Automatisches Entwurfs- und Entwicklungssystem für harte Echtzeitsysteme

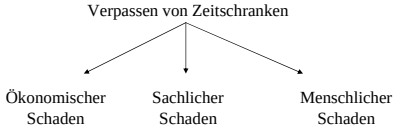
Jan Blumenthal
20.02.2003

Gliederung

- Einführung
- Bestandsaufnahme
- Konzept des Frameworks YASA
- Vorstellung einzelner Komponenten
  - Executives
  - Scheduler
- Zusammenfassung

Allgemeine Ziele

- Echtzeitsysteme:
  - funktionaler Nachweis
  - nicht-funktionaler (zeitlicher) Nachweis

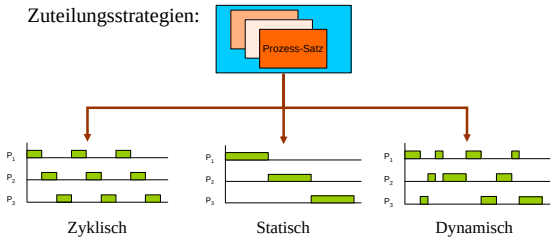


Konzentration auf Optimierung des Schedulingverhaltens

Scheduler

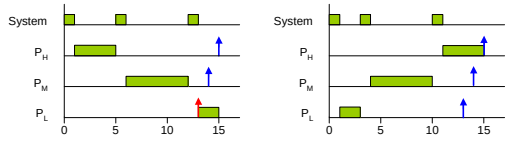
Aufgabe: Zuteilung der Prozesse auf die CPU

Zuteilungsstrategien:



Optimierung des Schedulings

Prioritätsbasiert: *Earliest Deadline First:*



- **Analyse des Verhaltens nötig**
- **Austausch des Schedulers erforderlich**

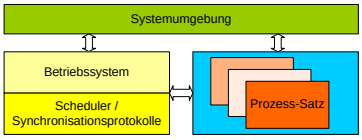
Synchronisationsprotokolle

Semaphoren:
- Datenaustausch von Prozessen
- Absicherung von kritischen Abschnitten
- Betriebsmittelanforderungen

Synchronisationsprotokolle:
- Vermeidung von Effekten bei Belegung von Semaphoren
  - Prioritäteninversion
  - Deadlocks
- Enge Kopplung an Scheduler

Austausch der Synchronisationsprotokolle erforderlich

Aktuelle Systeme



Nachteile:
- Scheduler fest mit Betriebssystem verbunden
- Keine dynamischen Scheduler bzw. Synchronisationsprotokolle
- Schedulingverlauf nicht überprüfbar

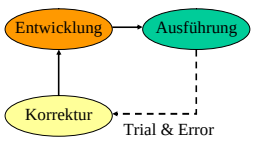
Spezielle Ziele

- Optimierung des Schedulingverhaltens in Echtzeitbetriebssystemen:
  - Umfangreiche Analyse des Schedulingverhaltens
  - Austausch der Scheduler
  - Nutzung weiterer Synchronisationsprotokolle
- Entwurf von harten Echtzeitsystemen
- Unterstützung des Hardware / Software-Codesigns

Entwicklungsframework YASA

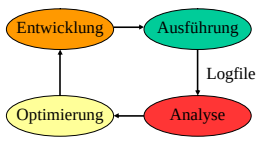
Anwendungsentwicklung

Aktueller Stand:



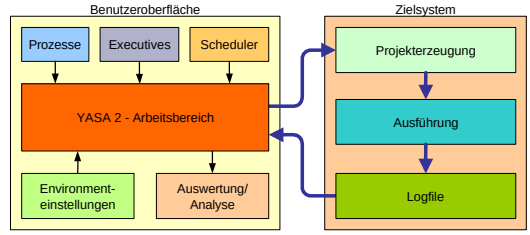
- Unbekannter Schedulingverlauf
- Keine Analyse
- Keine bzw. schlechte Fehlererkennung

YASA:



- Protokollierter Schedulingverlauf
- Analyse des Schedulingverlaufs möglich

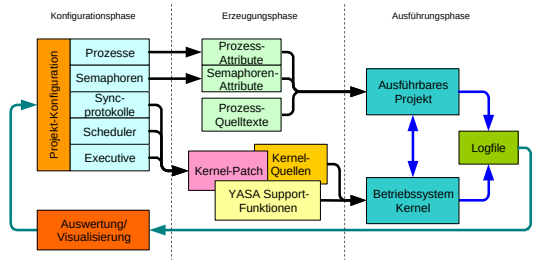
Gliederung von YASA



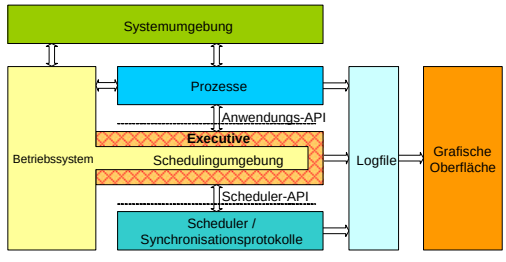
- Projektverwaltung & Konfiguration
- Quelltextgenerierung
- Auswertung

- Kompilierung
- Ausführung
- Logfile-Erstellung

Zyklus eines Projekts



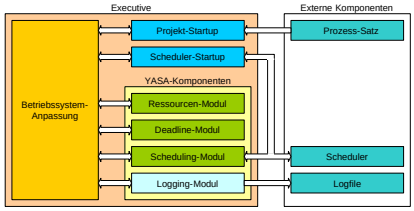
Arbeitsweise von YASA



➡ Kapselung der Schedulingumgebung durch Executives

Aufbau eines Executives

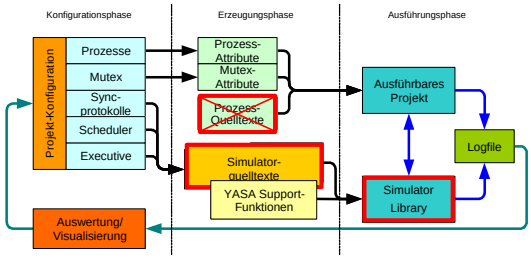
- Virtuelle Schedulingumgebung
- Schnittstellen für Scheduler und Prozesse
- Unterstützung portabler Scheduler



Simulator-Executive

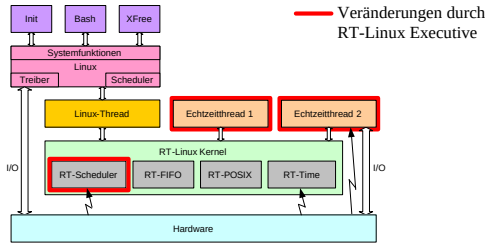
- Ausführung der Anwendung mit synthetischer Last
- Virtueller Programmlauf
  - Prozesszustände
  - Ressourcenanforderungen
- Multiprozessorfähig
- Plattformunabhängig
- Anwendungsgebiete:
  - Nachweis der theoretischer Analysen
  - Entwicklung von Schedulern

Projektzyklus im Simulator



Besonderheiten des Simulator-Executives

RT-Linux Executive



Scheduler

- Wechsel zur Laufzeit
- Implementierung
  - Executive-unabhängig
  - Plattformunabhängig
  - Keine Limitierung der Zeitauflösung, Prozessanzahl

Verfügbare Scheduler:

Statische Scheduler	Dynamische Scheduler	Zyklische Scheduler
Prioritätsbasiert	Earliest Deadline First	Round Robin
Rate Monotonic	Least Laxity First	First Come – First Serve
Deadline Monotonic	Enhanced Least Laxity First	

Synchronisationsprotokolle

- Synchronisation der Betriebsmittelanforderungen
  - Verhinderung der Prioritäteninversion
  - Deadlockvermeidung
- Einführung dynamischer Scheduling-Prioritäten
  - Entkopplung von Synchronisationsprotokollen und Schedulern

Verfügbare Synchronisationsprotokolle:

Statisch	Dynamisch
Priority Inheritance Protocol	Dynamic Priority Ceiling Protocol
Priority Ceiling Protocol	Stack Resource Policy
Ceiling Semaphore Protocol	

Dynamische Scheduling-Prioritäten

- Anpassung der Synchronisationsprotokolle an aktuellen Schedulerprioritätstyp
- Zugriff auf Variablen (Deadline, Priorität) durch Mappingmakros

