

# Betriebssystemtechnik

Adressräume: Trennung, Zugriff, Schutz

## I. Einleitung

Wolfgang Schröder-Preikschat

25. April 2019



# Gliederung

## Einführung

Motivation  
Grundlagen  
Inhalt

## Organisation

Voraussetzungen  
Veranstaltungsbetrieb  
Leistungsnachweise

## Anhang



# Schutz von Prozessen

~ **in räumlicher Hinsicht** ~> BST

## ■ Angriffssicherheit (*security*)

- Schutz einer Entität vor seiner Umgebung
- Immunität
- verhindern, in einen Adressraum einbrechen zu können

## ■ Betriebssicherheit (*safety*)

- Schutz der Umgebung vor einer Entität
- Isolation
- verhindern, aus einem Adressraum ausbrechen zu können

~ **in zeitlicher Hinsicht** ~> EZS[4]

## ■ Einhaltung von Terminen, Vermeidung von Interferenzen

~ **in energetischer Hinsicht** ~> EASY[3]

## ■ Abfederung von Energiespitzen, Einhaltung von Energiebudgets



# Adressraum

(vgl. auch [5])

## ■ realer ~

- reflektiert die phys(ikal)ischen Eigenschaften des Rechensystems
- nicht zu jeder Adresse gibt es einen Adressaten (Speicher, Geräte)
- die Bindung zwischen beiden ist fest, zur Laufzeit unveränderlich
  - **Vorsicht:** Speicherbankumschaltung
- ungültige Adressen implizieren undefiniertes/fehlerhaftes Verhalten

## ■ logischer ~

- reflektiert die strukturellen Eigenschaften eines Programms
- zu jeder Adresse gibt es immer einen (speicher-) residenten Adressaten
- die Bindung zwischen beiden ist jedoch lose, zur Laufzeit veränderlich
- ungültige Adressen – innerhalb des Adressraums – gibt es nicht
  - **Vorsicht:** Unterschied zwischen Segmentierung und Seitennummerierung

## ■ virtueller ~

- reflektiert die gegenwärtige/zukünftige Auslastung des Rechensystems
- zu einer Adresse kann es zeitweilig einen nichtresidenten Adressaten geben
- ansonsten „erben“ die Adressen alle Eigenschaften logischer Adressräume



## Adressraumtrennung

Segmentierung oder Eingrenzung von Programmen:

- **Maschinenprogrammzebene** (Ebene<sub>3</sub>)
  - die **Zentraleinheit**<sup>1</sup> ermöglicht Immunität/Isolation in Hardware
    - MMU (Abk. *memory management unit*) ..... logischer Adressraum
    - MPU (Abk. *memory protection unit*) ..... phys(ikal)ischer Adressraum
  - das **Betriebssystem**<sup>1</sup> programmiert diese Hardware problemspezifisch
- **Programmiersprachenebene** (Ebene<sub>5</sub>)
  - der **Kompilierer**<sup>1</sup> ermöglicht Immunität/Isolation in Software
  - Programme liegen in einer **typischeren Programmiersprache** vor

Prozesse können die durch ihren logischen Adressraum jew. definierte **Schutzdomäne** nicht oder nur kontrolliert verlassen

- Abwesenheit von Prozessor- und Speicherfehlern vorausgesetzt
  - je nach Abstraktionsebene aber mit unterschiedlichem Wirkungsfeld

<sup>1</sup>Prozessor

## Adressraumzugriff

Folge der Trennung: Adressraumgrenzen überschreitende Operationen

- durch **Wechsel** der Schutzdomäne
  - prozedurbasierte Technik
    - Systemaufruf, leichtgewichtiger Fernaufruf
    - Kontrollflussfortsetzung im anderen Adressraum
  - koroutinenbasierte Technik
    - Nachrichtenversenden, Fernaufruf
    - Kontrollflusswechsel hin zum anderen Adressraum
- durch **Mitbenutzung** (*sharing*) von Adressraumbereichen
  - Datenverbund (*data sharing*)
    - mit gleichförmigen oder ungleichförmigen Lese-/Schreibrechten
  - Gemeinschaftsbibliothek (*shared library*)
- durch Kombination beider Ansätze
  - Einrichtung eines Datenverbunds beim Wechsel der Schutzdomäne
    - aus dem „eingewechselten Adressraum“ heraus veranlasst
    - zum Lesen/Schreiben von Entitäten des „ausgewechselten Adressraums“

