

Betriebssystemtechnik

Adressräume: Trennung, Zugriff, Schutz

I. Einleitung

Wolfgang Schröder-Preikschat

26. April 2022



Einführung

Motivation

Grundlagen

Inhalt

Organisation

Voraussetzungen

Veranstaltungsbetrieb

Leistungsnachweise

Anhang



~ **in räumlicher Hinsicht** \rightsquigarrow 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** \rightsquigarrow EZS[3]

- Einhaltung von Terminen, Vermeidung von Interferenzen

~ **in energetischer Hinsicht**

- Abfederung von Energiespitzen, Einhaltung von Energiebudgets



■ 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



Segmentierung oder Eingrenzung von Programmen:

- **Maschinenprogrammzebene** (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 **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



Grenzen überschreitende Operationen als Folge der Trennung

- 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 **Kombinierung** beider Ansätze
 - Einrichtung eines Datenverbunds beim Wechsel der Schutzdomäne
 - aus dem „eingewechselten Adressraum“ heraus veranlasst
 - zum Lesen/Schreiben von Entitäten des „ausgewechselten Adressraums“



- **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



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 \rightsquigarrow mikrokern-ähnliches Betriebssystem

- *Anwenden* ausgewählter Vorlesungsinhalte für StuBS
- *Analyse* der Anforderungen an und Gegebenheiten von StuBS
- *Synthese* von Adressraumabstraktionen und StuBS
- *Evaluation* des erweiterten StuBS_{ml}: Vorher-nachher-Vergleich



- Einführung: vertikale (räumliche) Isolation ÜV

- Einflussfaktoren
 - Systemaufrufe: Befehlsformate der Maschinenprogrammzebene ÜV
 - Betriebssystemarchitektur: {Nano,Mikro,Makro,Exo}kern V
 - Hierarchien: Schichtenstruktur von Betriebssystemen V

- Adressraumkonzepte
 - seitenbasiert: ein-/mehrstufig, invertiert, überwacht ÜV
 - Segmentierung: seitenbasierte Hybride V



- Sprachbasierung: Systemprogrammiersprachen S

- Adressraummodelle
 - Mehradressraumsystem: total/partiell privat ÜV
 - Einadressraumsystem: Randomisierung, Befähigungen V

- Spezialfälle
 - adaptiver Hauptspeicherschutz: TLB, typsichere Programme ÜV
 - virtuell gemeinsamer Speicher: seitenbasierte „IPC“ ÜV
 - virtuell nicht-flüchtiger Hauptspeicher: NVRAM, Transparenz V
 - dynamisches Binden: Gemeinschaftsbibliotheken V

- Nachlese, Ausblick V



Einführung

Motivation

Grundlagen

Inhalt

Organisation

Voraussetzungen

Veranstaltungsbetrieb

Leistungsnachweise

Anhang



■ Voraussetzung

Softwaresysteme

- SP, SPiC
- BS \Leftrightarrow StuBS

Programmiersysteme

- C, C++, make
- ASM

Hardwaresysteme

- x86/x64
- 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



- Vorlesungs-, Übungs- und Rechnerzeiten:
 - auf `sys.cs.fau.de` dem Reiter „Lehre“ folgen

- Unterrichtssprache:



- Vorlesung und Übung



- Übung und Fachbegriffe

- informatische Fachsprache
 - Sachwortverzeichnis (in Arbeit und Überarbeitung)
 - `www4.cs.fau.de/~wosch/glossar.pdf`



- **Tafelübung** \leadsto „*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** \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!*



²Abk. für Webanmeldefrickelformular Enterprise Logic

5 ECTS

- BST und StuBS_{ml}
- 4 SWS (2 V + 2 Ü)

- Prüfungsgespräch
 - 20 Minuten
 - Stoff zu V + Ü

7,5 ECTS

- BST und MPStuBS_{ml}
- 6 SWS (2 V + 2 Ü + 2 EÜ)
 - erweiterte Übung
 - ggf. auch Extraaufgabe

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

- erfolgreiche Bearbeitung der Übungsaufgaben ist Voraussetzung
- Anmeldung zum Prüfungsgespräch per *E-Mail* an:
wosch@cs.fau.de
 - angeben ob 5 oder 7,5 ECTS
 - Termin oder Terminfenster mitsenden
 - Prüfungszeitraum (Ausnahmen bestätigen die Regel)



Einführung

Motivation

Grundlagen

Inhalt

Organisation

Voraussetzungen

Veranstaltungsbetrieb

Leistungsnachweise

Anhang





- Wolfgang Schröder-Preikschat, Prof. Dr.-Ing. habil.
 - Vorlesung
 - sys.cs.fau.de/person/wosch



- Bernhard Heinloth, M. Sc.
 - Übungen
 - sys.cs.fau.de/person/heinloth



- Phillip Raffeck, M. Sc.
 - Übungen
 - sys.cs.fau.de/person/raffeck



- Dustin Nguyen, M. Sc.
 - Übungen
 - sys.cs.fau.de/person/nguyen



- [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.

