Systemprogrammierung

Grundlage von Betriebssystemen

Teil A – I. Organisation

Jürgen Kleinöder
Wolfgang Schröder-Preikschat

12. April 2016

Agenda

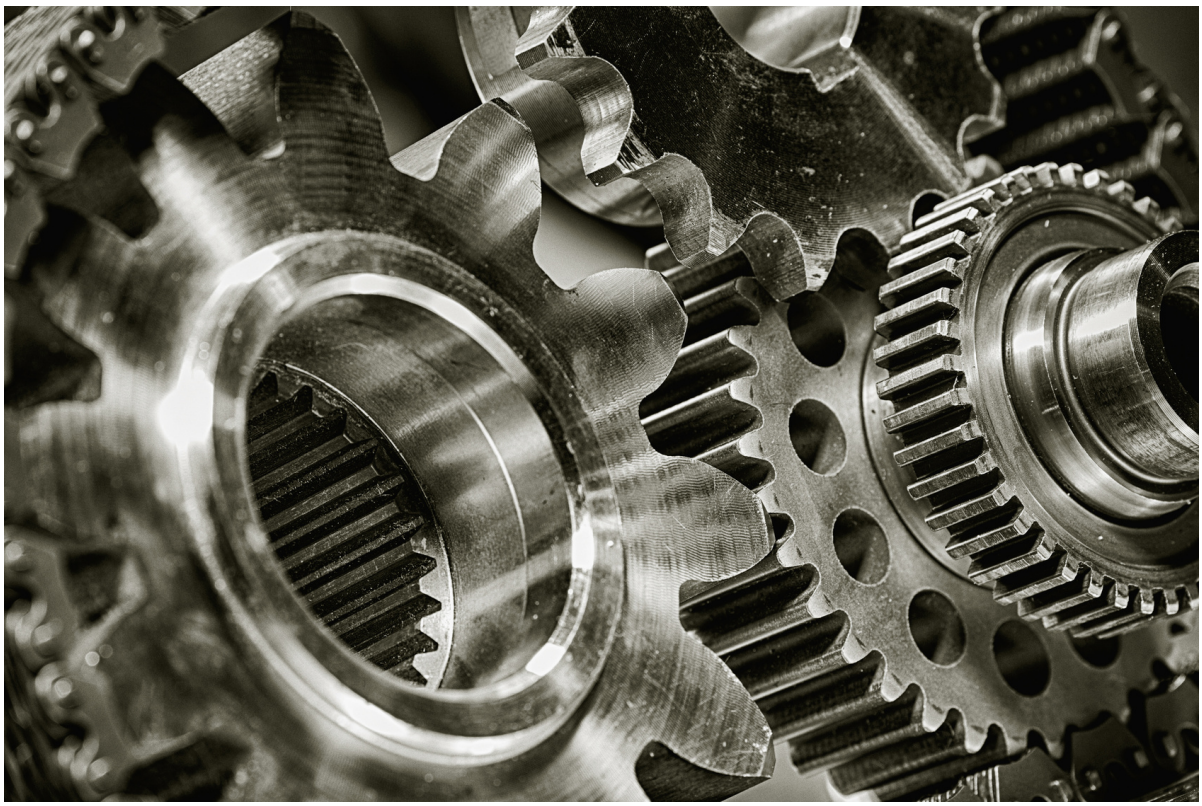
Einleitung

Konzept
- Lehrkanon
- Lehrziele
- Vorkenntnisse
- Veranstaltungsbetrieb
- Leistungsnachweise

Ausklang

Gliederung

Softwaregetriebe



Quelle: `fotalia.com`

Definition (Systemprogrammierung)

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

[a] *central processing unit* (CPU), ein-/mehrfach, ein-, mehr- oder vielkernig.
[b] Geräte zur Ein-/Ausgabe oder Steuerung/Regelung „externer Prozesse“.