## Adressraumschutz

Isolation schafft Immunität

- **hardwarebasiert**, segment- oder seitenorientiert
  - MMU/MPU vergleicht Zugriffsart und Zugriffsrecht
    - Lesen, Schreiben, Ausführen – auch in Kombination
  - CPU begeht Ausnahme von normaler Programmausführung
    - lässt den aktuellen Maschinenbefehl in die Falle (*trap*) laufen
    - bewirkt damit eine **synchrone Programmunterbrechung**
  - Betriebssystem führt Ausnahmebehandlung [2] durch
    - Wiederaufnahmmodell: Fortsetzung des unterbrochenen Prozesses
    - Beendigungsmodell: Abbruch des unterbrochenen Prozesses
- **softwarebasiert**, datentyporientiert ⇒ **sprachbasiert**
  - Laufzeitsystem – d. h., der Prozess selbst – führt o. g. Funktionen durch
  - bestimmte Überprüfungen nimmt jedoch bereits der Kompilierer vor
    - alle statisch, also vor Laufzeit, entscheidbaren Zugriffsoperationen
- in Synergie beider Ansätze: **Befähigung** (*capability*, [1])
  - befähigungsbasierte Systeme sind kompliziert – obwohl ideal zum Schutz

## Lernziele

### Vorlesung

- **Wissen** zu Adressraumkonzepten von Betriebssystemen vertiefen
- **Verstehen** über (logische) Adressräume festigen
  - inhaltliches Begreifen verschiedener Facetten von Adressräumen
  - intellektuelle Erfassung des Zusammenhangs, in dem Adressräume stehen

### Übung

↪ mikrokern-ähnliches Betriebssystem

- **Anwenden** ausgewählter Vorlesungsinhalte für OOSTuBS
- **Analyse** der Anforderungen an und Gegebenheiten von OOSTuBS
- **Synthese** von Adressraumabstraktionen und OOSTuBS
- **Evaluation** des erweiterten OOSTuBS: Vorher-nachher-Vergleich

# Überblick

- Systemaufruf; Befehlsformate der Maschinenprogrammzebene ÜV
- ↳ Betriebssystemarchitektur: {Nano,Mikro,Makro,Exo}kern V
- ↳ Schichtenstruktur von Betriebssystemen V
- Segmentierung, Seitennummerierung; Seitenkacheltablelle ÜV
- ↳ Adressraumverwaltung, Hardwaremerkmale V
- ↳ Adressraummodelle, Benutzer- und Kernadressraum V
- Interprozesskommunikation, Semantiken ÜV
- ↳ Kommunikationsabstraktionen V
- ↳ Mitbenutzung (*sharing*) V
- Gemeinschaftsbibliotheken, dynamisches Binden V
- Nachlese, Ausblick V



# Gliederung

Einführung  
Motivation  
Grundlagen  
Inhalt

Organisation  
Voraussetzungen  
Veranstaltungsbetrieb  
Leistungsnachweise

Anhang



# Anforderungen

Vorkenntnisse

## Voraussetzung

### Softwaresysteme

- SP, SPiC
- BS ⇔ OOSTuBS

### Programmiersysteme

- C, C++, make
- ASM

### Hardwaresysteme

- x86, IA64
- Mehrkerner

## Erfahrung

- in der hardwarenahen Programmierung
  - Gerätetreiber, Unterbrechungsbehandlung, Prozesswechsel
- in der Fehlersuche/-beseitigung (*debugging*) in Betriebssystemen
  - nichtsequentielle Programme, mehrkernige Prozessoren
- in der projektorientierten Entwicklung nativer Systemprogramme

## Erwartung

- intrinsische Motivation, kritisches Denken, positive Fehlerkultur



# Unterrichtstermine und -sprache

- Vorlesungs-, Übungs- und Rechnerzeiten:
  - auf [www4.cs.fau.de](http://www4.cs.fau.de) dem Reiter „Lehre“ folgen
- Unterrichtssprache:



- Vorlesung und Übung ■ Fachbegriffe
- informatische Fachsprache
  - Sachwortverzeichnis (in Arbeit und Überarbeitung)
    - [www4.cs.fau.de/~wosch/glossar.pdf](http://www4.cs.fau.de/~wosch/glossar.pdf)



## Übungsbetrieb

- **Tafelübung**  $\leadsto$  „*learning by exploring*“
  - Anmeldung über [WAFTEL](#)<sup>2</sup> (URL siehe Leitseite von BST)
    - Freischaltung erfolgt nach der Vorlesung, heute im Tagesverlauf
  - Übungsaufgaben sind bevorzugt in Gruppen zu bearbeiten
- **Rechnerarbeit**  $\leadsto$  „*learning by doing*“, **kein Tafelersatz**
  - Anmeldung ist nicht vorgesehen, reservierte Arbeitsplätze s.o.
  - bei Fragen zu den Übungsaufgaben, Übungsleiter konsultieren
    - Email senden bzw. einfach vorbeischaun...

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



<sup>2</sup>Abk. für Webanmeldefrickelformular Enterprise Logic

## Prüfungsrelevante Studienleistung

### 5 ECTS

- BST
- 4 SWS (2 V + 2 Ü)

- Prüfungsgespräch
  - 20 Minuten
  - Stoff zu V + Ü

### 7,5 ECTS

- BST++
- 6 SWS (2 V + 2 Ü + 2 EÜ)
  - erweiterte Übung
  - ggf. auch Extraaufgabe

- Prüfungsgespräch
  - 30 Minuten
  - Stoff zu V + Ü + EÜ

- Bearbeitung aller Übungsaufgaben wird dringendst empfohlen
- Anmeldung zum Prüfungsgespräch per *E-Mail* an: [wosch@cs.fau.de](mailto:wosch@cs.fau.de)
  - angeben ob BST oder BST++
  - Termin oder Terminfenster mitsenden
    - Prüfungszeitraum (Ausnahmen bestätigen die Regel)

## Gliederung

### Einführung

Motivation  
Grundlagen  
Inhalt

### Organisation

Voraussetzungen  
Veranstaltungsbetrieb  
Leistungsnachweise

### Anhang

## Kontakt



- Wolfgang Schröder-Preikschat, Prof. Dr.-Ing. habil.
  - Vorlesung
  - <http://www4.cs.fau.de/~wosch>



- Timo Hönig, Dr.-Ing.
  - Vorlesung, Übungen
  - <http://www4.cs.fau.de/~thoenig>



- Bernhard Heinloth, M. Sc.
  - Übungen
  - <http://www4.cs.fau.de/~heinloth>

- [1] DENNIS, J. B. ; HORN, E. C. V.:  
Programming Semantics for Multiprogrammed Computations.  
In: *Communications of the ACM* 9 (1966), März, Nr. 3, S. 143–155
- [2] GOODENOUGH, J. B.:  
Exception Handling: Issues and a Proposed Notation.  
In: *Communications of the ACM* 18 (1975), Nr. 12, S. 683–696
- [3] HÖNIG, T. :  
*Energy-Aware Computing Systems*.  
[http://www4.informatik.uni-erlangen.de/Lehre/WS18/V\\_EASY](http://www4.informatik.uni-erlangen.de/Lehre/WS18/V_EASY), 2018 ff.
- [4] SCHRÖDER-PREIKSCHAT, W. :  
*Echtzeitsysteme*.  
[http://www4.informatik.uni-erlangen.de/Lehre/WS05/V\\_EZS](http://www4.informatik.uni-erlangen.de/Lehre/WS05/V_EZS), 2005 ff.
- [5] SCHRÖDER-PREIKSCHAT, W. ; KLEINÖDER, J. :  
*Systemprogrammierung*.  
[http://www4.informatik.uni-erlangen.de/Lehre/WS08/V\\_SP](http://www4.informatik.uni-erlangen.de/Lehre/WS08/V_SP), 2008 ff.

