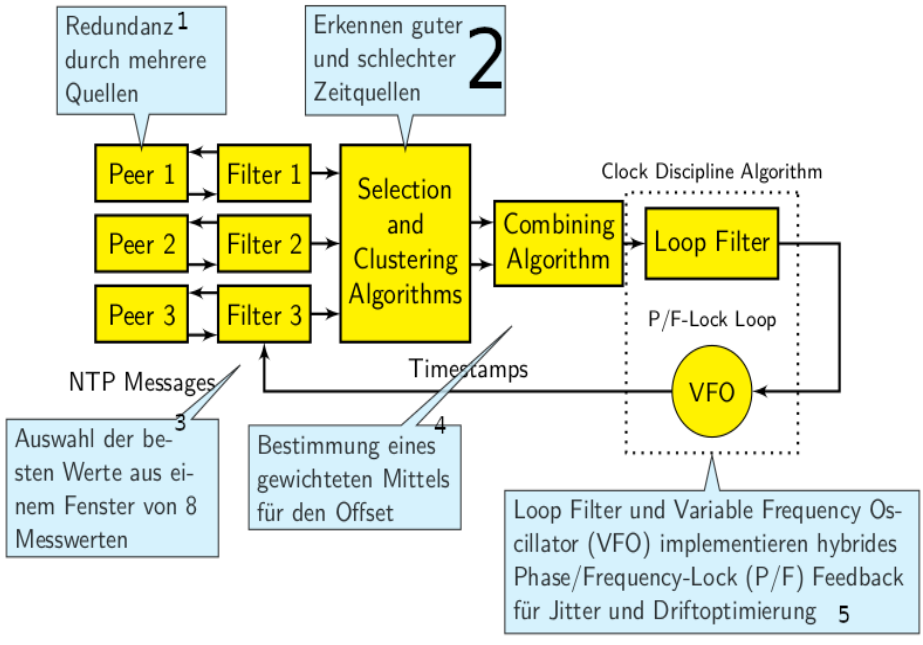
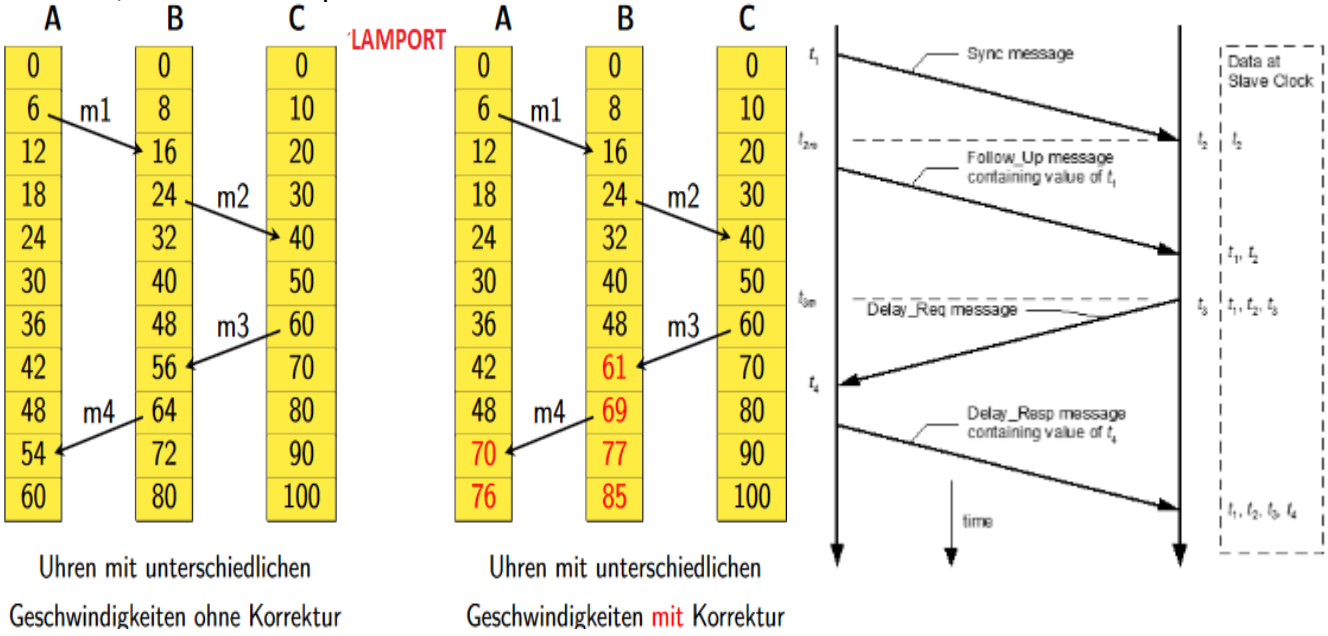
EZV 1) ECHTZEIT != ECHT SCHNELL
Systematisches Testen (gemäß Richtlinien) gefordert.
Deadline (Frist): Zeitpunkt, zu dem die Verarbeitungsergebnisse vorliegen müssen.
Echtzeitbetrieb (Def. nach DIN 44300, 1985): Ein Betrieb eines Rechensystems, bei dem Programme zur Verarbeitung anfallender Daten ständig bereit sind, derart, dass die Verarbeitungsergebnisse innerhalb einer vorgegebenen Zeitspanne verfügbar sind. Die Daten können je nach Anwendungsfall nach einer zeitlich zufälligen Verteilung oder zu vorherbestimmten Zeitpunkten anfallen.
⇒korrektes Systemverhalten erfordert damit auch, dass zeitliche
Vorgaben eingehalten werden.
Monitoring: nur Sensoren, keine Aktoren
Open Loop: nur Aktoren, keine Sensoren
Feedback Control: Sensoren und Aktoren
Reaktives System: Ereignisgetrieben ; Airbag (Echtzeit), Automat (nicht Echtzeit); Antwortzeit muss eingehalten werden.
Zeitgetriebenes System: kontrolliert durch Absolute Zeitpunkte, Perioden, Zeitdauern (bsp: ABS) ; Zeitplan muss eingehalten werden (kein Jitter).
Lastannahme: definiert die angenommene Spitzenbelastung (z.B. Anzahl Ereignisse je sec), die durch das externe System erzeugt wird.
Achtung:
-Mittelwerte sind unbrauchbar!
-Statistische Argumente bzgl. einer geringen Wahrscheinlichkeit des Auftretens unabhängiger Ereignisse sind unzulässig.
-In kritischen Situationen treten häufig sehr viele, zeitlich eng
korrelierte Ereignisse auf (→ „Ereignissturm“).
Fehlerannahme: definiert Art und Anzahl der angenommenen Fehler sowie welche Funktionalität das System unter diesen Annahmen aufrecht erhält.
