

# Systemprogrammierung

*Grundlage von Betriebssystemen*

## Teil A – I. Organisation

Wolfgang Schröder-Preikschat

23. April 2019



## Agenda

---

Einleitung

Konzept

Lehrkanon

Lehrziele

Vorkenntnisse

Veranstaltungsbetrieb

Leistungsnachweise

Ausklang



## Einleitung

### Konzept

Lehrkanon

Lehrziele

Vorkenntnisse

Veranstaltungsbetrieb

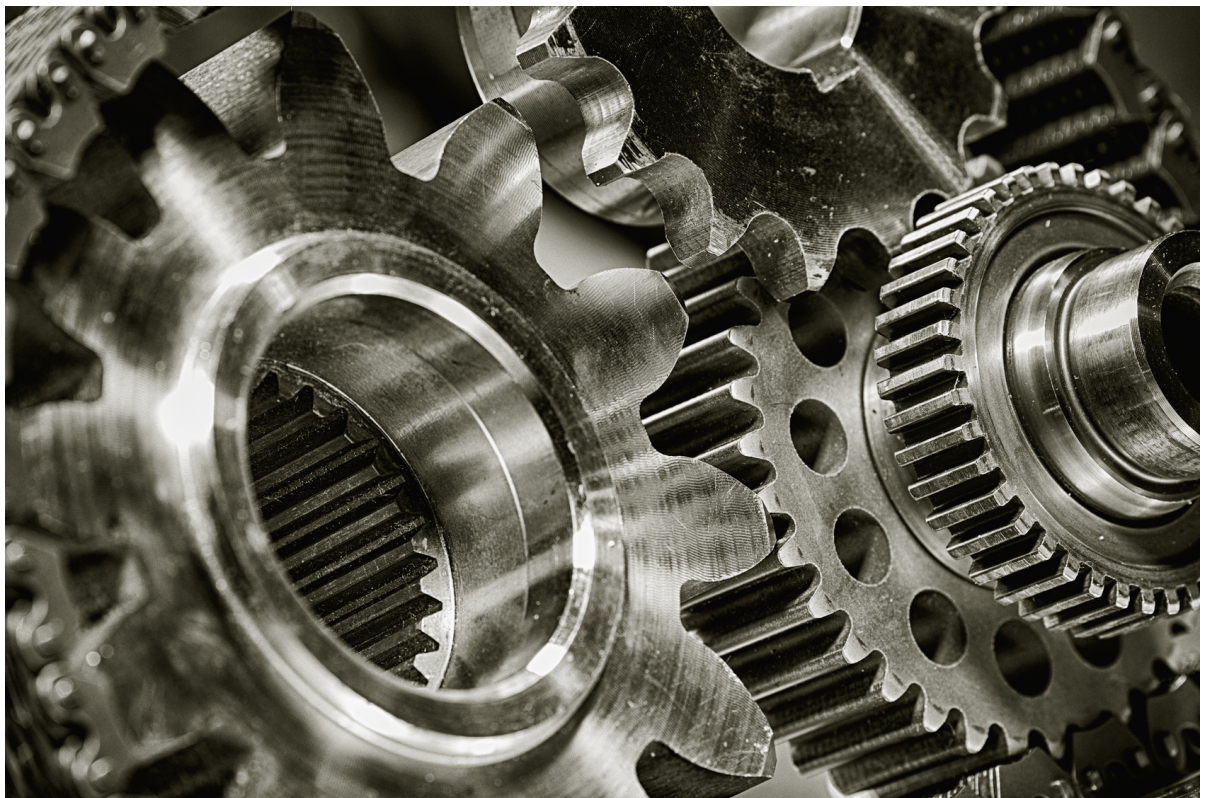
Leistungsnachweise

### Ausklang



## Softwaregetriebe

Infrastruktur



Quelle: fotolia.com



## Definition (Systemprogrammierung)

Erstellen von Softwareprogrammen, die Teile eines Betriebssystems sind beziehungsweise mit einem Betriebssystem direkt interagieren oder die Hardware (genauer: Zentraleinheit<sup>a</sup> und Peripherie<sup>b</sup>) eines Rechensystems betreiben müssen.

<sup>a</sup>central processing unit (CPU), ein-/mehrfach, ein-, mehr- oder vielkernig.

<sup>b</sup>Geräte zur Ein-/Ausgabe oder Steuerung/Regelung „externer Prozesse“.



# Systemnahe Software

## Definition (Systemprogrammierung)

Erstellen von Softwareprogrammen, die Teile eines Betriebssystems sind beziehungsweise mit einem Betriebssystem direkt interagieren oder die Hardware (genauer: Zentraleinheit<sup>a</sup> und Peripherie<sup>b</sup>) eines Rechensystems betreiben müssen.

<sup>a</sup>central processing unit (CPU), ein-/mehrfach, ein-, mehr- oder vielkernig.

<sup>b</sup>Geräte zur Ein-/Ausgabe oder Steuerung/Regelung „externer Prozesse“.

- einerseits **Anwendungssoftware** („oben“)
  - ermöglichen, unterstützen, nicht entgegenwirken
- andererseits **Plattformsysteme** („unten“)
  - anwendungsspezifisch verfügbar machen
  - problemorientiert betreiben



Quelle: arcadja.com, Franz Kott



## Definition (Systemprogrammierung)

Erstellen von Softwareprogrammen, die Teile eines Betriebssystems sind beziehungsweise mit einem Betriebssystem direkt interagieren oder die Hardware (genauer: Zentraleinheit<sup>a</sup> und Peripherie<sup>b</sup>) eines Rechensystems betreiben müssen.

<sup>a</sup>central processing unit (CPU), ein-/mehrfach, ein-, mehr- oder vielkernig.

<sup>b</sup>Geräte zur Ein-/Ausgabe oder Steuerung/Regelung „externer Prozesse“.

