Betriebssysteme (BS)

VL 3.1 – Unterbrechungen, Hardware – Grundlagen

Volkmar Sieh / **Daniel Lohmann**

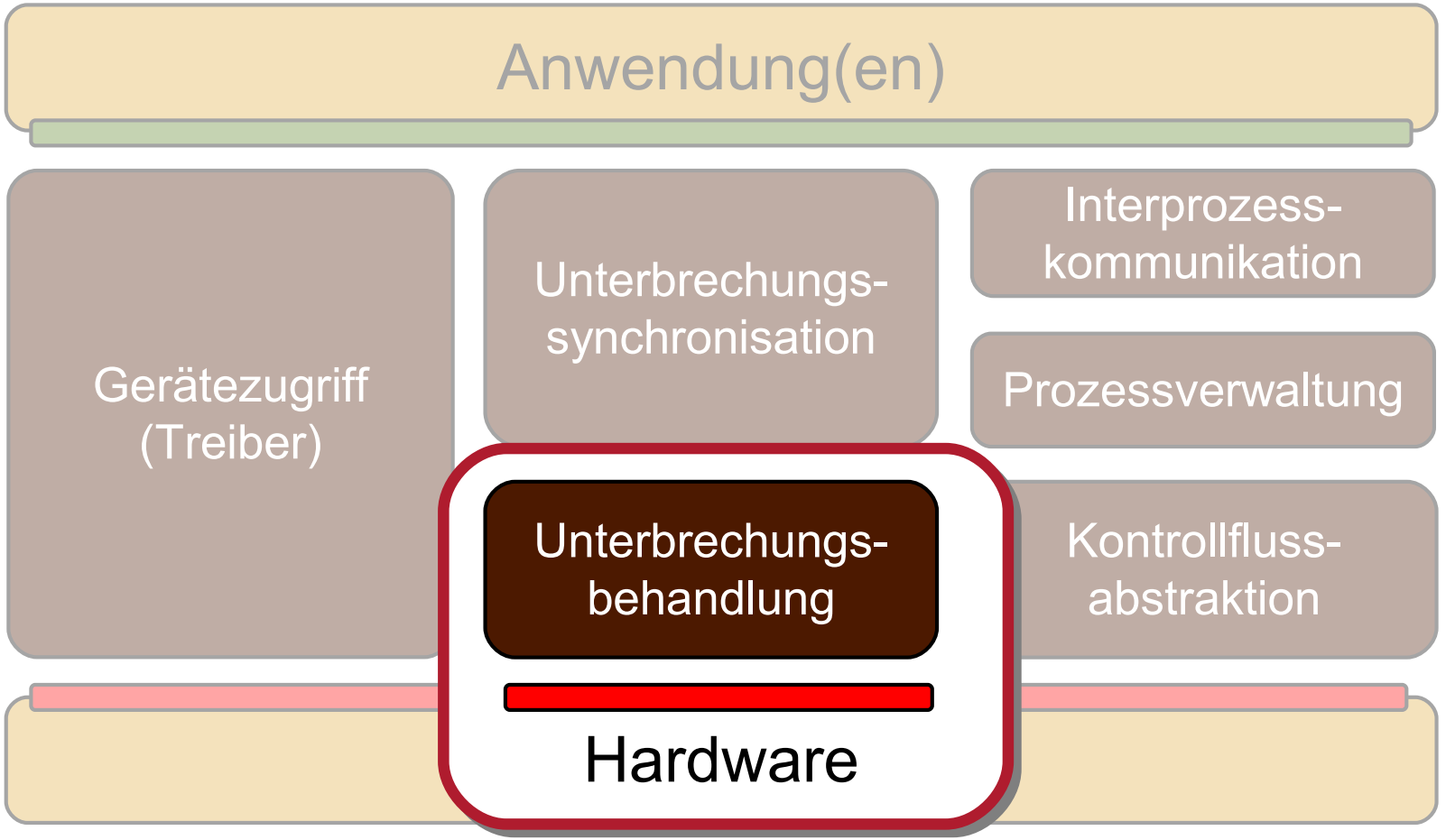
Lehrstuhl für Informatik 4
Verteilte Systeme und Betriebssysteme

