Betriebssystembau (BSB)

VL 1 – Lehrveranstaltungskonzept & Organisation

Alexander Lochmann

Lehrstuhl für Informatik 12 – Arbeitsgruppe Systemsoftware
Technische Universität Dortmund

`https://sys.cs.tu-dortmund.de/de/lehre/ws23/bsb`

WS 23 – 09. Oktober 2023

technische universität dortmund

arbeitsgruppe systemsoftware

Vorwort

Die Lehrveranstaltung ist grundsätzlich für alle Studierende offen. Sie verlangt allerdings gewisse Vorkenntnisse.

Lernziele

- **Vertiefen** des Wissens über die interne Funktionsweise von Betriebssystemen
  - Ausgangspunkt: Betriebssysteme
  - Schwerpunkt: Nebenläufigkeit und Synchronisation
- **Entwickeln** eines Betriebssystems *von der Pike auf*
  - OOStuBS Lehrbetriebssystem (freiwillig: MPStuBS)
  - Praktische Erfahrungen im Betriebsystembau machen
- **Verstehen** der technologischen Hardware-Grundlagen
  - PC-Technologie verstehen und einschätzen können
  - Schwerpunkt: x86-Architektur

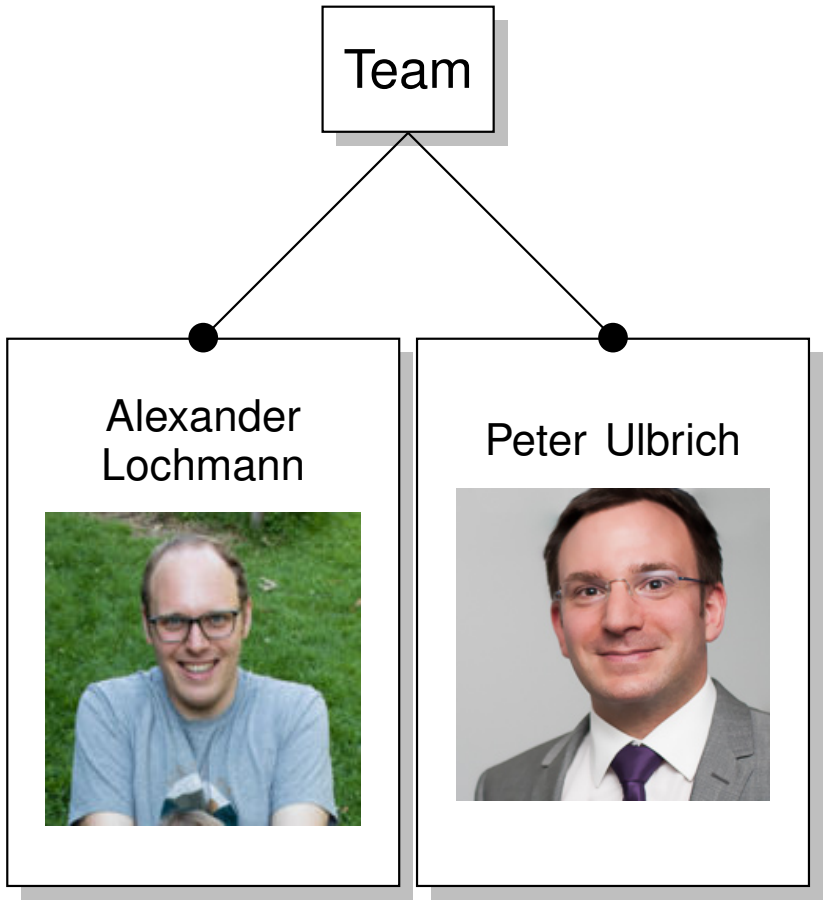
Voraussetzungen

- Rechnerarchitektur, **Betriebssysteme**
- C / **C++**, Assembler (x86)
- Ein gewisses Maß an **Durchhaltevermögen**
- Freude an systemnaher und **hardwarenaher Programmierung**

Wir arbeiten auf der “nackten Maschine” (*bare metal*)!

