Grundlagen echtzeitfähiger Systeme

Thema 1: Einführung

Robert Baumgartl

12. März 2024

Organisatorisches

. . . finden Sie unter

https://www.informatik.htw-dresden.de/~robge/ezs/ezs.html

Ressourcen

Folien, Bücher

- Vorlesungsfolien (im PDF), Hausaufgaben, aktuelle Informationen erscheinen im Laufe des Semesters unter *http://www.htw-dresden.de/~robge/ezs/ezs.html*
- Jane W. S. Liu. *Real-Time Systems*. Prentice Hall, 2000
- Alan Burns und Andy Wellings. *Real-Time Systems and Programming Languages*. 3. Aufl. Addison-Wesley, 2001
- Dieter Zöbel. *Echtzeitsysteme: Grundlagen der Planung*. Springer, 2008
- Jim Cooling. *Software Engineering for Real-Time Systems*. 1. Aufl. Addison-Wesley, 2003

Ressourcen

www

- Kompendium; aus meinen bisherigen Veranstaltungen kompiliert
- Doug Jensens Site: *http://www.real-time.org/*
- Computer-Related Incidents with Commercial Aircraft: *http://www.rvs.uni-bielefeld.de/publications/compendium/index.html*
- GI-Fachgruppe *Echtzeitsysteme*: *http://www.real-time.de/*
- Vorsicht – die deutsche Wikipedia taugt (bislang) nicht besonders zur Erklärung des Fachgebiets
- Beispiel: Erklärung zur *Prioritätsgrenze*

Was machen wir hier eigentlich?

Die Webseite besagt:

> *„Die LV Echtzeitsysteme ist eine einsemestrige Einführung in Theorie und Praxis von Rechensystemen, die zur Lösung zeitkritischer Probleme eingesetzt werden.“*

= Kennenlernen von Verfahren und Methoden zur/zum

- Entwurf,
- Realisierung und
- Analyse

von (Rechen-)systemen, die **zeitliche Garantien** geben können.

Thematische Übersicht

Prinzip: Anwendung des Echtzeitgedankens auf alle möglichen Aspekte/Komponenten herkömmlicher Rechensysteme.
Beispiele:

- *Echtzeit*-Scheduling
- *echtzeit*fähige Ressourcenverwaltung
- *Echtzeit*-Kommunikation
- *Echtzeit*-Betriebssysteme
- Programmiersprachen für *Echtzeitsysteme*
- Verifikation von *Echtzeit*systemen

Allerdings gibt es auch noch einige originäre Themen:

- Zeit
- Fehlertoleranz

Verwandte Diskursbereiche

- Eingebettete Systeme
- Rechnernetze und Betriebssysteme
- Schedulingtheorie (Mathematik!)
- Rechnerarchitektur

Voraussetzungen

- Grundlagen Programmierungstechnik (Zeigerkonzept, Listen, Bäume)
- Rechner- bzw. Prozessorarchitektur: Caches, Pipelines, Busse etc.
- Programmbeispiele in C, kaum Assembler
- Grundlagen Betriebssysteme (z.B. Scheduling, Semaphore, Deadlock, Virtueller Speicher)

Am allerwichtigsten ist jedoch **Interesse**!

Was bedeutet „(in) Echtzeit“ ?

umgangssprachlich: Transformation der Eingangsdaten erfolgt mit der Geschwindigkeit ihres Empfangs. („*xy* läuft in Echtzeit“; $xy \in$ {MP3-Encoder, Renderer, . . . })

Hermann Kopetz: “A real-time computer system is a computer system in which the correctness of the system behaviour depends not only on the logical results of the computations, but also on the physical instant at which these results are produced.” [1]

Jane Liu: “. . . a real-time system is required to complete its work and deliver its services on a timely basis.” [2]

[1] Hermann Kopetz. *Real-Time Systems*. 3. Norwell, MA: Kluwer Academic Publishers, 1999, S. 2.

[2] Jane W. S. Liu. *Real-Time Systems*. Prentice Hall, 2000, S. 1.

Was bedeutet „(in) Echtzeit“ ?