Achtung:
Das System muss im schlimmsten Fall (Worst Case) die maximale
Anzahl von Fehlern bei Spitzenlast handhaben können.
Geltungsbereich der Annahmen (Assumption Coverage):Wahrscheinlichkeit, dass die gemachten Annahmen mit der Wirklichkeit übereinstimmen.
Vorhersagbarkeit (Predictability): bedeutet, dass das Systemverhalten bei gegebenen Lastannahmen und Fehlerannahmen in Hinblick auf Funktionalität, zeitliches Verhalten (Rechtzeitigkeit) und Verlässlichkeit (Dependability) für den worst case eingehalten wird.
→ Es wird damit "vorhersagbar". Garantiertes Systemverhalten muss nicht optimal sein.
Häufig nimmt man z.B. eine etwas schlechtere Antwortzeit in Kauf, wenn man sicher ist, dass diese bestimmt nicht überschritten wird.
Harte Echtzeitsysteme (Hard Real-Time Systems):
-Mindestens eine zeitliche Anforderung (Deadline) an das Systemverhalten muss immer und unter allen Last- und Fehlersituationen eingehalten werden (s.o.).
-Es werden „Garantien“ gegeben, die z.B. durch formale Beweise oder analytische Modelle belegt werden.
-Z.B.: garantierte Antwortzeiten zwischen Eingabe vom Sensor und reaktionsbedingter Ausgabe an den Aktoren.
-Jedes Verhalten außerhalb der Garantien wird als Systemversagen eingestuft.
-Unterklasse Feste Echtzeitsysteme: Versagen macht zunächst nur die aktuelle Operation wertlos, Versagen bei Wiederholung
Weiche Echtzeitsysteme (Soft Real-Time Systems):
Die zeitlichen Anforderungen werden in der Regel / statistisch eingehalten, gelegentliche Ausnahmen dürfen aber vorkommen