Friedrich-Alexander-Universität
Erlangen Nürnberg

WS 20 – 9. November 2020

https://www4.cs.fau.de/Lehre/WS20/V_BS

Überblick: Einordnung dieser VL



Agenda

Einordnung
Grundlagen
Hardware-Architekturen
Zusammenfassung

Agenda

Einordnung

Grundlagen

Hardware-Architekturen

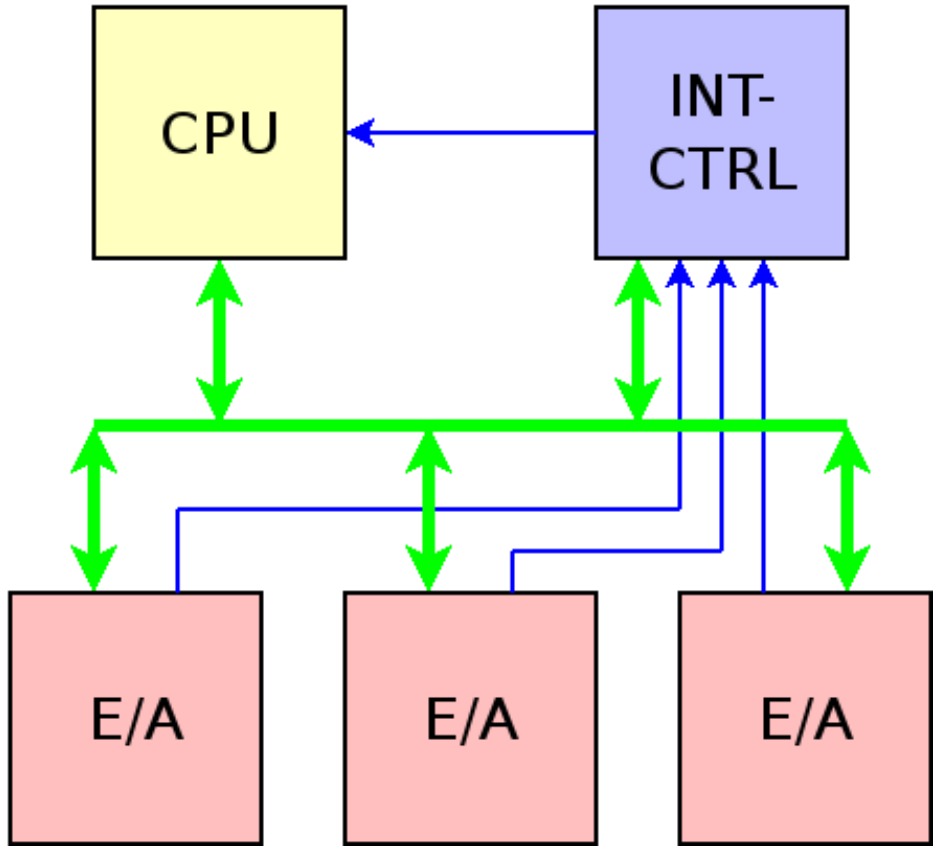
Zusammenfassung

Sinn und Zweck von Unterbrechungen

Ein Blick zurück in die Historie von Betriebssystemen...

- Überlappte Ein-/Ausgabe:
  - Eingaben: Verschwendung von anderweitig nutzbaren Prozessorzyklen bei (oft nicht vorhersagbar langem) *aktivem Warten*
  - Ausgaben: selbständiges Agieren der E/A-Geräte (z.B. durch *DMA*) entlastet die CPU
- Timesharing Betrieb:
  - Zeitgeber-Unterbrechungen geben dem Betriebssystem die Möglichkeit
    - zur *Verdrängung von Prozessen*
    - Aktivitäten zeitgesteuert zu starten

Interrupt-Hardware



Interrupt-Controller ist im einfachsten Fall ein „Oder”-Gatter...

Agenda

Priorisierung

- **Problem**:
  - Mehrere Unterbrechungsanforderungen können gleichzeitig signalisiert werden. Welche ist wichtiger?
  - Während die CPU auf die wichtigste Anforderung reagiert, können weitere Anforderungen signalisiert werden.

Priorisierung

- **Problem**:
  - Mehrere Unterbrechungsanforderungen können gleichzeitig signalisiert werden. Welche ist wichtiger?
  - Während die CPU auf die wichtigste Anforderung reagiert, können weitere Anforderungen signalisiert werden.
- **Lösung**: ein **Priorisierungsmechanismus** ...
  - **in Software**: die CPU hat nur einen IRQ (*interrupt request*) Eingang und fragt die Geräte in bestimmter Reihenfolge ab
  - **in Hardware**: eine Priorisierungsschaltung ordnet Geräten eine Priorität zu und leitet immer nur die dringendste Anforderung zur Behandlung weiter
  - **mit festen Prioritäten**: jedem Gerät wird statisch eine Priorität zugeordnet
  - **mit variablen Prioritäten**: Prioritäten sind dynamisch änderbar oder wechseln zum Beispiel zyklisch

