Echtzeitsysteme

Struktureller Aufbau von Echtzeitanwendungen

Sommersemester 2024

Peter Wägemann

Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU)
Lehrstuhl Informatik 4 (Systemsoftware)
https://sys.cs.fau.de

Lehrstuhl für Informatik 4
Systemsoftware

Friedrich-Alexander-Universität
Technische Fakultät

Fragestellungen

- Was sind die grundlegenden Bestandteile einer Echtzeitanwendung?
  - Was ist ein **Ereignis**, was eine **Aufgabe** und was ein **Arbeitsauftrag**?

Fragestellungen

- Was sind die grundlegenden Bestandteile einer Echtzeitanwendung?
  - Was ist ein **Ereignis**, was eine **Aufgabe** und was ein **Arbeitsauftrag**?

- Wie bildet man Aufgaben auf das kontrollierende Rechensystem ab?
  - Wie funktioniert die **Ablaufsteuerung**?
  - Welche Kosten verursacht sie?

Fragestellungen

- Was sind die grundlegenden Bestandteile einer Echtzeitanwendung?
  - Was ist ein **Ereignis**, was eine **Aufgabe** und was ein **Arbeitsauftrag**?
- Wie bildet man Aufgaben auf das kontrollierende Rechensystem ab?
  - Wie funktioniert die **Ablaufsteuerung**?
  - Welche Kosten verursacht sie?
- Was ist der zeitliche Zusammenhang zwischen physikalischem Objekt und Echtzeitanwendung?
  - Welche grundlegenden **Zeitparameter** gibt es?
  - Wie hängen Auslösezeit, Termin und Ausführungszeit zusammen?

Fragestellungen

- Was sind die grundlegenden Bestandteile einer Echtzeitanwendung?
  - Was ist ein **Ereignis**, was eine **Aufgabe** und was ein **Arbeitsauftrag**?
- Wie bildet man Aufgaben auf das kontrollierende Rechensystem ab?
  - Wie funktioniert die **Ablaufsteuerung**?
  - Welche Kosten verursacht sie?
- Was ist der zeitliche Zusammenhang zwischen physikalischem Objekt und Echtzeitanwendung?
  - Welche grundlegenden **Zeitparameter** gibt es?
  - Wie hängen Auslösezeit, Termin und Ausführungszeit zusammen?
- Was versteht man unter dem Begriff **Planbarkeit**?
  - Wie stellt man die Rechtzeitigkeit einer Echtzeitanwendung sicher?

Übersicht

1 Einplanungseinheit
- Elemente einer Echtzeitanwendung
- Ausführungsstränge und Arbeitsaufträge

2 Ablaufsteuerung
- Verwaltungsgemeinkosten
- Trennung von Belangen
- Grundsätzliche Verfahren

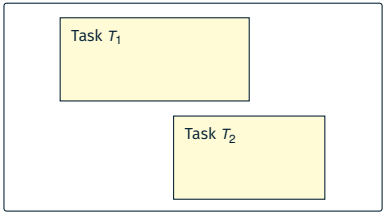
3 Zeitparameter

4 Planbarkeit
- Zulässigkeit und Gültigkeit
- Bewertung von Einplanungsalgorithmen

5 Zusammenfassung

⚠ Strukturelle Elemente

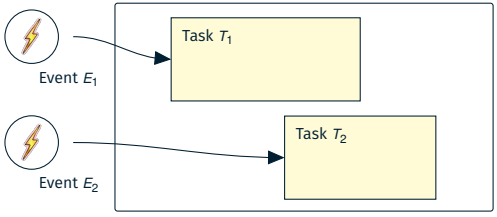
Definitionen siehe [2, S. 75], [3, S. 26], [4, S. 10], [5, S. 13]



- Echtzeitanwendungen bestehen aus *Aufgaben* T_i (engl. *tasks*)

⚠ Strukturelle Elemente

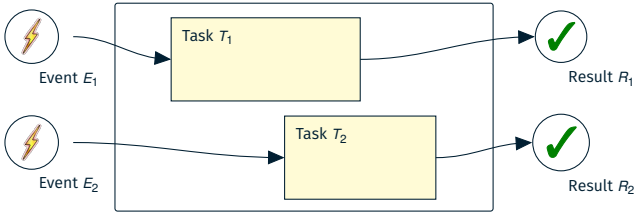
Definitionen siehe [2, S. 75], [3, S. 26], [4, S. 10], [5, S. 13]



- Echtzeitanwendungen bestehen aus *Aufgaben* T_i (engl. *tasks*)
- Aufgaben werden durch *Ereignisse* E_i (engl. *events*) aktiviert

⚠ Strukturelle Elemente

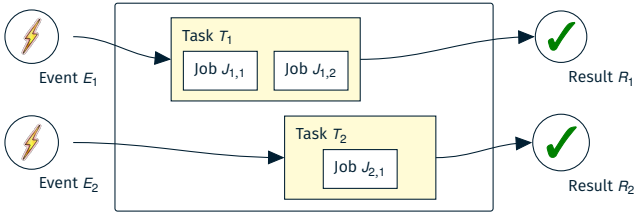
Definitionen siehe [2, S. 75], [3, S. 26], [4, S. 10], [5, S. 13]



- Echtzeitanwendungen bestehen aus *Aufgaben* T_i (engl. *tasks*)
- Aufgaben werden durch *Ereignisse* E_i (engl. *events*) aktiviert
- Aufgaben stellen *Ergebnisse* R_i (engl. *results*) bereit

⚠ Strukturelle Elemente

Definitionen siehe [2, S. 75], [3, S. 26], [4, S. 10], [5, S. 13]



- Echtzeitanwendungen bestehen aus *Aufgaben* T_i (engl. *tasks*)
- Aufgaben werden durch *Ereignisse* E_i (engl. *events*) aktiviert
- Aufgaben stellen *Ergebnisse* R_i (engl. *results*) bereit
- Aus Sicht der Ausführungsumgebung untergliedern sich Aufgaben in ausführbare *Arbeitsaufträge* $J_{i,j}$ (engl. *jobs*)

⚠ Ereignis (engl. *event*)

Ereignis

An ***event*** is a change of state, occurring at an instant. [4, S. 10]

- Ein Ereignis ist …

  bestimmt, falls sein Auftreten als Funktion der physikalischen Zeit beschrieben werden kann, und

  ungewiss, falls dies nicht zutrifft.

⚠ Ereignis (engl. *event*)

Ereignis

An ***event*** is a change of state, occurring at an instant. [4, S. 10]

- Ein Ereignis ist …
  bestimmt, falls sein Auftreten als Funktion der physikalischen Zeit beschrieben werden kann, und
  ungewiss, falls dies nicht zutrifft.
- *Auslöser* (engl. *trigger*) für Ereignisse lassen sich unterscheiden:
  event trigger Ereignisse werden von Zustandsänderungen in der physikalischen Umwelt oder dem kontrollierenden Rechensystem abgeleitet
  time trigger Ereignisse rühren ausschließlich vom Voranschreiten der physikalischen Zeit her[1]

[1] Das Voranschreiten der Zeit stellt natürlich auch eine Zustandsänderung in der Umwelt dar.

⚠ Aufgabe (engl. *task*) und Arbeitsauftrag (engl. *job*)

Aufgabe & Arbeitsauftrag

We call each unit of work that is scheduled and executed by the system a ***job*** and a set of related jobs which jointly provide some system function a ***task***. [3, S. 26]

⚠ Aufgabe (engl. *task*) und Arbeitsauftrag (engl. *job*)

Aufgabe & Arbeitsauftrag

> We call each unit of work that is scheduled and executed by the system a ***job*** and a set of related jobs which jointly provide some system function a ***task***. [3, S. 26]

- Tritt das Ereignis E_i ein und aktiviert die Aufgabe T_i, werden einer oder mehrere Arbeitsaufträge $J_{i,j}$ dieser Aufgabe ausgelöst.
  - → **1:n Beziehung** zwischen Aufgabe und Arbeitsaufträgen
    - Arbeitsaufträge erben die temporalen Eigenschaften der Aufgabe

⚠ Aufgabe (engl. *task*) und Arbeitsauftrag (engl. *job*)

Aufgabe & Arbeitsauftrag

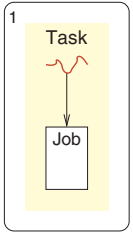
> We call each unit of work that is scheduled and executed by the system a ***job*** and a set of related jobs which jointly provide some system function a ***task***. [3, S. 26]

- Tritt das Ereignis E_i ein und aktiviert die Aufgabe T_i, werden einer oder mehrere Arbeitsaufträge $J_{i,j}$ dieser Aufgabe ausgelöst.
  - → **1:n Beziehung** zwischen Aufgabe und Arbeitsaufträgen
  - Arbeitsaufträge erben die temporalen Eigenschaften der Aufgabe