Systemnahe Software Bindeglied

Definition (Systemprogrammierung)

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

[a] *central processing unit* (CPU), ein-/mehrfach, ein-, mehr- oder vielkernig.
[b] 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)

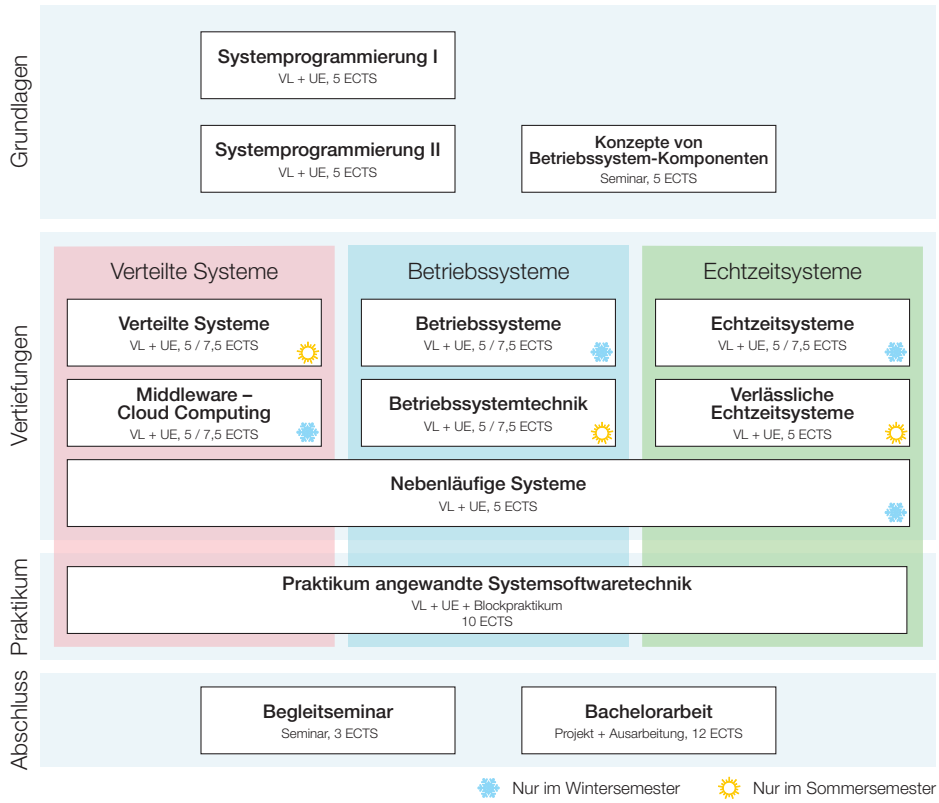
Erstellen von Softwareprogrammen, die Teile eines Betriebssystems sind beziehungsweise mit einem Betriebssystem direkt interagieren oder die Hardware (genauer: Zentraleinheit[a] und Peripherie[b]) eines Rechensystems betreiben müssen.

[a] *central processing unit* (CPU), ein-/mehrfach, ein-, mehr- oder vielkernig.
[b] 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



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

- der Stoff von SP2 ist „kausal abhängig“ vom Stoff von SP1
  - SP1 liefert Grundlagen für SP2, das wiederum den Stoff von SP1 vertieft
- 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

- der Stoff von SP2 ist „kausal abhängig“ vom Stoff von SP1
  - SP1 liefert Grundlagen für SP2, das wiederum den Stoff von SP1 vertieft
- beide Hälften sind Grundlage vor allem der „Betriebssysteme“-Säule

Grundlagen der Systemprogrammierung (GSP)