Die meisten sind überrascht, wie viel Spaß das macht :-)

https://sys.cs.tu-dortmund.de/{al,pu}



Veranstaltungskonzept

- **Wissensvermittlung**
  - Stoff der Vorlesung $\rightsquigarrow$ Präsenzbetrieb
  - Stoff der Tafelübung $\rightsquigarrow$ Präsenzbetrieb
  - Handzettel (engl. *handout*):
    `https://sys.cs.tu-dortmund.de/de/lehre/ws23/bsb/`
- **Praktische Arbeit**
  - Übungs- und Programmieraufgaben
  - In Präsenz und Heimarbeit
  - In 2er- oder 3er-Gruppen
- **Interaktion**
  - Vorlesung und Tafelübung $\mapsto$ Wöchentliche Präsenzveranstaltungen
  - Rechnerübung $\mapsto$ Betreute Laborarbeit, Heimarbeit (VC nach Absprache)
  - Matrix-Raum: `https://matrix.to/#/#bsb-helpdesk:fachschaften.org`

Bedeutung der Übungen

- Tafelübungen/Seminar $\rightsquigarrow$ „*learning by exploring*“
  - Besprechung der Übungsaufgaben, Skizzierung von Lösungswegen
  - Vertiefung des Vorlesungsstoffes, Klärung offener Fragen
- Rechnerarbeit $\rightsquigarrow$ „*learning by doing*“