⚠ Die Aufgabe ist die Entwurfseinheit, der Arbeitsauftrag die Einplanungseinheit

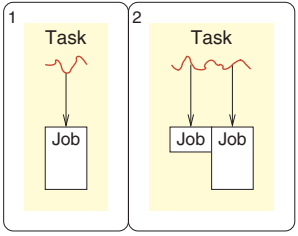
- Arbeitsaufträge unterliegen der Ablaufplanung
- Eigenschaften dieser Planung können sich zur Laufzeit dynamisch ändern

⚠ Aufgabe vs. Arbeitsauftrag vs. Faden (engl. *thread*)



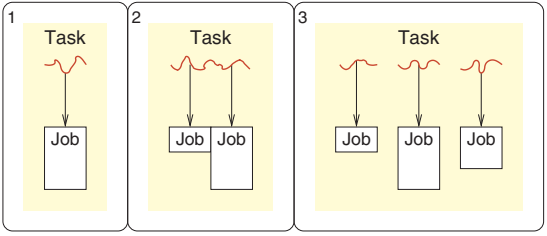
- Abbildung auf *Fäden* (engl. *threads*) des Betriebssystems
  1. Einfädige Aufgabe, ein Arbeitsauftrag

⚠ Aufgabe vs. Arbeitsauftrag vs. Faden (engl. *thread*)



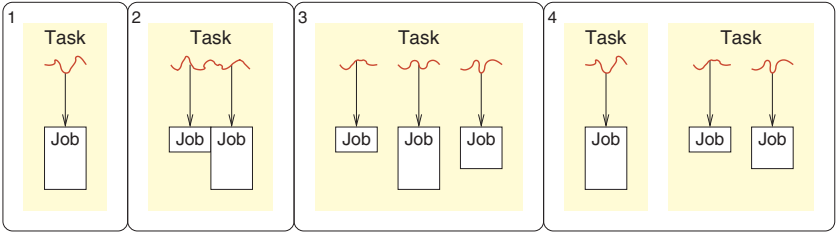
- Abbildung auf *Fäden* (engl. *threads*) des Betriebssystems
  1. Einfädige Aufgabe, ein Arbeitsauftrag
  2. Einfädige Aufgabe, zwei Arbeitsaufträge

⚠ Aufgabe vs. Arbeitsauftrag vs. Faden (engl. *thread*)



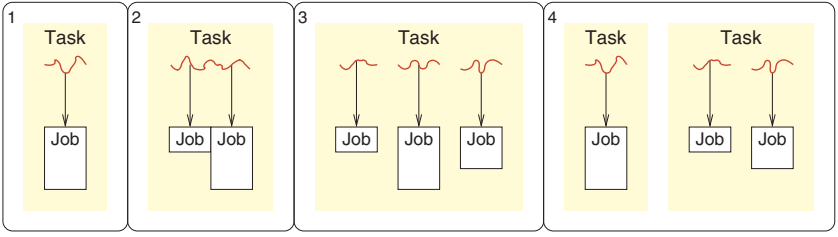
- Abbildung auf *Fäden* (engl. *threads*) des Betriebssystems
  1. Einfädige Aufgabe, ein Arbeitsauftrag
  2. Einfädige Aufgabe, zwei Arbeitsaufträge
  3. Mehrfädige Aufgabe, drei Arbeitsaufträge

⚠ Aufgabe vs. Arbeitsauftrag vs. Faden (engl. *thread*)



- Abbildung auf *Fäden* (engl. *threads*) des Betriebssystems
  1. Einfädige Aufgabe, ein Arbeitsauftrag
  2. Einfädige Aufgabe, zwei Arbeitsaufträge
  3. Mehrfädige Aufgabe, drei Arbeitsaufträge
  4. Zwei Aufgaben (ein- und mehrfädig), drei Arbeitsaufträge

⚠ Aufgabe vs. Arbeitsauftrag vs. Faden (engl. *thread*)



- Abbildung auf *Fäden* (engl. *threads*) des Betriebssystems
  1. Einfädige Aufgabe, ein Arbeitsauftrag ← **Regelfall!**
  2. Einfädige Aufgabe, zwei Arbeitsaufträge
  3. Mehrfädige Aufgabe, drei Arbeitsaufträge
  4. Zwei Aufgaben (ein- und mehrfädig), drei Arbeitsaufträge

⚠ **1:n:m Beziehung** (Aufgabe:Arbeitsaufträge:Fäden)

Übersicht

1 Einplanungseinheit
- Elemente einer Echtzeitanwendung
- Ausführungsstränge und Arbeitsaufträge

2 Ablaufsteuerung
- Verwaltungsgemeinkosten
- Trennung von Belangen
- Grundsätzliche Verfahren

3 Zeitparameter

4 Planbarkeit
- Zulässigkeit und Gültigkeit
- Bewertung von Einplanungsalgorithmen

5 Zusammenfassung

Ablaufsteuerung als zweiphasiger Prozess

Ablaufsteuerung

Wann wird welcher Arbeitsauftrag auf welcher Recheneinheit ausgeführt?

Ablaufsteuerung als zweiphasiger Prozess

Ablaufsteuerung

Wann wird welcher Arbeitsauftrag auf welcher Recheneinheit ausgeführt?

- Ziel ist die rechtzeitige Fertigstellung der Arbeitsaufträge
  - Einhaltung ihrer jeweiligen Termine (s. Folie 22)

Ablaufsteuerung als zweiphasiger Prozess

Ablaufsteuerung

Wann wird welcher Arbeitsauftrag auf welcher Recheneinheit ausgeführt?

- Ziel ist die rechtzeitige Fertigstellung der Arbeitsaufträge
  - Einhaltung ihrer jeweiligen Termine (s. Folie 22)

- **Phase 1:** Aufgaben/Arbeitsaufträge $\mapsto$ Fäden (1:n:m)
  - $\rightarrow$ Entspricht einer **räumlichen Allokation** von Betriebsmitteln (Fäden)
  - Abbildung erfolgt statisch zum Entwurfszeitpunkt
  - Impliziert ggf. Latenzen durch Sequentialisierung
  - 1:1-Abbildung von Arbeitsaufträgen auf Fäden typisch

Ablaufsteuerung als zweiphasiger Prozess (Forts.)

- **Phase 2:** Fäden $\mapsto$ Prozessor/Kerne (m:l)
  - $\rightarrow$ **Zeitliche Einplanung** (engl. scheduling) von Fäden
    - Nutzung des Prozessors im *zeitlichen Mehrfachbetrieb* (engl. *temporal multiplexing*) (typ. #Fäden >> #Prozessoren/Kerne)
    - Abbildung erfolgt **offline** oder **online**
      - Offline $\rightsquigarrow$ der komplette Ablauf wird vorab festgelegt
      - Online $\rightsquigarrow$ der Prozessor wird zur Laufzeit zugeteilt

Ablaufsteuerung als zweiphasiger Prozess (Forts.)

- **Phase 2:** Fäden $\mapsto$ Prozessor/Kerne (m:l)
  - $\rightarrow$ **Zeitliche Einplanung** (engl. scheduling) von Fäden
    - Nutzung des Prozessors im *zeitlichen Mehrfachbetrieb* (engl. *temporal multiplexing*) (typ. #Fäden >> #Prozessoren/Kerne)
    - Abbildung erfolgt **offline** oder **online**
      - Offline $\rightsquigarrow$ der komplette Ablauf wird vorab festgelegt
      - Online $\rightsquigarrow$ der Prozessor wird zur Laufzeit zugeteilt

⚠ Zeitlicher Mehrfachbetrieb impliziert ggf. den dynamischen Wechsel zwischen verschiedenen Fäden $\rightsquigarrow$ **zusätzlicher Aufwand**

- Mechanismus zum Abfertigen einzelner Fäden (vgl. Ausnahmebehandlung Folie III-1/16 ff)
- Strategie zur Auswahl des nächsten laufbereiten Fadens

Ablaufsteuerung als zweiphasiger Prozess (Forts.)

- **Phase 2:** Fäden $\mapsto$ Prozessor/Kerne (m:l)
  - $\rightarrow$ **Zeitliche Einplanung** (engl. scheduling) von Fäden
    - Nutzung des Prozessors im *zeitlichen Mehrfachbetrieb* (engl. *temporal multiplexing*) (typ. #Fäden >> #Prozessoren/Kerne)
    - Abbildung erfolgt **offline** oder **online**
      - Offline $\rightsquigarrow$ der komplette Ablauf wird vorab festgelegt
      - Online $\rightsquigarrow$ der Prozessor wird zur Laufzeit zugeteilt

⚠ Zeitlicher Mehrfachbetrieb impliziert ggf. den dynamischen Wechsel zwischen verschiedenen Fäden $\rightsquigarrow$ **zusätzlicher Aufwand**

- Mechanismus zum Abfertigen einzelner Fäden (vgl. Ausnahmebehandlung Folie III-1/16 ff)
- Strategie zur Auswahl des nächsten laufbereiten Fadens

- In Echtzeitsystemen **nicht vernachlässigbar**!

