

Systemprogrammierung

Grundlagen von Betriebssystemen

Teil A – I. Organisation

24. April 2025

Jürgen Kleinöder
Rüdiger Kapitza

(© Wolfgang Schröder-Preikschat, Rüdiger Kapitza)



Lehrstuhl für Informatik 4
Systemsoftware



Friedrich-Alexander-Universität
Technische Fakultät



Jürgen Kleinöder



Rüdiger Kapitza

Einleitung

Konzept

Lehrkanon

Lehrziele

Vorkenntnisse

Arrangement

Veranstaltungsbetrieb

Leistungsnachweise

Ausklang

Einleitung

Konzept

Lehrkanon

Lehrziele

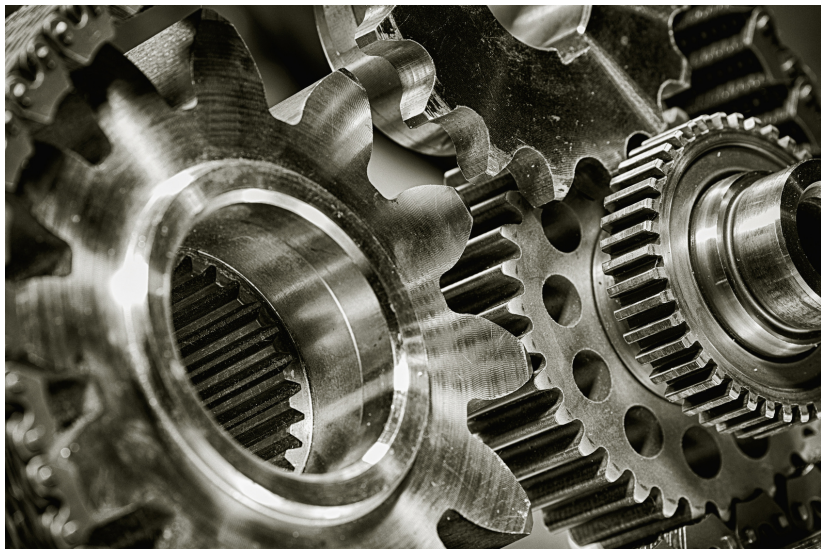
Vorkenntnisse

Arrangement

Veranstaltungsbetrieb

Leistungsnachweise

Ausklang



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.

^acentral processing unit (CPU), ein-/mehrfach, ein-, mehr- oder vielkernig.

^bGeräte zur Ein-/Ausgabe oder Steuerung/Regelung „externer Prozesse“.

Auch schon mal zwischen zwei Stühlen sitzend:

