UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY

REPORT

OPERATING SYSTEM - LINUX KERNEL MODULE

BẠCH GIA HUY - 21127615
NGUYỄN HỒNG THÁI - 21127689
HUỲNH SỈ KHA - 21127734

Lecturers:

Phạm Tuấn Sơn

Lê Viết Long

25th April 2023

Contents

Mức độ hoàn thành: Hoàn thành 100% các yêu cầu của đồ án.

Bảng phân công việc			
Tên	**MSSV**	**Công việc**	**Mức độ hoàn thành**
Bạch Gia Huy	21127615	Thiết kế, lên ý tưởng, viết báo cáo	100%
Nguyễn Hồng Thái	21127689	Thiết kế, lên ý tưởng thuật toán, viết báo cáo	100%
Huỳnh Sỉ Kha	21127734	Cài đặt chương trình, viết báo cáo	100%

1. Linux Kernel Module

1.1 Định nghĩa

Là một file có tên mở rộng là ".ko", được lắp vào hoặc tháo ra khỏi kernel khi cần thiết nên còn có tên gọi khác là Loadable kernel module. Một trong những kiểu phổ biến là driver.

Ưu điểm:

- Giảm kích thước kernel, tránh gây lãng phí bộ nhớ và giảm thời gian khởi động hệ thống.
- Không phải biên dịch lại kernel khi thêm mới hoặc thay đổi driver.
- Không cần khởi động lại hệ thống khi thêm mới driver.

1.2 Qui trình thêm module vào kernel space

Khi cần thêm mới một module chưa có trong kernel space, kernel sẽ đưa module vào với 1 qui trình tự động sau:

1. Kernel kích hoạt tiến trình **modprobe** cùng với tham số truyền vào là tên module.
2. Tiến trình modprobe kiểm tra file $/lib/modules/ < kernel - version > /modules.dep$ xem file module có phụ thuộc vào module khác không.
3. Tiến trình modprobe kích hoạt tiến trình **insmod** để đưa các module phụ thuộc vào trước, rồi mới đưa module cần thiết.

Cách kernel kích hoạt tiến trình modprobe: Có hai cách

1. Sử dụng **kmod** là thành phần của Linux kernel, hoạt động trong kernel space. Khi được gọi, nó sẽ truyền tên module cho hàm **request_module**, khi đó, hàm đó sẽ gọi **call_usermodehelper_setup** để sinh ra tiến trình modprobe.
2. Sử dụng **udevd** là tiến trình hoạt động trong user space.

2. Các thư viện dùng trong chương trình

1. linux/module.h: Chứa hai macro module_init() và module_exit().
    - module_init: Giúp xác định hàm nào sẽ được thực thi ngay sau khi lắp module vào kernel.

- module_exit: Giúp xác định hàm nào được thực thi ngay trước khi tháo module ra khỏi kernel.

2. linux/fs.h: Định nghĩa các hàm cấp phát và giải phóng device number.
3. linux/device.h: Định nghĩa các hàm phục vụ cho việc tạo device file.
4. linux/slab.h: Định nghĩa hàm kzalloc và kfree để cấp phát và thu hồi vùng nhớ của thiết bị.
5. linux/cdev.h: Định nghĩa các hàm dùng cho đối tượng cdev.
6. linux/random.h: Dùng để khởi tạo ra một số tự nhiên ngẫu nhiên.

3. Character Driver

3.1 Device file (device node)

Trong Linux kernel, file chia làm 6 loại: file thông thường, thư mục, pipe, socket, symbolic link và device file (device node). Tác dụng của device file là để đánh lừa các tiến trình rằng, các char và block device tương tự như file thông thường từ đó giúp việc đọc ghi dữ liệu trở nên dễ dàng hơn.

a. Tạo device file tự động

Dựa vào tiến trình **udevd** để tạo và huỷ các device file trong thư mục *dev*. Quá trình gửi sự kiện lên cho udevd gọi là uevent, sau đó, udevd sẽ tạo ra một device file trong thư mục */dev*. Quá trình khởi tạo device file gồm có 3 bước:

- Sử dụng thư viện *device.h*.
- Tạo một lớp các thiết bị.
- Tạo thiết bị trong lớp.

