

Betriebssystemtechnik

Adressräume: Trennung, Zugriff, Schutz

I. Einleitung

Wolfgang Schröder-Preikschat

11. April 2016



Gliederung

Einführung

Motivation
Grundlagen
Inhalt

Organisation

Voraussetzungen
Vorlesung und Übung
Modulkonzept
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[3]

■ Einhaltung von Terminen, Vermeidung von Interferenzen

~ **in energetischer Hinsicht** ~> ffs.

■ Abfederung von Energiespitzen, Einhaltung von Energiebudgets



Adressraum

(vgl. auch [4])

■ 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** (Ebene₃)
 - 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** (Ebene₅)
 - der **Kompilierer**¹ 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

¹Prozessor

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
- ↳ Adressraummodelle, Benutzer- und Kernadressraum V
- ↳ sprachbasierte Systeme, Systemprogrammiersprachen V
- Interprozesskommunikation, Semantiken ÜV
- ↳ Kommunikationsabstraktionen V
- ↳ Mitbenutzung (*sharing*) V
- Gemeinschaftsbibliotheken, dynamisches Binden V
- Nachlese, Ausblick V



Gliederung

Einführung
Motivation
Grundlagen
Inhalt

Organisation
Voraussetzungen
Vorlesung und Übung
Modulkonzept
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



Vorlesungsbetrieb und Lehrmaterialien

Vorlesungstermine bis KW 28

- Mo, 14:15 – 15:45, 0.031

„Weiße Flecke“

- 23.05. und evtl. 27.06.

Handzettel (*handout*) sind verfügbar wie folgt:

- www4.informatik.uni-erlangen.de/Lehre/SS16/V_BST
- Folienkopien werden vor der Vorlesung ausgegeben

Fachbegriffe der Informatik (Deutsch ↔ Englisch)

- www.babylonia.ork.uk
- www.inf.fu-berlin.de/inst/ag-ss/montagswort
- www.aktionlebendigesdeutsch.de



Übungsbetrieb

Termine ab KW 16

- T Mi, 10:15 – 11:45, 0.031/00.153
- R zwei Termine, siehe Web

Ausfälle in KW 20

- Pfingstferien/Bergwoche

Tafelübung (T)

- Anmeldung über [WAFFEL²](#) (URL siehe Webseite von BST)
 - Freischaltung erfolgt nach der Vorlesung, heute im Tagesverlauf
- Übungsaufgaben sind bevorzugt in Gruppen zu bearbeiten

Rechnerarbeit (R): liegt stark in Eigenverantwortung

- Anmeldung ist nicht vorgesehen, reservierte Arbeitsplätze s.o.
- bei Fragen zu den Übungsaufgaben, Übungsleiter konsultieren
 - Email senden bzw. einfach vorbeischaun. . .

²Abk. für Webanmeldefrickelformular Enterprise Logic

Bedeutung von Tafel- und Rechnerübungen

Tafelübungen \leadsto „learning by exploring“

- Besprechung der Übungsaufgaben, Skizzierung von Lösungswegen
- Vertiefung des Vorlesungsstoffes, Klärung offener Fragen

Rechnerarbeit \leadsto „learning by doing“

- selbstständiges Bearbeiten der Übungsaufgaben am Rechner
- der Rechner ist allerdings **kein Tafelersatz**

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



Systemsoftwaretechnik (SST)

Modul SST

- V + Ü
- 6 SWS (3 + 3)
- 7,5 ECTS (5 + 2,5)

=

Modul BST

- V + Ü
- 4 SWS
- 5 ECTS

+

Modul KSS

- V + Ü
- 2 SWS
- 2,5 ECTS

- BST motiviert Betriebssysteme als **variantenreiche Systemsoftware**
 - am Beispiel selbst entwickelter Experimentiersoftware (OOSTuBS)
 - mit Softwaretechnik im Hintergrund
- KSS lehrt **Variabilitätsverwaltung** in Betriebssystemproduktlinien
 - am Beispiel von Eigenentwicklungen (CiAO, Sloth) und Linux
 - mit Softwaretechnik im Vordergrund

Prüfungsrelevante Studienleistung

5 ECTS

- BST
- 4 SWS (2 V + 2 Ü)

7,5 ECTS

- SST
- 6 SWS (3 V + 3 Ü)
 - 4 SWS durch BST
 - 2 SWS durch KSS

■ Prüfungsgespräch

- 20 Minuten
- Stoff zu BST
- Prüfer² wosch, Beisitzer² dl

■ Prüfungsgespräch

- 30 Minuten
- Stoff zu BST + KSS
- Prüfer² wosch und dl

- Bearbeitung aller Übungsaufgaben wird dringendst empfohlen
- Anmeldung zum Prüfungsgespräch per *E-Mail* an:

vosch@cs.fau.de für SST oder BST

dl@cs.fau.de für SST

²Ausnahmen bestätigen die Regel.

Einführung

Motivation
Grundlagen
Inhalt

Organisation

Voraussetzungen
Vorlesung und Übung
Modulkonzept
Leistungsnachweise

Anhang



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



- Daniel Lohmann, Dr.-Ing.
 - Vorlesung KSS/SST
 - <http://www4.cs.fau.de/~lohmann>



- Christian Dietrich, Dipl. Inf.
 - Übungen BST + KSS
 - <http://www4.cs.fau.de/~stettberger>



- Valentin Rothberg, M. Sc.
 - Übungen BST + KSS
 - <http://www4.cs.fau.de/~gabor>



Literaturverzeichnis

- [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] SCHRÖDER-PREIKSCHAT, W. :
Echtzeitsysteme.
http://www4.informatik.uni-erlangen.de/Lehre/WS05/V_EZS, 2005 ff.
- [4] SCHRÖDER-PREIKSCHAT, W. ; KLEINÖDER, J. :
Systemprogrammierung.
http://www4.informatik.uni-erlangen.de/Lehre/WS08/V_SP, 2008 ff.