  - Selbstständiges Bearbeiten der Übungsaufgaben am Rechner
    - Abgabe der bearbeiteten Übungsaufgaben
    - Klärung von Unklarheiten/Problemen bei/mit den Übungsaufgaben

$\rightarrow$ Bereitet euch vor! Wir erwarten konkrete Fragen!

Der, die, das.
Wer, wie, was?
Wieso, weshalb, warum?
Wer nicht fragt, bleibt dumm!

Studien- und Prüfungsleistungen (1)

VL – Vorlesung **2,5**

Vorstellung und detailierte Behandlung des Lehrstoffs

+

Ü – Übung **2,5**

- Übung **OOStuBS** (Alternativ: MPStubBS)
- 6 Übungsaufgaben
- Abnahme alle 14 Tage

+

RÜ – Rechnerübung **0**

- **Betreutes** Arbeiten am Rechner
- Hilfe zu **OOStuBS** und MPStuBS

Studien- und Prüfungsleistungen (2)

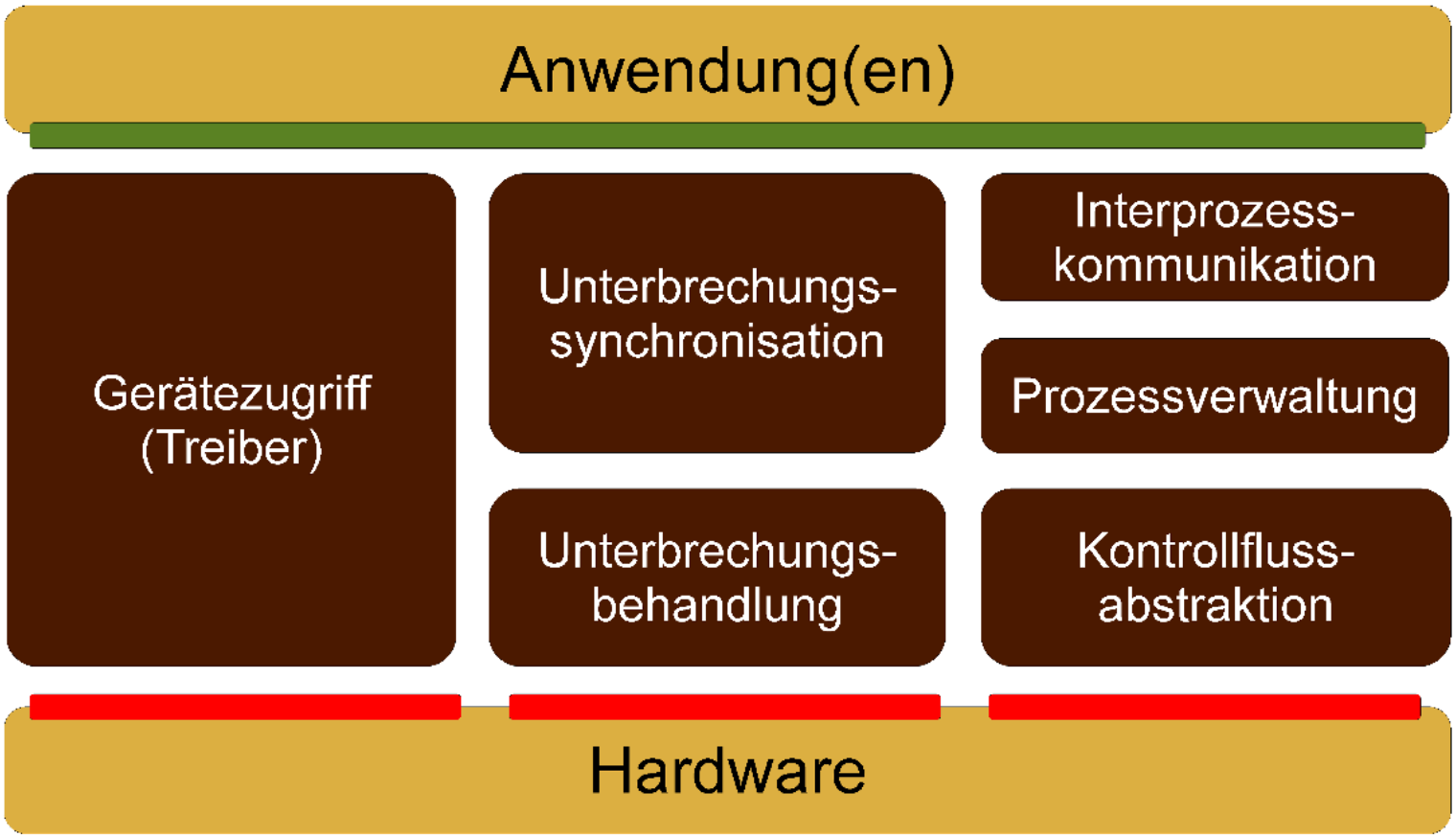
- **Studienleistung**[1]
  - Bearbeitung der Übungsaufgaben — Persönliche Abgabe
  - Keine Punkte im klassischen Sinne — Eigenverantwortung
- **Prüfungsleistung**
  - 30 min. mündliche Prüfung — Kompetenzorientiert
- **Berechnung der Modulnote**
  - Note der mündlichen Prüfung + “Übungsbonus” in Zweifelsfällen

