4.1 Problemstellung

Was heißt Echtzeit-Programmierung?

Kommerzielle Datenverarbeitung

⇨ Richtigkeit des Ergebnisses

Echtzeit-Datenverarbeitung

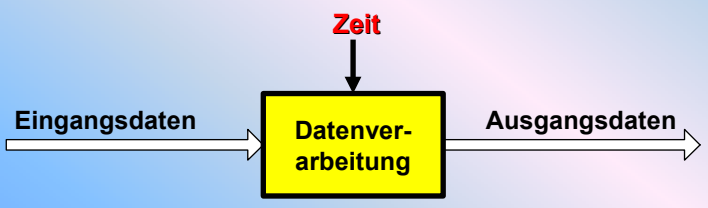
⇨ Richtigkeit des Ergebnisses

⇨ Rechtzeitigkeit des Ergebnisses

nicht zu früh, nicht zu spät

NICHT-ECHTZEIT-DATENVERARBEITUNG:

ECHTZEIT-DATENVERARBEITUNG:



Echtzeitprogrammierung (Realzeit-Programmierung)

Erstellung von Programmen so, daß bei der Datenverarbeitung im Computer die zeitlichen Anforderungen an die Erfassung der Eingabedaten, an die Verarbeitung im Computer und an die Ausgabe der Ausgabedaten erfüllt werden.

Zeitanforderungen

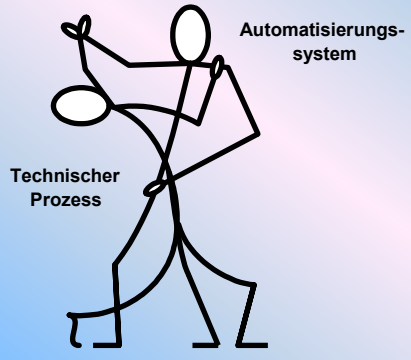
Zeitliche Anforderungen abhängig von den zeitlichen Abläufen im technischen Prozess

Koordination mit dem technischen Prozess

Echtzeit: den echten Zeitabläufen entsprechend, keine Zeitdehnung, keine Zeitraffung

Arten von Anforderungen an das zeitliche Verhalten der Datenverarbeitung

- Forderungen nach Rechtzeitigkeit
- Forderungen nach Gleichzeitigkeit



Unterschiede zwischen Informationssystemen und Echtzeitsystemen

Informationssysteme	Echtzeitsysteme
datengesteuert	ereignisgesteuert
komplexe Datenstrukturen	einfache Datenstrukturen
große Menge an Eingangs-daten	kleine Menge an Eingangs-daten
I / O -intensiv	rechenintensiv
maschinenunabhängig	hardwareabhängig

Wichtige Begriffe

Reaktive Systeme

Echtzeitsysteme, die auf Eingangssignale vom technischen Prozess reagieren und Ausgangssignale zur Beeinflussung des technischen Prozesses liefern

Beispiel: Automatisierungssysteme

Eingebettete Systeme

Integration des Automatisierungssystems in den technischen Prozess

physikalisch und logisch

Beispiel: Rasierapparat, Handy, Waschmaschine, Werkzeugmaschine

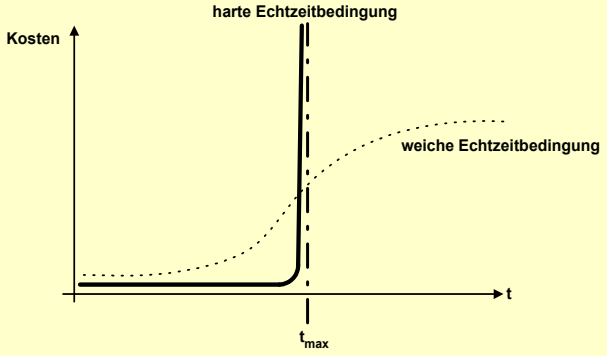
Harte Echtzeitsysteme

Einhaltung von strengen Zeitschranken (Deadlines) ist unabdingbar

Beispiel: ***DDC-Regelungen in Flugzeugen, Motorsteuerungen im Kraftfahrzeug***

Weiches Echtzeitsystem

Eine Verletzung von Zeitschranken kann toleriert werden



Forderung nach Rechtzeitigkeit

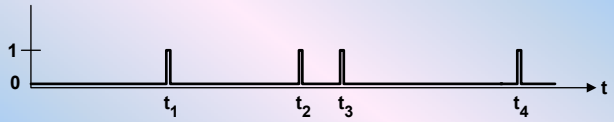
- rechtzeitiges Abrufen der Eingabedaten
- rechtzeitige Durchführung der Verarbeitung
- rechtzeitige Ausgabe der Ausgabedaten

Zeitbedingungen im Zusammenhang mit Rechtzeitigkeit

- ***Absolutzeitbedingungen***
  Bsp.: 11:45 Signal zur Abfahrt
- ***Relativzeitbedingungen***
  Bsp.: wenn ein Meßwert einen Grenzwert überschreitet, nach 10 Sekunden Abschaltsignal

Klassifizierung von Zeitbedingungen bei der Echtzeit- Programmierung (1)

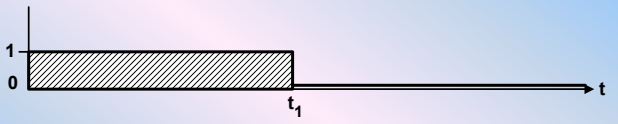
Ausführung einer Funktion zu festen Zeitpunkten t_1, t_2, t_3



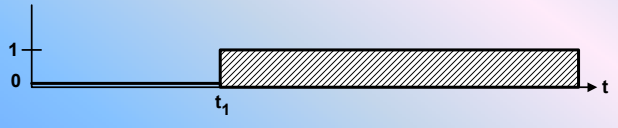
Ausführung einer Funktion in einem zu jedem Zeitpunkt t_1 zugeordneten Toleranzintervall

Klassifizierung von Zeitbedingungen bei der Echtzeit- Programmierung (2)

Ausführung einer Funktion in einem Zeitintervall bis zu einem spätesten Zeitpunkt t_1



Ausführung einer Funktion in einem Zeitintervall von einem frühesten Zeitpunkt t_1 an



Typische Anwendungsbeispiele aus der Prozess-automatisierung für die angegebenen Fälle von Zeitbedingungen

