

Betriebssysteme (BS)

VL 5.3 – Unterbrechungen, SoftIRQs

Volkmar Sieh / Daniel Lohmann

Lehrstuhl für Informatik 4
Verteilte Systeme und Betriebssysteme

Friedrich-Alexander-Universität
Erlangen Nürnberg

WS 21 – 22. November 2021

https://www4.cs.fau.de/Lehre/WS21/V_BS



Agenda

Einordnung

Beispiele

Aufteilung Interrupt-Bearbeitung

Implementierung

SoftIRQs

SoftIRQ-Beispiel

Verwandte Konzepte

Zusammenfassung



Hinweis: `keyboard_soft_irq` muss ggf. gar nicht aufgerufen werden:

- Eine Taste wurde losgelassen.
- Eine Shift/Alt/Ctrl/...-Taste wurde gedrückt.

Daher könnte `keyboard_hard_irq`-Methode Bit setzen, wenn `keyboard_soft_irq`-Methode ausgeführt werden soll.

=> Entspricht einem Interrupt-Mechanismus
(diesmal in Software)
sogenannte „**SoftIRQs**“



Typisches SoftIRQ-Interface (pro SoftIRQ):

configure: Registriere Behandlungsfunktion, die beim Auftreten des SoftIRQs aufgerufen werden soll.

trigger: Wenn SoftIRQ disabled: setze Pending-Bit, sonst: führe Behandlungsfunktion aus.

disable: Disable SoftIRQ.

enable: Enable SoftIRQ. Wenn Pending-Bit gesetzt, führe Behandlungsfunktion aus.



```
void disable() {
    enabled = false;
}

void enable() {
    while (pending) {
        pending = false;
        handler();
    }

    enabled = true;
}

void trigger() {
    if (enabled) {
        asm volatile ("sti\n");
        handler();
        asm volatile ("cli\n");
    } else {
        pending = true;
    }
}
```



SoftIRQ-Interface

Vorsicht! Race-Condition:

```
void disable() {
    enabled = false;
}

void enable() {
    while (pending) {
        pending = false;
        handler();
    }
    enabled = true;
}

void trigger() {
    if (enabled) {
        asm volatile ("sti\n");
        handler();
        asm volatile ("cli\n");
    } else {
        pending = true;
    }
}
```

<= trigger hat keinen Effekt!



trigger wird nur im Interrupt-Handler aufgerufen.
Kann daher mit cli/sti hart synchronisiert werden:

```
void enable() {  
    asm volatile("cli\n");  
    while (pending) {  
        pending = false;  
        asm volatile("sti\n");  
        handler();  
        asm volatile("cli\n");  
    }  
    enabled = true;  
    asm volatile("sti\n");  
}
```



SoftIRQ-Interface

Idee: *BSD: trigger wird nur im Interrupt-Handler aufgerufen und kennt damit gesicherten Instruktion-Pointer (IP).

```
void enable() {
    asm volatile("enable_begin:\n");
    while (pending) {
        pending = false;
        handler();
    }
    enabled = true;
    asm volatile("enable_end:\n");
}

void trigger() {
    if (enabled) {
        asm volatile("sti\n");
        handler();
        asm volatile("cli\n");
    } else {
        pending = 1;
        if (enable_begin <= IP && IP <= enable_end) {
            IP = enable_begin;
        }
    }
}
```



Hardware-IRQs werden beim Signalisieren eines Interrupts von der Hardware aus nur dann gestartet, wenn sie nicht schon laufen.

Kann durch weiteres Flag (`running`) nachgebildet werden.



Agenda

Einordnung

Beispiele

Aufteilung Interrupt-Bearbeitung

Implementierung

SoftIRQs

SoftIRQ-Beispiel

Verwandte Konzepte

Zusammenfassung



SoftIRQ-Beispiel

```
void keyboard_hard_irq() {
    uint8_t code = read_code_from_keyboard_controller();
    if (count < sizeof(buffer)) {
        buffer[head] = code;
        head = (head + 1) % sizeof(buffer);
        count++;
    }
    apic_disable_keyboard_irq();
    trigger();
    apic_enable_keyboard_irq();
}

void keyboard_soft_irq() {
    asm volatile ("cli\n");
    while (0 < count) {
        uint8_t code = buffer[tail];
        tail = (tail + 1) % sizeof(buffer);
        count--;
        asm volatile ("sti\n");
        print_to_screen(code);
        asm volatile ("cli\n");
    }
    asm volatile ("sti\n");
}
```



SoftIRQ-Beispiel

```
void keyboard_soft_irq() {
    asm volatile ("cli\n");
    while (0 < count) {
        uint8_t code = buffer[tail];
        tail = (tail + 1) % sizeof(buffer);
        count--;
        asm volatile ("sti\n");
        print_to_screen(code);
        asm volatile ("cli\n");
    }
    asm volatile ("sti\n");
}

void console_write(char code) {
    disable();
    print_to_screen(code);
    enable();
}
```

