

Betriebssystemtechnik

Adressräume: Trennung, Zugriff, Schutz

I. Einleitung

Wolfgang Schröder-Preikschat

25. April 2019

Gliederung

Einführung

Motivation

Grundlagen

Inhalt

Organisation

Voraussetzungen

Veranstaltungsbetrieb

Leistungsnachweise

Anhang



© wosch

BST (SS 2019, VL 1)

Einführung

2

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



© wosch

BST (SS 2019, VL 1)

Einführung – Motivation

3

© wosch

BST (SS 2019, VL 1)

Einführung – Grundlagen

4

Segmentierung oder Eingrenzung von Programmen:

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

¹Prozessor

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
 - Wiederaufnahmemodell: 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



Folge der Trennung: Adressraumgrenzen überschreitende Operationen

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



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 OStuBS
- **Analyse** der Anforderungen an und Gegebenheiten von OStuBS
- **Synthese** von Adressraumabstraktionen und OStuBS
- **Evaluation** des erweiterten OStuBS: Vorher-nachher-Vergleich



Überblick

- | | |
|---|----|
| ■ Systemaufruf; Befehlsformate der Maschinenprogrammebene | ÜV |
| ↳ Betriebssystemarchitektur: {Nano,Mikro,Makro,Exo}kern | V |
| ↳ Schichtenstruktur von Betriebssystemen | V |
| ■ Segmentierung, Seitennummerierung; Seitenkacheltabelle | ÜV |
| ↳ Adressraumverwaltung, Hardwaremerkmale | V |
| ↳ Adressraummodelle, Benutzer- und Kernadressraum | V |
| ■ Interprozesskommunikation, Semantiken | ÜV |
| ↳ Kommunikationsabstraktionen | V |
| ↳ Mitbenutzung (<i>sharing</i>) | V |
| ■ Gemeinschaftsbibliotheken, dynamisches Binden | V |
| ■ Nachlese, Ausblick | V |



© wosch

BST (SS 2019, VL 1)

Einführung – Inhalt

9

Anforderungen

Vorkenntnisse

Voraussetzung

Softwaresysteme

- SP, SPiC
- BS ⇔ OOSuBS

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



© wosch

BST (SS 2019, VL 1)

Organisation – Voraussetzungen

11

Gliederung

Einführung
Motivation
Grundlagen
Inhalt

Organisation
Voraussetzungen
Veranstaltungsbetrieb
Leistungsnachweise

Anhang



© wosch

BST (SS 2019, VL 1)

Organisation

10

Unterrichtstermine und -sprache

- Vorlesungs-, Übungs- und Rechnerzeiten:
 - auf 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



© wosch

BST (SS 2019, VL 1)

Organisation – Veranstaltungsbetrieb

12

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

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



²Abk. für Webanmeldefrickelformular Enterprise Logic

Gliederung

Einführung
Motivation
Grundlagen
Inhalt

Organisation
Voraussetzungen
Veranstaltungsbetrieb
Leistungsnachweise

Anhang

- | | |
|--|--|
| <p>5 ECTS</p> <ul style="list-style-type: none">■ BST■ 4 SWS (2 V + 2 Ü) | <p>7,5 ECTS</p> <ul style="list-style-type: none">■ BST++■ 6 SWS (2 V + 2 Ü + 2 EÜ)<ul style="list-style-type: none">■ erweiterte Übung■ ggf. auch Extraaufgabe |
| <p>■ Prüfungsgespräch</p> <ul style="list-style-type: none">■ 20 Minuten■ Stoff zu V + Ü | <p>■ Prüfungsgespräch</p> <ul style="list-style-type: none">■ 30 Minuten■ Stoff zu V + Ü + EÜ |
| <p>■ Bearbeitung aller Übungsaufgaben wird dringendst empfohlen</p> | |
| <p>■ Anmeldung zum Prüfungsgespräch per <i>E-Mail</i> an:
wosch@cs.fau.de</p> <ul style="list-style-type: none">■ angeben ob BST oder BST++■ Termin oder Terminfenster mitsenden■ Prüfungszeitraum (Ausnahmen bestätigen die Regel) | |

Kontakt



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



- Timo Höning, 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, 2018 ff.
- [4] SCHRÖDER-PREIKSCHAT, W. :
Echtzeitssysteme.
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, 2008 ff.