	Absolutzeit-Bedingungen	Relativzeit-Bedingungen
Ausführung einer Funktion zu festen Zeitpunkten	Kennfeldaufnahme an Prüfständen	Analyse von Stoffen in der Chemie
Ausführung einer Funktion in einem Toleranz-Zeit-intervall	Erfassung von Regel-grössen	Messwertüberwachung auf gleitende Grenzen
Ausführung einer Funktion in einem Zeitintervall bis zu einem spätesten Zeitpunkt	Erfassung von Datentele-grammen	Erfassung von Stückgut-Kennungen
Ausführung einer Funktion in einem Zeitintervall von einem frühesten Zeitpunkt an	Folgesteuerung bei Chargenprozessen	Erfassung von Signalen einer Lichtschranke

Forderung nach Gleichzeitigkeit

Vorgänge in Umwelt laufen gleichzeitig ab

- ⇨ Echtzeit-Systeme müssen darauf "gleichzeitig" reagieren
- ⇨ mehrere Datenverarbeitungsaufgaben müssen gleichzeitig durchgeführt werden

Bsp.:

- ❋ Reaktion auf gleichzeitige Fahrt mehrerer Züge
- ❋ Verarbeitung mehrerer gleichzeitig anfallender Messwerte bei einer Heizung
- ❋ Motorsteuerung und ABS-System gleichzeitig

Realisierung von Gleichzeitigkeit

- ❒ jede Datenverarbeitungsaufgabe durch getrennte Computer
- ❒ einen Computer für alle Datenverarbeitungsaufgaben

quasi gleichzeitig

Voraussetzung:

Vorgänge in der Umwelt langsam im Vergleich zum Ablauf der Programme im Rechner

Forderung nach Determiniertheit

Parallele bzw. quasiparallele Abläufe sind im allgemeinen nicht vorhersehbar

- ⇨ das zeitliche Verhalten ist nicht determiniert
- ⇨ keine Garantie der Sicherheit von Automatisierungssystemen

Echtzeitsystem heißt determiniert

Für jeden möglichen Zustand und für jede Menge an Eingangsinformationen gibt es eindeutige Menge von Ausgabeinformationen und einen eindeutigen nächsten Zustand

Voraussetzung: endliche Menge von Systemzuständen

Zeitlich determiniertes System

Antwortzeit für alle Ausgabeinformationen ist bekannt

Arten von Echtzeit-Systemen

- **Dialogsysteme**
- **Automatisierungssysteme**

Dialogsysteme

- ➔ Eingabedaten über entsprechende Eingabemedien
  - Tastatur
  - Lichtgriffe
  - Maus
- ➔ Warten auf Antwort, d.h. Ausgabe von Ergebnissen auf einem Ausgabemedium
  - Bildschirm
  - Drucker

Beispiele für Dialogsysteme

- Platzbuchungssysteme Luftfahrtgesellschaften
- Kontoführungssysteme bei Banken
- Lagerhaltungssysteme

Rechtzeitigkeit bei Dialogsystemen

zulässige Antwortzeit bei Dialog-Echtzeitsystemen in der Größenordnung Sekunden

Automatisierungssysteme

Zeitliche Reaktion abhängig von den Vorgängen im technischen Prozess

Rechtzeitigkeit bei Automatisierungssystemen

zulässige Antwortzeit bei Automatisierungssystemen in der Größenordnung

Millisekunden/Mikrosekunden

Methoden der Echtzeit-Programmierung ähnlich für

- Automatisierungssysteme
- Dialogsysteme

4.2 Echtzeit-Programmierverfahren

Arten des Vorgehens zur Erfüllung der Forderung nach Rechtzeitigkeit und Gleichzeitigkeit

⇨ **Synchrone Programmierung:**
Planung des zeitlichen Ablaufs vor der Ausführung der Programme

Planwirtschaft

⇨ **Asynchrone Programmierung (Parallelprogrammierung):**
Organisation des zeitlichen Ablaufs während der Ausführung der Programme

Marktwirtschaft

Bsp.: Zahnarztpraxis **= Realzeitsystem**

Patient	Automatisierungsprogramm
Behandlungsstuhl und Zahnarzt	Prozessor, der Programme ausführt
Planer	Entwerfer eines Programms
Organisator Sprechstundenhilfe	Betriebssystem
Planung des zeitlichen Ablaufs **vor** der Behandlung mit Hilfe eines Terminkalenders	Synchrone Programmierung
Organisation des zeitlichen Ablaufs **während** der Behandlung (Aufruf aus dem Wartezimmer)	Asynchrone Programmierung, Parallelprogrammierung

Verfahren der synchronen Programmierung

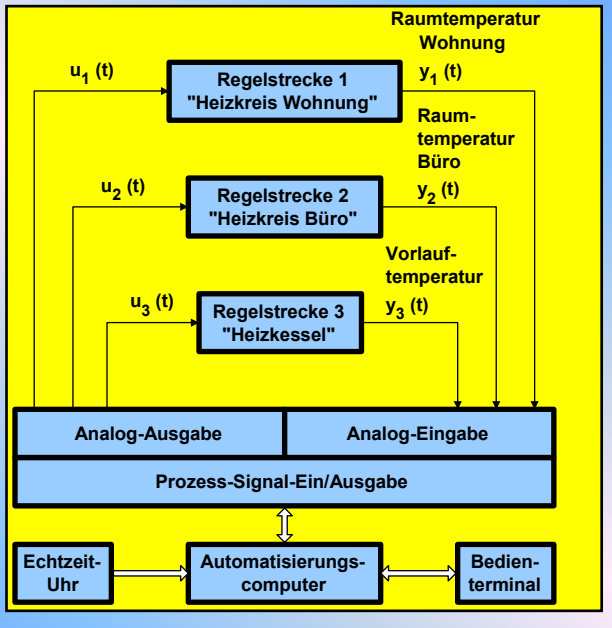
Planung des zeitlichen Verhaltens zyklisch auszuführender Teilprogramme vor der Ausführung

- Synchronisierung der zyklisch auszuführenden Teilprogramme mit einem Zeitraster
- Zeitraster über Echtzeit-Uhr, Unterbrechungssignal zum Aufruf über Teilprogramme

Time triggered

- fest vorgegebene Reihenfolge des Ablaufs der Teilprogramme

Prozess-automatisierungs-system zur Temperatur-regelung für eine Heizungsanlage



Fachtechnische (regelungstechnische) Konzeption

Planung

- ❒ Auslegung der Regelalgorithmen und Reglerparameter
- ❒ Festlegung der Abtastzeiten für die Regelkreise