- einerseits **Anwendungssoftware** („oben“)
  - ermöglichen, unterstützen, nicht entgegenwirken
- andererseits **Plattformsysteme** („unten“)
  - anwendungsspezifisch verfügbar machen
  - problemorientiert betreiben
  - bedingt verbergen
    - nachteilige Eigenschaften versuchen zu kaschieren



Quelle: arcadja.com, Franz Kott



## Gliederung

### Einleitung

### Konzept

Lehrkanon

Lehrziele

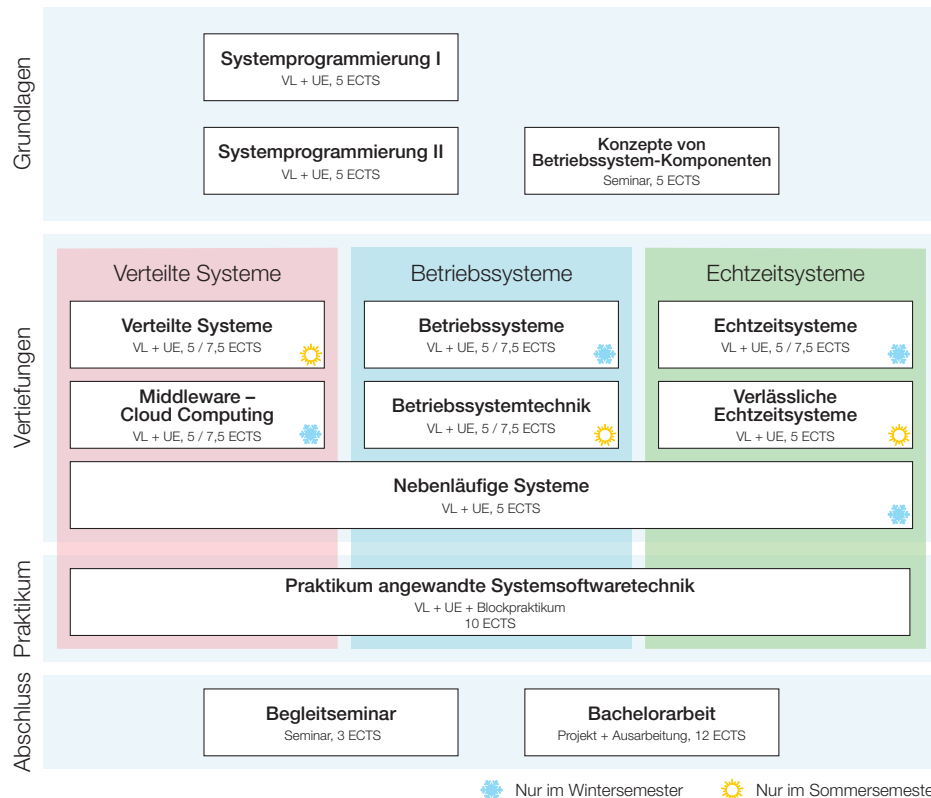
Vorkenntnisse

Veranstaltungsbetrieb

Leistungsnachweise

### Ausklang





## Module SP (10 ECTS) und GSP (5 ECTS)

### Systemprogrammierung (SP) ~ geteiltes Modul

- ↪ Systemprogrammierung I (SP1) ↦ Teile A und B 5 ECTS
- ↪ Systemprogrammierung II (SP2) ↦ Teil C 5 ECTS



## Module SP (10 ECTS) und GSP (5 ECTS)

### Systemprogrammierung (SP) ~ geteiltes Modul

|                                                |        |
|------------------------------------------------|--------|
| ↪ Systemprogrammierung I (SP1) ↦ Teile A und B | 5 ECTS |
| ↪ Systemprogrammierung II (SP2) ↦ Teil C       | 5 ECTS |

- SP1 geht in die **Breite**, liefert einen funktionalen Überblick
- SP2 geht in die **Tiefe**, behandelt ausgewählte Funktionen im Detail
- beide Hälften sind Grundlage vor allem der „Betriebssysteme“-Säule



## Module SP (10 ECTS) und GSP (5 ECTS)

### Systemprogrammierung (SP) ~ geteiltes Modul

|                                                |        |
|------------------------------------------------|--------|
| ↪ Systemprogrammierung I (SP1) ↦ Teile A und B | 5 ECTS |
| ↪ Systemprogrammierung II (SP2) ↦ Teil C       | 5 ECTS |

- SP1 geht in die **Breite**, liefert einen funktionalen Überblick
- SP2 geht in die **Tiefe**, behandelt ausgewählte Funktionen im Detail
- beide Hälften sind Grundlage vor allem der „Betriebssysteme“-Säule

### Grundlagen der Systemprogrammierung (GSP)

|                                |        |
|--------------------------------|--------|
| ↪ Systemprogrammierung I (SP1) | 5 ECTS |
|--------------------------------|--------|

- Export für Studiengänge, die Stoffausbau durch SP2 nicht erfordern



## Studiengänge und Zuordnung

| Abschluss | Studiengang                             | SP1 | SP2 |
|-----------|-----------------------------------------|-----|-----|
| Bachelor  | Informatik                              | ×   | ×   |
|           | Informations- und Kommunikationstechnik | ×   | ×   |
|           | Computational Engineering               | ×   | ×   |
|           | Wirtschaftsinformatik                   | ×   | ×   |
|           | Informatik, 2-Fach Bachelor             | ×   |     |
|           | Medizintechnik                          | GSP |     |
| Lehramt   | Informatik, Gymnasium                   | ×   | ×   |



## Studiengänge und Zuordnung

| Abschluss | Studiengang                             | SP1 | SP2 |
|-----------|-----------------------------------------|-----|-----|
| Bachelor  | Informatik                              | ×   | ×   |
|           | Informations- und Kommunikationstechnik | ×   | ×   |
|           | Computational Engineering               | ×   | ×   |
|           | Wirtschaftsinformatik                   | ×   | ×   |
|           | Informatik, 2-Fach Bachelor             | ×   |     |
|           | Medizintechnik                          | GSP |     |
| Lehramt   | Informatik, Gymnasium                   | ×   | ×   |