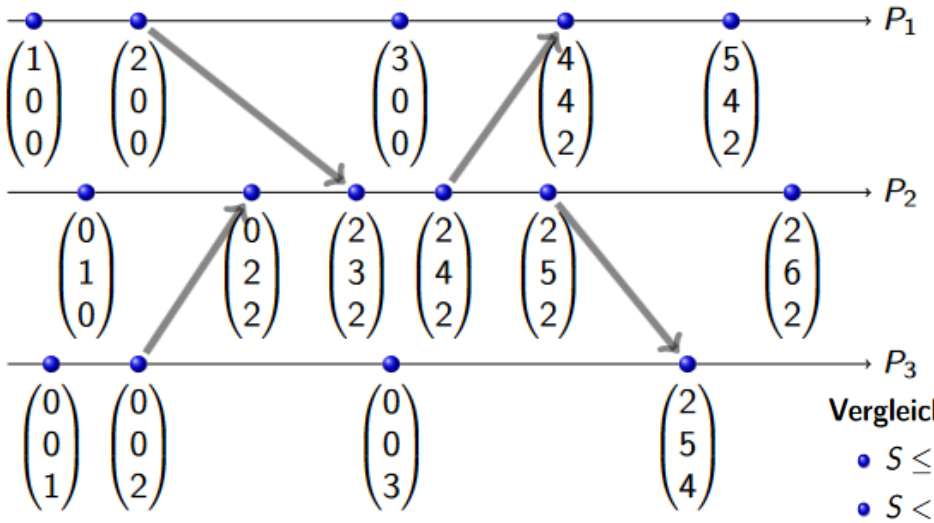
2) **Zeitsysteme**: GMT, UT, TAI, UTC (heutiger Standard: Abweichung 1 s in 300.000 Jahren)
Zeitverteildienste: Langwellen Radiosender, GEOS Satellitensystem (Genauigkeit 0.5 ms)
GPS basierte Uhr: -GPS Signal als Referenz einer PLL-Schaltung
-hochgenaue Sekundenimpulse (pps pulse-per-second)
-typische Genauigkeit: ca. 1 µS
Galileo-Satellitensystem (der EU/ESA): 30 Satelliten, max Genauigkeit 1m; Bewegungen 0.2 m/s
Begriffe (nach Kopetz): Zeit modelliert als Zeitstrahl: Unendliche Menge T von Zeitpunkten
-Zeitdauer als Intervall auf Zeitstrahl
-Ereignis findet zu einem Zeitpunkt statt
-partielle Ordnung (totale Ordnung mit zusätzlichem Kriterium zB. Knottennummer in VS)
Kausale Ordnung von Ereignissen: -Ursache/Wirkungs-Beziehung
-Aus kausaler Ordnung folgt termporale Ordnung der Ereignisse (Umkehrung gilt nicht)
Einheitliche Empfangsordnung von Ereignissen (Delivery Order): -Kommunikationssystem stellt einheitliche Ordnung aller Nachrichten bei allen Empfängern sicher
-Empfangsordnung muss nicht mit temporaler Ordnung oder Empfangsordnung übereinstimmen
Referenzzeit: Approximation der wahren physikalischen Zeit
Abweichung, Genauigkeit (Accuracy): absolute oder relative Differenz zu einer Referenzzeit
Auflösung/Granularität(Granularity):kleinste messbare Zeitdauer(2aufeinanderfolgende Zeitpunkte)
Stabilität(Stability): Frequenzschwankung einer Uhr; Drift als Frequenzdifferenz zwischen Uhren
Kommerzielle Time Server:Unterstützung Standardprotokolle(NTP SNTP), Zeitsynchronisation
Die **globale Zeit** heißt plausibel für die Granularität g, wenn die sie lokal implementierenden Uhren des Ensembles eine beschränkte Präzision Π besitzen (max. Unterschied je zweier Uhren) ; $g > \Pi$
Ab Unterschied von zwei globalen Ticks ist korrekte temporale Ordnung möglich und durch die Zeitmarken gegeben. Für eine konsistente globale temporale Ordnung zweier globaler Ereignisse durch zwei lokale Knoten ist ein Abstand der Ereignisse von 3g notwendig.
Algorithmus von Christian: Mittlere Roundtriplaufzeit messen und berücksichtigen
Schwächen: Rückwärtsgehen einer Uhr möglich ; Schwankungen in Nachrichtenlaufzeiten
TSP (Time Synchronisation): Master/Slave-Algorithmus
-aktiver Master fragt aktuelle Zeiten aller Knoten ab & berechnet Mittel & verteilt Differenz (Offset) an jeden Client
-Schwächen: Rückwärtsgehen einer Uhr möglich; keine Kompensation von Schwankungen in Nachrichtenlaufzeiten; keine Fehlerabschätzung; schlechte Skalierbarkeit
NTP (Network Time): heutiger Internet Standard: Genauigkeit LAN < 1ms; Internet < 10 ms
-Ziele: hohe Genauigkeit; Berücksichtigung schwankender Nachrichtenlaufzeiten und Rechnerausfällen; hohe Skalierbarkeit; eingeschränkte Authentifizierung, Verschlüsselung.