T Zeitabstand, in denen die Echtzeit-Uhr Impulse liefert

$T_1 = T$

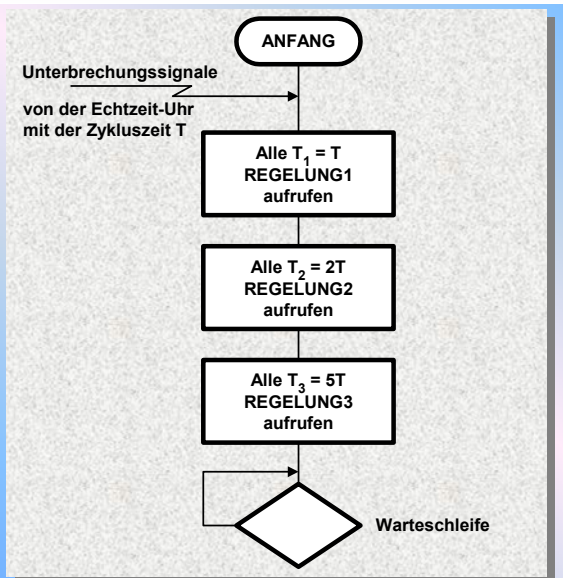
$T_2 = 2T$

$T_3 = 5T$

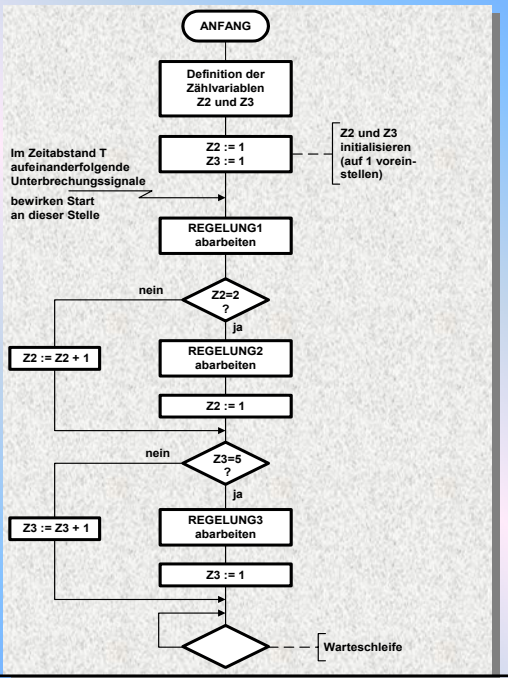
Zuordnung von Bezeichnern und Abtastzeiten zu den Teilprogrammen (Regelungsprogramme) des Prozessautomatisierungssystems

Teilprogramm	Bezeichner (Name)	Abtastzeit (Zykluszeit)
Temperatur-Regelung für die Regelstrecke “Heizkreis Wohnung“	REGELUNG1	$T_1 = T$
Temperatur-Regelung für die Regelstrecke „Heizkreis Büro“	REGELUNG2	$T_2 = 2T$
Temperatur-Regelung für die Regelstrecke „Heizkessel“	REGELUNG3	$T_3 = 5T$

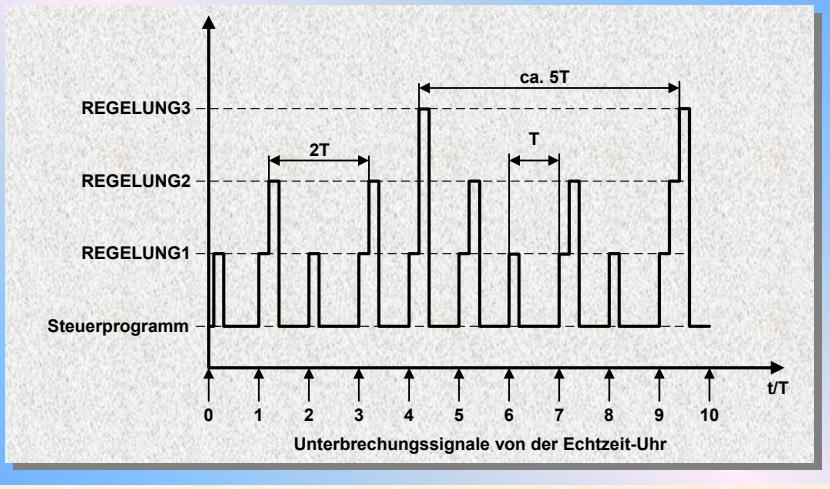
Grobentwurf des Steuerungsprogramms nach dem Verfahren der synchronen Programmierung



Feinentwurf des Steuer-programms nach dem Verfahren der synchronen Program-mierung



Zeitlicher Ablauf der drei Teilprogramme beim Verfahren der synchronen Programmierung



Annahme: ⇨ **Rechenzeit für Teilprogramme gleich groß**

⇨ **Summe der Rechenzeiten der drei Teilprogramme kleiner als Zykluszeit**

Eigenschaften des Verfahrens der synchronen Programmierung (1)

- ❒ Forderung nach Rechtzeitigkeit wird näherungsweise erfüllt

leichte Verschiebung

- ❒ Forderung nach Gleichzeitigkeit wird erfüllt, wenn Zykluszeit T klein gegenüber den Zeitabläufen im technischen Prozess
- ❒ Synchrone Programmierung gut für Echtzeit-Systeme mit zyklischen Progammabläufen

vorhersehbares Verhalten

- ❒ Synchrone Programmierung ungeeignet für die Reaktion auf zeitlich nicht vorhersehbare (asynchrone) Ereignisse
  - ❋ Erhöhung der Rechenzeit durch ständiges Abfragen
  - ❋ Verzögerung der Reaktion

Eigenschaften des Verfahrens der synchronen Programmierung (2)

❒ Nachteil der synchronen Programmierung:

Änderung der Aufgabenstellung bedeutet Änderung der gesamten Programmstruktur

- ❋ im Normalfall deterministisches Verhalten
- ❋ kein komplexes Organisationsprogramm
- ❋ etwas aufwendigere Planung (Entwicklung)

Bsp.: SPS-Programmierung

Verfahren der asynchronen Programmierung (Parallelprogrammierung)