Faden in Ausführung

Einheit der Einplanung (engl. *unit of scheduling*)

- Abstraktion (des Betriebssystems) von einem beliebigen Programm/Arbeitsauftrag in Ausführung ist der Prozess
  - Je Prozess existieren ggf. mehrere Fäden gleichzeitig
- Kontext eines Fadens manifestiert sich im Prozessorstatus
  - Inhalte der Arbeits- und Statusregister der CPU
  - ggf. erweitert um Segment-/Seitendeskriptoren von MMU bzw. TLB

[2] Ggf. lediglich „Prozeduraktivierung“ eines übergeordneten Programms, bei kooperativer Einplanung und Verzicht auf Synchronisationspunkte

Faden in Ausführung

Einheit der Einplanung (engl. *unit of scheduling*)

- Abstraktion (des Betriebssystems) von einem beliebigen Programm/Arbeitsauftrag in Ausführung ist der Prozess
  - Je Prozess existieren ggf. mehrere Fäden gleichzeitig
- Kontext eines Fadens manifestiert sich im Prozessorstatus
  - Inhalte der Arbeits- und Statusregister der CPU
  - ggf. erweitert um Segment-/Seitendeskriptoren von MMU bzw. TLB

⚠ *Einlastung* (engl. *dispatching*) eines Fadens ⇝ **Kontextwechsel** (vgl. Unterbrechungsbehandlung III-1/18)

[2] Ggf. lediglich „Prozeduraktivierung“ eines übergeordneten Programms, bei kooperativer Einplanung und Verzicht auf Synchronisationspunkte

Faden in Ausführung

Einheit der Einplanung (engl. *unit of scheduling*)

- Abstraktion (des Betriebssystems) von einem beliebigen Programm/Arbeitsauftrag in Ausführung ist der Prozess
  - Je Prozess existieren ggf. mehrere Fäden gleichzeitig
- Kontext eines Fadens manifestiert sich im Prozessorstatus
  - Inhalte der Arbeits- und Statusregister der CPU
  - ggf. erweitert um Segment-/Seitendeskriptoren von MMU bzw. TLB

⚠ *Einlastung* (engl. *dispatching*) eines Fadens $\rightsquigarrow$ **Kontextwechsel** (vgl. Unterbrechungsbehandlung III-1/18)

Prozessinstanz des Betriebssystems

Ein- oder mehrfädiges Programm, mit oder ohne eigenem Adressraum[2]

[2] Ggf. lediglich „Prozeduraktivierung“ eines übergeordneten Programms, bei kooperativer Einplanung und Verzicht auf Synchronisationspunkte

Verwaltungsgemeinkosten (engl. *overhead*)

Unterschiede[3] je nach Repräsentation von Aufgabe und Arbeitsauftrag

1. Einfädige Aufgabe $\leadsto$ fliegengewichtig
   - $\mathcal{O}(\textit{Prozeduraufruf})$
   - Auf- und Abbau vom Aktivierungsblock (engl. *activation record*)

[3] Die Landau-Notation dient hier zur Illustration der Laufzeitkomplexität/-kosten.

Verwaltungsgemeinkosten (engl. *overhead*)

Unterschiede[3] je nach Repräsentation von Aufgabe und Arbeitsauftrag

1. Einfädige Aufgabe $\rightsquigarrow$ fliegengewichtig
   - $\mathcal{O}(\mathit{Prozeduraufruf})$
   - Auf- und Abbau vom Aktivierungsblock (engl. *activation record*)
2. Mehrfädige Aufgabe

[3] Die Landau-Notation dient hier zur Illustration der Laufzeitkomplexität/-kosten.

Verwaltungsgemeinkosten (engl. *overhead*)

Unterschiede[3] je nach Repräsentation von Aufgabe und Arbeitsauftrag

1. Einfädige Aufgabe $\rightsquigarrow$ fliegengewichtig
   - $\mathcal{O}(\textit{Prozeduraufruf})$
   - Auf- und Abbau vom Aktivierungsblock (engl. *activation record*)
2. Mehrfädige Aufgabe
   2.1 Gemeinsamer Adressraum $\rightsquigarrow$ **federgewichtig**
   - $\mathcal{O}(\textit{Fadenwechsel}) + \mathcal{O}(1.)$
   - Austausch der Inhalte von Arbeits-/Statusregister der CPU

[3] Die Landau-Notation dient hier zur Illustration der Laufzeitkomplexität/-kosten.

Verwaltungsgemeinkosten (engl. *overhead*)

Unterschiede[3] je nach Repräsentation von Aufgabe und Arbeitsauftrag

1. Einfädige Aufgabe $\rightsquigarrow$ fliegengewichtig
   - $\mathcal{O}(\textit{Prozeduraufruf})$
   - Auf- und Abbau vom Aktivierungsblock (engl. *activation record*)
2. Mehrfädige Aufgabe
   - 2.1 Gemeinsamer Adressraum $\rightsquigarrow$ **federgewichtig**
     - $\mathcal{O}(\textit{Fadenwechsel}) + \mathcal{O}(1.)$
     - Austausch der Inhalte von Arbeits-/Statusregister der CPU
   - 2.2 Separierter Betriebssystemkern $\rightsquigarrow$ **leichtgewichtig**
     - $\mathcal{O}(\textit{Systemaufruf}) + \mathcal{O}(2.1.)$
     - Behandlung der synchronen Programmunterbrechung (engl. *trap*)

[3] Die Landau-Notation dient hier zur Illustration der Laufzeitkomplexität/-kosten.

Verwaltungsgemeinkosten (engl. *overhead*)

Unterschiede[3] je nach Repräsentation von Aufgabe und Arbeitsauftrag

1. Einfädige Aufgabe $\rightsquigarrow$ fliegengewichtig
   - $\mathcal{O}(\textit{Prozeduraufruf})$
   - Auf- und Abbau vom Aktivierungsblock (engl. *activation record*)
2. Mehrfädige Aufgabe
   - 2.1 Gemeinsamer Adressraum $\rightsquigarrow$ **federgewichtig**
     - $\mathcal{O}(\textit{Fadenwechsel}) + \mathcal{O}(1.)$
     - Austausch der Inhalte von Arbeits-/Statusregister der CPU
   - 2.2 Separierter Betriebssystemkern $\rightsquigarrow$ **leichtgewichtig**
     - $\mathcal{O}(\textit{Systemaufruf}) + \mathcal{O}(2.1.)$
     - Behandlung der synchronen Programmunterbrechung (engl. *trap*)
   - 2.3 getrennte Adressräume $\rightsquigarrow$ **schwergewichtig**
     - $\mathcal{O}(\textit{Adressraumwechsel}) + \mathcal{O}(2.2.)$
     - Löschen/Laden vom Zwischenspeicher (engl. *cache*) der **MMU**

[3] Die Landau-Notation dient hier zur Illustration der Laufzeitkomplexität/-kosten.

Verdrängbare Aufgabe

Unterbrechung & Wiederaufnahme eines Arbeitsauftrags

⚠ Laufende Arbeitsaufträge können ***Verdrängung* (engl. *preemption*)** erleiden

→ Dem Faden des laufenden Arbeitsauftrags wird die CPU entzogen

1. Asynchrone Programmunterbrechung tritt auf (vgl. III-1/12 ff)
2. Unterbrochene Faden wird als laufbereit (erneut) eingeplant
3. Ein (anderer) laufbereiter Faden wird ausgewählt und eingelastet

Verdrängbare Aufgabe

Unterbrechung & Wiederaufnahme eines Arbeitsauftrags

⚠ Laufende Arbeitsaufträge können ***Verdrängung* (engl. *preemption*)** erleiden

→ Dem Faden des laufenden Arbeitsauftrags wird die CPU entzogen

1. Asynchrone Programmunterbrechung tritt auf (vgl. III-1/12 ff)
2. Unterbrochene Faden wird als laufbereit (erneut) eingeplant
3. Ein (anderer) laufbereiter Faden wird ausgewählt und eingelastet

- Verdrängung ist eine **Systemfunktion** mit Nebenbedingungen:
  - Asynchrone Programmunterbrechungen sind möglich
  - Behandlungsroutine aktiviert den *Planer* (engl. *scheduler*)
  - Planer muss verdrängend arbeiten (engl. *preemptive scheduling*)
  - Mindestens ein (anderer) laufbereiter Faden steht zur Verfügung

Verdrängbare Aufgabe

Unterbrechung & Wiederaufnahme eines Arbeitsauftrags

⚠ Laufende Arbeitsaufträge können ***Verdrängung* (engl. *preemption*)** erleiden

→ Dem Faden des laufenden Arbeitsauftrags wird die CPU entzogen

1. Asynchrone Programmunterbrechung tritt auf (vgl. III-1/12 ff)
2. Unterbrochene Faden wird als laufbereit (erneut) eingeplant
3. Ein (anderer) laufbereiter Faden wird ausgewählt und eingelastet

- Verdrängung ist eine **Systemfunktion** mit Nebenbedingungen:
  - Asynchrone Programmunterbrechungen sind möglich
  - Behandlungsroutine aktiviert den *Planer* (engl. *scheduler*)
  - Planer muss verdrängend arbeiten (engl. *preemptive scheduling*)
  - Mindestens ein (anderer) laufbereiter Faden steht zur Verfügung

- Verdrängung ist eine nicht-funktionale Systemeigenschaft (vgl. III-1/2)
  - Impliziert an anderen Programmstellen **Synchronisationsbedarf**
  - Muss **transparent** für die betroffene Aufgabe sein (siehe Folie 13)

Verdrängbare Aufgabe (Forts.)