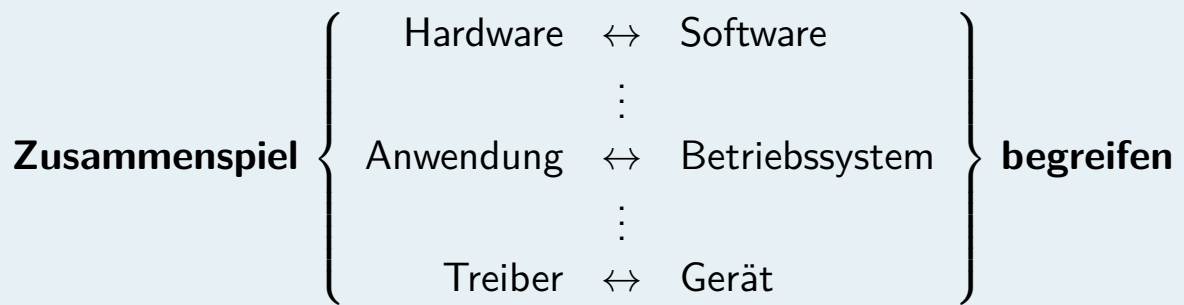
### ■ **Alternative** zu [Systemnahe Programmierung in C \(SPiC\)](#):

| Abschluss | Studiengang                      | SP1 | SP2 |
|-----------|----------------------------------|-----|-----|
| Bachelor  | Mathematik, Nebenfach Informatik | ×   |     |
|           | Technomathematik                 | ×   |     |



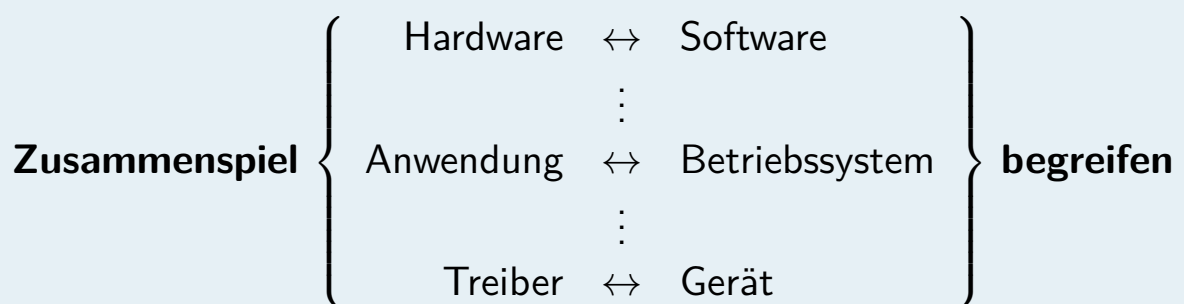
## Lernziele

- Vorgänge innerhalb von Rechensystemen **ganzheitlich** verstehen



## Lernziele

- Vorgänge innerhalb von Rechensystemen **ganzheitlich** verstehen



- imperative Systemprogrammierung (in C) in Grundzügen kennenlernen
  - im Kleinen für **Dienstprogramme** praktizieren
  - im Großen durch **Betriebssysteme** erfahren
- Beziehungen zwischen funktionalen und nicht-funktionalen Systemmerkmalen erfassen



Quelle: fotalia.com





# Lehrveranstaltungsformen

## ■ Vorlesung — Vorstellung und detaillierte Behandlung des Lehrstoffs

*Lehrveranstaltung an einer Universität, Hochschule, bei der ein Dozent, eine Dozentin über ein bestimmtes Thema im Zusammenhang vorträgt. [2]*

- Organisation (der Systemsoftware) von Rechensystemen
- Grundlagen von Betriebssystemen
- maschinennahe Programme



# Lehrveranstaltungsformen

## ■ Vorlesung — Vorstellung und detaillierte Behandlung des Lehrstoffs

*Lehrveranstaltung an einer Universität, Hochschule, bei der ein Dozent, eine Dozentin über ein bestimmtes Thema im Zusammenhang vorträgt. [2]*

- Organisation (der Systemsoftware) von Rechensystemen
- Grundlagen von Betriebssystemen
- maschinennahe Programme

## ■ Übung — Vertiefung, Aufgabenbesprechung, Tafelübungen

*Lehrveranstaltung an der Hochschule, in der etwas, besonders das Anwenden von Grundkenntnissen, von den Studierenden geübt wird. [2]*

- Systemprogrammierung in C
- Systemprogramme, -aufrufe, -funktionen von UNIX



# Lehrveranstaltungsformen

## ■ Vorlesung — Vorstellung und detaillierte Behandlung des Lehrstoffs

*Lehrveranstaltung an einer Universität, Hochschule, bei der ein Dozent, eine Dozentin über ein bestimmtes Thema im Zusammenhang vorträgt. [2]*

- Organisation (der Systemsoftware) von Rechensystemen
- Grundlagen von Betriebssystemen
- maschinennahe Programme

## ■ Übung — Vertiefung, Aufgabenbesprechung, Tafelübungen

*Lehrveranstaltung an der Hochschule, in der etwas, besonders das Anwenden von Grundkenntnissen, von den Studierenden geübt wird. [2]*

- Systemprogrammierung in C
- Systemprogramme, -aufrufe, -funktionen von UNIX

## ■ Rechnerarbeit — Programmierung, Fehlersuche/-beseitigung

- UNIX (Linux), CLI (*shell*), GNU (gcc, gdb, make), vi. . .