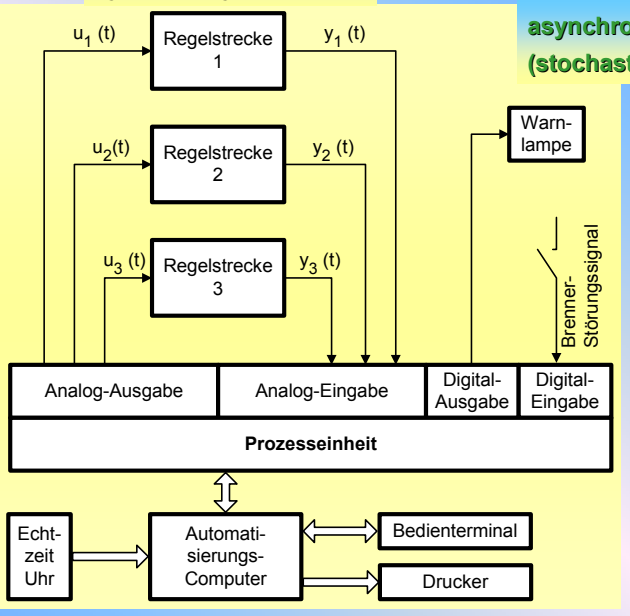
Organisationsprogramm (Echtzeitbetriebssystem) steuert während des Ablaufs der Teilprogramme den zeitlichen Aufruf

- ❒ Aufruf der Teilprogramme, wenn Zeitbedingungen erfüllt sind
- ❒ Gleichzeitige Ausführung wird nach bestimmter Strategie sequentialisiert
  - ❋ Zuordnung von Prioritätsnummern
  - ❋ Priorität desto größer, je niedriger Prioritäts-nummer ist

Prozess-automatisierungs-system für eine Heizungsanlage

synchron zyklisch

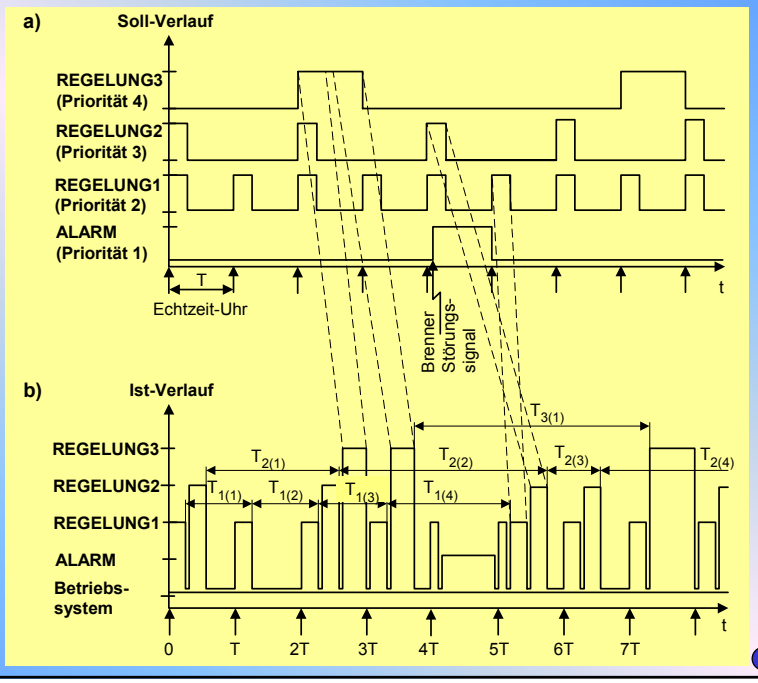
asynchron (stochastisch)



Zuordnung von Bezeichnern, Abtastzeiten und Prioritätsnummern zu den 4 Teilprogrammen des Prozessautomatisierungssystems

Teilprogramm	Bezeichner	Abtastzeit	Proritäts-nummer	Priorität
Reaktion auf Brennerstörung mit Alarmmeldung	ALARM	_	1	höchste
Temperatur-Regelung für Heizkreis 1	REGELUNG1	$T_1 = T$	2	zweit-höchste
Temperatur-Regelung für Heizkreis 2	REGELUNG2	$T_2 = 2T$	3	dritt-höchste
Temperatur-Regelung für Heizkreis 3	REGELUNG3	$T_3 = 5T$	4	niedrigste

Zeitlicher Verlauf der 4 Teil-programme für das Prozess-automati-sierungs-system



Eigenschaften des Verfahrens der asynchronen Programmierung (1)

❒ Forderung nach Rechtzeitigkeit nur näherungsweise erfüllt

schlecht für nieder priore Teilprogramme

! ❋ Zeitbedingungen um so besser erfüllt, je höher die Priorität des jeweiligen Teilprogramms

! ❋ Ist-Zeitablauf kann sich gegenüber Soll-Zeitablauf stark verschieben

Teilprogramme können sich gegenseitig überholen

Eigenschaften des Verfahrens der asynchronen Programmierung (2)

- Aufeinanderfolge der Teilprogramme ist nicht deterministisch, sondern stellt sich dynamisch ein

Bei Programmerstellung lässt sich nicht im Voraus angeben, welches Teilprogramm zu welchem Zeitpunkt ablaufen wird

- **einfache Entwicklung**
- **Komplexität im Verwaltungsprogramm**
- **Programmablauf nicht durchschaubar**

Ereignisgesteuerte vs. zeitgesteuerte Systeme

Ereignisgesteuerte Architekturen

asynchrone Programmierung

- Alle Aktivitäten als Folge von Ereignissen
  - Aktivierung von Tasks
  - Senden von Nachrichten
- Unterstützung durch Echtzeitbetriebssysteme
- nicht-deterministisches Verhalten
- flexibel bezüglich Veränderungen

Zeitgesteuerte Architekturen

synchrone Programmierung

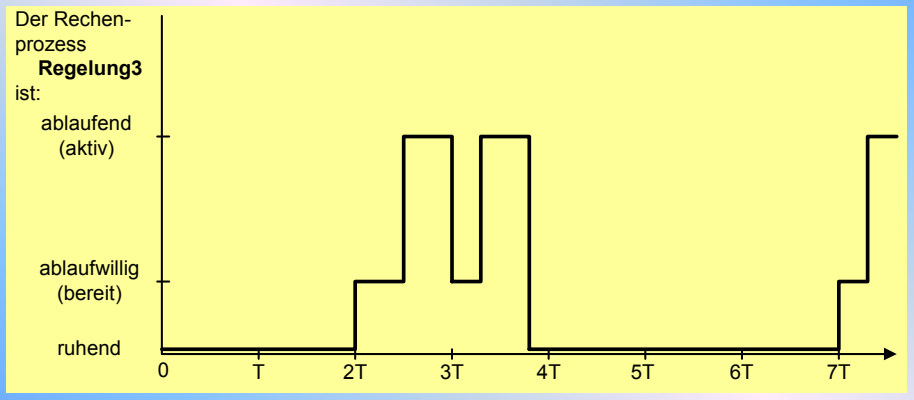
- Periodische Durchführung aller Tasks und Kommunikationsaktion
- Abtastung von externen Zustandsgrößen zu festgelegten Zeitpunkten
- wenig flexibel bei Änderungen
- einfach analysierbar