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



Quelle: arcadja.com, Franz Kott



Quelle: <https://www.bell-labs.com/usr/dmr/www>

Einleitung

Konzept

Lehrkanon

Lehrziele

Vorkenntnisse

Arrangement

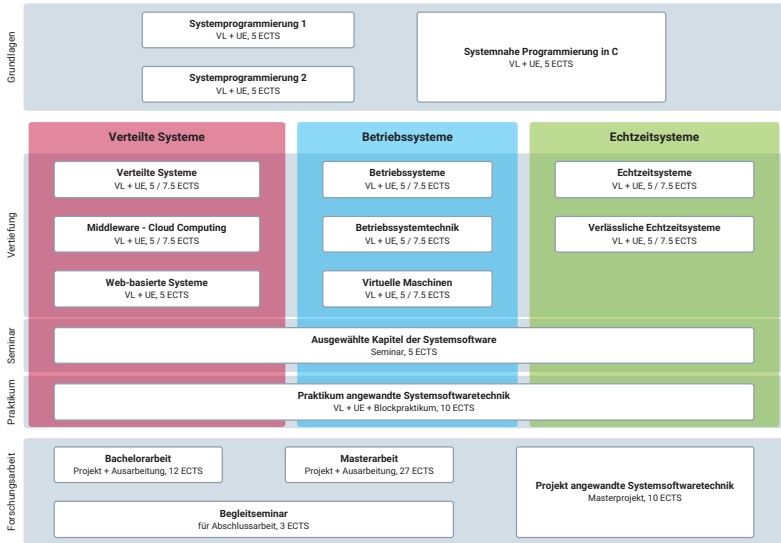
Veranstaltungsbetrieb

Leistungsnachweise

Ausklang

Konzept

Lehrkanon



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 spezifische Studiengänge

Studiengänge und Zuordnung

Abschluss	Studiengang	SP1	SP2
Bachelor	Informatik	×	×
	Informations- & Kommunikationstech.	×	×
	Computational Engineering	×	×
	Wirtschaftsinformatik	×	×
	Informatik, 2-Fach Bachelor	×	
	Medizintechnik	GSP	
Lehramt	Informatik, Gymnasium	×	×

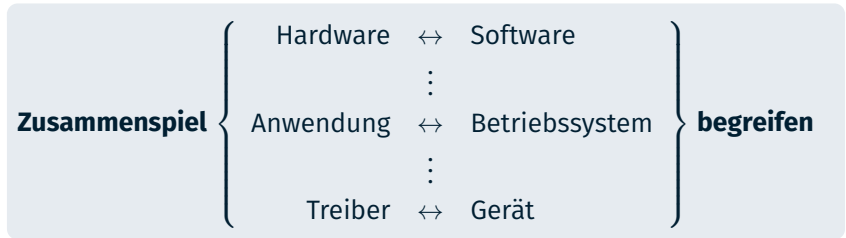
■ **Alternative** zu Systemnahe Programmierung in C (SPiC):

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

Konzept

Lehrziele

- Vorgänge in Rechensystemen **ganzheitlich** verstehen



- 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: fotolia.com

- **Vorlesung** — Detaillierte Behandlung des Lehrstoffs
 - Organisation (der Systemsoftware) von Rechensystemen
 - Grundlagen von Betriebssystemen
 - maschinennahe Programme
- **Übung** — Vertiefung, Aufgabenbesprechung, Tafelübungen
 - Systemprogrammierung in C
 - Systemprogramme, -aufrufe, -funktionen von UNIX
- **Rechnerarbeit** — Programmierung, Fehlersuche/-beseitigung
 - Linux/UNIX, CLI (*shell*), GNU (*gcc*, *gdb*, *make*), vi...

I. Überblick

Teil A ~ C-Programmierung

II. Einführung in C

III. Programm $\mapsto$ Prozess

Teil B ~ Grundlagen

IV. Einleitung

V. Abstraktionen (UNIX)

VI. Rechnerorganisation

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

Konzept

Vorkenntnisse

Voraussetzungen zum Verständnis des Lehrstoffs

■ obligatorisch: **Grundlagen der Programmierung** $\mapsto$ AuD bzw. GdP

- Datentypen, Kontrollkonstrukte, Prozeduren
- statische und dynamische Datenstrukturen
- „Programmierung im Kleinen“

$\hookrightarrow$ vor allem für die Übung, weniger für die Vorlesung

■ wünschenswert: **Technische Informatik** $\mapsto$ GTI, etwas 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

■ aktuelle Sekundärliteratur

- Heinrich Müller und Frank Weichert [2]: Vorkurs Informatik (aus GdP)
- William Stallings [3]: Computer Organization and Architecture

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

Einleitung

Konzept

Lehrkanon

Lehrziele

Vorkenntnisse

Arrangement

Veranstaltungsbetrieb

Leistungsnachweise

Ausklang

Arrangement

Veranstaltungsbetrieb

- Vorlesungs-, Übungs- und Rechnerzeiten:
 - auf `sys.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/~wosch/glossar.pdf`

- Aneignung von neuem Wissen
 - an der Präsentation teilnehmen, ihr zuhören, Fragen stellen
 - behandelte Themen untereinander diskutieren und nachbereiten
- mit bisherigem/anderem/zukünftigen Wissen in Beziehung bringen:
 - GdP** ■ 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 zur Verfügung
 - PDF: auf `sys.cs.fau.de` dem Reiter „Lehre“ folgen
 - Anzahl und „Füllungsdichte“ sind bewusst eher hoch gehalten:
 - schriftlich fixierte Gedankenstränge als Hilfe zur Nachbearbeitung