[1] Nicht verpflichtend

Terminübersicht Wintersemester 2023/24

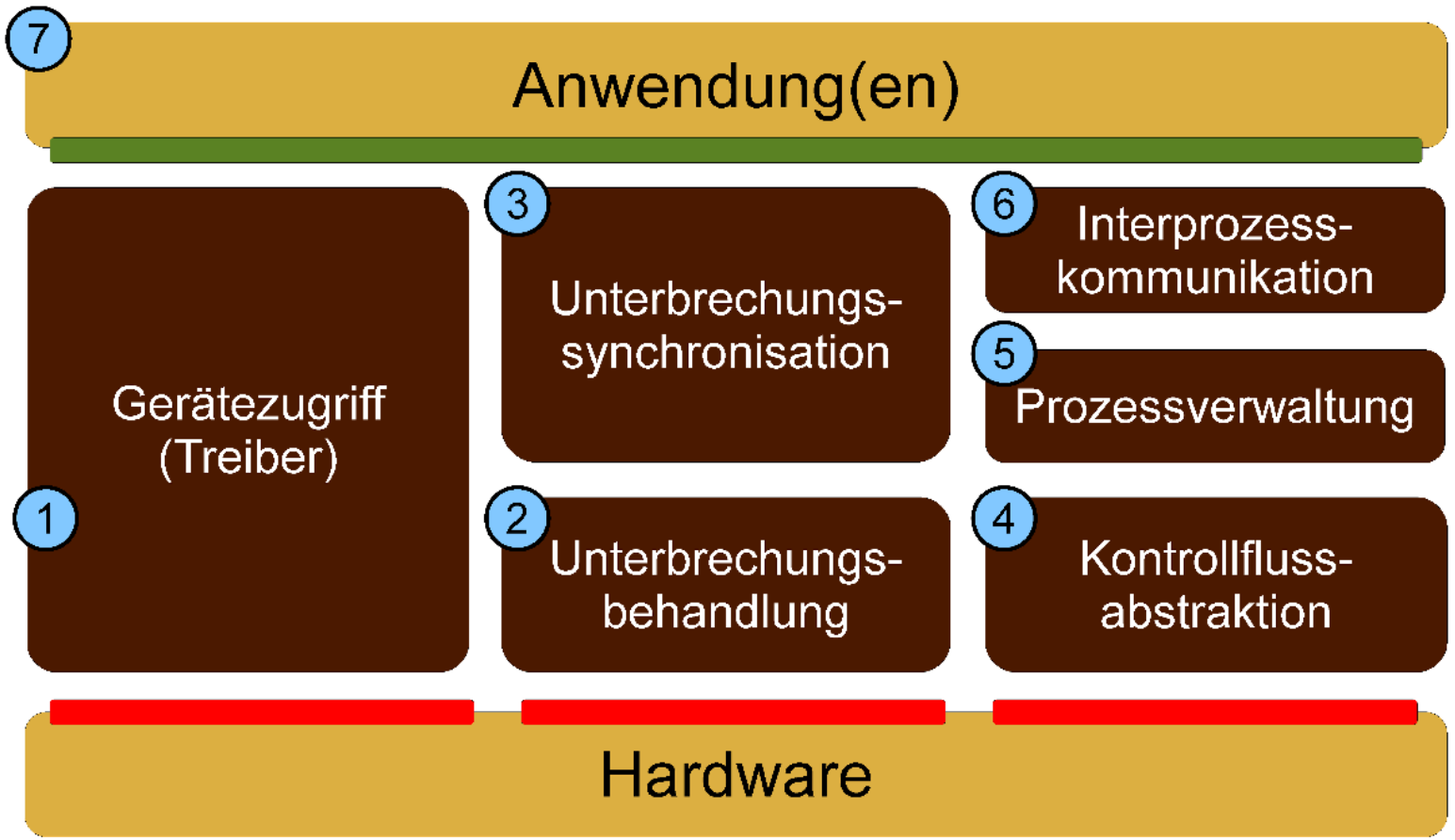
KW	Mo 14-16	Di 08:30-10	Di 14-16	Raum
41	VL_{1-2}	$Ü_1$		Vorlesung
42	VL_3	RÜ	RÜ	E.003, OH12
43	VL_4	RÜ	RÜ	
44	VL_5	$Ü_2$	A_1	Tafelübung
45	VL_6	$Sem_{2?}$	RÜ	R205, OH16
46	VL_7	$Ü_3$	A_2	R205, OH16
47	VL_8	$Sem_{3?}$	RÜ	
48	VL_9	$Ü_4$	A_3	Rechnerübung
49	VL_{10}	RÜ	RÜ	E07, OH16
50	VL_{11}	$Ü_5$	A_4	E07, OH16
51	VL_{12}	RÜ	RÜ	
52				
01				
02	VL_{13}	$Ü_6$	A_5	
03	VL_{14}	RÜ	RÜ	
04		$Ü_7$	A_6	
05			RÜ	
06	VL_{15}		A_7	