SPS-Systeme sind zeitgesteuerte Echtzeit-Systeme

4.3 Rechenprozesse (Tasks)

- Zeitlicher Vorgang der Abarbeitung eines sequentiellen Programms auf einem Rechner
- Rechenprozess existiert nicht nur während der Ausführung der Befehle, sondern auch während geplanter oder erzwungener Wartezeiten
- Rechenprozess beginnt mit Eintrag in eine Liste des Echtzeit-Betriebssystems und endet mit dem Löschen aus dieser Liste

Verlauf des Rechenprozesses "Regelung3" beim Verfahren der asynchronen Programmierung



Unterscheidung

- Programm (Anzahl von Befehlen)
- Ausführung des Programms
  (einmalige Ausführung der Befehlsfolge)
- Rechenprozess

Aufruf von Unterprogrammen

- Ausführung des aufrufenden Programms wird unterbrochen
- Ausführung des Unterprogramms
- Fortsetzung des aufrufenden Programms

Aufruf eines Rechenprozesses

- gleichzeitige Ausführung des aufrufenden Programms und des aufgerufenen Rechenprozesses

Rechenprozess

ein von einem Echtzeit-Betriebssystem gesteuerter Vorgang der Abarbeitung eines sequentiellen Programms

Hierarchie von parallel ausführbaren Objekten

- Task
  - beinhaltet ein oder mehrere Threads
  - eigener virtueller Adressraum
  - starke Abschottung
- Thread (Leichtgewichtsprozess)
  - Element einer Task
  - gemeinsamer Adressraum

Unterschiede zwischen Task und Thread

Task	Thread
Besitzer von Betriebsmitteln	Kann außer Prozessor selbst keine Betriebsmittel besitzen, verfügt über alle Betriebsmittel der Task, der er angehört
Eigener Adressraum	Adressraum der Task, der er angehört
Enthält einen oder mehrere Threads	Element einer Task
Kommunikation über die Task-grenzen hinaus, bevorzugt über Botschaften	Kommunikation zwischen den Threads, bevorzugt über gleiche Daten

Zustandsmodelle von Rechenprozessen

4 Grundzustände

- Zustand “laufend” (running)
  - das Teilprogramm ist in Bearbeitung

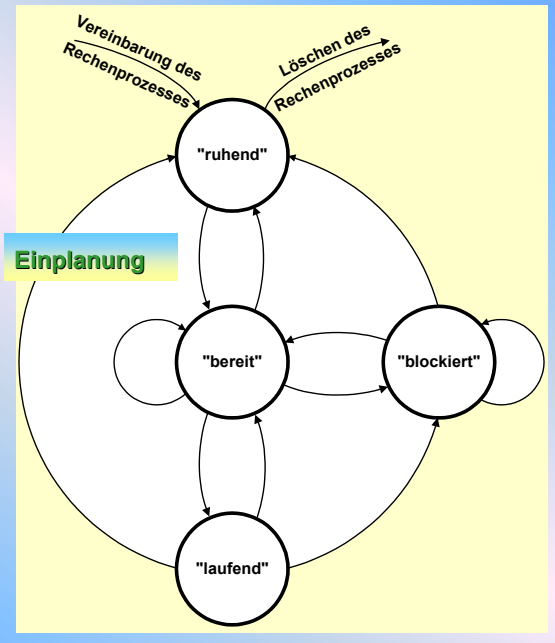
- Zustand “bereit” oder “ablaufwillig” (runable)
  - alle Zeitbedingungen für den Ablauf sind erfüllt
  - es fehlt der Start durch das Betriebssystem

d.h. die Ausführung

- Zustand “blockiert” (suspended)
  - Rechenprozess wartet auf den Eintritt eines Ereignisses
  - Wenn das Ereignis eingetreten ist, Übergang aus dem Zustand “blockiert” in den Zustand “bereit”

- Zustand “ruhend” (dormant):
  - Rechenprozess ist nicht ablaufbereit, weil Zeitbedingungen oder sonstige Voraussetzungen nicht erfüllt sind.

Zustands-diagramm eines Rechenprozesses



Einplanung von Rechenprozessen

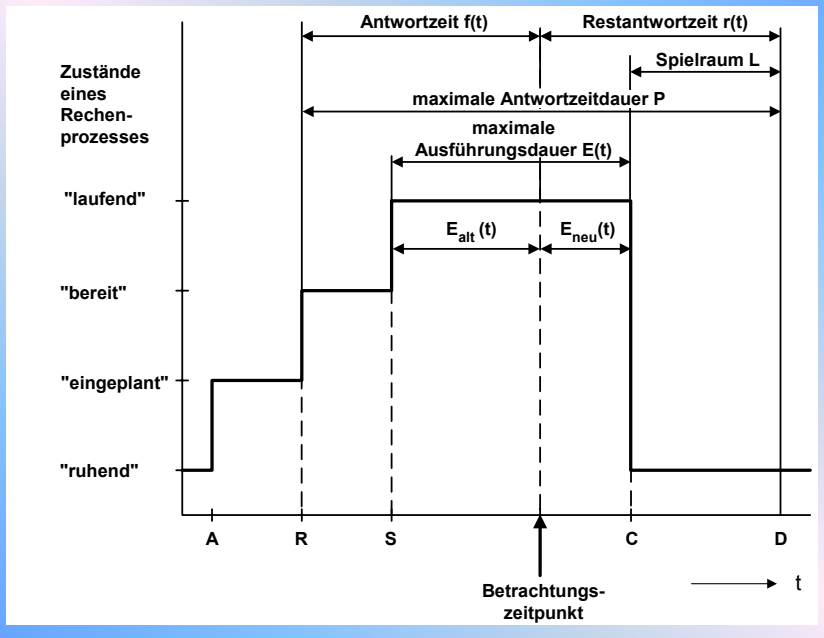
- Beauftragung eines Rechenprozesses zyklisch oder zu bestimmten Uhrzeiten
- Einplanung ist für die Möglichkeit des Übergangs vom Zustand “ruhend” in den Zustand “bereit”

Zeitparameter eines Rechenprozesses:

A:	Arrival time (Ankunftszeitpunkt)
R:	Request time (Einplanungszeitpunkt)
S:	Start time (Startzeit, Zuteilung eines Betriebsmittels)
C:	Completion time (Beendigungszeitpunkt)
D:	Deadline (Maximalzeit)
E:	Execution time (maximale Ausführungsdauer)
P:	Period time (maximale Antwortzeit)
L:	Laxity (Spielraum)
r:	Restantwortzeit
f:	Flow time (Antwortzeit)

Prioritätsvergabe für Rechenprozesse

- statische Prioritätsvergabe
- dynamische Prioritätsvergabe
  - Verwendung von Deadlines (Zeitparameter)



Zusammenhang zwischen den Zeitparametern eines Rechenprozesses

$A <= R <= S <= C - E <= D - E$

Maximale Ausführungsdauer (Rechenzeitdauer)

$E(t) = E_{alt}(t) + E_{neu}(t)$

$E_{alt}(t)$: bisherige Ausführungsdauer zum Betrachtungszeitpunkt

$E_{neu}(t)$: verbleibende Ausführungsdauer

Zeitlicher Spielraum (laxity)

für die Ausführung eines Rechenprozesses
$L = D - S - E$

4.4 Zeitliche Koordinierung (Synchronisierung) von Rechenprozessen

Klassifikation von Aktionen eines Rechenprozesses

- **Zwei Aktionen in Rechenprozessen heißen parallel, wenn sie gleichzeitig ablaufen können**
- **Zwei Aktionen heißen sequentiell, wenn sie in einer bestimmten Reihenfolge angeordnet sind**
- **Zwei Aktionen aus zwei verschiedenen Rechenprozessen heißen nebenläufig, wenn sie gleichzeitig ablaufen können (äußere Parallelität)**
- **Zwei Aktionen eines Rechenprozesses heißen simultan, wenn sie gleichzeitig ausgeführt werden können**

Threads

Synchronisierung von Rechenprozessen

Anforderung an die Durchführung von Rechenprozessen

Synchronität zwischen Rechenprozessen und den Vorgängen im technischen Prozess

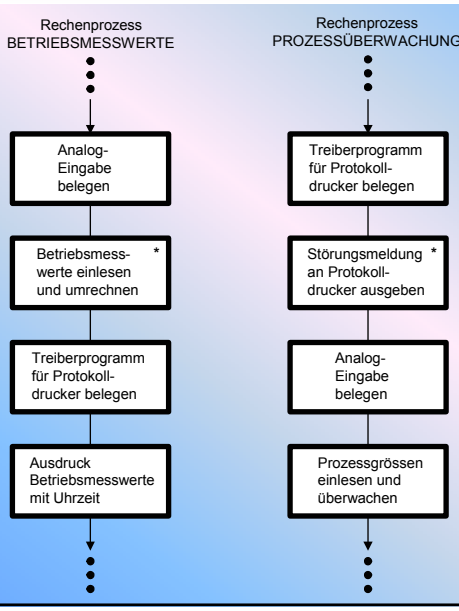
Abhängigkeiten zwischen Rechenprozessen

- Logische Abhängigkeiten aufgrund der Vorgänge im technischen Prozess
- Abhängigkeiten durch die gemeinsame Benutzung von Betriebsmitteln

Beipiel für logische Abhängigkeiten

- REGELUNG1 und REGELUNG2 müssen abwechselnd durchgeführt werden
- Bevor die Sollwerte für die Regelung verwendet werden können, müssen sie mindestens einmal definiert werden

Beispiel für die Abhängigkeit von Rechenprozessen aufgrund gemeinsam benutzter Betriebsmittel



Gemeinsame Betriebsmittel:
- Protokolldrucker
- Analog-Eingabe

Deadlock-möglichkeit!

Synchronisierungseigenschaften

Verklemmung (deadlock):

zwei oder mehrere Rechenprozesse blockieren sich gegenseitig

⇒ zeitliche Koordinierung der Rechenprozesse

= Synchronisierung von Rechenprozessen

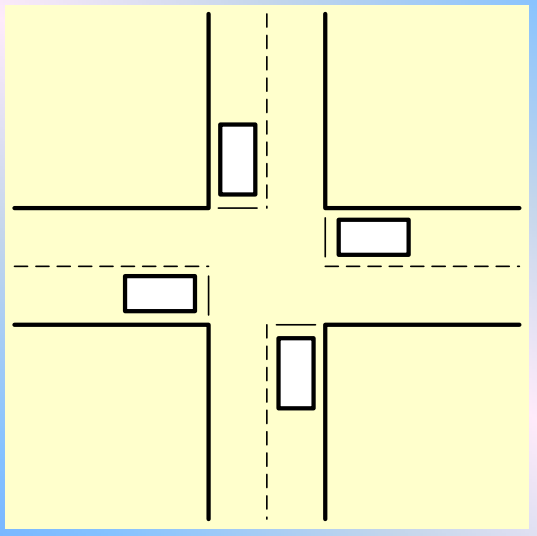
= Einschränkung des freien parallelen Ablaufs

Synchronisierung von Rechenprozessen ist gleichbedeutend mit Synchronisierung ihrer Aktionen

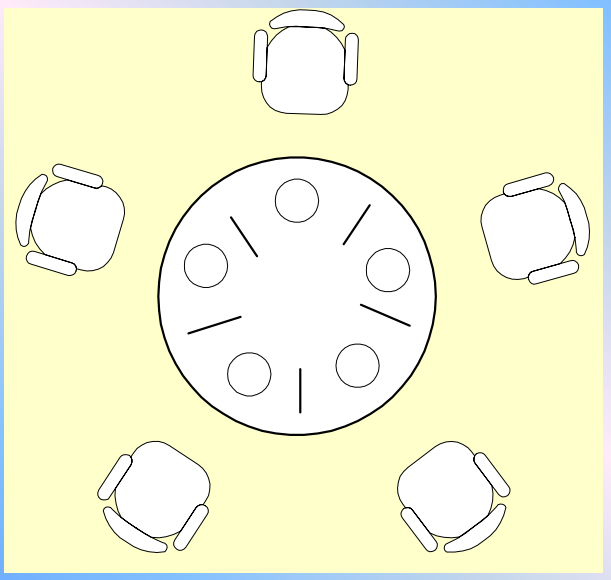
Permanente Blockierung (livelock, starvation)

Konspiration von Rechenprozessen blockiert einen Rechenprozess

Beispiel für einen Deadlock



Beispiel für einen Livelock



Hauptformen der Synchronisierung

- Logische Synchronisierung (aufgabenorientierte oder prozess-orientierte Synchronisierung)
- Betriebsmittelorientierte Synchronisierung