- *Transparenz* (engl. *transparency*) von Verdrängung bedingt:

Verdrängbare Aufgabe (Forts.)

- *Transparenz* (engl. *transparency*) von Verdrängung bedingt:
  1. Zustand eines unterbrochenen Fadens ist **invariant**
     - → Sicherung und Wiederherstellung des Prozessorstatus
     - – Maßnahmen zur **Einlastung** **(engl. *dispatching*)** von Fäden

Verdrängbare Aufgabe (Forts.)

- *Transparenz* (engl. *transparency*) von Verdrängung bedingt:
  1. Zustand eines unterbrochenen Fadens ist **invariant**
     - → Sicherung und Wiederherstellung des Prozessorstatus
     - – Maßnahmen zur **Einlastung** **(engl. *dispatching*)** von Fäden
  2. Verzögerte Ausführung des Fadens verletzt **keine Termine**
     - → Vergabe, Überwachung und Einhaltung von Fristen (engl. *deadline monitoring*)
     - – Zuordnung von statischer/dynamischer *Priorität* (engl. *priority*)
     - – Maßnahmen zur **Einplanung** **(engl. *scheduling*)** von Fäden

Verdrängbare Aufgabe (Forts.)

- *Transparenz* (engl. *transparency*) von Verdrängung bedingt:
  1. Zustand eines unterbrochenen Fadens ist **invariant**
     - → Sicherung und Wiederherstellung des Prozessorstatus
     - – Maßnahmen zur **Einlastung (engl. *dispatching*)** von Fäden
  2. Verzögerte Ausführung des Fadens verletzt **keine Termine**
     - → Vergabe, Überwachung und Einhaltung von Fristen (engl. *deadline monitoring*)
     - – Zuordnung von statischer/dynamischer *Priorität* (engl. *priority*)
     - – Maßnahmen zur **Einplanung (engl. *scheduling*)** von Fäden

Kostenminimierung (Komplexität von Aufgaben siehe Folie 23)

- Einfache nicht-verdrängbare Aufgaben können auf 1. verzichten
  - Einfach verdrängbare oder komplexe jedoch nicht
- Echtzeitrechensysteme können auf 2. ggf. sogar verzichten
  - Sofern die Umgebung keine harten Echtzeitbedingungen vorgibt

- ***Einplanung* (engl. *scheduling*) ↦ Strategie**
  - → Festlegung einer Einlastungsreihenfolge
  - Erstellung des Ablaufplans von Arbeitsaufträgen
  - In Bezug zur Aufgabenbearbeitung:

    entkoppelt (engl. *off-line*) ⇝ zur Entwurfszeit

    gekoppelt (engl. *on-line*) ⇝ zur Laufzeit[4]

[4] Vorlage ist ggf. ein vor Beginn der Aufgabenbearbeitung offline erstellter Ablaufplan, der während der Aufgabenbearbeitung online fortgeschrieben wird.

- ***Einplanung* (engl. *scheduling*) ↦ Strategie**
  - → Festlegung einer Einlastungsreihenfolge
  - Erstellung des Ablaufplans von Arbeitsaufträgen
  - In Bezug zur Aufgabenbearbeitung:

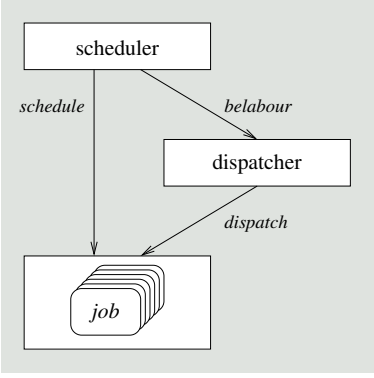
    entkoppelt (engl. *off-line*) ⇝ zur Entwurfszeit

    gekoppelt (engl. *on-line*) ⇝ zur Laufzeit[4]

- ***Einlastung* (engl. *dispatching*) ↦ Mechanismus**
  - → Umsetzung der Einplanungsentscheidungen
  - Abarbeitung des Ablaufplans von Arbeitsaufträgen
  - Ist immer gekoppelt mit der Aufgabenbearbeitung
    - Ablaufpläne können nur *online* befolgt werden

[4] Vorlage ist ggf. ein vor Beginn der Aufgabenbearbeitung offline erstellter Ablaufplan, der während der Aufgabenbearbeitung online fortgeschrieben wird.

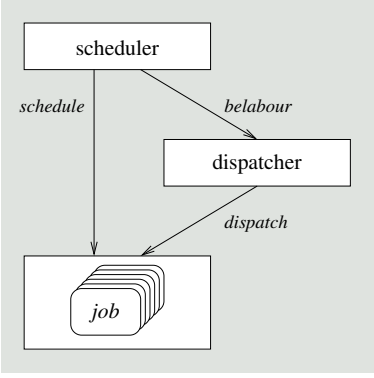
Einplanung und Einlastung



Gekoppeltes System

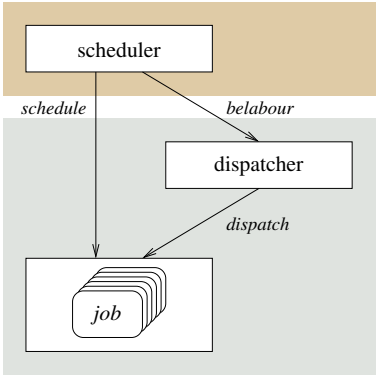
- Zeit- und örtlich gekoppelt
  - Zur Laufzeit
  - Integriert in einem System
    - Auf *einem* Rechner

Einplanung und Einlastung



Gekoppeltes System

- Zeit- und örtlich gekoppelt
  - Zur Laufzeit
  - Integriert in einem System
    - Auf *einem* Rechner



Entkoppeltes System

- Zeit- oder örtlich entkoppelt
  - Vor und zur Laufzeit
  - Separiert in zwei Systeme
    - Ggf. auf zwei Rechner

⚠ Einplanungszeitpunkte

- **On-line scheduling** (zur Laufzeit) $\rightsquigarrow$ *kein à priori Wissen* nötig
  - Einzige Option bei *unbekannter* zukünftiger Auslastung
    - Lastparameter sind erst zur Joblaufzeit bekannt
  - Getroffenen Entscheidungen sind häufig nur suboptimal
    - Eingeschränkte Fähigkeit, Betriebsmittel maximal zu nutzen
  - Ermöglicht/unterstützt jedoch ein **flexibles System**

- **On-line scheduling** (zur Laufzeit) $\rightsquigarrow$ *kein à priori Wissen* nötig
  - Einzige Option bei *unbekannter* zukünftiger Auslastung
    - Lastparameter sind erst zur Joblaufzeit bekannt
  - Getroffenen Entscheidungen sind häufig nur suboptimal
    - Eingeschränkte Fähigkeit, Betriebsmittel maximal zu nutzen
  - Ermöglicht/unterstützt jedoch ein **flexibles System**
- **Off-line scheduling** (zur Entwurfszeit) $\rightsquigarrow$ *à priori Wissen* nötig
  - Voraussetzung ist ein vollständig bekanntes System, d.h.:
    - Lastparameter sind vor Joblaufzeit vollständig bekannt
    - Ein fester Satz von Systemfunktionen ist gegeben
  - **Zur Laufzeit ist Lösung von NP-schweren Problemen unpraktikabel**

    $\rightarrow$ Also einen Ablaufplan zu finden, der alle Task/Job-Fristen einhält
  - Änderungen am System bedeuten **Neuberechnung** des Ablaufplans
    - Gilt für *alle* Änderungen/Rekonfigurationen an Software *und* Hardware

Zeitgesteuert (engl. *time-triggered*, auch *time-driven*) ✓

- Einlastung nur zu *festen Zeitpunkten*
  - → Vorgegeben durch das *Echtzeitrechensystem*
- Offline (entkoppelte) Einplanung