Aufbau eines Betriebssystems



Themenübersicht Übung

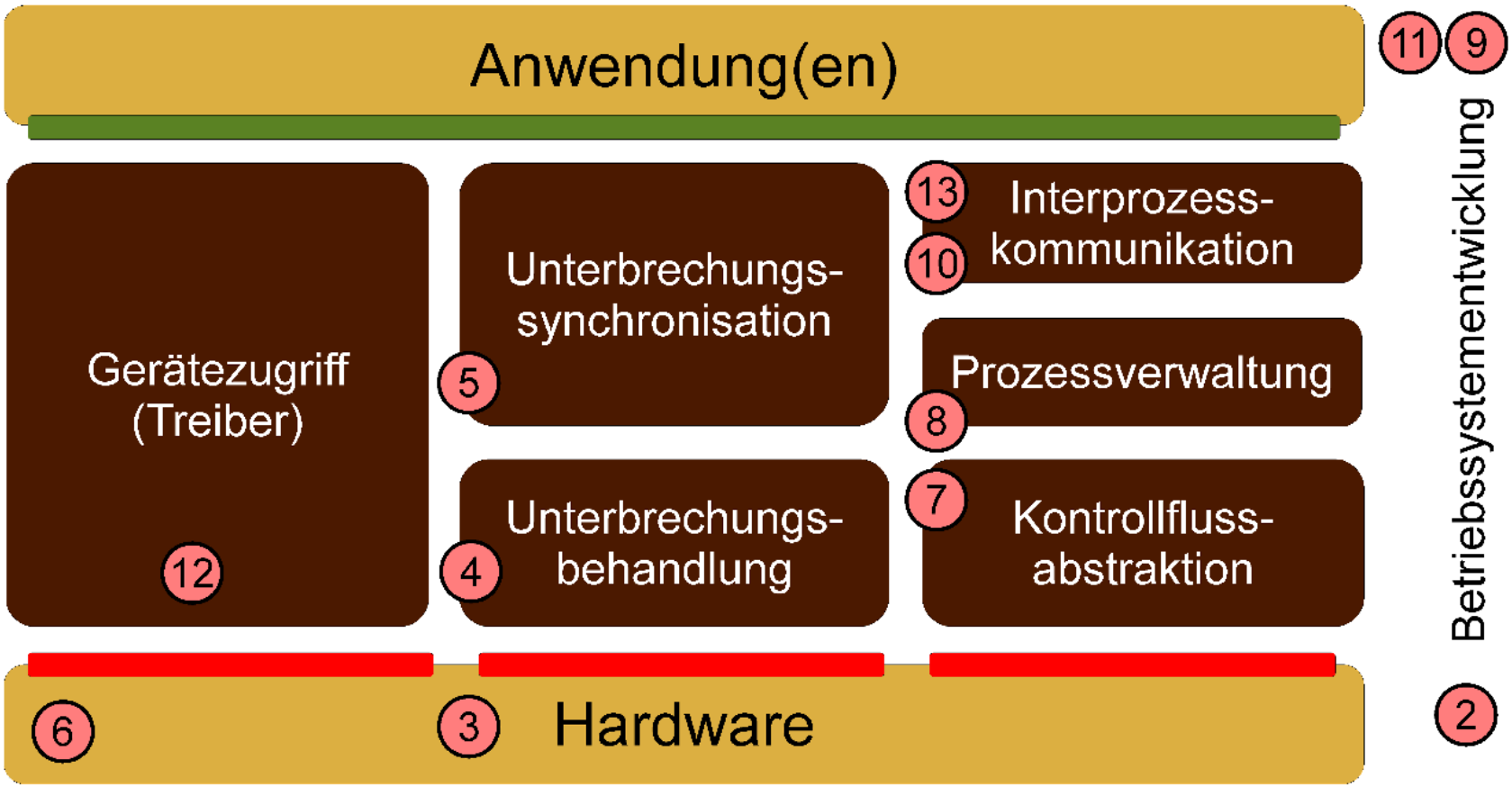
Am Beispiel von: OOStuBS, MPStuBS



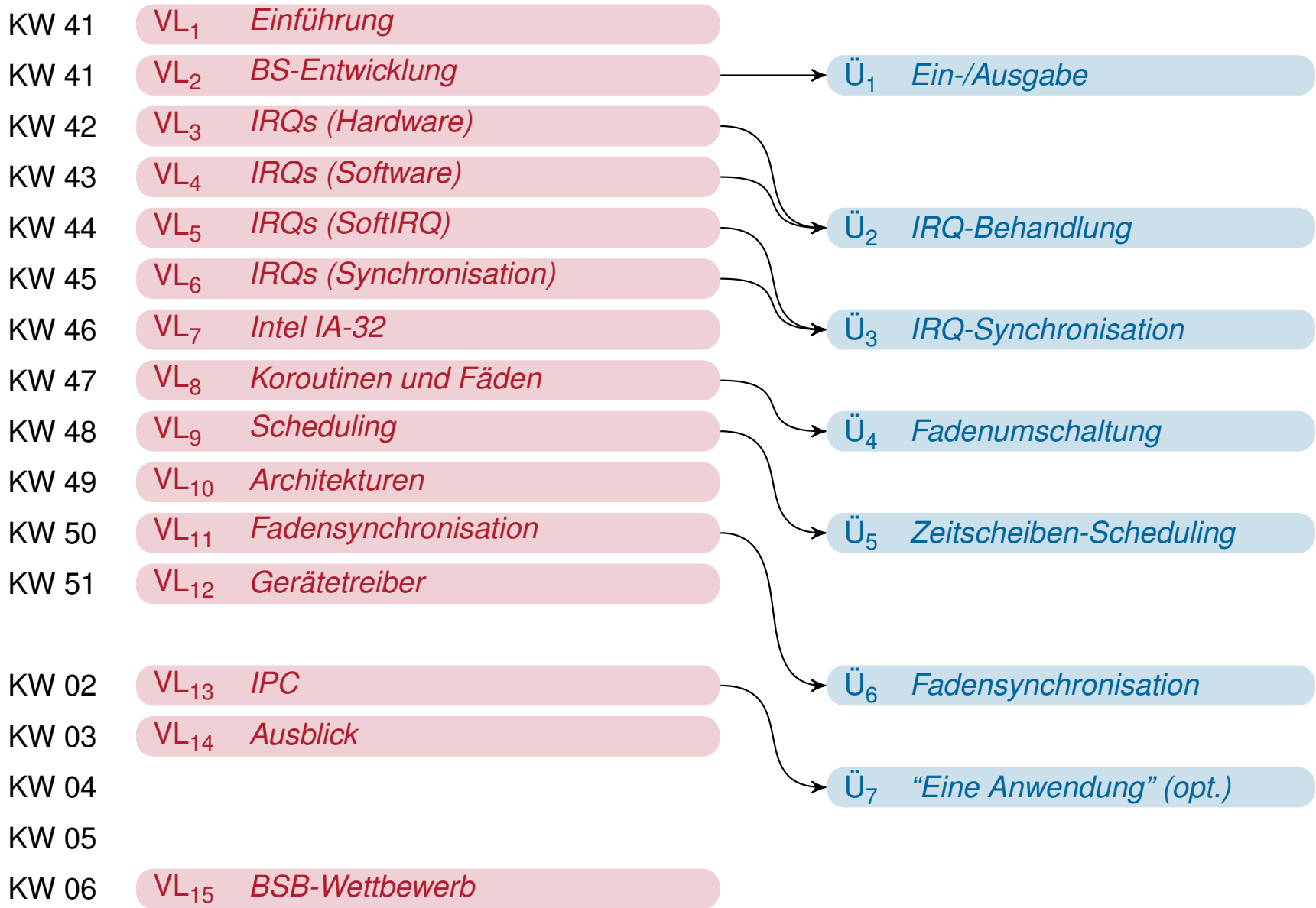
Betriebssystementwicklung

Themenübersicht Vorlesung

Am Beispiel von: x86, MC68k, TriCore; Windows, Linux



Verzahnung von Vorlesung und Übungsaufgaben



VL_2 Betriebssystementwicklung

(oft ein harter Kampf!)

- Erste Schritte

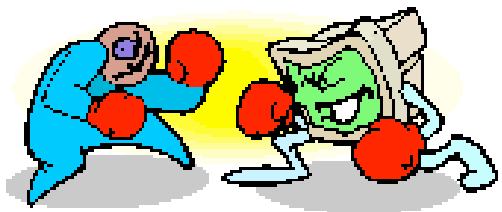
  Wie bringt man sein System auf die Zielhardware?

  - Übersetzen und Linken für “nackte Hardware”
  - Bootvorgang

- Testen und Fehlersuche

  Was tun, wenn das System nicht reagiert?

  - “printf”-*Debugging*
  - Simulatoren
  - *Debugger*
  - *Remote debugging*
  - Hardwareunterstützung



$VL_{3–5}$ Programmunterbrechungen

- im Prinzip
  - Unterbrechungen, *Traps* und Ausnahmen
  - Vektortabellen
  - geschachtelte Unterbrechungen
  - *spurious interrupts*
- beim PC
  - CPU und APIC
  - Unterbrechungen in Multiprozessorsystemen
- Behandlung im Betriebssystem
  - Kopplungsfunktion
  - Zustandssicherung

VL_6 Unterbrechungssynchronisation

- Zusammenspiel zwischen Unterbrechungsbehandlung und “normalem” Kontrollfluss
  - Ursache und Problem
  - Kontrollflussebenenmodell
- Hardware-Mechanismen zur “harten Synchronisation”
  - `cli` und `sti`
  - Unterbrechungsebenen
- Software-Mechanismen zur “weichen Synchronisation”
  - Pro-/Epilogmodell und Varianten
  - Unterbrechungstransparente Algorithmen

