

# Betriebssysteme (BS)

## VL 1 – Einführung

**Daniel Lohmann**

Lehrstuhl für Informatik 4  
Verteilte Systeme und Betriebssysteme

Friedrich-Alexander-Universität  
Erlangen Nürnberg

WS 14 – 8. Oktober 2014

[https://www4.cs.fau.de/Lehre/WS14/V\\_BS](https://www4.cs.fau.de/Lehre/WS14/V_BS)



*Die Lehrveranstaltung ist grundsätzlich für alle Studiengänge offen. Sie verlangt allerdings gewisse Vorkenntnisse. Diese müssen nicht durch Teilnahme an den Lehrveranstaltungen von I4 erworben worden sein.*



- **Vertiefen** des Wissens über die interne Funktionsweise von Betriebssystemen
  - Ausgangspunkt: Systemprogrammierung
  - Schwerpunkt: Nebenläufigkeit und Synchronisation
- **Entwickeln** eines Betriebssystems *von der Pike auf*
  - OOSTuBS / MPStuBS Lehrbetriebssysteme
  - **Praktische** Erfahrungen im Betriebssystembau machen
- **Verstehen** der technologischen Hardware-Grundlagen
  - PC-Technologie verstehen und einschätzen können
  - Schwerpunkt: Intel x86 / IA-32



# Voraussetzungen

---

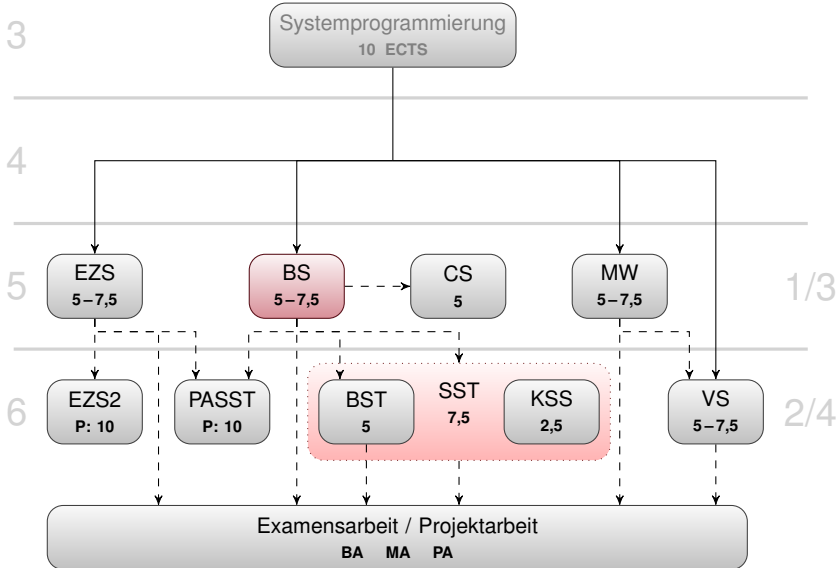
- Rechnerarchitektur, **Systemprogrammierung**
- C / **C++**, Assembler (x86)
- Ein gewisses Maß an **Durchhaltevermögen**
- Freude an systemnaher und **hardwarenaher Programmierung**

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

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



# Einpassung in den Musterstudienplan (Bachelor/Master)



## VL – Vorlesung

2,5

Vorstellung und detaillierte Behandlung des Lehrstoffs

+

## Ü – Übung

2,5

- Übung **OOSTuBS**
- 6 – 7 Übungsaufgaben
- Abnahme alle 14 Tage

oder

## EÜ – Erweiterte Übung

5

- Übung **MPStuBS**
- erweiterte Aufgaben
- Rechnerübung “Pflicht”

+

## RÜ – Rechnerübung

0

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



## ■ **Wahlpflichtmodul** (Bachelor/Master) der Vertiefungsrichtung **Verteilte Systeme und Betriebssysteme**

- eigenständig (nur BS) VL + Ü oder VL + EÜ
- mit weiteren Veranstaltungen VL oder VL + Ü oder VL + EÜ

## ■ Studien- und Prüfungsleistungen

- Bachelor Prüfungsleistung
- Master Prüfungsleistung  
erworben durch
  - erfolgreiche Teilnahme an den Übungen
  - erfolgreiche Bearbeitung aller Übungsaufgaben
  - 30 min. mündliche Prüfung