# Inhaltsüberblick

Kapitelzuordnung und -folge

## I. Lehrveranstaltungsüberblick



## I. Lehrveranstaltungsüberblick

### Teil A ~ C-Programmierung

- II. Einführung in C
- III. Programm  $\mapsto$  Prozess



## I. Lehrveranstaltungsüberblick

### Teil A ~ C-Programmierung

- II. Einführung in C
- III. Programm  $\mapsto$  Prozess

### Teil B ~ Grundlagen

- IV. Einleitung
- V. Rechnerorganisation
- VI. Abstraktionen (UNIX)
- VII. Betriebsarten



## I. Lehrveranstaltungsüberblick

### Teil A ~ C-Programmierung

- II. Einführung in C
- III. Programm  $\mapsto$  Prozess

### Teil B ~ Grundlagen

- IV. Einleitung
- V. Rechnerorganisation
- VI. Abstraktionen (UNIX)
- VII. Betriebsarten

### VIII. Zwischenbilanz SP1



## I. Lehrveranstaltungsüberblick

### Teil A ~ C-Programmierung

- II. Einführung in C
- III. Programm  $\mapsto$  Prozess

### Teil B ~ Grundlagen

- IV. Einleitung
- V. Rechnerorganisation
- VI. Abstraktionen (UNIX)
- VII. Betriebsarten

### VIII. Zwischenbilanz SP1

### Teil C ~ Vertiefung

- IX. Prozessverwaltung
  - Einplanung
  - Einlastung
- X. Koordinierung
  - Synchronisation
- XI. Betriebsmittelverwaltung
- XII. Speicherverwaltung
  - Adressräume
  - Arbeitsspeicher
- XIII. Dateisysteme
  - Speicherung
  - Fehlererholung



## I. Lehrveranstaltungsüberblick

### Teil A ~ C-Programmierung

- II. Einführung in C
- III. Programm  $\mapsto$  Prozess

### Teil B ~ Grundlagen

- IV. Einleitung
- V. Rechnerorganisation
- VI. Abstraktionen (UNIX)
- VII. Betriebsarten

### VIII. Zwischenbilanz SP1

### XIV. Fragestunde SP1 & SP2

## Teil C ~ Vertiefung

- IX. Prozessverwaltung
  - Einplanung
  - Einlastung
- X. Koordinierung
  - Synchronisation
- XI. Betriebsmittelverwaltung
- XII. Speicherverwaltung
  - Adressräume
  - Arbeitsspeicher
- XIII. Dateisysteme
  - Speicherung
  - Fehlererholung



## Voraussetzungen zum Verständnis des Lehrstoffs

- obligatorisch: **Grundlagen der Programmierung**  $\mapsto$  AuD
  - Datentypen, Kontrollkonstrukte, Prozeduren
  - statische und dynamische Datenstrukturen
  - „Programmierung im Kleinen“



## Voraussetzungen zum Verständnis des Lehrstoffs

- obligatorisch: **Grundlagen der Programmierung**  $\mapsto$  AuD
    - Datentypen, Kontrollkonstrukte, Prozeduren
    - statische und dynamische Datenstrukturen
    - „Programmierung im Kleinen“
- vor allem für die Übung, weniger für die Vorlesung



## Voraussetzungen zum Verständnis des Lehrstoffs

- obligatorisch: **Grundlagen der Programmierung**  $\mapsto$  AuD
    - Datentypen, Kontrollkonstrukte, Prozeduren
    - statische und dynamische Datenstrukturen
    - „Programmierung im Kleinen“
- vor allem für die Übung, weniger für die Vorlesung
- wünschenswert: **Technische Informatik**  $\mapsto$  GTI, GRA
    - „Von-Neumann-Architektur“
      - Operationsbefehle, Befehlsoperanden, Adressierungsarten
      - Unterbrechungssteuerung (Pegel kontra Flanke)
    - Assemblerprogrammierung
      - Pseudo- und Maschinenbefehle (IA32)
      - Binär-, Oktal-, Hexadezimalcode
    - CPU, DMA, FPU, IRQ, MCU, MMU, NMI, PIC, TLB



# Voraussetzungen zum Verständnis des Lehrstoffs

- obligatorisch: **Grundlagen der Programmierung**  $\mapsto$  AuD
  - Datentypen, Kontrollkonstrukte, Prozeduren
  - statische und dynamische Datenstrukturen
  - „Programmierung im Kleinen“ $\rightarrow$  vor allem für die Übung, weniger für die Vorlesung
- wünschenswert: **Technische Informatik**  $\mapsto$  GTI, GRA
  - „Von-Neumann-Architektur“
    - Operationsbefehle, Befehlsoperanden, Adressierungsarten
    - Unterbrechungssteuerung (Pegel kontra Flanke)
  - Assemblerprogrammierung
    - Pseudo- und Maschinenbefehle (IA32)
    - Binär-, Oktal-, Hexadezimalcode
  - CPU, DMA, FPU, IRQ, MCU, MMU, NMI, PIC, TLB
- altbewährte und nach wie vor aktuelle Sekundärliteratur: [4, 5, 3]



## Abhängigkeiten zwischen den Vorlesungsteilen

### Systemprogrammierung I

- Teil A ■ setzt grundlegende Programmierkenntnisse voraus
- vermittelt Grundlagen der **Programmierung in C**
- Teil B ■ setzt grundlegende Programmierkenntnisse in C voraus
- vermittelt **Operationsprinzipien** von Betriebssystemen



## Abhängigkeiten zwischen den Vorlesungsteilen

### Systemprogrammierung I

- Teil A
  - setzt grundlegende Programmierkenntnisse voraus
  - vermittelt Grundlagen der **Programmierung in C**
- Teil B
  - setzt grundlegende Programmierkenntnisse in C voraus
  - vermittelt **Operationsprinzipien** von Betriebssystemen

### Systemprogrammierung II

- Teil C
  - setzt Kenntnisse dieser Operationsprinzipien voraus
  - vermittelt **interne Funktionsweisen** von Betriebssystemen



## Abhängigkeiten zwischen den Vorlesungsteilen

### Systemprogrammierung I

- Teil A
  - setzt grundlegende Programmierkenntnisse voraus
  - vermittelt Grundlagen der **Programmierung in C**
- Teil B
  - setzt grundlegende Programmierkenntnisse in C voraus
  - vermittelt **Operationsprinzipien** von Betriebssystemen

### Systemprogrammierung II

- Teil C
  - setzt Kenntnisse dieser Operationsprinzipien voraus
  - vermittelt **interne Funktionsweisen** von Betriebssystemen

- Erlangung der benötigten Vorkenntnisse:
  - i durch Vorlesungsteilnahme
    - empfohlene sequentielle Belegung der Vorlesungsteile
  - ii durch Lehrbuchlektüre, aus anderen Lehrveranstaltungen, ...





