

Betriebssysteme (BS)

VL 5.2 – Unterbrechungen, SoftIRQs – Aufteilung

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



```
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++;
    }
}

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

void keyboard_irq() {
    keyboard_hard_irq();
    asm volatile ("sti\n");
    keyboard_soft_irq();
}
```



```
void keyboard_irq() {  
    keyboard_hard_irq();  
    asm volatile ("sti\n");  
    keyboard_soft_irq();  
}
```



```
void keyboard_hard_irq();  
asm volatile ("sti\n");  
keyboard_soft_irq();  
}
```

Funktioniert so eventuell nicht!



```
void keyboard_hard_irq();  
asm volatile ("sti\n");  
keyboard_soft_irq();  
}
```

Funktioniert so eventuell nicht!

- Problem: nach sti kann Stack überlaufen!



```
void keyboard_irq() {  
    keyboard_hard_irq();  
    asm volatile ("sti\n");  
    keyboard_soft_irq();  
}
```

Funktioniert so eventuell nicht!

- Problem: nach sti kann Stack überlaufen!
- Daher (ähnlich Linux):

```
void keyboard_irq() {  
    keyboard_hard_irq();  
    apic_disable_keyboard_irq();  
    asm volatile ("sti\n");  
    keyboard_soft_irq();  
    asm volatile ("cli\n");  
    apic_enable_keyboard_irq();  
}
```



```
apic_enable_keyboard_irq();
```

```
apic_disable_keyboard_irq();
```

können ggf. alle Interrupts niedrigerer Priorität ein- bzw. ausschalten.

=> selbstdefinierte Interrupt-Prioritäten



Nachteile:

- Schreiben in / Lesen aus Puffer kostet Zeit.
- Disable/Enable von Interrupts kostet Zeit.

Vorteil:

- Andere(!), ggf. wichtige(!), Interrupts werden früher bearbeitet.



Im Falle von „shared interrupts“ müssen

- erst von allen beteiligten Geräten ggf. Daten ausgelesen und zwischengespeichert,
- dann Interrupts wieder zugelassen,
- dann von allen beteiligten Geräten die zwischengespeicherten Daten weiterverarbeitet werden

