BÀI 20: CHARACTER DEVICE DRIVER

Trình điều khiển thiết bị ký tự

?? Mục tiêu bài học

1. **Phân loại Driver:** Char Driver vs Block Driver vs Network Driver.
2. **Định danh:** Major Number (Loại driver) & Minor Number (Thứ tự thiết bị).
3. **Device Node:** File trong `/dev` được tạo ra như thế nào?
4. **File Operations:** Cấu trúc `file_operations` để map lệnh `open/read` từ User xuống Kernel.

1. Các loại thiết bị trong Linux

- **Character Device (Char):** Truy cập tuần tự theo byte (stream). Ví dụ: Bàn phím, Chuột, Cổng Serial, Sound Card. (Chiếm 80% driver).
- **Block Device:** Truy cập theo khối (block), có thể seek. Ví dụ: Ổ cứng, Thẻ nhớ, USB.
- **Network Interface:** Không có device node, quản lý qua socket.

2. Major & Minor Number

Mỗi driver được quản lý bằng một cặp số:

- **Major Number (Mã định danh lớp):** Xác định driver nào sẽ xử lý. (Ví dụ: Tất cả cổng Serial có Major = 4).
- **Minor Number (Mã thiết bị cụ thể):** Xác định thiết bị nào trong lớp đó. (Ví dụ: COM1 là minor 0, COM2 là minor 1).

> **Lệnh kiểm tra:** `ls -l /dev/ttyS0`
>
> `crw-rw---- 1 root dialout 4, 64 ...` (Major 4, Minor 64)

3. Cấu trúc `file_operations` (fops)

Đây là "trái tim" của Char Driver. Nó ánh xạ System Call của User sang hàm của Driver.

```
static struct file_operations fops = {
    .owner = THIS_MODULE,
    .open = my_open,
    .release = my_close,
    .read = my_read,
    .write = my_write,
};
```

Khi User gọi `read()`, Kernel sẽ tìm trong `fops` xem hàm `.read` trỏ tới đâu và gọi hàm đó.

4. Quy trình đăng ký Driver

1. **Cấp phát Major Number:**
   - Tĩnh: `register_chrdev_region()` (Cổ điển).
   - Động: `alloc_chrdev_region()` (Khuyên dùng).
2. **Tạo Cdev (Character Device structure):** `cdev_init()`, `cdev_add()`.
3. **Tạo Device Node (Tự động):**
   - Tạo Class: `class_create()`.
   - Tạo Device: `device_create()`.
     -> Tự sinh ra file `/dev/my_device`.

🛠️ PHẦN THỰC HÀNH (LAB 20)

Viết Driver tạo file `/dev/dummy_driver`

Yêu cầu

1. Viết Kernel Module đăng ký một Char Driver.
2. Trong hàm `my_open` và `my_close`: In log `printk` báo hiệu.
3. Tự động tạo file `/dev/dummy_driver` khi `insmod`.
4. Viết app C ở User Space: * `open("/dev/dummy_driver")`. * `close()`.
5. Quan sát `dmesg` xem driver có phản hồi không.

Q & A

Hẹn gặp lại ở Bài 21: Trao đổi dữ liệu!