↪ Systemprogrammierung I (SP1) — 5 ECTS

- SP1 geht inhaltlich in die Breite, liefert einen funktionalen Überblick

Studiengänge und Zuordnung

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

Studiengänge und Zuordnung

Abschluss	Studiengang	SP1	SP2
Bachelor	Informatik	×	×
	Informations- und Kommunikationstechnik	×	×
	Computational Engineering	×	×
	Wirtschaftsinformatik	×	×
	Informatik, 2-Fach Bachelor	×	
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

$$\textbf{Zusammenspiel} \left\{ \begin{array}{rcl} \text{Hardware} & \leftrightarrow & \text{Software} \\ & \vdots & \\ \text{Anwendung} & \leftrightarrow & \text{Betriebssystem} \\ & \vdots & \\ \text{Treiber} & \leftrightarrow & \text{Gerät} \end{array} \right\} \textbf{begreifen}$$

Lernziele

- Vorgänge innerhalb von Rechensystemen **ganzheitlich** verstehen

$$\textbf{Zusammenspiel} \left\{ \begin{array}{rcl} \text{Hardware} & \leftrightarrow & \text{Software} \\ & \vdots & \\ \text{Anwendung} & \leftrightarrow & \text{Betriebssystem} \\ & \vdots & \\ \text{Treiber} & \leftrightarrow & \text{Gerät} \end{array} \right\} \textbf{begreifen}$$

- 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

Inhaltsüberblick

Inhaltsüberblick

Kapitelzuordnung und -folge

I. Lehrveranstaltungsüberblick

Teil A ∼ C-Programmierung

II. Einführung in C

III. Programm ↦ Prozess

Inhaltsüberblick

Inhaltsüberblick

Kapitelzuordnung und -folge

I. Lehrveranstaltungsüberblick

Teil A ∼ C-Programmierung
- II. Einführung in C
- III. Programm ↦ 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

Inhaltsüberblick

Voraussetzungen zum Verständis des Lehrstoffs

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

Voraussetzungen zum Verständis des Lehrstoffs

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

Voraussetzungen zum Verständis des Lehrstoffs

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

Abhängigkeiten zwischen den Vorlesungsteilen

Systemprogammierung 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

Systemprogammierung 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

Systemprogammierung 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 und -sprache

- Vorlesungs-, Übungs- und Rechnerzeiten:
  - auf `https://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 `https://www4.cs.fau.de` dem Reiter „Lehre“ folgen
  - Sondertermine am Semesteranfang für den *Crash*-Kurs über C
- Unterrichtssprache:

  - Vorlesung und Übung
  - Fachbegriffe, soweit sinnvoll
    - `www.babylonia.ork.uk`
    - `www.inf.fu-berlin.de/inst/ag-ss/montagswort`
    - `www.aktionlebendigesdeutsch.de`

Unterrichtstermine und -sprache

- Vorlesungs-, Übungs- und Rechnerzeiten:
  - auf `https://www4.cs.fau.de` dem Reiter „Lehre“ folgen
  - Sondertermine am Semesteranfang für den *Crash*-Kurs über C
- Unterrichtssprache:

  - Vorlesung und Übung
  - Fachbegriffe

  - Fachbegriffe, soweit sinnvoll
    - `www.babylonia.ork.uk`
    - `www.inf.fu-berlin.de/inst/ag-ss/montagswort`
    - `www.aktionlebendigesdeutsch.de`

Vorlesung

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

Vorlesung

Kumulatives Lernen

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

© wosch SP (SS 2016, A – I) 2.4 Konzept – Veranstaltungsbetrieb I/16

Vorlesung

- mit bisherigem/anderem Wissen in Beziehung bringen:
  - AuD: Grundlagen der Programmierung in einer **Hochsprache**
  - PFP: Grundlagen der parallelen Programmierung
  - GRA: Rechnerorganisation oder -architektur

Vorlesung

Kumulatives Lernen

- 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 **Assembliersprache**

Vorlesung

- 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 **Assembliersprache**
- im Hörsaal präsentiertes Lehrmaterial: **Vorlesungsfolien**
  - gedruckte Kopien stehen als Handzettel kostenlos zur Verfügung
    - PDF: auf `https://www4.cs.fau.de` dem Reiter „Lehre“ folgen