Logische Synchronisierung

- Anpassung des Ablaufs der Rechenprozesse an den Ablauf der Vorgänge im technischen Prozess
- Synchronisierung bedeutet
  - Erfüllung von Anforderungen bezüglich der Reihenfolge von Aktionen
  - Berücksichtigung vorgegebener Zeitpunkte bzw. Zeitabstände
  - Reaktionen auf Unterbrechungsmeldungen aus dem technischen Prozess

Betriebsmittelorientierte Synchronisierung

- Einhaltung von Bedingungen bezüglich der Verwendung gemeinsam benutzter Betriebsmittel (Ressourcen)

Synchronisierungsverfahren

- Semaphore
- kritische Regionen
- Rendez-vous-Konzept

Grundgedanke bei allen Synchronisierungsverfahren

⇨ Rechenprozess muss warten, bis ein bestimmtes Signal bzw. Ereignis eintrifft

⇨ Einfügen von Wartebedingungen an den kritischen Stellen

Semaphorkonzept

Synchronisierung von Rechenprozessen durch Signale (Dijkstra)

Semaphorvariable: positive, ganzzahlige Werte

Semaphoroperationen

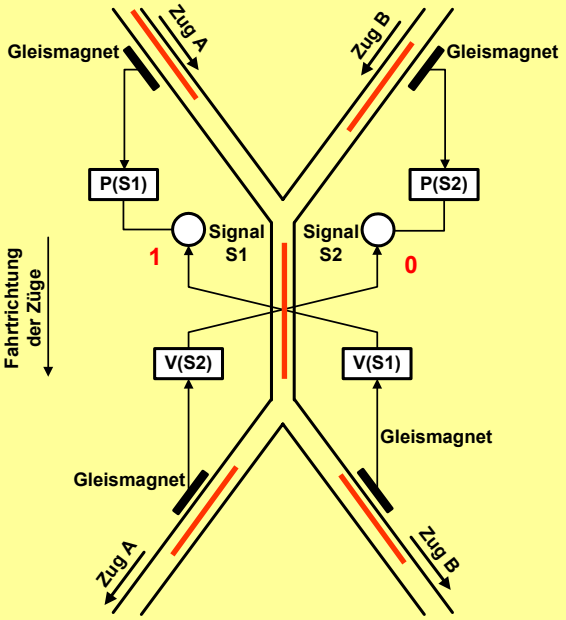
V(S_i): Operation V (S_i) erhöht den Wert der Semaphorvariable S_i um 1

P(Si): Operation P(S_i) bestimmt Wert von S_i

- ist Wert von $S_i > 0$ wird S_i um 1 erniedrigt
- ist $S_i = 0$ wird gewartet, bis $S_i > 0$ geworden ist

unteilbar !

Beispiel: Zugverkehr über eine eingleisige Strecke



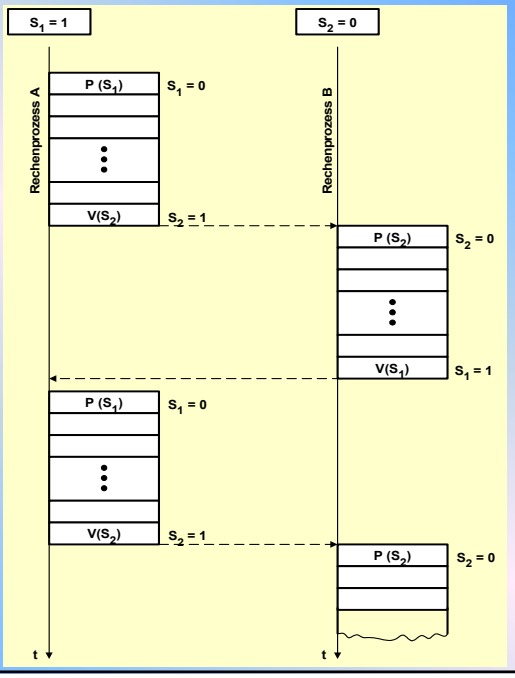
Operation

Variable

Synchronisie-rungsstelle

Operation

Beispiel für die Anwendung von Semaphoroperationen auf zwei Rechenprozesse, die stets abwechselnd ablaufen sollen



4.5 Kommunikation zwischen Rechenprozessen

Synchronisierung: Erfüllung zeitlicher und logischer Randbedingungen beim parallelen Ablauf von Rechenprozessen

Kommunikation: Austausch von Daten zwischen parallel ablaufenden Rechenprozessen

Wechselwirkung Synchronisierung - Kommunikation

Synchronisierung:	informationslose Kommunikation
Kommunikation:	Synchronisierung zum Informationsaustausch

Möglichkeiten der Datenkommunikation

- Gemeinsam benutzter Speicher (shared memory)
  - gemeinsame Variable
  - gemeinsame komplexe Datenstruktur
- Versenden von Nachrichten (messages)
  - Übertragung von Nachrichten von einem Rechenprozessor zu einem anderen (message passing)
  - vor allem bei verteilten Systemen

Arten der Kommunikation

- **Synchrone Kommunikation**
  - sendender und empfangender Rechenprozess kommunizieren an einer definierten Stelle im Programmablauf

Warten durch Blockierung

- **Asynchrone Kommunikation**
  - Daten werden gepuffert

keine Wartezeiten

4.6 Strategien zur Zuteilung des Prozessors an ablaufbereite Rechenprozesse (Scheduling-Verfahren)

Scheduling-Problem

- Rechenprozesse benötigen Ressourcen (Prozessor, Ein-/Ausgabegeräte usw.)
- Ressourcen sind knapp
- Rechenprozesse konkurrieren um Ressourcen
- Zuteilung der Ressourcen muss verwaltet werden

Beispiel: Gleise im Bahnhofsbereich

Scheduling: Vergabe des Prozessors an ablaufbereite Rechenprozesse nach einem festgelegten Algorithmus (Scheduling-Verfahren)

Problem:

1. Gibt es für eine Menge von Tasks (Taskset) einen ausführbaren Plan (Schedule)
2. Gibt es einen Algorithmus, der einen ausführbaren Schedule findet