Grundsätzliche Verfahren

Vorangetrieben durch interne oder externe Ereignisse

Zeitgesteuert (engl. *time-triggered*, auch *time-driven*) ✓

- Einlastung nur zu *festen Zeitpunkten*
  - → Vorgegeben durch das *Echtzeitrechensystem*
- Offline (entkoppelte) Einplanung

Ereignisgesteuert (engl. *event-triggered*, auch *event-driven*) ✓

- Einlastung zu Ereigniszeitpunkten
  - → Vorgegeben durch das *kontrollierte Objekt*
- Online (gekoppelte) Einplanung

Grundsätzliche Verfahren

Vorangetrieben durch interne oder externe Ereignisse

Zeitgesteuert (engl. *time-triggered*, auch *time-driven*) ✓

- Einlastung nur zu *festen Zeitpunkten*
  - → Vorgegeben durch das *Echtzeitrechensystem*
- Offline (entkoppelte) Einplanung

Reihum gewichtet (engl. *weighted round robin*)

- Echtzeitverkehr in Hochgeschwindigkeitsnetzen
  - im Koppelnetz (engl. *switched network*)
- Untypisch für die Einplanung von Arbeitsaufträgen

Ereignisgesteuert (engl. *event-triggered*, auch *event-driven*) ✓

- Einlastung zu Ereigniszeitpunkten
  - → Vorgegeben durch das *kontrollierte Objekt*
- Online (gekoppelte) Einplanung

⚠ Taktsteuerung

Zeitgesteuertes (engl. *time-triggered*) Echtzeitsystem

⚠ Einlastungszeitpunkte von Arbeitsaufträgen *à priori* bestimmt

- Alle Parameter aller Arbeitsaufträge sind *off-line* bekannt
- WCET, Betriebsmittelbedarf (z.B. Speicher, Fäden, Energie), …

- Overheads zur Laufzeit sind **minimal**

⚠ Einlastungszeitpunkte von Arbeitsaufträgen *à priori* bestimmt
- Alle Parameter aller Arbeitsaufträge sind *off-line* bekannt
- WCET, Betriebsmittelbedarf (z.B. Speicher, Fäden, Energie), …

- Overheads zur Laufzeit sind **minimal**

- Einlastung erfolgt in **festen Intervallen**
  - *Variables Intervall* durch *Zeitgeber* (engl. *timer*) mit der Länge des jeweils einzulastenden Arbeitsauftrags programmiert ↦ WCET (siehe Kapitel III-3)
    - Jeder Zeitablauf bewirkt eine asynchrone Programmunterbrechung
    - Als Folge findet die Einlastung des nächsten Arbeitsauftrags statt
  - *Festes Intervall* mittels regelmäßiger Unterbrechungen (Zeitgeber)
    - Festes Zeitraster liegt über die Ausführung der Arbeitsaufträge
    - Dient z.B. dem Abfragen (engl. *polling*) von Sensoren/Geräten

⚠ Einplanung und Einlastungszeitpunkte **vorab nicht bekannt**

- Asynchrone Programmunterbrechungen: Hardwareereignisse
  - Zeitsignal, Bereitstellung von Sensordaten, Beendigung von E/A
- Synchronisationspunkte: ein-/mehrseitige Synchronisation
  - Schlossvariable, Semaphor, Monitor

⚠ Vorrangsteuerung

Ereignisgesteuertes (engl. *event-triggered*) Echtzeitsystem

⚠ Einplanung und Einlastungszeitpunkte **vorab nicht bekannt**

- Asynchrone Programmunterbrechungen: Hardwareereignisse
  - Zeitsignal, Bereitstellung von Sensordaten, Beendigung von E/A
- Synchronisationspunkte: ein-/mehrseitige Synchronisation
  - Schlossvariable, Semaphor, Monitor

- Ereignisse haben **Prioritäten**
- Prioritäten $\mapsto$ **zeitliche Dringlichkeit** ($\neq$ *Kritikalität* [1], siehe VL 4-2)
  - Zuteilung von Betriebsmitteln erfolgt **prioritätsorientiert**
    - Arbeitsaufträge höherer Priorität haben Vorrang
  - Prioritäten werden *offline* vergeben und ggf. *online* fortgeschrieben
    - Arbeitsaufträge haben eine statische oder dynamische Priorität
  - Betriebsmittel (insb. CPU) bleiben niemals absichtlich ungenutzt
    - Im Gegensatz zur Taktsteuerung, die Betriebsmittel brach liegen lässt

Übersicht

1 Einplanungseinheit
- Elemente einer Echtzeitanwendung
- Ausführungsstränge und Arbeitsaufträge

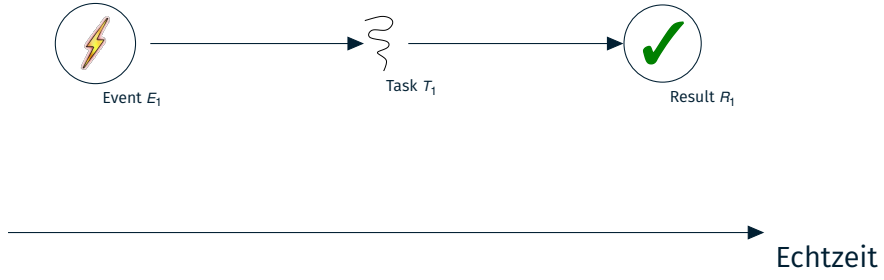
2 Ablaufsteuerung
- Verwaltungsgemeinkosten
- Trennung von Belangen
- Grundsätzliche Verfahren

3 Zeitparameter

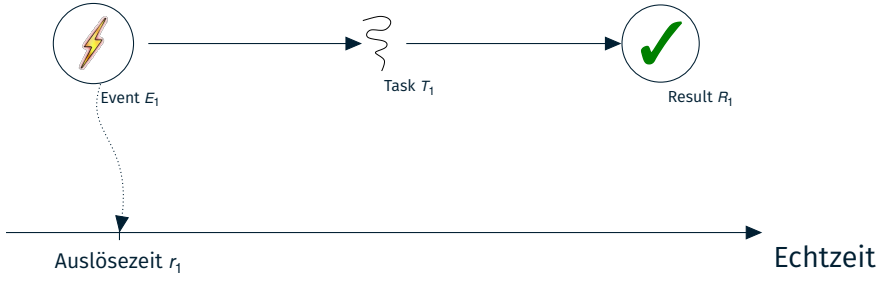
4 Planbarkeit
- Zulässigkeit und Gültigkeit
- Bewertung von Einplanungsalgorithmen

5 Zusammenfassung

Punkte auf der Echtzeitachse

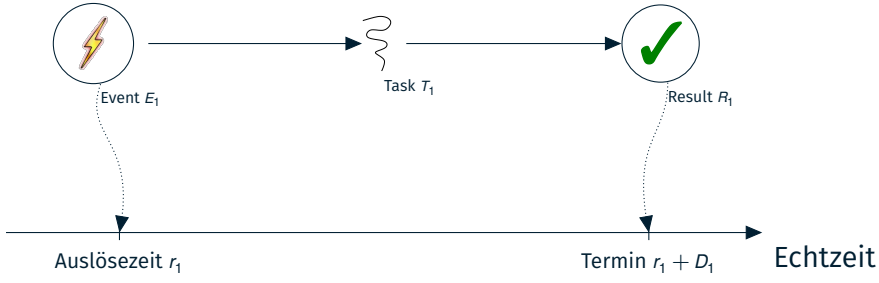


Punkte auf der Echtzeitachse



- Das Ereignis E_i löst zum **Auslösezeitpunkt** r_i den Arbeitsauftrag $J_{i,j}$ der Aufgabe T_i aus

Punkte auf der Echtzeitachse



- Das Ereignis E_i löst zum **Auslösezeitpunkt** r_i den Arbeitsauftrag $J_{i,j}$ der Aufgabe T_i aus
- Das Ergebnis R_i muss bis zum **Termin** D_i vorliegen

- ***Auslösezeit*** r_i **(engl. *release time*)** $\mapsto$ Arbeitsauftrag steht zur Ausführung bereit
  - Damit ist die Einlastung des betreffenden Auftrags möglich
    - Voraussetzung: Abhängigkeitsbedingungen[5] sind erfüllt
  - Ggf. verzögert Einplanung/Koordinierung die Einlastung des Jobs