32 bit für Sekunden seit 1.1.70 ; 32 bit Sekundenbruchteil.
Dynamisch festgelegte Verbindungsstruktur mit Backup Verbindungen

DTS (Distributed Time Service): Etablieren eines Zeitintervalls, das UTC enthält und Ungenauigkeiten minimiert.
Precision Time (PTP): höhrere Genauigkeit als NTP; Master Slave Verfahren ; Genauigkeit im ns Bereich, in Software in µS Bereich.





- kausal abhängige Ereignisse, z.B. $(0,0,1) \rightarrow (5,4,2)$, $(1,0,0) \rightarrow (2,6,2)$
- nebenläufige Ereignisse, z.B. $(0,0,3) || (5,4,2)$

Vergleich von Zeitmarkenvektoren

- $S \leq T :\Rightarrow S[i] \leq T[i]$ für alle i
- $S < T :\Rightarrow S \leq T$ und $S \neq T$
- $S || T :\Rightarrow \neg(S < T)$ und $\neg(T < S)$

Nebenläufigkeit

- Ereignisse a und b sind nebenläufig $\Leftrightarrow VC(a) || VC(b)$

Kausalität

- $a \rightarrow b \Leftrightarrow VC(a) < VC(b)$

nachrichtenbasierte Kommunikation

jeder Prozess P_i besitzt Uhr VC_i als Vektor von Zeitmarken

lokales Ereignis in P_i:

- $VC_i[i] := VC_i[i] + 1$, sonst unverändert

Sendeereignis in P_i:

- $VC_i[i] := VC_i[i] + 1$ (Erhöhe eigenen Ereigniszähler)
- Versende Nachricht mit eigener Vektorzeit $vt = VC_i$

Empfangsereignis in P_k:

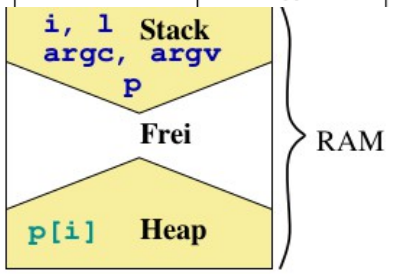
- $VC_k[j] := max\{VC_k[j], vt[j]\}$ für alle j
- $VC_k[k] := VC_k[k] + 1$ (Erhöhe eigenen Ereigniszähler)

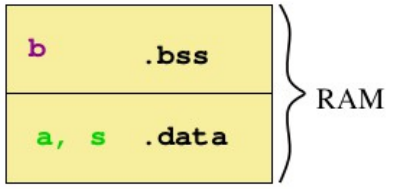
3)

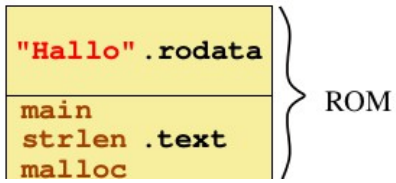
Klasse	Lesen	Schreiben	Ausführen	Initialisiert	Sektion
Programmcode	x	-	x	x	.text
initialisierte Daten	x	x	-	x	.data
uninitialsierte Daten	x	x	-	-	.bss
nur-lese Daten	x	-	-	x	.rodata

```
static int   a = 5;
char *s =    "Hallo";
int b;

main(int argc, char *argv[])
{
    int i;
    int l = strlen(s);
    char *p;
    p = malloc(l + 1);
    for(i = 0; i < l; i++)
       p[i] = s[i];
}
```