## Unterrichtstermine

- Vorlesungs-, Übungs- und Rechnerzeiten:
  - auf [www4.cs.fau.de](http://www4.cs.fau.de) dem Reiter „Lehre“ folgen
  - Sondertermine am Semesteranfang für den *Crash*-Kurs über C



## Unterrichtstermine und -sprache

- Vorlesungs-, Übungs- und Rechnerzeiten:
  - auf [www4.cs.fau.de](http://www4.cs.fau.de) dem Reiter „Lehre“ folgen
  - Sondertermine am Semesteranfang für den *Crash*-Kurs über C

- Unterrichtssprache:



- Vorlesung und Übung



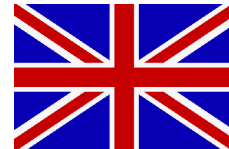
## Unterrichtstermine und -sprache

- Vorlesungs-, Übungs- und Rechnerzeiten:
  - auf [www4.cs.fau.de](http://www4.cs.fau.de) dem Reiter „Lehre“ folgen
  - Sondertermine am Semesteranfang für den *Crash*-Kurs über C

- Unterrichtssprache:



- Vorlesung und Übung



- Fachbegriffe



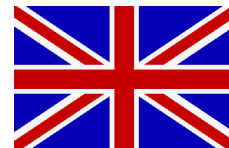
## Unterrichtstermine und -sprache

- Vorlesungs-, Übungs- und Rechnerzeiten:
  - auf [www4.cs.fau.de](http://www4.cs.fau.de) dem Reiter „Lehre“ folgen
  - Sondertermine am Semesteranfang für den *Crash*-Kurs über C

- Unterrichtssprache:



- Vorlesung und Übung



- Fachbegriffe

- Sachwortverzeichnis (in Arbeit und Überarbeitung)

- [www4.cs.fau.de/~woch/glossar.pdf](http://www4.cs.fau.de/~woch/glossar.pdf)



- Aneignung von neuem Wissen
- mit bisherigem/anderem Wissen in Beziehung bringen:



- Aneignung von neuem Wissen
  - selbständig die jeweils nächste Vorlesung vorbereiten
  - an der Präsentation teilnehmen, ihr zuhören, Fragen stellen
  - behandelte Themen untereinander diskutieren, Lehrstoff nachbereiten



- mit bisherigem/anderem Wissen in Beziehung bringen:

**AuD** ■ Grundlagen der Programmierung in einer **Hochsprache**

**PFP** ■ Grundlagen der parallelen Programmierung

**GRA** ■ Rechnerorganisation oder -architektur



- mit bisherigem/anderem Wissen in Beziehung bringen:

**AuD** ■ Grundlagen der Programmierung in einer **Hochsprache**

**PFP** ■ Grundlagen der parallelen Programmierung

**GRA** ■ Rechnerorganisation oder -architektur

- Grundlagen der Programmierung in **Assemblersprache**



- Aneignung von neuem Wissen
  - selbständig die jeweils nächste Vorlesung vorbereiten
  - an der Präsentation teilnehmen, ihr zuhören, Fragen stellen
  - behandelte Themen untereinander diskutieren, Lehrstoff nachbereiten
- mit bisherigem/anderem Wissen in Beziehung bringen:
  - AuD ■ Grundlagen der Programmierung in einer **Hochsprache**
  - PFP ■ Grundlagen der parallelen Programmierung
  - GRA ■ Rechnerorganisation oder -architektur
    - Grundlagen der Programmierung in **Assemblersprache**
- im Hörsaal präsentiertes Lehrmaterial: **Vorlesungsfolien**
  - stehen animiert und in Handzettelform elektronisch zur Verfügung
    - PDF: auf [www4.cs.fau.de](http://www4.cs.fau.de) dem Reiter „Lehre“ folgen



- Aneignung von neuem Wissen
  - selbständig die jeweils nächste Vorlesung vorbereiten
  - an der Präsentation teilnehmen, ihr zuhören, Fragen stellen
  - behandelte Themen untereinander diskutieren, Lehrstoff nachbereiten
- mit bisherigem/anderem Wissen in Beziehung bringen:
  - AuD ■ Grundlagen der Programmierung in einer **Hochsprache**
  - PFP ■ Grundlagen der parallelen Programmierung
  - GRA ■ Rechnerorganisation oder -architektur
    - Grundlagen der Programmierung in **Assemblersprache**
- im Hörsaal präsentiertes Lehrmaterial: **Vorlesungsfolien**
  - stehen animiert und in Handzettelform elektronisch zur Verfügung
    - PDF: auf [www4.cs.fau.de](http://www4.cs.fau.de) dem Reiter „Lehre“ folgen
  - Anzahl und „Füllungsdichte“ sind bewusst eher hoch gehalten:
    - i obligatorischer und optionaler (Anhang) Vorlesungsstoff
    - ii schriftlich fixierte Gedankenstränge als Hilfe zur Nachbearbeitung