Verlust von IRQs

- **Problem**:
  - während der Behandlung oder Sperrung von Unterbrechungen, kann die CPU keine neuen Unterbrechungen behandeln
  - die Speicherkapazität für Unterbrechungsanforderungen ist endlich.
    - i.d.R. ein Bit pro Unterbrechungseingang

Verlust von IRQs

- **Problem**:
  - während der Behandlung oder Sperrung von Unterbrechungen, kann die CPU keine neuen Unterbrechungen behandeln
  - die Speicherkapazität für Unterbrechungsanforderungen ist endlich.
    - i.d.R. ein Bit pro Unterbrechungseingang
- **Lösung**: in Software
  - die Unterbrechungsbehandlungsroutine sollte möglichst kurz sein (zeitlich!), um die Wahrscheinlichkeit von Verlusten zu minimieren
  - Unterbrechungen sollten nicht unnötig lange gesperrt werden
  - jeder Gerätetreiber sollte davon ausgehen, dass **eine** Unterbrechung **mehr als eine** abgeschlossene E/A Operation anzeigen kann

Zuordnung einer Behandlungsroutine

- **Problem**:
  - die Software soll mit möglichst wenig Aufwand herausfinden können, welches Gerät die Unterbrechung ausgelöst hat
    - eine sequentielle Abfrage der Geräte kostet nicht nur Zeit, sondern verändert die Zustände von E/A Bussen und unbeteiligten Geräten

Zuordnung einer Behandlungsroutine

- **Problem**:
  - die Software soll mit möglichst wenig Aufwand herausfinden können, welches Gerät die Unterbrechung ausgelöst hat
    - eine sequentielle Abfrage der Geräte kostet nicht nur Zeit, sondern verändert die Zustände von E/A Bussen und unbeteiligten Geräten
- **Lösung**:
  - jeder Unterbrechung wird eine Nummer zugeordnet, die als Index in eine Vektortabelle verwendet wird
    - die Vektornummer hat nicht zwangsläufig etwas mit der Priorität zu tun
    - es kommt in der Praxis leider vor, dass Geräte sich eine Vektornummer teilen müssen (*interrupt sharing*)
  - der Aufbau der Vektortabelle variiert je nach Prozessortyp
    - meist enthält sie Zeiger auf Funktionen
    - seltener sind die Einträge selbst bereits Instruktionen

Zustandssicherung

- **Problem**:
  - nach der Ausführung der Behandlungsroutine soll zum normalen Kontext zurückgekehrt werden können
  - die Behandlung soll quasi unbemerkt ablaufen (*transparency*)

Zustandssicherung

- **Problem**:
  - nach der Ausführung der Behandlungsroutine soll zum normalen Kontext zurückgekehrt werden können
  - die Behandlung soll quasi unbemerkt ablaufen (*transparency*)
- **Lösung**:
  - Zustandssicherung durch Hardware
    - nur das Notwendigste: z.B. Rücksprungadresse u. Prozessorstatuswort
    - Wiederherstellung durch speziellen Befehl, z.B. iret, rte, ...
  - Zustandssicherung durch Software
    - da Unterbrechungen jederzeit auftreten können, muss auch die Behandlungsroutine Zustände sichern und wiederherstellen

Geschachtelte Behandlung

- **Problem**:
  - um auf sehr wichtige Ereignisse schnell reagieren zu können, soll auch eine Unterbrechungsbehandlung unterbrechbar sein
  - eine unbegrenzte Schachtelungstiefe muss aber vermieden werden

Geschachtelte Behandlung

- **Problem**:
  - um auf sehr wichtige Ereignisse schnell reagieren zu können, soll auch eine Unterbrechungsbehandlung unterbrechbar sein
  - eine unbegrenzte Schachtelungstiefe muss aber vermieden werden
- **Lösung**:
  - die CPU erlaubt immer nur Unterbrechungen mit höherer Priorität
  - die aktuelle Priorität wird im Prozessorstatuswort gespeichert
  - die vorherige Priorität wird auf einem Stapel abgelegt

Interrupt-Ablauf

HW: E/A-Gerät signalisiert Interrupt an Interrupt-Controller

HW: Interrupt-Controller signalisiert Interrupt an CPU

HW: CPU sichert einige Register (PC, IE, ...)