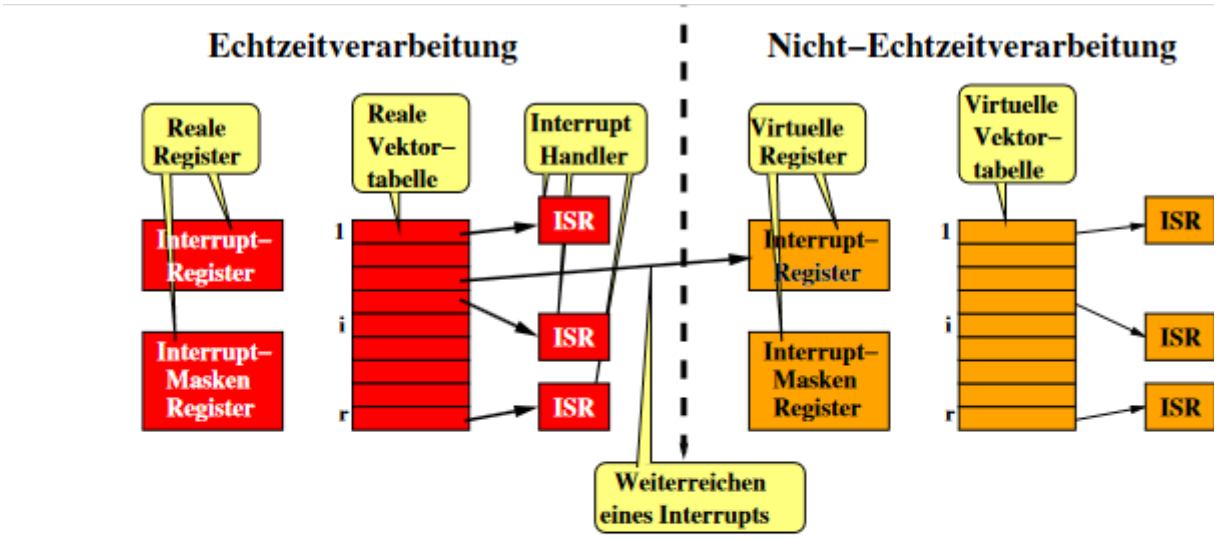




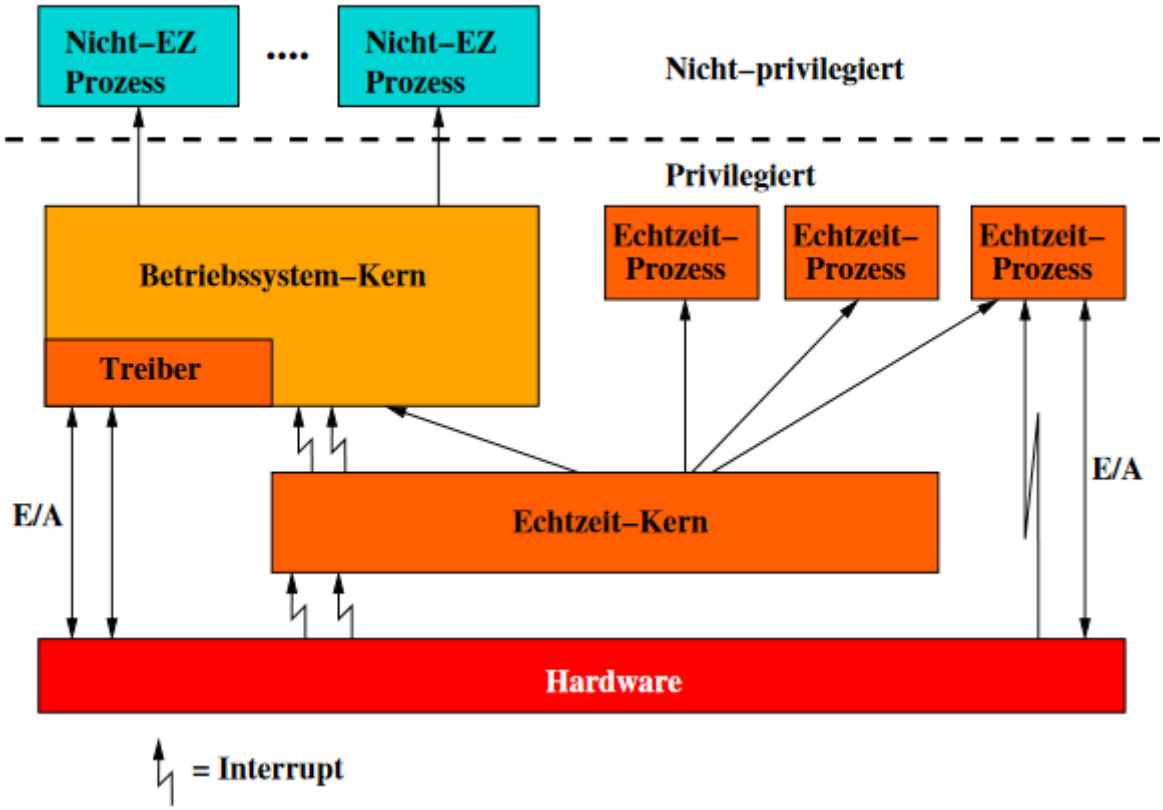
Diese 4 Sektionen müssen vor Programmstart initialisiert werden. Betriebssystem lädt die Speicherinhalte einer ausführbaren Datei ins RAM.
C Startup Code: .data Sektion initialisieren; .bss nullen; argv aufbereiten.

Setjmp/longjmp: Bsp: Direkte Rückkehr statt Durchreichen der Fehlerbedingung

4)



