ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN
KHOA CÔNG NGHỆ THÔNG TIN

* * * * *

BÁO CÁO ĐỒ ÁN: KERNEL MODULE

MÔN HỌC: HỆ ĐIỀU HÀNH

LỚP : 18CNTN

SINH VIÊN THỰC HIỆN : 18120019 – Nguyễn Hoàng Dũng

18120052 – Lê Hạnh Linh

Thành phố Hồ Chí Minh, ngày 14 tháng 12 năm 2020

1. ĐỌC HIỂU VẤN ĐỀ:

1.1. **Character Device là gì ?**

Character Device là bất kì thiết bị nào mà khi tương tác với nó, Kernel có khả năng đọc hoặc ghi từng kí tự (1 byte dữ liệu) đến thiết bị đó.

Để điều khiển Character Device, chúng ta cần một Character Driver tương ứng để gửi hoặc nhận từng byte kí tự.

Character Driver là một dạng của Kernel Module.

Trong bài này, Character Driver thực thi trên một vùng nhớ như cách các Driver khác tương tác với một Device thực, nên thực chất Character Device có thể hiểu là vùng nhớ được sử dụng bởi Character Driver. Device ở đây đơn giản là một Interface giữa Kernel và Character Driver và cho phép người dùng sử dụng các dịch vụ.

1.2. **Character Device File là gì ?**

Character Device File là một thực thế nắm giữ Character Device tương ứng. Device File xuất hiện trong File System như một File thông thường. Bằng việc tạo ra Device File, Linux Kernel đánh lừa các tiến trình rằng các character device cũng chỉ là các file thông thường. Device file cũng có các System Call Interface như open, close, read, write,…và device sẽ "map" các lời gọi này với hàm tương ứng driver cung cấp bên dưới.

Khi một tiến trình ở User Space gọi System Call để tương tác với Device File, hệ thống file ảo – Virtual File System sẽ giải mã file được gọi để biết được rằng đây là một Device File.

1.3. **Làm sao để tương tác với Character Driver:**

Việc sử dụng Character Driver được thực hiện thông qua Device File tương ứng. Mỗi thao tác file – File Operations được tiến trình ở User Space gọi đối với Device File sẽ được Virtual File System diễn giải thành 1 hàm tương ứng trong Character Driver.

Như vậy, Device File chính là một Interface của Device Driver.

Để làm được điều này, Virtual File System cần được thông báo về các thao tác file và hàm được đăng ký tương ứng. Điều này được hiện trong mã nguồn của Driver, hay Kernel Module ta cần cài đặt.

1.4. **Làm sao để Kernel biết Driver nào tương ứng với Device File ?**

Kernel sử dụng Device Number để biết Device Driver nào tương ứng với Device File. Device Number là một bộ gồm 2 số:

- Major Number: Thể hiện kết nối giữa Device File và Device Driver.
- Minor Number: Giúp Device Driver nhận biết phải điều khiển thiết bị nào trong trường hợp Driver đó đang điều khiển nhiều thiết bị.

1.5. **Làm sao để tương tác với Kernel Module từ User Space ?**

Khi một tiến trình trên User Space muốn tương tác với một Module, trước tiên ta cần phải đã nạp được Module đó vào Kernel Space.

Việc nạp ở đây được diễn ra theo cách thủ công, bằng cách sử dụng lệnh insmod, với mặc định đã có sẵn file Kernel Object được build cho Kernel Module tương ứng.

Trong quá trình sau khi nạp Module vào Kernel Space, hàm module_init() sẽ được gọi với tham số là con trỏ của hàm khởi tạo do chúng ta cài đặt. Hàm khởi tạo là một hàm có macro __init đi kèm ở đầu tên hàm.

Nhiệm vụ của hàm khởi tạo này bao gồm việc đăng ký số hiệu Major Number và tạo Device File tương ứng trong thư mục /dev.

Như vậy, sau khi nạp thành công một Module, Device File sẽ xuất hiện trong thư mục /dev và trở thành một Interface, cho phép tiến trình User giao tiếp với Module. Số hiệu Major Number sẽ giúp Kernel xác định được Module tương ứng với Device File.

1.6. **Cách đăng ký Device Number:**

Sử dụng hàm register_chrdev_region()

*int register_chrdev_region(dev_t first, unsigned int count, char *name)*

Mục tiêu của hàm này là đăng ký 1 hoặc nhiều device number để làm việc. Trong đó, first là số hiệu của device number đầu tiên trong dãy các số ta mong muốn đăng ký. Count là tổng số device number ta đang yêu cầu.

Kernel sẽ kiểm tra nếu các device numbers đó chưa được đăng ký thì sẽ trả về 0, nếu không thành công thì negative error code được trả về, có nghĩa là ta không được truy cập vào dãy các device number đó.

Trong trường hợp này ta phải tự chọn một bộ device numbers với major number mà “có vẻ” chưa được dùng.

Tuy nhiên, để tránh conflict với các thiết bị khác, và chúng ta nhiều khi không biết rõ major number nào còn hay không, ta nên sử dụng giải pháp xin cấp phát động như sau:

*int alloc_chrdev_region(dev_t *dev, unsigned int first,*
*unsigned int count, char *name)*

Hàm này chỉ cung cấp số device numbers là count mà ta mong muốn, kernel sẽ tự tìm major number và trả về thông qua tham số dev truyền vào.

Tuy nhiên, ta mới chỉ yêu cầu Kernel cấp cho ta Device Number mà chưa thông báo với Kernel rằng sẽ dùng các Device Number này vào mục đích gì. Trước khi một tiến trình của User Space có thể access vào Device Number, Module của ta cần phải liên kết các function của nó với các thao tác của Device.

1.7. **Cách tạo một Device File:**

Khi được cấp device number, ta sẽ tạo ra một device file để user có thể truy cập. Trước tiên cần tạo ra lớp thiết bị device class là một lớp ảo thông qua hàm *class_create()*.

Sau đó tạo ra một device file là một thể hiện của lớp vừa được tạo bằng hàm *device_create*().

Hàm này sẽ tạo ra một Device File được gán vào major number và minor number của device liên kết với nó và đăng ký nó với hệ thống, nên trong File System sẽ xuất hiện file này.

Tuy nhiên, lúc này chưa thể thao tác trên file này, do bên dưới Kernel chưa được cài đặt để xử lý các thao tác trên file đặc biệt này.

1.8. **Cách tạo Character Device trong Kernel Space:**

Sử dụng hàm cdev_init()

*void cdev_init(struct cdev *cdev, struct file_operations *fops)*

Khi đã tạo 1 Device thì ta cũng cần phải liên kết các thao tác trên Device đến các hàm tương ứng trên Driver quản lý.

cdev là một cấu trúc trong Kernel Space đại diện cho 1 Character Device. Bất cứ khi nào 1 inode trỏ đến 1 Device File thì cấu trúc này sẽ trỏ đến Character Device tương ứng.

file_operation là một cấu trúc cho phép 1 Device File khi được mở sẽ có các thao tác File được liên kết với hàm trong Driver tương ứng.

Tiếp theo ta cần thông báo với Kernel về sự xuất hiện của Character Device này bằng hàm cdev_add()

*int cdev_add(struct cdev *dev, dev_t num, unsigned int count)*

Khi này, Kernel sẽ quản lý Character Device này thông qua Device Number.