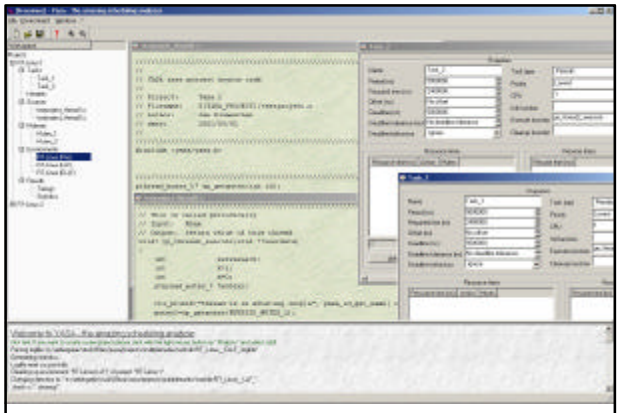
Name	Typ	Datentyp	Vergleichsoperation
Priorität	Statisch	int	$P_1 > P_2$
Schlupf	Dynamisch	YASA_TIME	$P_1 < P_2$
Deadline	Dynamisch	YASA_TIME	$P_1 < P_2$
Benötigte Zeit	Dynamisch	YASA_TIME	$P_1 > P_2$
Kürzeste Zeit	Dynamisch	YASA_TIME	$P_1 < P_2$

Benutzeroberfläche YASA

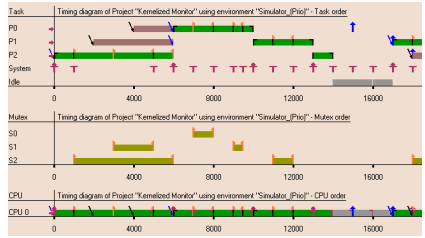
- Objekt-orientierte Programmierung (261 Klassen)
- Plattformunabhängig durch Klassenbibliothek Qt
- Nutzung von Entwurfsmustern
- Multilingual (UNICODE)
- Vollständige Projekt-Konfiguration
- Auswertung in grafischer und tabellarischer Form

Taskdiagramm
Mutexdiagramm
Prozessordiagramm

Tasktabelle
Mutextabelle
Prozessortabelle

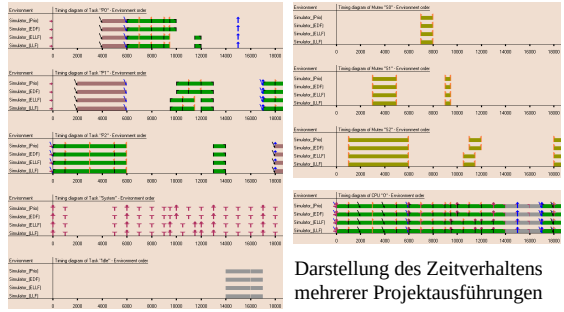


Grafische Auswertung I



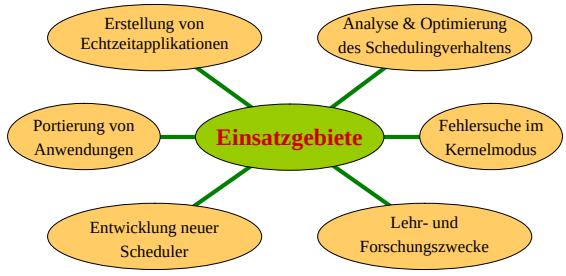
Darstellung des Zeitverhaltens einer Projektausführung

Grafische Auswertung II



Darstellung des Zeitverhaltens mehrerer Projektausführungen

Einsatzgebiete



Zusammenfassung YASA

- Entwicklungsframework für bestehende Echtzeitbetriebssysteme
  - Optimierung des Schedulingverhaltens von Echtzeitanwendungen
  - Virtuelle Schedulingumgebung
  - Plattformunabhängige Scheduler & Synchronisationsprotokolle
  - Komponenten zur Protokollierung & Deadlinebehandlung
  - Umfangreiche Auswertungen in der Benutzeroberfläche
- Ergebnisse
  - Veröffentlichung: 13th IEEE International RSP Workshop, Darmstadt, 2002
  - Anfragen der Industrie (ABB)
- Homepage
  http://yasa.e-technik.uni-rostock.de
  http://sourceforge.net/projects/yasa

Vielen Dank