- Aneignung von neuem Wissen
  - selbständig die jeweils nächste Vorlesung vorbereiten
  - an der Präsentation teilnehmen, ihr zuhören, Fragen stellen
  - behandelte Themen untereinander diskutieren, Lehrstoff nachbereiten
- mit bisherigem/anderem Wissen in Beziehung bringen:
  - AuD ■ Grundlagen der Programmierung in einer **Hochsprache**
  - PFP ■ Grundlagen der parallelen Programmierung
  - GRA ■ Rechnerorganisation oder -architektur
    - Grundlagen der Programmierung in **Assemblersprache**
- im Hörsaal präsentiertes Lehrmaterial: **Vorlesungsfolien**
  - stehen animiert und in Handzettelform elektronisch zur Verfügung
    - PDF: auf [www4.cs.fau.de](http://www4.cs.fau.de) dem Reiter „Lehre“ folgen
  - Anzahl und „Füllungsdichte“ sind bewusst eher hoch gehalten:
    - i obligatorischer und optionaler (Anhang) Vorlesungsstoff
    - ii schriftlich fixierte Gedankenstränge als Hilfe zur Nachbearbeitung
  - Anhänge und **ergänzende Materialien** sind keine Klausuraufgaben



# Übung

- Wissen durch **direkte Erfahrung** vertiefen

*Tugendhaftes Verhalten und fachliches Können wird weniger durch einfache Belehrung als durch praktisches Nachmachen, Üben, Anwenden erlernt. (Aristoteles [1])*



- Wissen durch **direkte Erfahrung** vertiefen

*Tugendhaftes Verhalten und fachliches Können wird weniger durch einfache Belehrung als durch praktisches Nachmachen, Üben, Anwenden erlernt. (Aristoteles [1])*

- Diskussion der Übungsaufgaben, Lösungsansätze ausarbeiten
- Vorlesungsstoff festigen, offene Fragen klären



- Wissen durch **direkte Erfahrung** vertiefen

*Tugendhaftes Verhalten und fachliches Können wird weniger durch einfache Belehrung als durch praktisches Nachmachen, Üben, Anwenden erlernt. (Aristoteles [1])*

- Diskussion der Übungsaufgaben, Lösungsansätze ausarbeiten
- Vorlesungsstoff festigen, offene Fragen klären

- **Tafelübung** unter Anleitung einer/s Übungsleiterin/s

- Anmeldung durch [WAFFEL](#)<sup>1</sup> (URL siehe Webseite von SP)
- Übungsaufgaben sind in Gruppen zu bearbeiten: Kannvorschrift
  - ist abhängig von der Teilnehmeranzahl



<sup>1</sup>Abk. Webanmeldefrickelformular Enterprise Logic

- Wissen durch **direkte Erfahrung** vertiefen

*Tugendhaftes Verhalten und fachliches Können wird weniger durch einfache Belehrung als durch praktisches Nachmachen, Üben, Anwenden erlernt. (Aristoteles [1])*

- Diskussion der Übungsaufgaben, Lösungsansätze ausarbeiten
- Vorlesungsstoff festigen, offene Fragen klären

- **Tafelübung** unter Anleitung einer/s Übungsleiterin/s

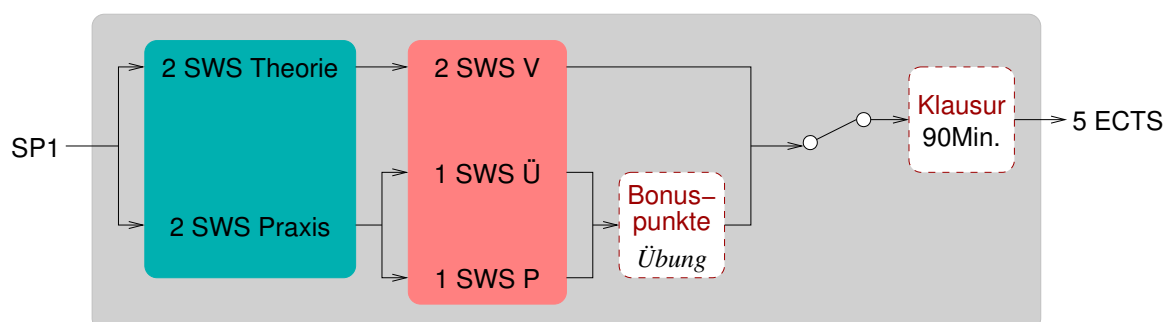
- Anmeldung durch [WAFFEL](#)<sup>1</sup> (URL siehe Webseite von SP)
- Übungsaufgaben sind in Gruppen zu bearbeiten: Kannvorschrift
  - ist abhängig von der Teilnehmeranzahl

- **Rechnerarbeit** in Eigenverantwortung

- ohne Anmeldung, reservierte Arbeitsplätze stehen zur Verfügung
- bei Fragen sich an die Übungsleiter/innen von SP wenden

