Echtzeitscheduling (1)

- Scheduling in Betriebssystemen
  - Ressourcenausteilung (CPU, Speicher, Kommunikation)
  - Faire Ressourcenvergabe, insbesondere CPU
  - Hohe Interaktivität / kurze Reaktionszeit für interaktive Prozesse
  - Hoher Durchsatz, hohe Prozessorauslastung für Hintergrundjobs

- Idee der Echtzeitsysteme
  - Bereitstellen einer Ausgabe innerhalb einer bestimmten Frist
  - → Rechtzeitigkeit
  - Absolute Geschwindigkeit

- Harte/weiche Echtzeit
  - Harte Echtzeit: Verpassen einer Deadline ist kritisch, z.B. bezüglich Personenschaden
  - Weiche Echtzeit: das Überschreiten einer Deadline kann in einem gewissen Maß toleriert werden

Echtzeitscheduling (2)

Klassifizierung von Echtzeitscheduling-Verfahren

Statische Verfahren

- Kalenderbasierte Verfahren / tabellengesteuert
  - Statische (offline) Analyse der Durchführbarkeit von Scheduling-Plänen
  - Fester Plan, wann welche Task beginnt
  - Planung für periodische Tasks basierend auf Superperiode
  - Vorhersagbar, unflexibel

- Prioritätengesteuerte Verfahren
  - statische (offline) Analyse der Durchführbarkeit von Scheduling-Plänen
  - Scheduling-Plan wird indirekt über Prioritäten abgebildet
  - Vorrangunterbrechung (basierend auf vergebenen Prioritäten)
  - Z.B. Rate-Monotonic-Scheduling (RMS)

Echtzeitscheduling (3)

Klassifizierung von Echtzeitscheduling-Verfahren

Dynamische Verfahren

- Nicht adaptive Verfahren
  - Offline Analyse der Task-Laufzeit
  - Online Analyse der Durchführbarkeit von Scheduling-Plänen
  - Bei Ankunft einer Task wird ihr basierend auf ihren Eigenschaften eine Priorität zugewiesen
  - Beispiele: Planung nach Zeitschranken (Earliest Deadline first, EDF), Planung nach Spielräumen (Least Laxity First, LLF)

- Adaptive Verfahren
  - Offline Analyse der Task-Laufzeit; Schätzung der durchschn. Ausführungszeit
  - Online Analyse der Durchführbarkeit von Scheduling-Plänen
  - Fehlertoleranz notwendig, da keine Garantie gegeben werden kann
  - Beispiel: Task-Pair-Scheduling, Anytime-Scheduling

Echtzeitscheduling (4)

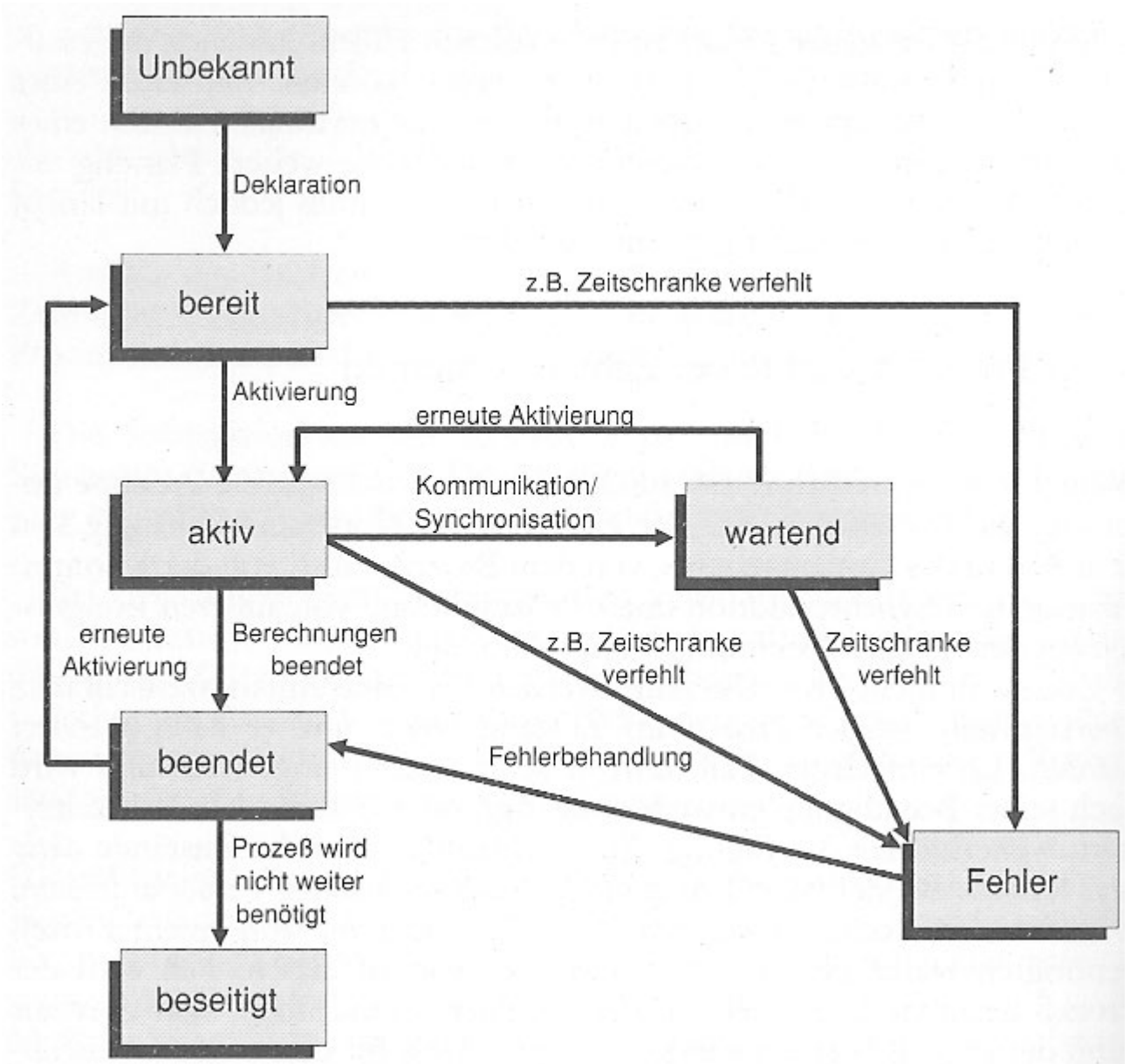
Klassifizierung von Echtzeitscheduling-Verfahren