DIN 44300: „Echtzeitbetrieb ist ein Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig betriebsbereit sind derart, daß die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind.“

dt. Wikipedia: „Von Echtzeitsystemen [...] spricht man, wenn ein System ein Ergebnis innerhalb eines vorher fest definierten Zeitintervalles garantiert berechnet, also bevor eine bestimmte Zeitschranke erreicht ist. Die Größe des Zeitintervalles spielt dabei keine Rolle [...].“

Schlussfolgerung

Das bedeutet:

- *sowohl* die Korrektheit,
- *als auch* die Dauer der Ermittlung

der Ergebnisse wird *garantiert*.

In Nicht-Echtzeit-Systemen wird i. a. nur die Korrektheit garantiert.

Vereinfachung: Eine bestimmte zeitliche *Schranke* (Deadline) wird nicht überschritten.

Verhalten eigentlich auch in „normalen“ Systemen erwünscht, z.B. bei der Reaktion auf Benutzereingaben (Problem: Aufwand).

Nochmals . . .

Was bedeutet „die Dauer der Operation wird garantiert“?

Ganz einfach: eine Spezifikation legt fest, wie lange die Operation maximal dauern darf, und das Echtzeitsystem garantiert, dass diese maximale Dauer niemals überschritten wird, was immer auch

- innerhalb des Systems und
- in der Umwelt

passieren mag.

Die Maximaldauer nennen wir (relative) Deadline (der Operation).

Häufige Mißverständnisse und Fehlannahmen

- Echtzeitsysteme sind sehr schnelle Rechner.

→ Widerspruch: Prozessoren in Echtzeitsystemen sind häufig niedriggetaktete Mikrocontroller.

- *Präemptive* Prozessorvergabe ermöglicht prinzipiell ein Echtzeitsystem.

→ reicht nicht aus; weder Windows noch Linux sind von Haus aus echtzeitfähig.

- Moore's Law bewirkt, dass Echtzeitsysteme überflüssig werden.

→ Falsche Annahme, dass „Echtzeit“ *schnell* bedeutet

- Es kann sowieso keine Garantie der Funktionsfähigkeit eines Rechners gegeben werden, daher sind Echtzeitsysteme nicht notwendig.
- Echtzeitsysteme werden in Assembler programmiert.

Häufige Mißverständnisse und Fehlannahmen

- Echtzeitsysteme sind sehr schnelle Rechner.

→ Widerspruch: Prozessoren in Echtzeitsystemen sind häufig niedriggetaktete Mikrocontroller.

- *Präemptive* Prozessorvergabe ermöglicht prinzipiell ein Echtzeitsystem.

→ reicht nicht aus; weder Windows noch Linux sind von Haus aus echtzeitfähig.

- Moore's Law bewirkt, dass Echtzeitsysteme überflüssig werden.

→ Falsche Annahme, dass „Echtzeit" *schnell* bedeutet

- Es kann sowieso keine Garantie der Funktionsfähigkeit eines Rechners gegeben werden, daher sind Echtzeitsysteme nicht notwendig.
- Echtzeitsysteme werden in Assembler programmiert.

Häufige Mißverständnisse und Fehlannahmen

- Echtzeitsysteme sind sehr schnelle Rechner.

→ Widerspruch: Prozessoren in Echtzeitsystemen sind häufig niedriggetaktete Mikrocontroller.

- *Präemptive* Prozessorvergabe ermöglicht prinzipiell ein Echtzeitsystem.

→ reicht nicht aus; weder Windows noch Linux sind von Haus aus echtzeitfähig.

- Moore's Law bewirkt, dass Echtzeitsysteme überflüssig werden.

→ Falsche Annahme, dass „Echtzeit" *schnell* bedeutet

- Es kann sowieso keine Garantie der Funktionsfähigkeit eines Rechners gegeben werden, daher sind Echtzeitsysteme nicht notwendig.
- Echtzeitsysteme werden in Assembler programmiert.

Häufige Mißverständnisse und Fehlannahmen

- Echtzeitsysteme sind sehr schnelle Rechner.