- Wissen durch **direkte Erfahrung** vertiefen

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

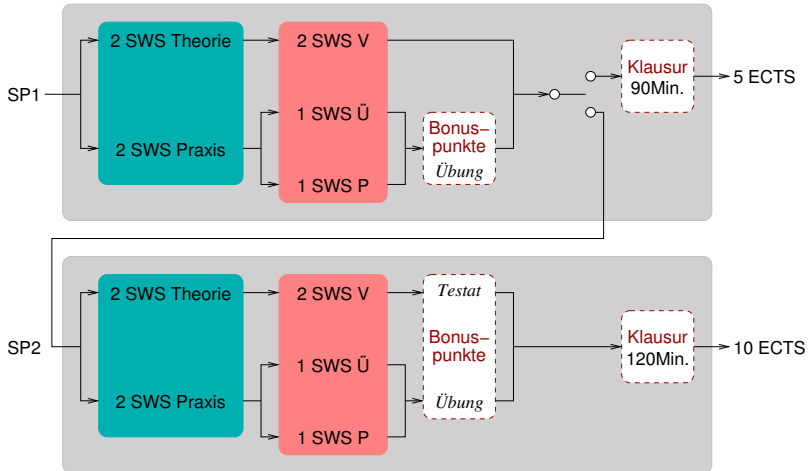
- Diskussion der Übungsaufgaben, Lösungsansätze ausarbeiten
- Vorlesungsstoff festigen, offene Fragen klären
- **Tafelübung** unter Anleitung einer/s Übungsleiterin/s
 - Anmeldung durch **WAFFEL**¹ (URL siehe Webseite von SP)
 - Übungsaufgaben sind in Gruppen zu bearbeiten: Kannvorschrift
 - ist abhängig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

¹Abk. Webanmeldefrickelformular Enterprise Logic

Arrangement

Leistungsnachweise

Studien- und Prüfungsleistungen



- **Übungsaufgaben:** 6 (SP1) + 5 (SP2) Programmieraufgaben
 - abgegebene Programme werden korrigiert und mit Punkten bewertet
 - **unzureichende Erklärung der Lösung** ergibt **0 Punkte**
 - Nichtanwesenheit impliziert unzureichende Erklärung
- 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 (*multiple choice*)

Notenbonus für die Klausur (auch: „Maxiklausur“)

- alle Übungspunkte gehen in die Bonusberechnung ein
- maximal ist ein Notenbonus von 0,7 erreichbar

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

■ **Notenbonus** nur auf Basis der Übungen **des letzten SP-Moduls**

- beeinflusst die Punkte-Notenskala der Klausur nicht, er wird 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 mit Klausurpunkten erfolgen

■ **Klausur:** Termin noch offen, Anfang vorlesungsfreie Zeit

- GSP** ■ Struktur analog Testat (S. 27), 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
- Im Team entwickeln, aber selbst zum Laufen bringen

Einleitung

Konzept

Lehrkanon

Lehrziele

Vorkenntnisse

Arrangement

Veranstaltungsbetrieb

Leistungsnachweise

Ausklang

Ausklang

Kontakt

Dozenten

- Jürgen Kleinöder
- Rüdiger Kapitza

Mitarbeiter

- Luis Gerhorst
- Thomas Preisner
- Tobias Häberlein

Tutoren

- Lukas Bilstein, Dana Elsmann, Kevin K., Maximilian Rostock, Julian Z., Christian Halder, Stefan Schmitt, Nova Ruff, Kilian Hanich, Luca Kellermann



www.augsburger-puppenkiste.de



Quelle: qmediasolutions.com

Ausklang

Bibliographie

[1] ARISTOTELES:

Nikomachische Ethik.

c. 334 BC

[2] MÜLLER, H. ; WEICHERT, F. :

Vorkurs Informatik – Der Einstieg ins Informatikstudium.

Springer Fachmedien Wiesbaden GmbH, 2017. –

392 S. –

ISBN 978-3-658-16141-5

[3] STALLINGS, W. :

Computer Organization and Architecture.

Pearson, 2021. –

896 S. –

ISBN 9781292420103