## ■ Berechnung der Modulnote

- Note der mündlichen Prüfung + “Übungsbonus” in Zweifelsfällen



## Übung

Raum 0.031 (Aquarium), Abgaben in 00.153 (CIP3)

- Zwei Termine zur Auswahl
  - Do, 10:00 – 11:30 (0.031) *oder* Do, 12:30 – 14:00 (0.031)
- Übungsaufgaben sind in 2er-Gruppen zu bearbeiten
- Anmeldung über **WAFFEL** (URL siehe Webseite)
  - Freischaltung erfolgt nach der Vorlesung, heute im Tagesverlauf

## Rechnerübung

Raum 00.153 (CIP3)

- Zwei Termine zur Auswahl
  - Mo, 14:00 – 15:30 (00.153) *oder* Mi, 14:00 – 15:30 (00.153)
- Betreuer können auch jederzeit direkt angesprochen werden





# Terminübersicht Wintersemester 2014

KW	Mo 14-16	Di 10-12	Mi 14-14	Do 10-12	Do 12-14	Raum
06.10.		VL <sub>1</sub>		Ü <sub>1</sub>	Ü <sub>1</sub>	0.031
13.10.		VL <sub>2</sub>		RÜ	RÜ	0.031
20.10.	RÜ	VL <sub>3</sub>	RÜ	Ü <sub>2</sub>	Ü <sub>2</sub>	00.153
27.10.	RÜ	VL <sub>4</sub>	RÜ	A <sub>1</sub>	A <sub>1</sub>	00.153
03.11.	RÜ	VL <sub>5</sub>	RÜ			
10.11.	RÜ	VL <sub>6</sub>	RÜ	Ü <sub>3</sub>	Ü <sub>3</sub>	
17.11.	RÜ	VL <sub>7</sub>	RÜ	A <sub>2</sub>	A <sub>2</sub>	
24.11.	RÜ	VL <sub>8</sub>	RÜ	Ü <sub>4</sub>	Ü <sub>4</sub>	
01.12.	RÜ	VL <sub>9</sub>	RÜ	A <sub>3</sub>	A <sub>3</sub>	
08.12.	RÜ	VL <sub>10</sub>	RÜ	Ü <sub>5</sub>	Ü <sub>5</sub>	
15.12.	RÜ	VL <sub>11</sub>	RÜ	A <sub>4</sub>	A <sub>4</sub>	
12.01.	RÜ	VL <sub>12</sub>	RÜ	Ü <sub>6</sub>	Ü <sub>6</sub>	
19.01.	RÜ	VL <sub>13</sub>	RÜ	A <sub>5</sub>	A <sub>5</sub>	
26.01.	RÜ	VL <sub>14</sub>	RÜ	A <sub>6</sub>	A <sub>6</sub>	
03.02.				A <sub>7</sub>	A <sub>7</sub>	