HW: CPU disabled Interrupt-Eingang (IE = 0)

HW: CPU holt sich von Interrupt-Controller IRQ-Nummer

HW: CPU lädt Vektor aus Interrupt-Vektor-Tabelle in PC

SW: Interrupt-Handler sichert weitere Register

SW: Interrupt-Handler behandelt Interrupt

SW: Interrupt-Handler restauriert Register

SW: Interrupt-Handler ruft `iret` auf

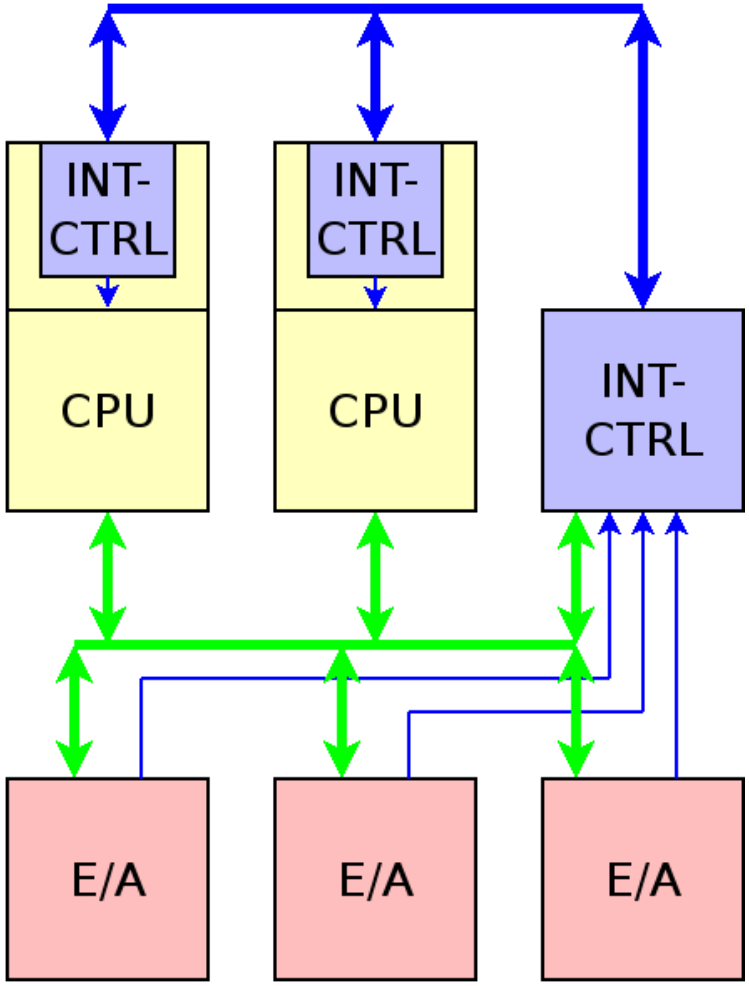
HW: CPU restauriert PC, IE, ...
Rücksprung in das unterbrochene Haupt-Programm,
Interrupt-Eingang wieder enabled

Multiprozessorsysteme

- **Problem**:
  - Unterbrechungen können immer nur von einer CPU behandelt werden. Aber welche?
  - es gibt eine weitere Kategorie von Unterbrechungen: die Interprozessor-Unterbrechungen
- **Lösung**: die Hardware zur Unterbrechungsbehandlung auf Multiprozessorsystemen muss komplexer ausgelegt sein. Es gibt viele Entwurfsvarianten ...
  - feste Zuordnung
  - zufällige Zuordnung
  - programmierbare Zuordnung
  - Zuordnung unter Berücksichtigung der Prozessorlast

  ... und Kombinationen davon.

Interrupt-Hardware (SMP)



Interrupt-Controller sind selbständige, konfigurierbare Einheiten...

Gefahr: „unechte Unterbrechungen“

(„*spurious interrupts*“)

- **Problem**: ein technischer Mechanismus zur Unterbrechungsbehandlung birgt die Gefahr von fehlerhaften Unterbrechungsanforderungen, z.B. durch ...
  - Hardwarefehler
  - fehlerhaft programmierte Geräte
- **Lösung**:
  - Hardware- und Softwarefehler vermeiden 
  - Betriebssystem „defensiv“ programmieren
    - mit unechten Unterbrechungen rechnen

Gefahr: „Unterbrechungsstürme“

(„*interrupt storms*“)

- **Problem**:
  - hochfrequente Unterbrechungsanforderungen können einen Rechner lahm legen
  - es handelt sich entweder um unechte Unterbrechungen oder der Rechner ist mit der E/A Last überfordert
  - kann leicht mit Seitenflattern (*thrashing*) verwechselt werden
- **Lösung**: durch das Betriebssystem
  - Unterbrechungsstürme erkennen
  - das verursachende Gerät deaktivieren