[5] Daten-/Kontrollfluss bzgl. kontrolliertem Objekt/anderer Arbeitsaufträge

- ***Auslösezeit*** r_i **(engl. *release time*)** $\mapsto$ Arbeitsauftrag steht zur Ausführung bereit
  - Damit ist die Einlastung des betreffenden Auftrags möglich
    - Voraussetzung: Abhängigkeitsbedingungen[5] sind erfüllt
  - Ggf. verzögert Einplanung/Koordinierung die Einlastung des Jobs
- *Termin* D_i (engl. *deadline*) $\mapsto$ Arbeitsauftrag soll/muss seine Ausführung beendet haben
  - Differenzierung nach der Art seines Bezugszeitpunktes

    relativ (engl. *relative deadline*) zur Auslösezeit oder

    absolut (engl. *absolute deadline*) als Echtzeit

    $\rightarrow$ absoluter Termin $=$ Auslösezeit $r_i +$ relativer Termin D_i
  - Je nach Anforderung weich, fest oder hart (vgl. Folie II/12)
    - Wert ∞ gibt keine Frist für den betreffenden Auftrag vor

[5] Daten-/Kontrollfluss bzgl. kontrolliertem Objekt/anderer Arbeitsaufträge

Einschub: Einfache und komplexe Aufgaben

- Einfache Aufgabe (engl. *simple task*)
  - → Ohne Synchronisationspunkt
    - Ausführung ist blockadefrei
    - Unabhängig vom Fortschritt anderer Aufgaben

Einschub: Einfache und komplexe Aufgaben

- Einfache Aufgabe (engl. *simple task*)
  - → Ohne Synchronisationspunkt
    - Ausführung ist blockadefrei
    - Unabhängig vom Fortschritt anderer Aufgaben
- Kann lokal betrachtet werden
  - Keine Beeinflussung von oder durch andere Aufgaben

Einschub: Einfache und komplexe Aufgaben

- Einfache Aufgabe (engl. *simple task*)
  - → Ohne Synchronisationspunkt
  - Ausführung ist blockadefrei
  - Unabhängig vom Fortschritt anderer Aufgaben
- Kann lokal betrachtet werden
  - Keine Beeinflussung von oder durch andere Aufgaben
- Komplexe Aufgabe (engl. *complex task*)
  - → Mit Synchronisationspunkt(en)
  - Benötigt Betriebsmitteln (BM) (neben der CPU)
  - Fortschritt hängt von anderer Aufgaben ab

Einschub: Einfache und komplexe Aufgaben

- Einfache Aufgabe (engl. *simple task*)
  - → Ohne Synchronisationspunkt
  - Ausführung ist blockadefrei
  - Unabhängig vom Fortschritt anderer Aufgaben
- Kann lokal betrachtet werden
  - Keine Beeinflussung von oder durch andere Aufgaben
- Komplexe Aufgabe (engl. *complex task*)
  - → Mit Synchronisationspunkt(en)
  - Benötigt Betriebsmitteln (BM) (neben der CPU)
  - Fortschritt hängt von anderer Aufgaben ab

⚠ Muss global betrachtet werden ⇝ **systemweite Analyse erforderlich**

- Analyse nur im Verbund mit allen Aufgaben des Systems

Jobphasen auf der Echtzeitachse

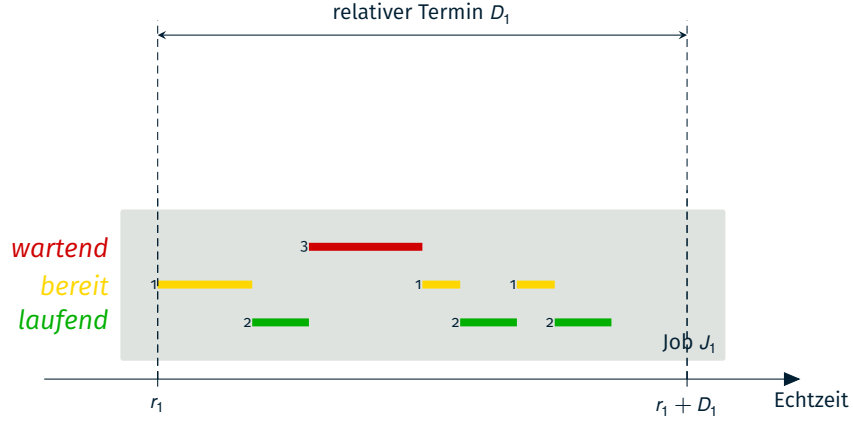
Arbeitsauftrag einer komplexen Aufgabe

- (1) Einplanung, (2) Einlastung, (3) Synchronisation

Jobphasen auf der Echtzeitachse

Arbeitsauftrag einer komplexen Aufgabe

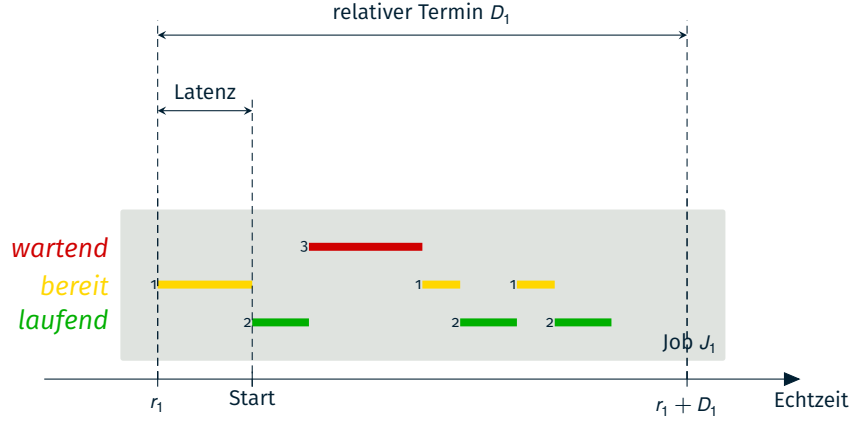
- (1) Einplanung, (2) Einlastung, (3) Synchronisation



Jobphasen auf der Echtzeitachse

Arbeitsauftrag einer komplexen Aufgabe

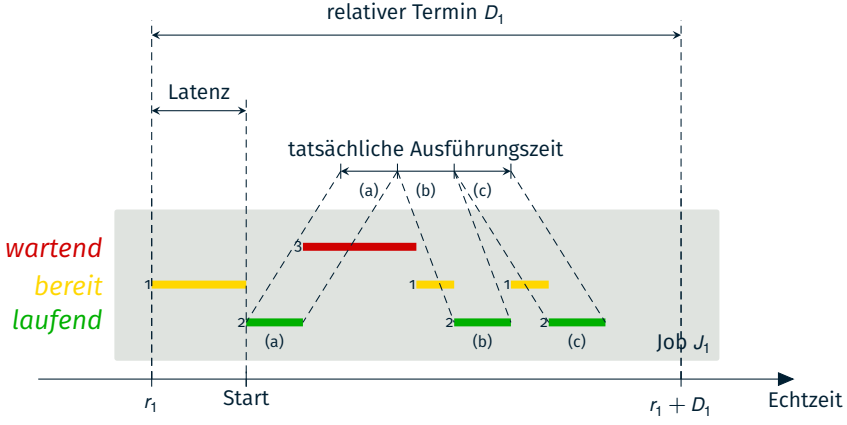
- (1) Einplanung, (2) Einlastung, (3) Synchronisation



Jobphasen auf der Echtzeitachse

Arbeitsauftrag einer komplexen Aufgabe

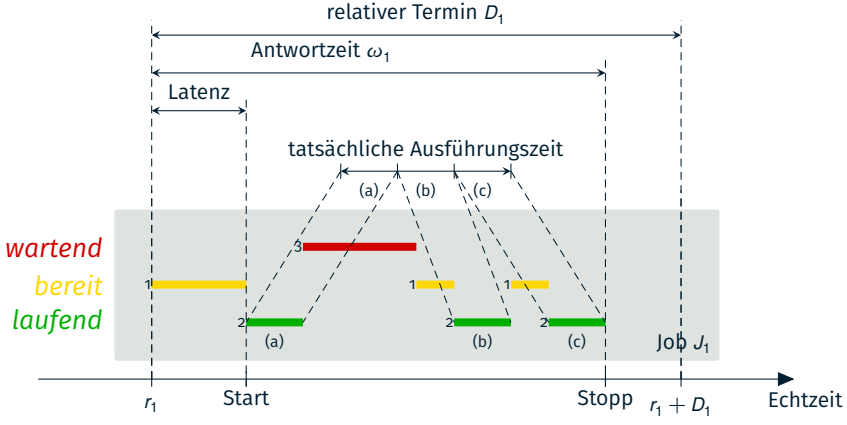
- (1) Einplanung, (2) Einlastung, (3) Synchronisation



Jobphasen auf der Echtzeitachse

Arbeitsauftrag einer komplexen Aufgabe

- (1) Einplanung, (2) Einlastung, (3) Synchronisation



Intervalle auf der Echtzeitachse

- *Latenz* (engl. *latency*) ↦ Zeitdauer zwischen Auslösezeit und Beginn der Abarbeitung