Schedulingverfahren	Planung	Laufzeitanalyse
Statisch	offline	offline
Dynamisch, nicht adaptiv	online	offline
Dynamisch, adaptiv	online	online

	Statisches Scheduling	Dynamisches Scheduling	
		Nicht adaptiv	Adaptiv
Ohne Garantie bzw. Akzeptanztest	keine Echtzeit, nur best effort		
Mit Garantie bzw. Akzeptanztest	z.B. Rate Monotonic	z.B. Spring Kernel	Nicht sinnvoll
Mit Garantie bzw. Akzeptanztest und Fehlertoleranz	z.B. kalenderbasiert + Task Pair Scheduling	z.B. EDF + Task Pair Scheduling	Fehlertoleranz erforderlich, z.B. TAFT

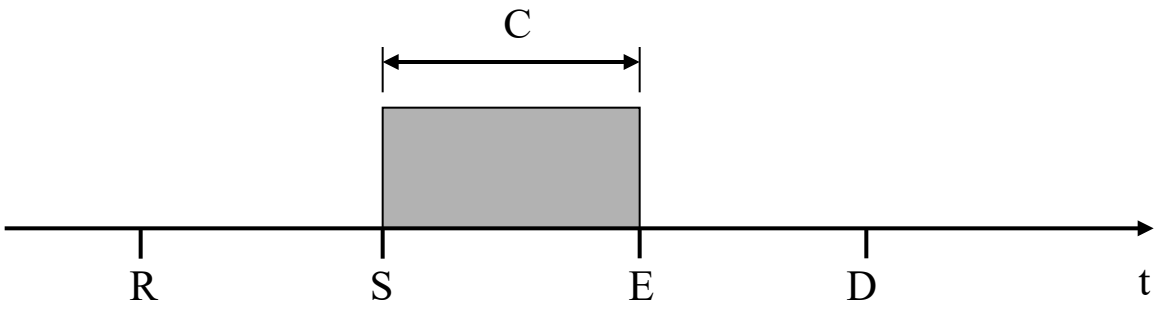
Echtzeitscheduling (5)

Prozesszustände



Echtzeitscheduling (6)

Prozesszeiten



R – frühester Startzeitpunkt (release time)

- Zeitmarke, an der der Prozess frühestens aktiviert werden kann
- Aktivierung durch externes Ereignis oder Programmaufruf

D – spätester Endzeitpunkt (deadline)

- Zeitpunkt, an dem der Prozess vollkommen ausgeführt sein muss
- Prozess muss so aktiviert werden, dass er nach seinem frühesten Startzeitpunkt aktiv wird und vor seinem spätesten Endzeitpunkt beendet ist
- Statt des spätesten Endzeitpunktes kann auch Restantwortzeit a(t) angegeben werden. Es gilt:

$$a(t) = D - t$$

Echtzeitscheduling (7)

Prozesszeiten

S – tatsächlicher Startzeitpunkt (start time)

- Zeitpunkt, an dem der Prozess aktiviert wird
- Frühester Startzeitpunkt und tatsächlicher Startzeitpunkt können zusammenfallen

