BÀI 24: PLATFORM DRIVER MODEL

Mô hình Driver hiện đại

🎯 Mục tiêu bài học

1. **Vấn đề:** Tại sao viết driver kiểu cũ (hardcode địa chỉ) là không tốt?
2. **Mô hình:** Bus - Device - Driver.
3. **Platform Bus:** Bus ảo dành cho các thiết bị tích hợp trong SoC (SoC Internal Peripherals).
4. **Cấu trúc:** `platform_driver` (Probe & Remove).

1. Linux Device Model

Linux quản lý thiết bị theo mô hình tách biệt:

- **Device (Thiết bị):** "Tôi là ai? Tôi có tài nguyên gì (IRQ, Register)?" -> Mô tả trong Device Tree.
- **Driver (Trình điều khiển):** "Làm thế nào để điều khiển?" -> Code C.
- **Bus:** Đường dây kết nối. Khi Device và Driver có tên trùng nhau (Match), Bus sẽ gọi hàm `probe()` của Driver.

2. Platform Driver Structure

Thay vì dùng `module_init`, ta đăng ký một struct:

```c
#include <linux/platform_device.h>

// Hàm chạy khi tìm thấy thiết bị khớp (Thay cho init)
static int my_probe(struct platform_device *pdev) {
    printk("Da tim thay thiet bi!\n");
    return 0;
}

// Hàm chạy khi thiết bị bị gỡ bỏ (Thay cho exit)
static int my_remove(struct platform_device *pdev) {
    printk("Da go thiet bi!\n");
    return 0;
}

static struct platform_driver my_driver = {
    .probe = my_probe,
    .remove = my_remove,
    .driver = {
        .name = "my_platform_device",
        .owner = THIS_MODULE
    },
};
```

3. Cơ chế Matching (Ghép đôi)

Làm sao Driver biết nó phải lái Device nào?

1. **Name Matching (Cũ):** Tên trong `.name` của Driver trùng với tên của Device.
2. **Device Tree Matching (Mới):** Dùng chuỗi `compatible`.
   - Driver khai báo: "Tôi lái được thiết bị compatible = `ti,omap-uart`".
   - Device Tree khai báo: Node UART có `compatible = "ti,omap-uart"`.
   - Kernel thấy khớp -> Gọi `probe()`.

🛠️ PHẦN THỰC HÀNH (LAB 24)

Đăng ký Platform Driver đơn giản

Yêu cầu

1. **Tạo 2 Module riêng biệt:**
   - `my_device.ko`: Đăng ký `platform_device` có tên "fake_device".
   - `my_driver.ko`: Đăng ký `platform_driver` lắng nghe tên "fake_device".
2. **Kịch bản:**
   - Nạp `my_driver.ko` trước -> Chưa có gì xảy ra.
   - Nạp `my_device.ko` sau -> Hàm `probe()` của driver lập tức được gọi.
   - Gỡ `my_device.ko` -> Hàm `remove()` được gọi.

> **Ý nghĩa:** Driver có thể được nạp sẵn, khi nào cắm thiết bị vào thì nó mới hoạt động (Hot-plug).

Q & A

Hẹn gặp lại ở Bài 25: Parsing Device Tree!