VL_7 Das x86 Programmiermodell

- Die Entwicklung der x86 CPU-Familie
  - vom 8086 bis zum Core i7
- Relikte und Eigenarten (*quirks*)
  - *Real Mode*
  - *A20 Gate*
- Neuerungen des *Protected Mode*
  - Ringe und Schutzmodell
  - *Task*-Modell
- Hardwarevirtualisierung

VL_8 Programmfäden

- Realisierung von Programmfäden
  - beim MC68k, Infinion TriCore, Intel x86
  - Fortsetzungen und Koroutinen als Basis
  - Implementierung des Kontextwechsels
- Fadenmodelle
  - leicht vs. schwer vs. federgewichtig vs. . . .
  - Umsetzung in einer Systemfamilie

VL_9 Ablaufplanung

- Kurze Wiederholung und Vertiefung
  - Grundprinzipien
  - Klassifikation
  - neue Strategien
- Beispiele aus der Praxis
  - Windows
  - Linux
  - Scheduling auf Multiprozessor-Systemen
- Herausforderungen beim Betriebssystembau
  - Zusammenspiel Ablaufplanung $\rightleftharpoons$ Unterbrechungssynchronisation

VL_{10} Architekturen

- Wie organisiert man ein Betriebssystem: Architekturmodelle
  - Bibliotheken
  - Monolithen
  - Mikrokerne
  - Exokerne
  - Hypervisor
- Geschichte: Revolutionen, Religionen . . . und die Realität
  - Bewertungskriterien
  - Erfolgs- und Misserfolgsgeschichten
- Beispiele aus der Praxis
  - OS360, Unix, Linux, L4, Windows
  - exoKernel, xen, vmware
  - . . .

VL_{11} Fadensynchronisation

- Grundsätzliches
  - Vorraussetzungen
  - aktives und passives Warten
- Synchronisationsprimitiven
  - *Mutex*, *Semaphore* und *Condition*
  - aus der Sicht des BS-Entwicklers
- spezielle Probleme
  - Wechselwirkung Synchronisation $\rightleftharpoons$ Ablaufplanung
  - Fortschrittsgarantie und Verklemmung
- Beispiele aus der Praxis
  - Synchronisationsprimitiven in Windows

VL_{12} Geräteprogrammierung

- Treiber und ihre Bedeutung
  - Vielfalt von Geräten
  - Probleme
- Komponentenmodell für Treiber
  - Struktur eines E/A-Systems
  - Treiberklassen und -schnittstellen
- Beispiele aus der Praxis
  - Windows
  - Linux

VL_{13} Interprozesskommunikation (IPC)

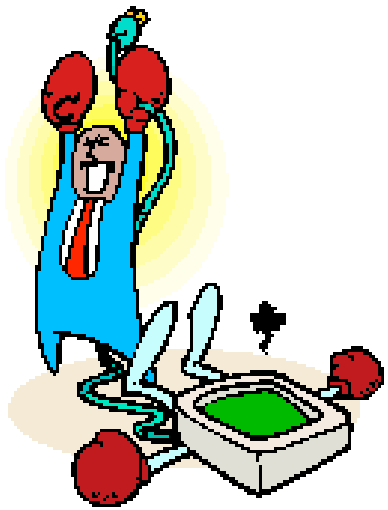
- Grundsätzliches
  - Wechselwirkung $\rightleftharpoons$ Synchronisation
  - implizite und explizite Synchronisation
- Abstraktionen jenseits von *Semaphore*
  - gemeinsamer und verteilter Speicher
  - Fern- und Nahaufrufe
- Dualität nachrichtenbasierter und prozeduraler Systeme
  - konkrete Beispiele
  - Mikrokern $\rightleftharpoons$ Monolith

VL_{14} Zusammenfassung und Ausblick

Quo Vadis Betriebssysteme?

- Zusammenfassung
  - Zusammenfassung des Lernstoffes
  - Diskussion der Evaluationsergebnisse
  - Tipps und Hinweise für die Prüfung
- Ausblick: Neue Herausforderungen
  - Multi- und Manycore Systeme
  - Heterogene Hardware
- Ausblick: Systeme aus der Forschung
  - Corey
  - Barrelfish/Multikernel
  - Factored OS
  - TxOS
  - OctoPOS/Invasive Computing
  - ...

Das Endziel. . .



Viel Spaß!