BÀI 25: TƯƠNG TÁC DRIVER VỚI DEVICE TREE

Parsing DT in Kernel

🎯 Mục tiêu bài học

1. **Kết hợp:** Ghép kiến thức Bài 9 (DTS) và Bài 24 (Platform Driver).
2. **API:** Sử dụng các hàm `of_*` (Open Firmware) để đọc dữ liệu từ DTB.
3. **Thực hành:** Viết Driver lấy số hiệu GPIO từ file DTS (Không hardcode nữa).

1. Khai báo Match Table

Trong Driver, ta cần khai báo bảng ID để Kernel biết driver này tương thích với node nào trong DTS.

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

static const struct of_device_id my_of_ids[] = {
    { .compatible = "my,led-gpio" },  // Chuỗi này phải có trong file DTS
    { },
};

MODULE_DEVICE_TABLE(of, my_of_ids);

static struct platform_driver my_driver = {
    .driver = {
        .name = "my_dt_driver",
        .of_match_table = my_of_ids,  // Gán bảng vào đây
    },
};
```

2. Lấy thông tin trong hàm Probe

Khi `probe(struct platform_device *pdev)` được gọi, `pdev` chứa con trỏ tới node trong Device Tree.

```
// Lấy GPIO từ property "gpios" trong DTS
struct gpio_desc *my_led;
my_led = gpiod_get(&pdev->dev, "led", GPIOD_OUT_LOW);
if (IS_ERR(my_led)) {
    printk("Loi khong lay duoc GPIO\n");
    return -1;
}

// Lấy một số nguyên từ property tùy chỉnh "my-prop"
u32 val;
of_property_read_u32(pdev->dev.of_node, "my-prop", &val);
```

3. Chuẩn bị file DTS (Overlay)

Ta cần thêm một node vào file DTS của board và biên dịch lại (hoặc dùng Overlay).

```
/ {
    my_led_device {
        compatible = "my,led-gpio";
        led-gpios = <&gpio1 21 0>;  /* GPIO1_21 */
        status = "okay";
    };
};
```

🛠️ PHẦN THỰC HÀNH (LAB 25)

Universal LED Driver

Yêu cầu

1. Sửa file DTS của board, thêm node cho LED.
2. Viết Platform Driver:
   - Dùng `of_match_table` để khớp với chuỗi `compatible`.
   - Trong `probe`: Lấy GPIO descriptor từ DTS.
   - Tạo file `/dev/dt_led`.
3. Khi ghi `1` vào `/dev/dt_led` -> Đèn sáng.

> **Ưu điểm:** Nếu chuyển sang board khác (Raspberry Pi), chỉ cần sửa file DTS, code Driver giữ nguyên 100%. Đây là triết lý "Write Once, Run Anywhere" của Linux Kernel.

Q & A

Hẹn gặp lại ở Bài 26: Kernel Synchronization!