→ Widerspruch: Prozessoren in Echtzeitsystemen sind häufig niedriggetaktete Mikrocontroller.

- *Präemptive* Prozessorvergabe ermöglicht prinzipiell ein Echtzeitsystem.

→ reicht nicht aus; weder Windows noch Linux sind von Haus aus echtzeitfähig.

- Moore's Law bewirkt, dass Echtzeitsysteme überflüssig werden.

→ Falsche Annahme, dass „Echtzeit" *schnell* bedeutet

- Es kann sowieso keine Garantie der Funktionsfähigkeit eines Rechners gegeben werden, daher sind Echtzeitsysteme nicht notwendig.
- Echtzeitsysteme werden in Assembler programmiert.

Häufige Mißverständnisse und Fehlannahmen

- Echtzeitsysteme sind sehr schnelle Rechner.

→ Widerspruch: Prozessoren in Echtzeitsystemen sind häufig niedriggetaktete Mikrocontroller.

- *Präemptive* Prozessorvergabe ermöglicht prinzipiell ein Echtzeitsystem.

→ reicht nicht aus; weder Windows noch Linux sind von Haus aus echtzeitfähig.

- Moore's Law bewirkt, dass Echtzeitsysteme überflüssig werden.

→ Falsche Annahme, dass „Echtzeit" *schnell* bedeutet

- Es kann sowieso keine Garantie der Funktionsfähigkeit eines Rechners gegeben werden, daher sind Echtzeitsysteme nicht notwendig.
- Echtzeitsysteme werden in Assembler programmiert.

Häufige Mißverständnisse und Fehlannahmen

- Echtzeitsysteme sind sehr schnelle Rechner.

→ Widerspruch: Prozessoren in Echtzeitsystemen sind häufig niedriggetaktete Mikrocontroller.

- *Präemptive* Prozessorvergabe ermöglicht prinzipiell ein Echtzeitsystem.

→ reicht nicht aus; weder Windows noch Linux sind von Haus aus echtzeitfähig.

- Moore's Law bewirkt, dass Echtzeitsysteme überflüssig werden.

→ Falsche Annahme, dass „Echtzeit" *schnell* bedeutet

- Es kann sowieso keine Garantie der Funktionsfähigkeit eines Rechners gegeben werden, daher sind Echtzeitsysteme nicht notwendig.
- Echtzeitsysteme werden in Assembler programmiert.

Häufige Mißverständnisse und Fehlannahmen

- Echtzeitsysteme sind sehr schnelle Rechner.

→ Widerspruch: Prozessoren in Echtzeitsystemen sind häufig niedriggetaktete Mikrocontroller.

- *Präemptive* Prozessorvergabe ermöglicht prinzipiell ein Echtzeitsystem.

→ reicht nicht aus; weder Windows noch Linux sind von Haus aus echtzeitfähig.

- Moore's Law bewirkt, dass Echtzeitsysteme überflüssig werden.

→ Falsche Annahme, dass „Echtzeit“ *schnell* bedeutet

- Es kann sowieso keine Garantie der Funktionsfähigkeit eines Rechners gegeben werden, daher sind Echtzeitsysteme nicht notwendig.
- Echtzeitsysteme werden in Assembler programmiert.

Häufige Mißverständnisse und Fehlannahmen

- Echtzeitsysteme sind sehr schnelle Rechner.

→ Widerspruch: Prozessoren in Echtzeitsystemen sind häufig niedriggetaktete Mikrocontroller.

- *Präemptive* Prozessorvergabe ermöglicht prinzipiell ein Echtzeitsystem.

→ reicht nicht aus; weder Windows noch Linux sind von Haus aus echtzeitfähig.

- Moore's Law bewirkt, dass Echtzeitsysteme überflüssig werden.

→ Falsche Annahme, dass „Echtzeit“ *schnell* bedeutet

- Es kann sowieso keine Garantie der Funktionsfähigkeit eines Rechners gegeben werden, daher sind Echtzeitsysteme nicht notwendig.
- Echtzeitsysteme werden in Assembler programmiert.