Vorlesung

Kumulatives Lernen

- 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 **Assembliersprache**
- im Hörsaal präsentiertes Lehrmaterial: **Vorlesungsfolien**
  - gedruckte Kopien stehen als Handzettel kostenlos zur Verfügung
    - PDF: auf `https://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

Vorlesung

- 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 **Assembliersprache**

- im Hörsaal präsentiertes Lehrmaterial: **Vorlesungsfolien**
  - gedruckte Kopien stehen als Handzettel kostenlos zur Verfügung
    - PDF: auf `https://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])*

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])*

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

Übung

Lernen durch Handeln

- 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[1] (URL siehe Webseite von SP)
  - Übungsaufgaben sind in Gruppen zu bearbeiten: Kannvorschrift
    - ist abhängig von der Teilnehmeranzahl

[1] Abk. *Webanmeldefrickelformular Enterprise Logic*

Übung

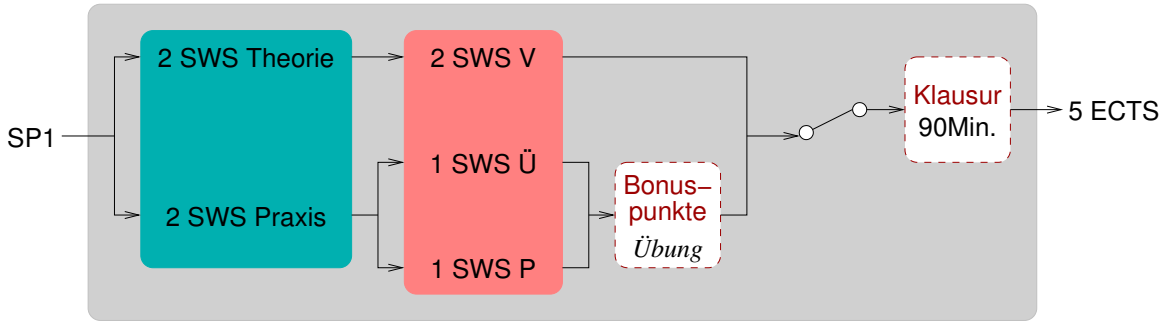
- Wissen durch **direkte Erfahrung** vertiefen