Intervalle auf der Echtzeitachse

- *Latenz* (engl. *latency*) ↦ Zeitdauer zwischen Auslösezeit und Beginn der Abarbeitung
- *Tatsächliche Ausführungszeit* (engl. *elapsed time*) ↦ durch den ausgeführten Arbeitsauftrag beanspruchte Rechenzeit
  - Wird durch die **maximale Ausführungszeit** e_i **(engl. *worst-case execution time*, WCET)** der Aufgabe T_i beschränkt
  - Die WCET ist prinzipiell unabhängig von anderen Arbeitsaufträgen

Intervalle auf der Echtzeitachse

- *Latenz* (engl. *latency*) $\mapsto$ Zeitdauer zwischen Auslösezeit und Beginn der Abarbeitung
- *Tatsächliche Ausführungszeit* (engl. *elapsed time*) $\mapsto$ durch den ausgeführten Arbeitsauftrag beanspruchte Rechenzeit
  - Wird durch die **maximale Ausführungszeit** e_i **(engl. *worst-case execution time*, WCET)** der Aufgabe T_i beschränkt
  - Die WCET ist prinzipiell unabhängig von anderen Arbeitsaufträgen
- ***maximale Antwortzeit*** ω_i **(engl. *worst-case response time, WCRT*)** $\mapsto$ Zeitdauer zwischen Auslösung und Terminierung des Arbeitsauftrags (genauer: bis das Ergebnis bereitgestellt wurde)
  - Die maximale Antwortzeit wird durch einen relativen Termin beschränkt: Antwortzeit $\omega_i \leq$ relativer Termin D_i

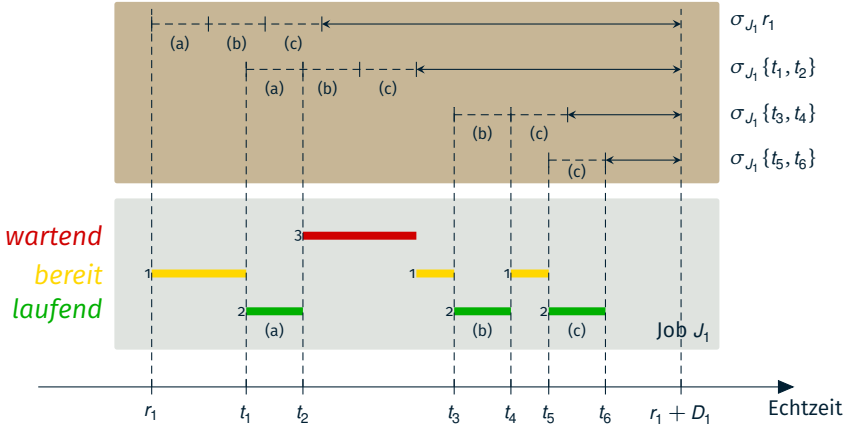
- ***Schlupfzeit*** $\sigma_{J_i} t$ **(engl. *slack time*)** Zeitdauer zwischen dem voraussichtlichen Ausführungsende und Termin eines sich in Bearbeitung befindlichen Arbeitsauftrags
  - Unter der Annahme, dass der Arbeitsauftrag nicht mehr blockiert oder unterbrochen wird

$$\begin{aligned} \sigma_{J_i} t &= r_i + D_i - t - \mathit{maturity}(J_i, t) \\ \mathit{maturity}(J_i, t) &= e_i - \mathit{elapsed\ time}(J_i, t) \end{aligned}$$

  - Ziel: Rechtzeitige, nicht möglichst schnelle Fertigstellung von Aufträgen

⚠ Gibt der Einplanung *Spielraum* zur Einlastung eines Auftrags

- In Phasen der Untätigkeit *verringert* sich die Schlupfzeit
- Während der Ausführung bleibt die Schlupfzeit *konstant*



Übersicht

1 Einplanungseinheit
- Elemente einer Echtzeitanwendung
- Ausführungsstränge und Arbeitsaufträge

2 Ablaufsteuerung
- Verwaltungsgemeinkosten
- Trennung von Belangen
- Grundsätzliche Verfahren

3 Zeitparameter

4 Planbarkeit
- Zulässigkeit und Gültigkeit
- Bewertung von Einplanungsalgorithmen

5 Zusammenfassung

Aufgabenstellung

Gegeben sei eine Menge Aufgaben T_i einer Echtzeitanwendung mit

- D_i dem relativen Termin (engl. *deadline*)
- e_i der maximalen Ausführungszeit (WCET)

der jeweiligen Aufgabe.

Aufgabenstellung

Gegeben sei eine Menge Aufgaben T_i einer Echtzeitanwendung mit

- D_i dem relativen Termin (engl. *deadline*)
- e_i der maximalen Ausführungszeit (WCET)

der jeweiligen Aufgabe.

Fragestellung:

Ist diese Menge von Aufgaben zulässig (engl. *feasible* oder *schedulable*)?

Zulässigkeit

(engl. *feasibility* oder *schedulability*)

- Ein Ablaufplan ist ***gültig* (engl. *valid*)**, falls gewisse **strukturelle Vorgaben** eingehalten werden:
  1. Zu jedem Zeitpunkt max. ein Arbeitsauftrag je CPU

Zulässigkeit

(engl. *feasibility* oder *schedulability*)

- Ein Ablaufplan ist ***gültig* (engl. *valid*)**, falls gewisse **strukturelle Vorgaben** eingehalten werden:
  1. Zu jedem Zeitpunkt max. ein Arbeitsauftrag je CPU
  2. Zu jedem Zeitpunkt max. eine CPU je Arbeitsauftrag

Zulässigkeit

(engl. *feasibility* oder *schedulability*)

- Ein Ablaufplan ist ***gültig* (engl. *valid*)**, falls gewisse **strukturelle Vorgaben** eingehalten werden:
  1. Zu jedem Zeitpunkt max. ein Arbeitsauftrag je CPU
  2. Zu jedem Zeitpunkt max. eine CPU je Arbeitsauftrag
  3. Keine Einplanung vor dem Auslösezeitpunkt

Zulässigkeit

(engl. *feasibility* oder *schedulability*)

- Ein Ablaufplan ist ***gültig* (engl. *valid*)**, falls gewisse **strukturelle Vorgaben** eingehalten werden:
  1. Zu jedem Zeitpunkt max. ein Arbeitsauftrag je CPU
  2. Zu jedem Zeitpunkt max. eine CPU je Arbeitsauftrag
  3. Keine Einplanung vor dem Auslösezeitpunkt
  4. Zuteilung der tatsächliche oder maximale Ausführungszeit

Zulässigkeit

(engl. *feasibility* oder *schedulability*)

- Ein Ablaufplan ist ***gültig* (engl. *valid*)**, falls gewisse **strukturelle Vorgaben** eingehalten werden:
  1. Zu jedem Zeitpunkt max. ein Arbeitsauftrag je CPU
  2. Zu jedem Zeitpunkt max. eine CPU je Arbeitsauftrag
  3. Keine Einplanung vor dem Auslösezeitpunkt
  4. Zuteilung der tatsächliche oder maximale Ausführungszeit
  5. Alle (un)gerichteten Abhängigkeiten werden erfüllt

Zulässigkeit

(engl. *feasibility* oder *schedulability*)

- Ein Ablaufplan ist ***gültig* (engl. *valid*)**, falls gewisse **strukturelle Vorgaben** eingehalten werden:
  1. Zu jedem Zeitpunkt max. ein Arbeitsauftrag je CPU
  2. Zu jedem Zeitpunkt max. eine CPU je Arbeitsauftrag
  3. Keine Einplanung vor dem Auslösezeitpunkt
  4. Zuteilung der tatsächliche oder maximale Ausführungszeit
  5. Alle (un)gerichteten Abhängigkeiten werden erfüllt
- Ein Ablaufplan ist ***zulässig* (engl. *feasible/schedulable*)**, falls
  1. Der Ablaufplan gültig ist
  2. Alle Arbeitsaufträge termingerecht eingeplant werden

$\forall i$: maximale Antwortzeit $\omega_i \leq$ relativer Termin D_i

Zulässigkeit (Forts.)

- Eine Menge von Aufgaben ist *zulässig* (engl. *feasible*)
  - hinsichtlich eines **Einplanungsalgorithmus**,
  - falls dieser Algorithmus einen zulässigen Ablaufplan erzeugt.

Zulässigkeit (Forts.)

- Eine Menge von Aufgaben ist *zulässig* (engl. *feasible*)
  - hinsichtlich eines **Einplanungsalgorithmus**,
  - falls dieser Algorithmus einen zulässigen Ablaufplan erzeugt.

- Die Planbarkeit einer Menge von Aufgaben hängt somit
  - vom verwendeten Einplanungsalgorithmus

Zulässigkeit (Forts.)