## Dozenten Vorlesung



Daniel Lohmann

## Tafel- und Rechnerübung



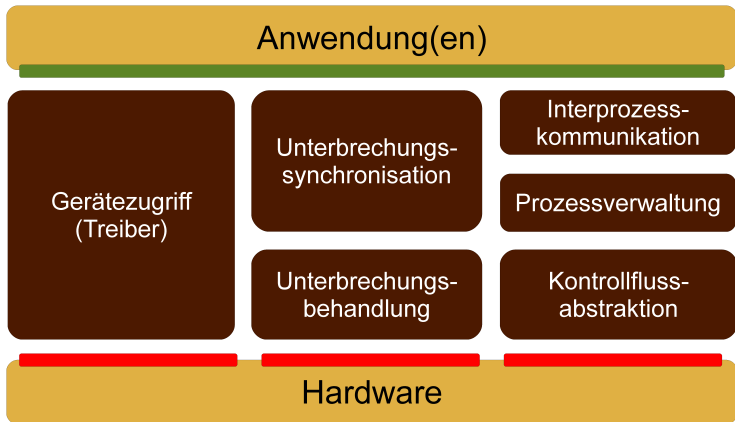
Gabor Drescher



Daniel Danner

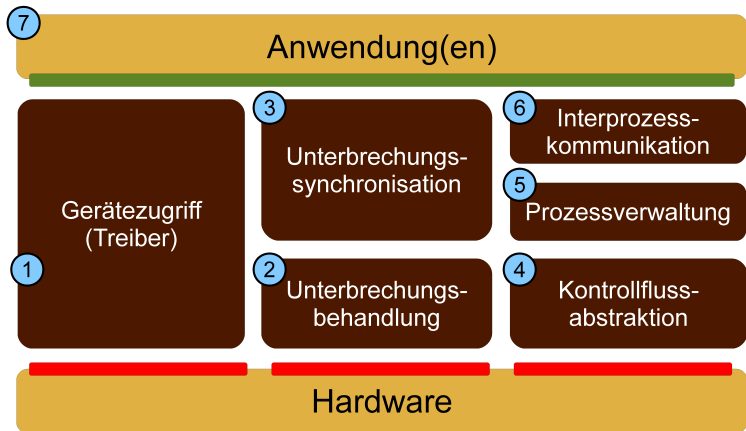


# Aufbau eines Betriebssystem



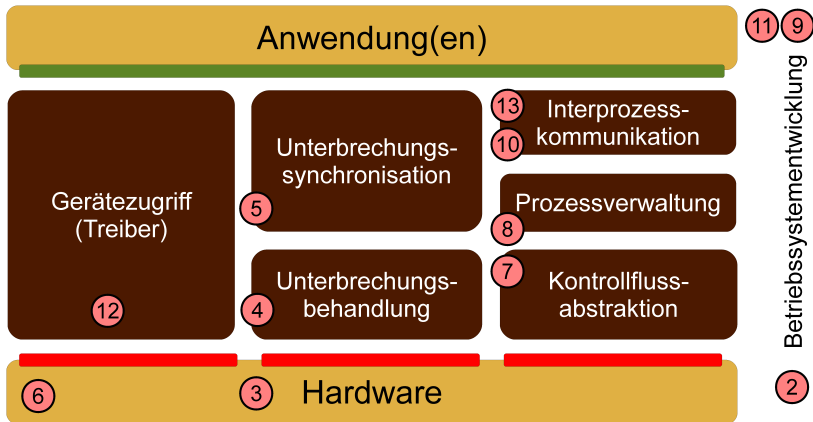
# Themenübersicht Übung

Am Beispiel von: OOSTuBS, MPStuBS



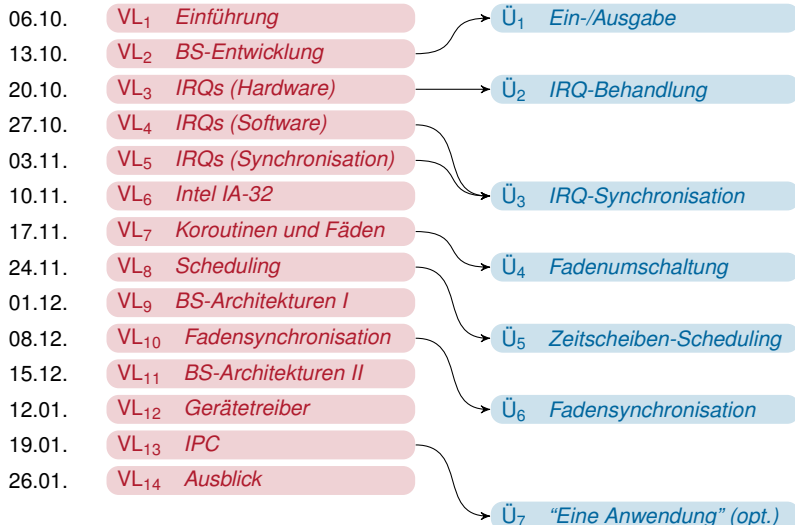
# Themenübersicht Vorlesung

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



# Verzahnung von Vorlesung und Übungsaufgaben

## KW



## ■ 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



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





- 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



- 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



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



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



- 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
  - ...



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



- Neue Herausforderungen
  - Multi- und Manycore Systeme
  - Heterogene Hardware
  
- Systeme aus der Forschung
  - Corey
  - Barrelfish/Multikernel
  - Factored OS
  - TxOS
  - OctoPOS/Invasive Computing
  - ...



- 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





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



- Zusammenfassung des Lernstoffes
- Diskussion der Evaluationsergebnisse
- Tipps und Hinweise für die Prüfung
- Ausblick





Viel Spaß!