Sekundäre Anforderungen an Echtzeitsysteme

- hohe Arbeitsgeschwindigkeit
- Fehlertoleranz
- Kosteneffizienz
  - niedriger Energiebedarf
  - hohe Ressourcenauslastung (Prozessor, Speicher)

$\longrightarrow$ Kompromiss je nach Einsatzfall nötig!

Härte eines Echtzeitsystems

a) *hartes* EZ-System: Verletzung einer Deadline hat katastrophale Konsequenzen für das System bzw. die Umwelt.

b) *weiches* EZ-System: Verletzung einer Deadline führt zum Absinken des Wertes des Ergebnisses, Einfluß auf die „Güte“ des Dienstes, Systemleistung sinkt mit der Anzahl verletzter Deadlines.

c) „mittelhartes“ EZ-System: Nach Überschreiten der Deadline ist das Ergebnis wertlos, jedoch keine Katastrophe (sog. “firm deadline”)

Härte eines Echtzeitsystems

Was passiert bei Verletzung einer Deadline?

3 Möglichkeiten, je nach „Wert“ des Ergebnisses nach Überschreitung der Deadline t_d:

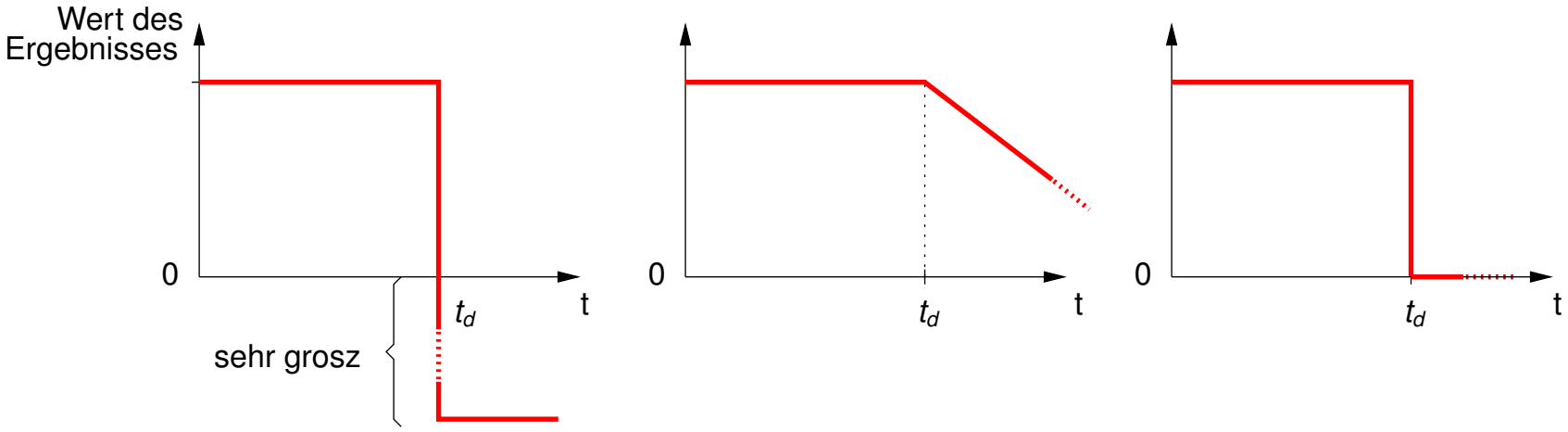

Abbildung: Hartes, weiches und „mittelhartes“ Echtzeitsystem

Härte eines Echtzeitsystems

Anmerkungen

- Unterscheidung zwischen harten und weichen EZ-Systemen nicht immer trivial, insbesondere der Übergang zwischen weichen und Nicht-Echtzeit-Systemen ist oftmals fließend.
- Fehlertoleranz bei harten EZ-Systemen meist ausgeprägter
- Idee der “Flexible Computations”: Aktivität wird geteilt in eine essentielle und eine wahlfreie Phase. Die essentielle Phase muß unbedingt bis zur Deadline abgeschlossen sein. Wahlfreie Phase erhöht die Güte des Resultats. Beispiel: Ermittlung der Raumkoordinaten einer Raumsonde.
- Komplettierung *vor* Deadline oftmals unerwünscht

