

Betriebssysteme (BS)

VL 5 – Unterbrechungen, Software

Volkmar Sieh / Daniel Lohmann

Lehrstuhl für Informatik 4
Verteilte Systeme und Betriebssysteme

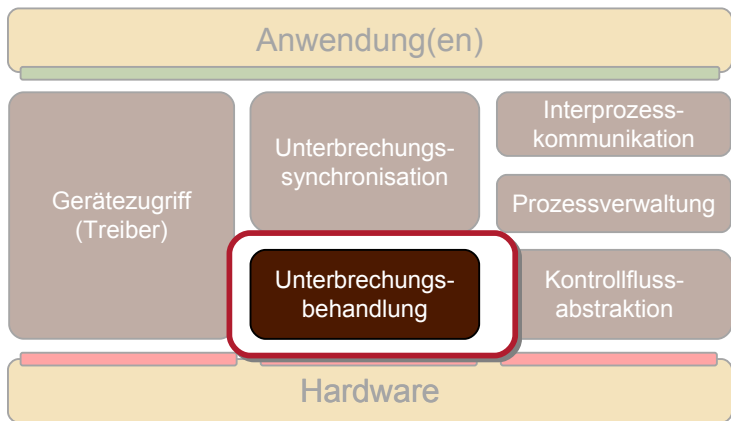
Friedrich-Alexander-Universität
Erlangen Nürnberg

WS 24 – 14. November 2024

<https://sys.cs.fau.de/lehre/ws24/bs>



Überblick: Einordnung dieser VL



Betriebssystementwicklung



Agenda

- Einordnung
- Beispiele
- Aufteilung Interrupt-Bearbeitung
- Implementierung
- SoftIRQs
- SoftIRQ-Beispiel
- Verwandte Konzepte
- Zusammenfassung



Agenda

Einordnung

Beispiele

Aufteilung Interrupt-Bearbeitung

Implementierung

SoftIRQs

SoftIRQ-Beispiel

Verwandte Konzepte

Zusammenfassung



Beispiel Console

Was passiert beim Tippen eines Zeichens in der Console?

1. Benutzer tippt Taste.
2. Keyboard-Controller signalisiert Interrupt.
3. CPU liest Key-Code aus.
4. CPU übersetzt Key-Code in ASCII-Zeichen.
5. ASCII-Zeichen wird im Eingabe-Puffer abgelegt.
6. Wenn „ECHO“-Modus eingeschaltet ist, wird Zeichen auf Bildschirm dargestellt (evntl. Scrollen, Cursor verschieben, Pixel gemäß Zeichensatz setzen/löschen).
7. Die CPU weckt ggf. einen wartenden Prozess auf.



Beispiel Netzwerk

Was passiert beim Empfang einer Ethernet-TCP-Nachricht?

1. Eine Netzwerk-Karte empfängt das Ethernet-Paket.
2. Die Netzwerk-Karte signalisiert einen Interrupt.
3. Die CPU leitet das Paket z.B. an die IP-Schicht weiter.
4. CPU überprüft die IP-Check-Summe.
5. In der IP-Schicht werden ggf. mehrere IP-Pakete zu einem Gesamt-Paket zusammengesetzt.
6. Die CPU leitet das Paket z.B. an die TCP-Schicht weiter.
7. In der TCP-Schicht werden z.B. ACK-Nachrichten generiert.
8. Die Daten werden in die Warteschlange des entsprechenden Ports eingehängt.
9. Die CPU weckt ggf. einen wartenden Prozess auf.



Beispiel NFS-Server

Was passiert, wenn NFS-Client beim Server Daten anfordert?

1. IP-Nachrichteneingang s.o.
2. IP-Nachricht wird in die UDP-Schicht weitergeleitet.
3. Gemäß dem Auftrag im Paket werden z.B. von Platte Daten gelesen.
4. Daten und Ziel-/Empfänger-Adresse werden in ein neues UDP-Paket kopiert.
5. UDP-Paket wird in IP-Schicht weitergeleitet.
6. IP-Paket wird ggf. in mehrere kleinere IP-Paket zerteilt.
7. Für jedes Paket wird eine Check-Summe erstellt.
8. Alle IP-Pakete werden in die Ethernet-Schicht weitergeleitet.
9. Netzwerk-Karte verschickt die Ethernet-Pakete.



