to user function
#include <linux/slab.h>           // Required for kmalloc/kfree functions
#include <linux/random.h>         // Required for the get_random_bytes function
```

Hình 2: Các thư viện sử dụng

```
#define  DEVICE_NAME "randomness_generator"    ///< The device will appear at /dev/randomness... using this value
#define  CLASS_NAME  "first_project"        ///< The device class -- this is a character device driver

MODULE_LICENSE("GPL");            ///< The license type -- this affects available functionality
MODULE_AUTHOR("Quoc-Cuong TRAN");    // modinfo
MODULE_DESCRIPTION("First Project in Operating system course");
MODULE_VERSION("1.0");
```

Hình 3: Khai báo các thông tin

- Các thao tác mà chúng ta sẽ xử lý

```
module_init(RNG_init);
module_exit(RNG_exit);
```

Hình 4: Khai báo hàm nào dùng để khởi tạo và thoát ở cuối file code của kernel module

3 Nguồn tham khảo

- Writing a Linux Kernel Module — Part 1: Introduction
- Writing a Linux Kernel Module — Part 2: A Character Device