E – tatsächlicher Endzeitpunkt (completion time)

- Muss vor dem Endzeitpunkt des Prozesses liegen

C – Laufzeit (computation time)

- Zeitdauer, die ein Prozess zur Berechnung benötigt
- Wartezeiten, Zeit für Synchronisation und Kommunikation zählen nicht dazu

Echtzeitscheduling (8)

Prozesskoordination (schedule)

Unter einer Prozesskoordination versteht man die zeitliche Anordnung von Prozessen

Durchführbare Prozesskoordination (feasible study)

Eine Prozesskoordination $P'=P_{i1},P_{i2},...,P_{in}$ bezüglich einer Prozessmenge $P=P_1,P_2,...,P_n$ heißt durchführbar (feasible), wenn alle Prozesse in der Prozessmenge so koordiniert sind, dass sie ihre Zeitschranken einhalten.

Stabiler Scheduling-Algorithmus (stable scheduling algorithm)

Ein Scheduling-Algorithmus heißt stabil (stable) wenn bei einer transienten Überlastung eines Prozessors von einer Prozessmenge $P=P_1,P_2,...,P_n$ eine Teilmenge $P'=P_{i1},P_{i2},...,P_{im}$ mit $m \leq n$ durchführbar koordinieren kann.

Optimaler Scheduling-Algorithmus

Ein Scheduling-Algorithmus heißt optimal, wenn er immer zu einer Prozessmenge $P=P_1,P_2,...,P_n$ eine durchführbare Prozesskoordination liefert, wenn es eine durchführbare Prozesskoordination gibt.

Echtzeitscheduling (9)

Prozessorauslastungsfaktor (utilization factor)

$$U = \sum_{i=1}^{n} \frac{C_i^{t_1,t_2}}{\Delta t}$$

Mit $C_i^{t_1,t_2}$ – Berechnungszeit für Prozess im Teitintervall $[t_1,t_2]$

$\Delta t = t_2 - t_1$

Bei zyklischen Prozessen gilt:

$$U = \sum_{i=1}^{n} \frac{C_i}{T_i}$$

Mit C_i – Gesamtberechnungszeit für Prozess i im Zeitintervall T_i

T_i – Periodendauer für Prozess i

Echtzeitscheduling (10)

Rate Monotonic Scheduling, RMS

- Planung mit festen Raten

Prozessmodell

- Unterbrechenbare (*preemptive*) Prozesse
- Statische Prioritätenvergabe
- Prozesse mit harten Zeitschranken sind periodisch mit konstanter Periodendauer T
- Ein periodischer Prozess kann erst nach seiner Abarbeitung erneut gestartet werden
- Die periodischen Prozesse sind von einander unabhängig
- Die Laufzeit C eines periodischen Prozesses ist bekannt und konstant (ohne Prozessunterbrechungen)
- Nicht periodische Prozesse haben keine harten Zeitschranken
- Ein-Prozessor-Prozess-Scheduling

Einschränkung auf einander unabhängige Prozesse für kompl. EZS i. allg. nicht Akzeptierbar.

Echtzeitscheduling (11)

Rate Monotonic Scheduling, RMS

Verfahren

- Jeder zyklische Prozess erhält eine statische Priorität
- Prozesspriorität indirekt proportional zur Periodendauer

$$Priorität_{Prozess} = \frac{1}{T_i}$$

→ Prozesse mit kürzerer Periode haben eine höhere Priorität

- Es wird jeweils der Prozess mit der höchsten Priorität aktiviert
- Prozesse sind unterbrechbar
- Prozess mit höherer Priorität unterbricht Prozess mit niedrigerer Priorität

Echtzeitscheduling (12)

Rate Monotonic Scheduling, RMS

Eigenschaften

- Der Rate-Monotonic-Scheduling-Algorithmus ist optimal.
- Eine Menge von n unabhängigen periodischen Prozessen kann mit dem Rate-Monotonic-Scheduling-Algorithmus so koordiniert werden, dass alle Prozesse ihre Zeitschranken einhalten, wenn gilt:

$$\sum_{i=1}^{n} \frac{C_i}{T_i} \leq n \cdot \left(2^{\frac{1}{n}} - 1\right) = U(n)$$

- Mit steigender Prozesszahl sinkt die maximale Prozessorauslastung, bei der die Prozessmenge auf jeden Fall koordinierbar ist
- Für n $\rightarrow \infty$ gilt:

$$\lim_{n \to \infty} U(n) = \ln(2) \approx 69\%$$

- Grenzwert stellt eine Worst-Case-Abschätzung bezügl. Der koordinierbaren Prozessanzahl dar, d.h. die Koordinierbarkeit bei höhere Prozessorauslastungen ist möglich, aber nicht garantiert