Probleme:

- Während der ganzen Unterbrechungslaufzeit sind alle weiteren bzw. alle niederprioren Unterbrechungen gesperrt.
 - Gefahr von großer Interrupt-Verzögerung
 - Gefahr von Datenverlusten
- Unterbrechungsbehandlung kann nicht passiv warten.
 - Interrupts sind gesperrt



Einordnung

Beispiele

Aufteilung Interrupt-Bearbeitung

Implementierung

SoftIRQs

SoftIRQ-Beispiel

Verwandte Konzepte

Zusammenfassung



Was passiert beim Tippen eines Zeichens in der Console?

1. Benutzer tippt Taste.
2. Keyboard-Controller signalisiert Interrupt.
3. CPU liest Key-Code aus.

<= Hardware-IRQ abgearbeitet!

4. CPU übersetzt Key-Code in ASCII-Zeichen.
5. ...

Nach Punkt 3 könnten IRQs wieder erlaubt werden!



Unterbrechungsbehandlung zweigeteilt:

1. Teil, der Zeichen/Pakete/... bei der Hardware ausliest und in einen Puffer kopiert.
 - Interagiert nur minimal mit dem Rest des Systems.
 - Kann (fast) immer ablaufen.
2. Teil, der Zeichen/Pakete/... aus Puffer ausliest und weiterverarbeitet.
 - Ist weitgehend Hardware-unabhängig.
 - Kann (fast) immer unterbrochen werden.



Unterbrechungsbehandlung zweigeteilt:

1. Teil ggf. leer

Beispiel: Timer/Uhr-Interrupt

(Typisch: alle „Edge-Triggered“-Interrupts)



Multiprozessoren:

- 1. und 2. Teil können auf verschiedenen Kernen ausgeführt werden (Beispiel: E/A-Interrupts; Last-Verteilung).
oder
- 1. und 2. Teil müssen auf gleichem Kern ausgeführt werden (Beispiel: Schedule-Interrupt).



Agenda

Einordnung

Beispiele

Aufteilung Interrupt-Bearbeitung

Implementierung

SoftIRQs

SoftIRQ-Beispiel

Verwandte Konzepte

Zusammenfassung



Beispiel / Idee

```
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();
}
```

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



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.



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");
}
```



```
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();
}
```



Agenda

- Einordnung
- Beispiele
- Aufteilung Interrupt-Bearbeitung
- Implementierung
- SoftIRQs
- SoftIRQ-Beispiel
- Verwandte Konzepte**
- Zusammenfassung



Idee:

- Interrupt-Handler liest Daten aus Gerät in Puffer und **setzt Bit**
- vor **Rückkehr in den User-Mode** werden Puffer ausgelesen und die Daten verarbeitet, **wenn Bit gesetzt**

(Linux)



Idee:

- `while (pending) { pending = false; handler(); }`-Schleife wird nur begrenzte Anzahl von Malen durchlaufen.
- Danach wird „`ksoftirqd`“ (**Kernel-Thread**) aufgeweckt, der SoftIRQ-Abarbeitung übernimmt.

Vorteil: Interrupts können CPUs nicht monopolisieren.
(Linux)



Idee:

Hard-Interrupt-Handler triggert nicht (per Nummer) die Ausführung einer bestimmten Methode, sondern fügt eine beliebige Methode in eine Liste ein. Liste wird dann vor Rückkehr in User-Mode abgearbeitet. (Linux)



Idee:

- Interrupt-Handler liest Daten aus Gerät in Puffer
- weckt Thread auf
- Thread liest Puffer aus und ...

Threads können Prioritäten besitzen.

Threads können warten.



Idee:

- Interrupt-Handler fügen Aufträge in **Work-Queue** ein.
- **Work-Queue** wird von **Worker-Thread-Pool** abgearbeitet.



Agenda

- Einordnung
- Beispiele
- Aufteilung Interrupt-Bearbeitung
- Implementierung
- SoftIRQs
- SoftIRQ-Beispiel
- Verwandte Konzepte
- Zusammenfassung**



- Viele verschiedene Möglichkeiten, Interrupts abzuarbeiten.
- Immer Ziel: schnelle Reaktion auf (wichtige) Interrupts.
- Fast immer: „Hard“- und „Soft“-Interrupt-Teil.
- SoftIRQs: „Interrupts“ in Software.
- In aktuellen BS existieren weitere, von SoftIRQs abgeleitete Konzepte.