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

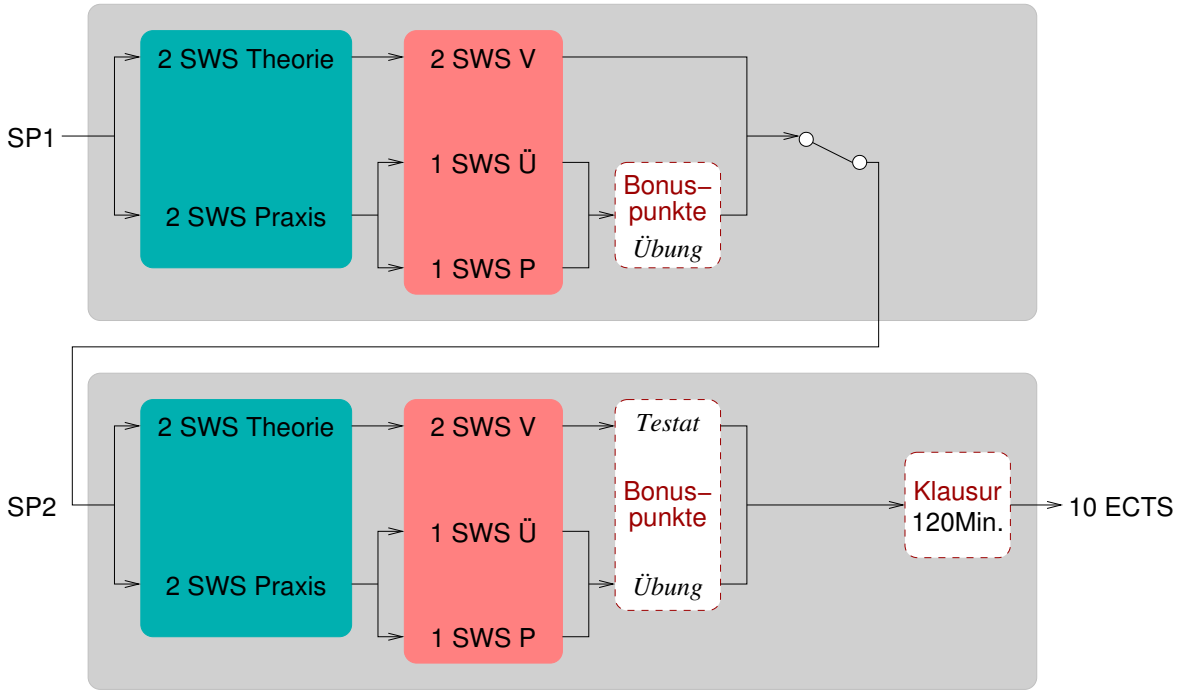
  - Diskussion der Übungsaufgaben, Lösungsansätze ausarbeiten
  - Vorlesungsstoff festigen, offene Fragen klären
- **Tafelübung** unter Anleitung einer/s Übungsleiterin/s
  - Anmeldung durch WAFFEL[1] (URL siehe Webseite von SP)
  - Übungsaufgaben sind in Gruppen zu bearbeiten: Kannvorschrift
    - ist abhängig von der Teilnehmeranzahl
- **Rechnerarbeit** in Eigenverantwortung
  - ohne Anmeldung, reservierte Arbeitsplätze stehen zur Verfügung
  - bei Fragen sich an die Übungsleiter/innen von SP wenden