- Eine Menge von Aufgaben ist *zulässig* (engl. *feasible*)
  - hinsichtlich eines **Einplanungsalgorithmus**,
  - falls dieser Algorithmus einen zulässigen Ablaufplan erzeugt.

- Die Planbarkeit einer Menge von Aufgaben hängt somit
  - vom verwendeten Einplanungsalgorithmus
  - sowie von den **Eigenschaften der Aufgaben** ab.
    - z.B. periodisch, verdrängbar, frei von Abhängigkeiten, …

⚠ Oft verwenden Algorithmen **Einschränkungen** der Eigenschaften von Aufgaben
  - Dies **vereinfacht** die Frage der Zulässigkeit oft beträchtlich
  - Analysen können Einschränkungen lockern/aufheben

Bewertung von Einplanungsalgorithmen

Optimalität (engl. _optimality_)

Ein Einplanungsalgorithmus ist *optimal* (engl. *optimal*) für eine gewisse Klasse von Aufgaben, falls er für eine Menge solcher Aufgaben einen zulässigen Ablaufplan findet, sofern ein zulässiger Ablaufplan existiert.

- Solch ein **Algorithmus stellt eine Referenz** dar
  - → Schafft es dieser Algorithmus nicht, schafft es keiner!
- Die (generelle) Zulässigkeit einer Menge von Aufgaben
  - Kann auf die Zulässigkeit für diesen Algorithmus reduziert werden
  - → Sofern ein entsprechendes Zulässigkeitskriterium existiert

Einhaltung von Terminen

Zeitgesteuerte Systeme $\rightsquigarrow$ konstruktiv (correct-by-construction)

- Alle Lastparameter sind à priori bekannt
- Die Konstruktion einer Ablauftabelle trägt ihnen Rechnung
- Abhängigkeiten können berücksichtigt werden

Einhaltung von Terminen

Zeitgesteuerte Systeme $\rightsquigarrow$ konstruktiv (correct-by-construction)

- Alle Lastparameter sind à priori bekannt
- Die Konstruktion einer Ablauftabelle trägt ihnen Rechnung
- Abhängigkeiten können berücksichtigt werden

Alle Termine werden eingehalten

- Wenn eine **zulässige Ablauftabelle** erzeugt werden kann

Einhaltung von Terminen

Zeitgesteuerte Systeme $\leadsto$ konstruktiv (correct-by-construction)

- Alle Lastparameter sind à priori bekannt
- Die Konstruktion einer Ablauftabelle trägt ihnen Rechnung
- Abhängigkeiten können berücksichtigt werden

Alle Termine werden eingehalten

- Wenn eine **zulässige Ablauftabelle** erzeugt werden kann

Ereignisgesteuerte Systeme $\leadsto$ analytisch (nachträglich)

- Lastparameter sind nicht vollständig bekannt
- Ablauf wird erst zur Laufzeit berechnet
- Abhängigkeiten müssen explizit gesichert werden

Einhaltung von Terminen

Zeitgesteuerte Systeme $\rightsquigarrow$ konstruktiv (correct-by-construction)

- Alle Lastparameter sind à priori bekannt
- Die Konstruktion einer Ablauftabelle trägt ihnen Rechnung
- Abhängigkeiten können berücksichtigt werden

Alle Termine werden eingehalten

- Wenn eine **zulässige Ablauftabelle** erzeugt werden kann

Ereignisgesteuerte Systeme $\rightsquigarrow$ analytisch (nachträglich)

- Lastparameter sind nicht vollständig bekannt
- Ablauf wird erst zur Laufzeit berechnet
- Abhängigkeiten müssen explizit gesichert werden

- Einhaltung von Terminen muss **immer explizit überprüft** werden

Übersicht

1 Einplanungseinheit
- Elemente einer Echtzeitanwendung
- Ausführungsstränge und Arbeitsaufträge

2 Ablaufsteuerung
- Verwaltungsgemeinkosten
- Trennung von Belangen
- Grundsätzliche Verfahren

3 Zeitparameter

4 Planbarkeit
- Zulässigkeit und Gültigkeit
- Bewertung von Einplanungsalgorithmen

5 Zusammenfassung

Resümee

- **Einplanungseinheit** $\mapsto$ Prozedur, Faden und/oder Fadengruppe
  - Aufgaben (*Tasks*) und Arbeitsaufträgen (*Jobs*)
  - Verwaltungsgemeinkosten ein- und mehrfädiger Aufgaben
  - Einplanung als zweiphasiger Prozess

Resümee

- **Einplanungseinheit** ↦ Prozedur, Faden und/oder Fadengruppe
  - Aufgaben (*Tasks*) und Arbeitsaufträgen (*Jobs*)
  - Verwaltungsgemeinkosten ein- und mehrfädiger Aufgaben
  - Einplanung als zweiphasiger Prozess
- **Ablaufsteuerung** ↦ Strategie & Mechanismus
  - Einplanung ist die Strategie, Einlastung ist der Mechanismus
  - entkoppelt vs. gekoppelt, Taktsteuerung vs. Vorrangsteuerung

Resümee

- **Einplanungseinheit** ↦ Prozedur, Faden und/oder Fadengruppe
  - Aufgaben (*Tasks*) und Arbeitsaufträgen (*Jobs*)
  - Verwaltungsgemeinkosten ein- und mehrfädiger Aufgaben
  - Einplanung als zweiphasiger Prozess
- **Ablaufsteuerung** ↦ Strategie & Mechanismus
  - Einplanung ist die Strategie, Einlastung ist der Mechanismus
  - entkoppelt vs. gekoppelt, Taktsteuerung vs. Vorrangsteuerung
- **Zeitparameter** sind Punkte und Intervalle auf der Echtzeitachse
  - Auslösezeit, (absoluter) Termin
  - Antwortzeit, relativer Termin, Schlupfzeit, Ausführungszeit

Resümee

- **Einplanungseinheit** ↦ Prozedur, Faden und/oder Fadengruppe
  - Aufgaben (*Tasks*) und Arbeitsaufträgen (*Jobs*)
  - Verwaltungsgemeinkosten ein- und mehrfädiger Aufgaben
  - Einplanung als zweiphasiger Prozess
- **Ablaufsteuerung** ↦ Strategie & Mechanismus
  - Einplanung ist die Strategie, Einlastung ist der Mechanismus
  - entkoppelt vs. gekoppelt, Taktsteuerung vs. Vorrangsteuerung
- **Zeitparameter** sind Punkte und Intervalle auf der Echtzeitachse
  - Auslösezeit, (absoluter) Termin
  - Antwortzeit, relativer Termin, Schlupfzeit, Ausführungszeit
- **Planbarkeit** sichert Rechtzeitigkeit der Echtzeitanwendung
  - gültige und zulässige Ablaufpläne
  - optimale Einplanungsalgorithmen
  - konstruktive vs. analytische Überprüfung der Planbarkeit

Literaturverzeichnis (1)

[1] A. Burns and R. I. Davis.
A survey of research into mixed criticality systems.
ACM Computing Surveys (CSUR), 50(6):82:1–82:37, 2017.

[2] Hermann Kopetz.
Real-Time Systems: Design Principles for Distributed Embedded Applications.
Kluwer Academic Publishers, first edition, 1997.

[3] Jane W. S. Liu.
Real-Time Systems.
Prentice Hall PTR, Englewood Cliffs, NJ, USA, 2000.

Literaturverzeichnis (2)

[4] Roman Obermaisser.
Event-Triggered and Time-Triggered Control Paradigms.
Springer-Verlag, 2005.

[5] John A. Stankovic, Marco Spuri, Krithi Ramamritham, and Giorgio C. Buttazzo.
Deadline Scheduling for Real-Time Systems.
Kluwer Academic Publishers, 1998.

EZS – Cheat Sheet

Typographische Konvention

Der erste Index gibt die Aufgabe an (z. B. D_i), der Zweite (optional) bezieht sich auf den Arbeitsauftrag (z. B. $d_{i,j}$). Exponenten zeigen verschiedene Varianten einer Eigenschaft an (z. B. T^{HI}, T^{MED}, T^{LO}). Funktionen beschreiben zeitlich variierende Eigenschaften (z. B. $P(t)$).

Eigenschaften

t (Real-)Zeit
d Zeitverzögerung (engl. delay)

Strukturelemente

E_i Ereignis (engl. event)
R_i Ergebnis (engl. result)
T_i Aufgabe (engl. task)
$J_{i,j}$ Arbeitsauftrag (engl. job) der Aufgabe T_i

Temporale Eigenschaften

Allgemein

r_i Auslösezeitpunkt (engl. release time)
e_i Maximale Ausführungszeit (WCET)
D_i Relativer Termin (engl. deadline)
d_i Absoluter Termin
ω_i Antwortzeit (engl. response time)
σ_i Schlupf (engl. slack)