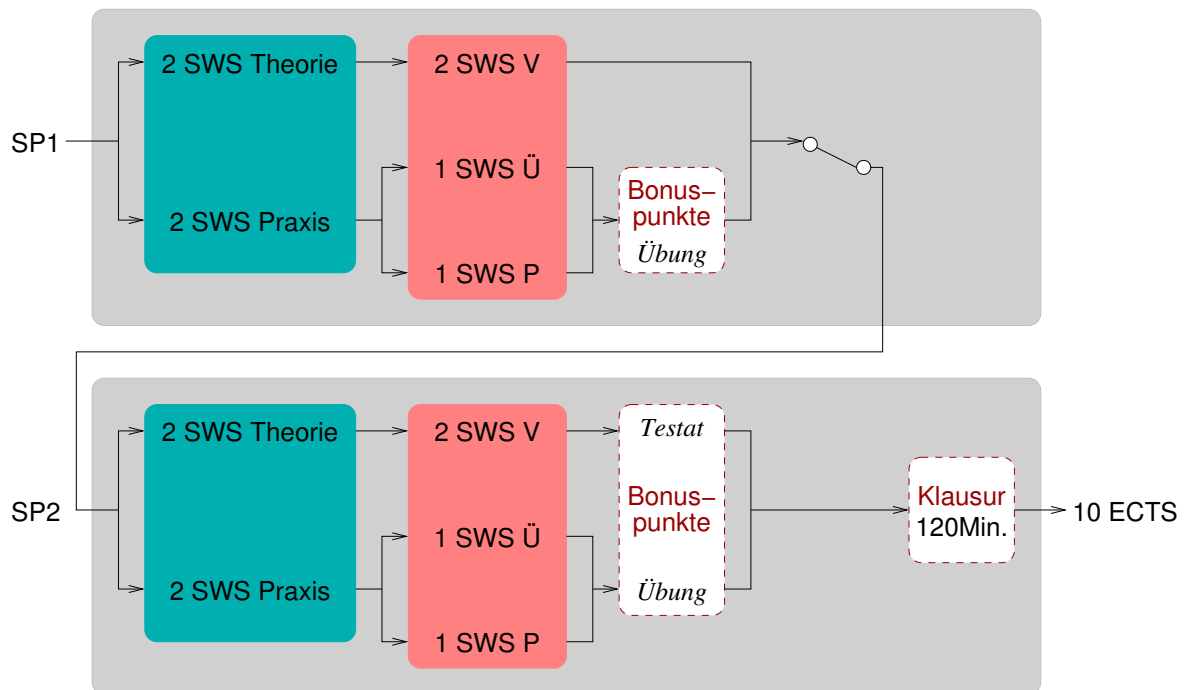
<sup>1</sup>Abk. *Webanmeldefrickelformular Enterprise Logic*

## Studien- und Prüfungsleistungen

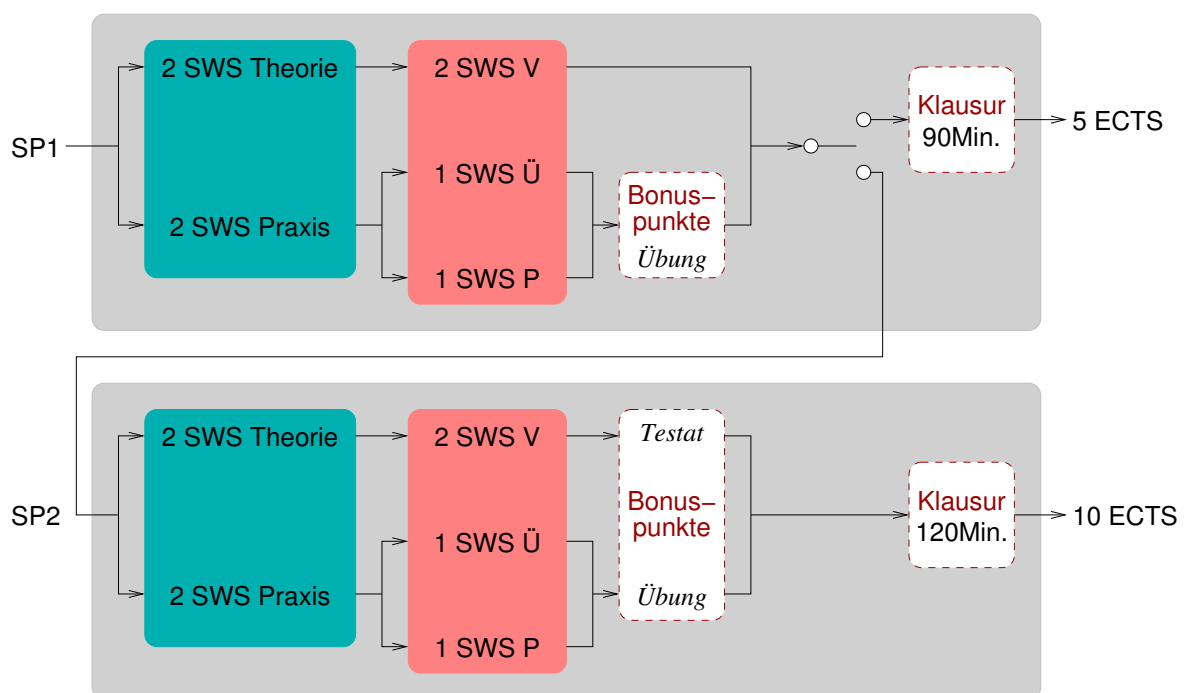




## Studien- und Prüfungsleistungen



## Studien- und Prüfungsleistungen



- **Übungsaufgaben:** 6 (SP1) + 5 (SP2) Programmieraufgaben
  - abgegebene Programme werden korrigiert und mit Punkten bewertet
  - unzureichende Erklärung der vorgestellten Lösung ergibt 0 Punkte
    - Nichtanwesenheit impliziert, die Lösung nur unzureichend erklären zu können



- **Übungsaufgaben:** 6 (SP1) + 5 (SP2) Programmieraufgaben
  - abgegebene Programme werden korrigiert und mit Punkten bewertet
  - unzureichende Erklärung der vorgestellten Lösung ergibt 0 Punkte
    - Nichtanwesenheit impliziert, die Lösung nur unzureichend erklären zu können
- ein **Antestat**<sup>2</sup> (auch: „Miniklausur“) zum Aufwärmen für SP2
  - geprüft wird Stoff von Vorlesung und Übung, 30 Minuten
    - Fragen zu **Teil A** und **Teil B** der Vorlesung
    - Trockenaufgabe als **Lückentest** in der Programmiersprache C
  - mit Aufgabenanteilen als **Mehrfachauswahl** (*multiple choice*)

<sup>2</sup>Allgemein eine mündliche oder schriftliche Prüfung in naturwissenschaftlichen Studienfächern am Anfang eines Semesters. Schriftlich ausgeführt im Fall von SP.