Gesamtarchitektur



AUTOSAR: Standard Software-Architektur für Automotive Systeme. (Basis: POSIX oder OSEK)
-RTE (Runtime Environmen) Schnittstelle für Applikationen.
-Ortstransparenz durch Sicht als „Virtual Function Bus“
OSEK: Offene Systeme und deren Schnittstellen für die Elektronik im Kraftfahrzeug.
OSEK-OS Spezifikation für (offen seit 1997; ISO seit 2005). Vielzahl von Implementierungen.
Idee: **statisches System: -**dynamischer Speicher & Erzeugen/Verwerfen von Objekten verboten
-Kein Ändern von Prioritäten zur Laufzeit
-Alle Objekte (Tasks, Events, Timer, …) und deren Parameter müssen zur Konfigurationszeit deklariert sein. (Spezielle Sprache dazu: OIL).
Skalierbarkeit: -Verfügbar auf einem weiten Bereich von Plattformen (8bit – 64 bit)
- Der Standard definiert 4 „Konformanzklassen (Scheduler Funktionalität)
-Validierung von Funktionsargumenten zur Laufzeit ist während der Entwicklung unverzichtbar.
-Einige OSEK Implementierungen sind < 1kB Codegröße
-Plattformspezifische Dinge (IO) nicht im OSEK OS Standard definiert
-**OSEK OS Processing Levels:** 1. Interrupts, 2. Scheduler, 3. Tasks
-OSEK OS Funktionsgruppen, Taskverwaltung, Alarme, Messages, Fehlerbehandlung, Synchronisation, Interrupts
-**OSEK OS Tasks**: Basic Tasks geben Kontrolle ab wenn sie terminieren, ein Interrupt auftritt oder ein höherpriorer Task rechenwillig wird ; Extended Tasks können auch „auf Event warten“. Tasks können Scheduler „reserverieren“ um zu verhindern, dass sie von anderen Tasks verdrängt werden
-**Non-preemptive Scheduling**: Taskwechsel nur möglich wenn folgende Funktionen aufgerufen werden: TerminateTask, Schedule, WaitEvent oder ein Task beendet wird und ein Nachfolgetask (mit ChainTask) aktiviert wird.
-**Full preemptive**: Ausnahme Task hält Scheduler Ressource
-**Mixed Preemptive**: Non Preemptive + Preemptive
Interruptkategorien: 1. Interrupthandler verwendet keine OSEK OS Systemdienste ; 2.Handler verwendet eine Teilmenge der OSEK OS Dienste; 3. wie 1. jedoch können nach Aufruf von EnterISR auch OSEK OS Systemdienste verwendet werden. Handler muss LeaveIsr aufrufen.
Event: -Mittel zur Tasksynchronisation, fest einer extended Task zugeordnet (zum Verlassen oder eintretten in den Wartezustand); extended Task kann viele Events besitzen.
Ressourcenverwaltung: -gegenseitiger Ausschluss: eine Ressource kann immer nur von einer Task belegt sein & OSEK-Ressourcen verhindern Prioritätsinversion und Deadlocks
-Zugriff auf Ressourcen führt niemals zu einem Wartezustand
-Der Scheduler wird in OSEK als Ressource behandelt: durch Belegen der Scheduler-Ressource kann eine Task ihre Verdrängung(Preemption) durch andere Tasks verhindern exi
Priority Ceiling Protokoll: Ceiling-Priorität:
≥ höchste aller Prioritäten der Tasks, die auf die Ressource zugreifen
< niedrigste aller Prioritäten der Tasks, die nicht auf die Ressource zugreifen, und deren Priorität höher liegt, als die höchstpriore der Tasks, die darauf zugreifen.
-Wenn ein Task eine Ressource beansprucht, und ihre Priorität < Ceiling-Priorität, wird ihre Priorität = Ceiling Priorität, bis die Ressource freigegeben wird. Danach wird sie wieder gesenkt.

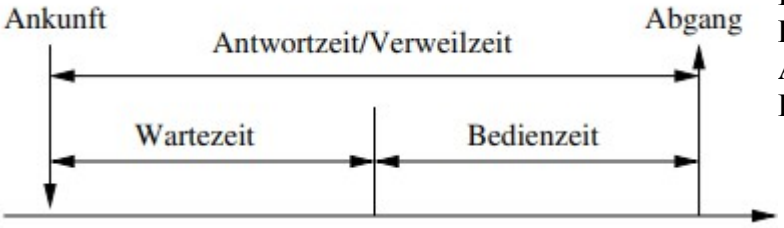
Alarm:
Messages:
Hooks:

POSIX ECHTZEITERWEITERUNGEN (Portable Operating System Interface extensions): prioritätengesteuertes Scheduling, Echtzeit-Signale, Clocks und Timer, Semaphore, Messages, Memory Mapped Files und Shared Memory, Asynchrone Ein/Ausgabe, Synchrone Ein/Ausgabe, Memory Locking

5) **Steuerung (Ohne Rückkopplung, bzw. Regler)**:
-offener „Regelkreis“ (Bsp: Wecker) ; keine Reaktion auf Änderungen und Störgrößen
Stabilität: „Aufschaukeln“ von Aktionen ist möglich (Ursache & Wirkung verstärken sich positive Rückkopplung) ; Eingang oder Ausgang enthalten Eigenfrequenz des Systems => Resonanz

PD-Regler: -Schnelle Reaktion auf Ankündigungen von Veränderungen Unruhig
PI-Regler: -Kombination schnelle Reaktion und exakte Ausregelung ; genau und mittelschnell
PID-Regler: -Universalregler Genau und schnell

6) **Prozess (kleinste einplanbare Einheit)**: endliche Ausführungsfolge von Maschinenbefehlen
Echtzeitplanung: Prozesse so schedulen, sodass alle Zeitbedingungen eingehalten werden
Periodischer Prozess: periodisch aufrufen
Prozess als Typ ; Instanzen als konkrete Prozesse mit privaten Eingangs- & Zielvariablen.
Dauer der Prozessumschaltung wird in den Planungsverfahren i.d.R. nicht berücksichtigt oder als Konstante den Ausführungszeiten zugeschlagen.
Bedienzeit: Zeitdauer für die reine Bearbeitung eines Auftrags die Bedienstation (hier Prozessor)



Fairness: „gerechte“ Behandlung aller Aufträge
Durchsatz: # erledigte Aufträge pro Zeiteinheit
Auslastung: Anteil der Zeit im Zustand „belegt“
Bereitzeit: Frühester möglicher Zeitpunkt

Variantent der Echtzeitplannung: Statisches/Dynamisches Scheduling
Explizite Plannung: Dem Rechensystem wird ein vollständiger Ausführungsplan (Schedule) übergeben, der zur Laufzeit befolgt wird (Umfang kann extrem groß werden)
implizite Planung: Dem Rechensystem werden nur die Planungsregeln übergeben
Phasen der Echtzeitplanung: - Phase1: Einplanbarkeitsanalyse
-Es wird geprüft ob ein brauchbarer (feasible) Ausführungsplan (Schedule) existiert.
-Phase2: Planerstellung (Alle Informationen die zur Laufzeit notwendig sind zum Schedulen)
-Phase3: Prozessorzuteilung
Echtzeit-Planungsverfahren: -**Durchsuchen des Lösungsraumes**
- Durchsuchen des Lösungsraumes ohne Beachtung von Strategien oder Heuristiken stellt das „einfachste” Planungsverfahren dar (Branch-and-Bound-Verfahren)
(A) Es werden zunächst alle möglichen Kombination von nacheinander ausgeführten Prozessen ohne Berücksichtigung von Bereitzeiten und Fristen generiert.(n! Scheduling Möglichkeiten;O(n!))
Nicht alle nach (A) erzeugten Pläne sind gut, wenn Bereitzeiten & Fristen berücksichtigt werden
(B) Heuristik: Einplanen eines Prozesses so früh wie möglich, d.h. s-r → min
Satz:Wenn es brauchbare Pläne für eine Prozessmenge gibt, so wird ein solcher durch (B) gefunden
-**Planen nach Fristen (Earliest Deadline First)**: -Der Prozessor wird demjenigen Prozess i zugeteilt, dessen Frist di den kleinsten Wert hat (am nächsten ist)
-anwendbar auf nicht-unterbrechbare und unterbrechbare Prozesse
-anwendbar auf statischen und dynamischen Planungsverfahren
-optimal wenn bei nicht-unterbrechbaren Prozessen, wenn alle Bereitzeiten gleich sind
-Falls EDF keinen brauchbaren Plan liefert, gibt es keinen