Klassifizierungsmerkmale von Echtzeitsystemen

1. „Härte“ der Deadline
2. Zuverlässigkeit und Fehlertoleranz (Sicherheit, Fehlererkennung)
3. Verteilung: zentralisiertes oder verteiltes Echtzeitsystem
4. zeitgesteuertes oder ereignisgesteuertes System
5. interaktives oder autonomes System
6. hierarchisches oder Einzelsystem
7. zyklische oder/und asynchrone Aktivitäten

Eingebettete Systeme

Echtzeitsysteme konstituieren häufig *Eingebettete Systeme* (embedded systems):

> *"Embedded computers are defined to be those where the computer is used as a component within a system: not as a computing engine in its own right. [...]"*[3]

d.h. der Computer ist Teil eines Geräts (und das Gerät ist kein Computer im klassischen Sinne).

[3] Jim Cooling. *Software Engineering for Real-Time Systems*. 1. Aufl. Addison-Wesley, 2003, S. 12.

Eingebettete Systeme

Merkmale

- komplexe Interaktion mit Umwelt und anderen eingebetteten Systemen
- werden häufig autonom betrieben
- hohe Anforderungen an Zuverlässigkeit
- ersetzen zunehmend mechanische Systeme
- Hard- und Software werden gemeinsam entworfen (HW/SW-Codesign).
- Einsatz eines Prozessors „von außen“ häufig nicht erkennbar
- arbeiten häufig unter Echtzeit-Bedingungen
- Kosten für Reparatur übersteigen (häufig) Kosten des Systems.

Weitere Einsatzfelder

- Luft- und Raumfahrt: Flugkontrollsystem, Steuerung von Raketenmotoren, Satelliten
- Digitale Steuerungen: **P**roportional **I**ntegral **D**erivative Controller, Robotersteuerungen, Computertomograph
- Automotive-Bereich: ABS, ESP, Airbag-Steuerung, Navigationssysteme, Motormanagement, CAN-und FlexRay-Bus
- Signalverarbeitung: RADAR, Verfolgung (Tracking) von Objekten, Bilderkennung, Unterhaltungselektronik
- Multimedia-Systeme: En- und Dekodierung von Audio/Video-Strömen, Klang- und Sprachsynthese, VoIP, Video Storage Server
- Echtzeit-Datenbanken

Aktuelle Forschungsgebiete

- Mixed Criticality Systems
- Smart Infrastructure
- Worst Case Execution Time (WCET) Analysis
- Verifikation echtzeitfähiger Systeme
- ...

Wichtige Konferenzen:

- Real-Time Systems Symposium (RTSS)
- European Conference on Real-Time Systems (ECRTS)

Zusammenfassung: Was haben wir gelernt?

1. Echtzeitsysteme sichern die Rechtzeitigkeit aller relevanten Operationen.
2. harte vs. weiche vs. „mittelharte“ Echtzeitsysteme
3. Begriffe: Deadline, eingebettetes System
4. Wenn etwas (umgspr.) *in Echtzeit läuft*, dann ist die Eingangsdatenrate gleich der Ausgangsdatenrate. (Beispiel: Echtzeit-3D-Darstellung)
5. Echtzeitsysteme sind außerordentlich interessant.

Vertiefende Literatur

Die hier aufgeführten Texte sind Lese*vorschläge*. Da diese dem Urheberrecht unterliegen (die Texte, nicht die Vorschläge), kann ich sie nicht einfach ins Netz stellen, sondern stelle sie Ihnen auf Anfrage zur Verfügung.

- John A. Stankovic. “Strategic Directions in Real-Time and Embedded Systems”. In: *ACM Computing Surveys* 28.4 (Dez. 1996), S. 751–763
- John Stankovic. “Misconceptions About Real-Time Computing”. In: *IEEE Computer* 21.10 (Okt. 1988), S. 10–19