3.2 Device number

Kernel dùng device number để xác định device driver tương ứng với device file. Mỗi device driver là 1 bộ gồm 2 số: major và minor number.

- Major number: Giúp kernel nhận ra device driver tương ứng.
- Minor number: Giúp device driver nhận biết thiết bị cần điều khiển nếu đang điều khiển nhiều thiết bị.

Cấp phát động device number

- Gọi hàm *alloc_chrdev_region* thuộc thư viện *fs.h* để yêu cầu kernel tìm kiếm device number phù hợp.
- Trong hàm huỷ, gọi hàm *unregister_chrdev_region* để giải phóng device number.

Công dụng: Giúp chương trình có thể chạy trên nhiều máy tính khác nhau

3.3 Mã nguồn

Mã nguồn của character driver được tổ chức thành hai phần chính:

- Phần OS specific: gồm các hàm khởi tạo, kết thúc driver và các hàm entry point của driver.
- Phần device specific: gồm các hàm khởi tạo/giải phóng thiết bị, đọc ghi các thanh ghi và hàm xử lý tín hiệu ngắt đến từ thiết bị.

4. Cấp phát bộ nhớ và khởi tạo thiết bị vật lý

4.1 Cấp phát bộ nhớ

Sử dụng hàm **kzalloc** thuộc thư việc *slab.h* để cấp phát bộ nhớ, hàm này không chỉ cấp phát bộ nhớ dưới kernel space mà còn thiết lập tất cả bytes của vùng nhớ bằng 0.

Để giải phóng vùng nhớ đã cấp phát, gọi hàm **kfree** với tham số truyền vào là địa chỉ vùng nhớ.

4.2 Khởi tạo thiết bị vật lý

Thiết bị vật lý được khởi tạo nằm trên Ram, gồm có 1 thanh ghi điều khiển, 5 thanh ghi trạng thái và 256 thanh ghi dữ liệu, mỗi thanh ghi có kích thước 1 byte:

- Thanh ghi điều khiển: bật, tắt thiết bị, tạm dừng, khôi phục hoạt động của thiết bị.
- Thanh ghi trạng thái: Kiểm tra thiết bị có đang hoạt động, có đang bận không, kiểm tra lỗi trong quá trình hoạt động.
- Thanh ghi dữ liệu: ghi dữ liệu vào thanh ghi này.

5. Đăng kí các entry point trong chương trình

5.1 Entry point open và release

Cấu trúc mô tả các character device trong Linux là **cdev**. Khi char driver muốn điều khiển một character device, thì nó phải nộp cấu trúc này cho kernel. Cấu trúc cdev thuộc kiểu cấu trúc **file_operations**.

a. Cấu trúc file_operations

Cấu trúc này được định nghĩa trong thư viện *fs.h*. Nhiệm vụ là để gán các hàm của char driver vào các con trỏ hàm.

Sau khi được gán thì các hàm trở thành entry point của char driver. Bây giờ, kernel sẽ thiết lập mối quan hệ 11 giữa system call với entry point (syscall open tương ứng với entry point open).

5.2 Entry point read

Việc đọc dữ liệu của tiến trình từ một char device gồm các bước:

- Gọi syscall open để mở open file tương ứng của char device.
- Gọi syscall read để đọc dữ liệu từ device file này. Khi đó, kernel sẽ kích hoạt entry point của device file tương ứng.
- Entry point read đọc dữ liệu từ char device rồi trả về cho tiến trình trên user space.

6. Tài liệu tham khảo:

- Tài liệu hướng dẫn đồ án lập trình Linux kernel module - Môn hệ điều hành - Phạm Tuấn Sơn, Lê Viết Long.
- Hướng dẫn lập trình Linux kernel module.