BÀI 23: XỬ LÝ NGẮT TRONG KERNEL

Interrupt Handling: Top-half & Bottom-half

🎯 Mục tiêu bài học

1. **Cơ chế:** Ngắt (Interrupt) vs Hỏi vòng (Polling).
2. **API:** Đăng ký hàm xử lý ngắt (`request_irq`).
3. **Kiến trúc:** Top-half (Xử lý nhanh) vs Bottom-half (Xử lý chậm).
4. **Hạn chế:** Những điều CẤM làm trong hàm ngắt (Ngủ, cấp phát bộ nhớ động).

1. Tại sao cần ngắt?

- **Polling (Hỏi vòng):** CPU liên tục kiểm tra trạng thái nút nhấn.
  - *Ưu điểm:* Dễ code.
  - *Nhược điểm:* Tốn CPU (CPU Load 100%), lãng phí năng lượng.
- **Interrupt (Ngắt):** CPU đi làm việc khác. Khi nút nhấn được bấm, phần cứng "chọt" CPU một cái để báo hiệu.
  - *Ưu điểm:* Tiết kiệm tài nguyên, phản hồi tức thì.

2. Đăng ký Ngắt (`request_irq`)

Để Kernel biết phải gọi hàm nào khi có ngắt xảy ra:

```
#include <linux/interrupt.h>

// Hàm xử lý ngắt (ISR)
static irqreturn_t my_irq_handler(int irq, void *dev_id) {
    printk("Nut da duoc nhan!\n");
    return IRQ_HANDLED;
}

// Trong hàm Init:
// irq_num: Số hiệu ngắt (Lấy từ GPIO)
// IRQF_TRIGGER_RISING: Ngắt cạnh lên
request_irq(irq_num, my_irq_handler, IRQF_TRIGGER_RISING, "my_button", NULL);
```

3. Top-half vs Bottom-half

Hàm ngắt (`my_irq_handler`) chạy trong **Interrupt Context**, nó chặn (block) tất cả các tác vụ khác. Nếu nó chạy quá lâu -> Treo máy.

- **Top-half (Hard IRQ):**
  - Chính là hàm `my_irq_handler` .
  - Nhiệm vụ: Làm cực nhanh (Ghi nhận sự kiện, xóa cờ ngắt).
  - Tuyệt đối **KHÔNG ĐƯỢC NGỦ** (`msleep` , `mutex` , `copy_to_user`).
- **Bottom-half (Soft IRQ / Tasklet / Workqueue):**
  - Nhiệm vụ: Xử lý nặng (Tính toán, gửi dữ liệu, đọc I2C).
  - Chạy sau khi Top-half kết thúc, có thể bị ngắt bởi ngắt khác.

4. Workqueue (Hàng đợi công việc)

Cách phổ biến nhất để xử lý Bottom-half (vì nó cho phép ngủ).

```
static struct work_struct my_work;

// Hàm Bottom-half (Được phép ngủ)
void my_work_func(struct work_struct *work) {
    msleep(100);  // Ngủ thoải mái
    printk("Xu ly xong viec nang!\n");
}

// Trong hàm ngắt (Top-half)
static irqreturn_t my_irq_handler(int irq, void *dev_id) {
    schedule_work(&my_work);  // Đẩy việc xuống hàng đợi
    return IRQ_HANDLED;
}
```

🛠️ PHẦN THỰC HÀNH (LAB 23)

Button Interrupt Driver

Yêu cầu

1. Nối một nút nhấn vào chân GPIO (ví dụ GPIO 60).
2. Chuyển đổi số GPIO sang số IRQ: `gpio_to_irq(60)`.
3. Đăng ký ngắt cạnh xuống (`IRQF_TRIGGER_FALLING`).
4. Khi nhấn nút:
    - In ra `dmesg` : "Button Pressed!".
    - Đảo trạng thái một biến `count` .
5. Gỡ ngắt (`free_irq`) khi rmmod.

Q & A

Hẹn gặp lại ở Bài 24: Platform Driver!