Beispiel: Vorlesung - Raum - Zeit - Zuordnung

Klassifikation von Scheduling-Verfahren

Klassifizierung in Abhängigkeit vom Zeitpunkt der Planung

- Statisches Scheduling
  - Planung des zeitlichen Ablaufs der Tasks erfolgt vor der Ausführung (dispatching table)
  - Berücksichtigung von Informationen über Taskset, Deadlines, Ausführungszeiten, Reihenfolgebeziehungen, Ressourcen
  - Dispatcher macht Zuteilung gemäß dispatching table
  - Laufzeit- Overhead minimal
  - determiniertes Verhalten

unflexibel bei Änderungen

- Dynamisches Scheduling
  - Organisation des zeitlichen Ablaufs während der Ausführung der Tasks
  - erheblicher Laufzeit-Overhead

flexibel bei Änderungen

Klassifizierung nach Art der Durchführung

- **Preemptives Scheduling**
  - laufende Task kann unterbrochen werden
  - höher priore Task verdrängt nieder priore Task

kooperatives Scheduling

- **Nicht-preemptives Scheduling**
  - laufende Task kann nicht unterbrochen werden
  - Prozessorfreigabe durch die Task selbst

Scheduling Verfahren

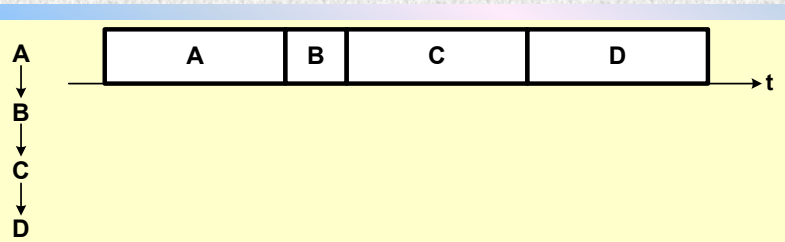
- FIFO-Scheduling (First-in-first-out)
- Scheduling mit festen (unveränderlichen) Prioritäten
- Zeitscheibenverfahren
- Verfahren der kleinsten Restanwortzeit
- Verfahren des kleinsten Spielraums

FIFO - Scheduling

- Nicht-preemptives Scheduling
- Task, deren Einplanung am weitesten zurückliegt bekommt Prozessor
- einfache Realisierung

ungeeignet für harte Echtzeitsysteme

Die Tasks werden in der Reihenfolge ausgeführt, wie sie ablauffähig werden



Scheduling mit festen Prioritäten

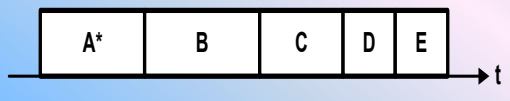
- Prioritäten werden statisch vergeben
- Task mit höchster Priorität bekommt Prozessor
- preemptive und nicht-preemptive Strategie ist möglich
- einfache Implementierung

für harte Echtzeitsysteme nur bedingt geeignet

Taskliste (sortiert nach steigender Priorität)

Task	Prio	
A	1	höchste
B	1	Prio
C	2	
D	3	
E	3	

Je nach Strategie läuft eine der beiden Tasks A bzw. B so lange sie die höchste Priorität haben.



*** FIFO - Scheduling: A war zuerst ablauffähig**

Zeitscheibenverfahren (Round-Robin-Verfahren)

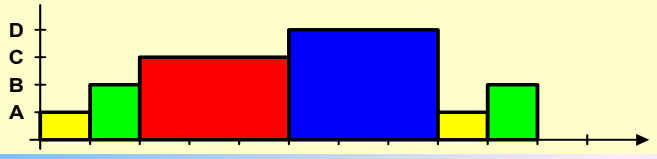
- Jede Task bekommt einen festgelegten Zeitschlitz, zu der sie den Prozessor bekommt
- Reihenfolge wird statisch festgelegt
- Abarbeitung einer Task “Stück für Stück”
- Verwendung bei Dialogsystemen (Multi-Tasking-Systeme)

für harte Echtzeitsysteme untauglich

Beispiel

Task A:	**14 ms**	⟺	**10,6%**
Task B:	**18 ms**	⟺	**13,6%**
Task C:	**50 ms**	⟺	**37,9%**
Task D:	**50 ms**	⟺	**37,9%**

Ablauf:



Verfahren der kleinsten Restantwortzeit (Earliest-Deadline-First-Verfahren)

- Task mit kleinster Restanwortzeit bekommt den Prozessor
- preemptives Vorgehen
- hoher Rechenaufwand für das Scheduling
- Einhaltung von zeitlichen Anforderungen wird speziell unterstützt

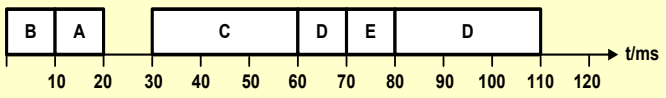
Beispiel

Task	Dauer	T_{min}	T_{max}
A	10 ms	0 ms	40 ms
B	10 ms	0 ms	30 ms
C	30 ms	30 ms	100 ms
D	40 ms	50 ms	200 ms
E	10 ms	70 ms	90 ms

t_{min}: frühester Termin

t_{max}: spätester Termin

Ablauf:



Begründung: Deadline von B ist früher als von A
Deadline von E ist früher als von D
⇨ Unterbrechung

Verfahren des kleinsten Spielraums (least laxity)

- Task mit kleinstem Spielraum erhält Prozessor
- Berücksichtigung von Deadlines und Ausführungsdauern
- sehr aufwendiges Verfahren

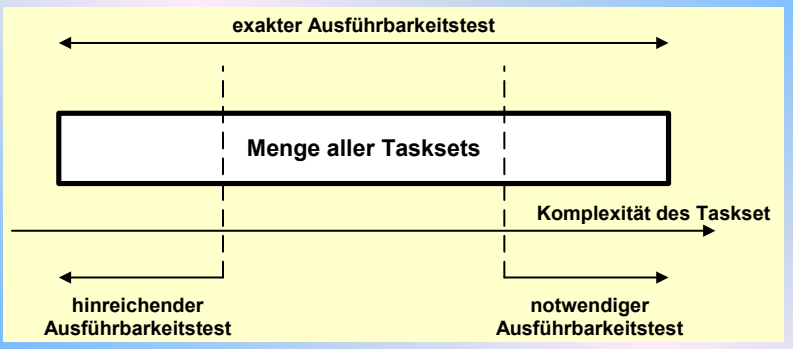
für harte Echtzeitsysteme am besten geeignet

Test zur Prüfung der Ausführbarkeit von Schedules

Ausführbarkeitstest (schedulability test)

Prüfung, ob der zeitliche Ablauf für eine Menge von Tasks (Taskset) so geplant werden kann, dass jede Task ihre Deadline einhält

Klassen von Ausführbarkeitstests



Beispiel: Notwendiger Ausführbarkeitstest für periodische Tasks:

Summe der Prozessorauslastung muss kleiner als 100% sein