- **Übungsaufgaben:** 6 (SP1) + 5 (SP2) Programmieraufgaben
  - abgegebene Programme werden korrigiert und mit Punkten bewertet
  - unzureichende Erklärung der vorgestellten Lösung ergibt 0 Punkte
    - Nichtanwesenheit impliziert, die Lösung nur unzureichend erklären zu können
- ein **Antestat**<sup>2</sup> (auch: „Miniklausur“) zum Aufwärmen für SP2
  - geprüft wird Stoff von Vorlesung und Übung, 30 Minuten
    - Fragen zu **Teil A** und **Teil B** der Vorlesung
    - Trockenaufgabe als **Lückentest** in der Programmiersprache C
  - mit Aufgabenanteilen als **Mehrfachauswahl** (*multiple choice*)

### Notenbonus für die Klausur (auch: „Maxiklausur“)

- bei 50 % der Punkte aus „Übungsaufgaben + Testat“
- Punkte darüberhinaus gehen in die Bonusberechnung ein
- maximal ist ein Notenbonus von 0,7 erreichbar

<sup>2</sup>Allgemein eine mündliche oder schriftliche Prüfung in naturwissenschaftlichen Studienfächern am Anfang eines Semesters. Schriftlich ausgeführt im Fall von SP.



## Kür und Pflicht

- **Notenbonus** nur auf Basis der Übungen **des laufenden SP-Moduls**
    - beeinflusst die Punkte-Notenskala der Klausur nicht, er wird allerdings bei bestandener Klausur auf die Klausurnote angewendet (abgezogen)
    - kann die Note einer bestandenen Klausur verbessern, nicht jedoch den Ausschlag zum Bestehen der Klausur geben
- Erreichen der Bestehensgrenze muss also immer mit regulär erworbenen Klausurpunkten erfolgen



## Kür und Pflicht

- **Notenbonus** nur auf Basis der Übungen **des laufenden SP-Moduls**
  - beeinflusst die Punkte-Notenskala der Klausur nicht, er wird allerdings bei bestandener Klausur auf die Klausurnote angewendet (abgezogen)
  - kann die Note einer bestandenen Klausur verbessern, nicht jedoch den Ausschlag zum Bestehen der Klausur geben
- Erreichen der Bestehensgrenze muss also immer mit regulär erworbenen Klausurpunkten erfolgen
- **Klausur:** Termin noch offen, Anfang vorlesungsfreie Zeit
  - GSP ■ Struktur analog Testat (S. 19), jedoch 90 Minuten Dauer
  - SP ■ Struktur analog GSP, jedoch 120 Minuten Dauer
    - zusätzlich Fragen zu **Teil C** der Vorlesung



## Kür und Pflicht

- **Notenbonus** nur auf Basis der Übungen **des laufenden SP-Moduls**
  - beeinflusst die Punkte-Notenskala der Klausur nicht, er wird allerdings bei bestandener Klausur auf die Klausurnote angewendet (abgezogen)
  - kann die Note einer bestandenen Klausur verbessern, nicht jedoch den Ausschlag zum Bestehen der Klausur geben
- Erreichen der Bestehensgrenze muss also immer mit regulär erworbenen Klausurpunkten erfolgen
- **Klausur:** Termin noch offen, Anfang vorlesungsfreie Zeit
  - GSP ■ Struktur analog Testat (S. 19), jedoch 90 Minuten Dauer
  - SP ■ Struktur analog GSP, jedoch 120 Minuten Dauer
    - zusätzlich Fragen zu **Teil C** der Vorlesung

Präsenz und **aktive Mitarbeit** machen die Klausur „leicht“

- Besuch der Vorlesung, zuhören und Fragen stellen
- Teilnahme an den Tafelübungen, Übungsaufgaben bearbeiten
- Programme im Team entwickeln, aber selbst zum Laufen bringen



## Einleitung

## Konzept

Lehrkanon  
Lehrziele  
Vorkenntnisse  
Veranstaltungsbetrieb  
Leistungsnachweise

## Ausklang



[www4.cs.fau.de/\\*](http://www4.cs.fau.de/*)



[www.augsburger-puppenkiste.de](http://www.augsburger-puppenkiste.de)

## Dozenten

- Jürgen Kleinöder (~jklein)
- Wolfgang Schröder-Preikschat (~wosch)

## Mitarbeiter

- Christian Eichler (~eichler)
- Dustin Nguyen (~nguyen)
- Simon Ruderich (~ruderich)

## Tutoren

- |                   |                    |                     |
|-------------------|--------------------|---------------------|
| ■ Ruben Baecker   | ■ Kai Friedrich    | ■ Tobias Schleier   |
| ■ Kilian Bender   | ■ Helene Gsänger   | ■ Jonas Schreiner   |
| ■ Paul Bergmann   | ■ Florian Güthlein | ■ Stefan Schöninger |
| ■ Fabian Bläse    | ■ Felix Hanika     | ■ Milan Stephan     |
| ■ Julian Brandner | ■ Lou Knauer       | ■ Johannes Weidner  |
| ■ Lukas Braun     | ■ Jenny Ottmann    | ■ Felix Windsheimer |
| ■ Eva Dengler     | ■ Timur Perst      | ■ Johannes Zahn     |
| ■ Anna Feiler     | ■ Julian Preis     |                     |





**DON'T PANIC**



Quelle: qmediasolutions.com

## Literaturverzeichnis

- [1] ARISTOTELES:  
*Nikomachische Ethik.*  
c. 334 BC
- [2] BIBLIOGRAPHISCHES INSTITUT GMBH:  
*Duden online.*  
<http://www.duden.de>, 2013
- [3] TANENBAUM, A. S.:  
*Structured Computer Organization.*  
Prentice-Hall, Inc., 1979. –  
443 S. –  
ISBN 0-130-95990-1
- [4] WIRTH, N. :  
*Systematisches Programmieren.*  
Teubner-Studienbücher, 1972. –  
160 S. –  
ISBN 3-519-02375-X
- [5] WIRTH, N. :  
*Algorithmen und Datenstrukturen.*  
Teubner-Studienbücher, 1975. –  
376 S. –  
ISBN 3-519-02330-X