[1] Abk. *Webanmeldefrickelformular Enterprise Logic*

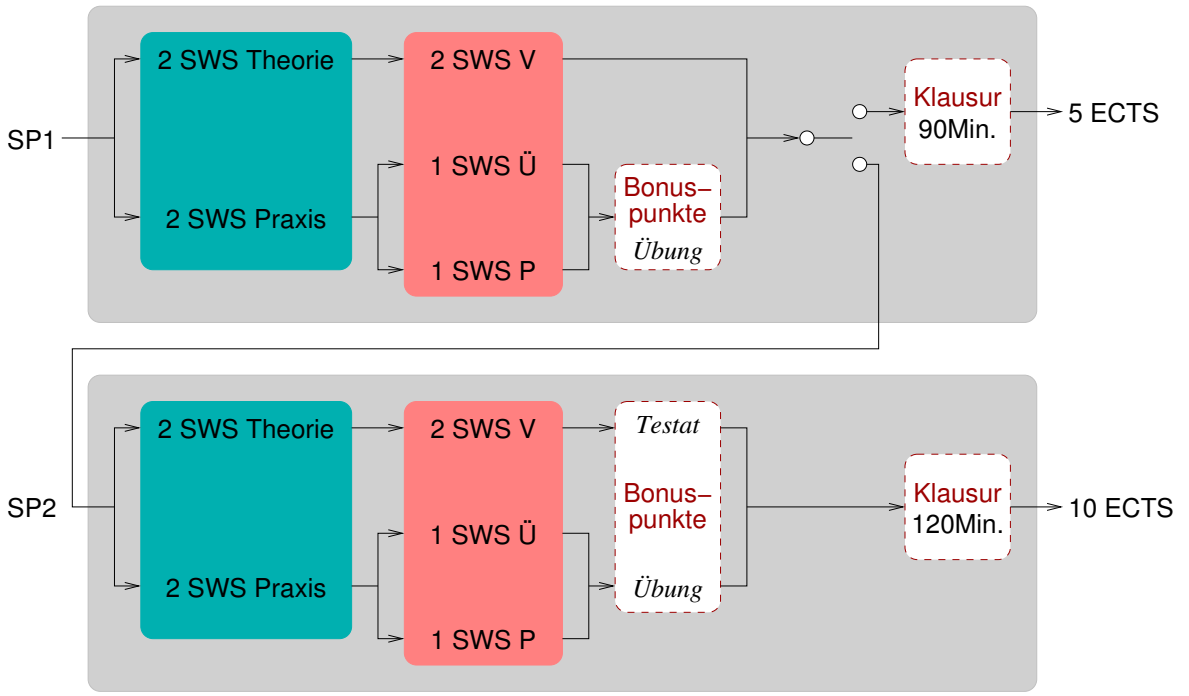
Studien- und Prüfungsleistungen



Studien- und Prüfungsleistungen



Studien- und Prüfungsleistungen



Kür

- **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

Kür

- **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**[2] (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 (engl. *multiple choice*)

[2] Allgemein eine mündliche oder schriftliche Prüfung in naturwissenschaftlichen Studienfächern am Anfang eines Semesters. Schriftlich ausgeführt im Fall von SP.

Kür

- **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**[2] (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 (engl. *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

[2] 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

Gliederung

www4.cs.fau.de/*

www.augsburger-puppenkiste.de

Dozenten

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

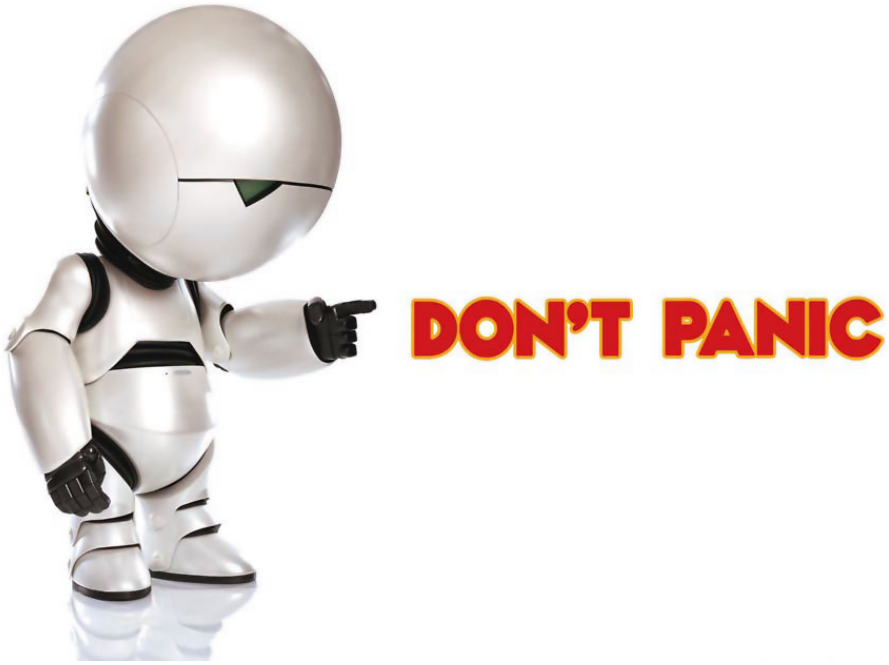
Mitarbeiter

- Stefan Reif (~reif)
- Andreas Ziegler (~ziegler)

Tutoren

- Hans-Peter Deifel
- Maximilian Eschenbacher
- Anna Feiler
- Stephan Gabert
- Helene Gsänger
- Máté Horváth
- Maximilian Krüger
- Daniel Laffling
- Lukas Lehnert
- Marvin Lunz
- Nicolas Pfeiffer
- Thomas Preisner
- Christian Schlumberger
- Milan Stephan
- Christian Strate
- Daniel Ziegler

Fragen. . .



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