Real-Time Operating Systems – Ein Überblick

Stefan Tittel

Universität Dortmund

Proseminar: Werkzeuge und Techniken zur Spezifikation, Simulation und Implementierung von eingebetteten Systemen, 2004

Inhaltsübersicht

Inhaltsübersicht

Inhaltsübersicht

Inhaltsübersicht

Was ist ein Operating System?

Definition

Das Operating System ist das Programm, welches beim Start eines Rechners geladen wird. Es dient als eine Schnittstelle zwischen den Rechnerkomponenten wie Hardware und BIOS und den Anwendungen, die auf dem Rechner laufen. Es bietet außerdem grundlegende Funktionen für die Verwaltung und Pflege des Operating System und Dateisystems.

Aufgaben eines Operating System

Verwaltet:

- Rechenleistung (Scheduling)
- Speicher
- Dateisystem
- Geräte

Stellt bereit:

- Hardwareabstraktion gegenüber Applikationen
- Benutzerschnittstelle gegenüber Anwendern

Berücksichtigt dabei:

- Effizienz
- Sicherheit

Aufgaben eines Operating System

Verwaltet:

- Rechenleistung (Scheduling)
- Speicher
- Dateisystem
- Geräte

Stellt bereit:

- Hardwareabstraktion gegenüber Applikationen
- Benutzerschnittstelle gegenüber Anwendern

Berücksichtigt dabei:

- Effizienz
- Sicherheit

Aufgaben eines Operating System

Verwaltet:

- Rechenleistung (Scheduling)
- Speicher
- Dateisystem
- Geräte

Stellt bereit:

- Hardwareabstraktion gegenüber Applikationen
- Benutzerschnittstelle gegenüber Anwendern

Berücksichtigt dabei:

- Effizienz
- Sicherheit

Aufgaben eines Operating System

Verwaltet:

- Rechenleistung (Scheduling)
- Speicher
- Dateisystem
- Geräte

Stellt bereit:

- Hardwareabstraktion gegenüber Applikationen
- Benutzerschnittstelle gegenüber Anwendern

Berücksichtigt dabei:

- Effizienz
- Sicherheit

Aufgaben eines Operating System

Verwaltet:

- Rechenleistung (Scheduling)
- Speicher
- Dateisystem
- Geräte

Stellt bereit:

- Hardwareabstraktion gegenüber Applikationen
- Benutzerschnittstelle gegenüber Anwendern

Berücksichtigt dabei:

- Effizienz
- Sicherheit

Aufgaben eines Operating System

Verwaltet:

- Rechenleistung (Scheduling)
- Speicher
- Dateisystem
- Geräte

Stellt bereit:

- Hardwareabstraktion gegenüber Applikationen
- Benutzerschnittstelle gegenüber Anwendern

Berücksichtigt dabei:

- Effizienz
- Sicherheit

Aufgaben eines Operating System

Verwaltet:

- Rechenleistung (Scheduling)
- Speicher
- Dateisystem
- Geräte

Stellt bereit:

- Hardwareabstraktion gegenüber Applikationen
- Benutzerschnittstelle gegenüber Anwendern

Berücksichtigt dabei:

- Effizienz
- Sicherheit

Aufgaben eines Operating System

Verwaltet:

- Rechenleistung (Scheduling)
- Speicher
- Dateisystem
- Geräte

Stellt bereit:

- Hardwareabstraktion gegenüber Applikationen
- Benutzerschnittstelle gegenüber Anwendern

Berücksichtigt dabei:

- Effizienz
- Sicherheit

Aufgaben eines Operating System

Verwaltet:

- Rechenleistung (Scheduling)
- Speicher
- Dateisystem
- Geräte

Stellt bereit:

- Hardwareabstraktion gegenüber Applikationen
- Benutzerschnittstelle gegenüber Anwendern

Berücksichtigt dabei:

- Effizienz
- Sicherheit

Aufgaben eines Operating System

Verwaltet:

- Rechenleistung (Scheduling)
- Speicher
- Dateisystem
- Geräte

Stellt bereit:

- Hardwareabstraktion gegenüber Applikationen
- Benutzerschnittstelle gegenüber Anwendern

Berücksichtigt dabei:

- Effizienz
- Sicherheit

Aufgaben eines Operating System

Verwaltet:

- Rechenleistung (Scheduling)
- Speicher
- Dateisystem
- Geräte

Stellt bereit:

- Hardwareabstraktion gegenüber Applikationen
- Benutzerschnittstelle gegenüber Anwendern

Berücksichtigt dabei:

- Effizienz
- Sicherheit

Eigenschaften eines Operating System

- Ein Operating System ist im Allgemeinen sehr komplex.
- Ein Operating System versucht im Allgemeinen unterschiedliche, teils gegensätzliche Anforderungen gut zu erfüllen (z. B. Effizienz trotz Sicherheit).
- Das Verhalten des Operating System ist dadurch nicht mit vertretbarem Aufwand determinierbar.

Eigenschaften eines Operating System

- Ein Operating System ist im Allgemeinen sehr komplex.
- Ein Operating System versucht im Allgemeinen unterschiedliche, teils gegensätzliche Anforderungen gut zu erfüllen (z. B. Effizienz trotz Sicherheit).
- Das Verhalten des Operating System ist dadurch nicht mit vertretbarem Aufwand determinierbar.

Eigenschaften eines Operating System

- Ein Operating System ist im Allgemeinen sehr komplex.
- Ein Operating System versucht im Allgemeinen unterschiedliche, teils gegensätzliche Anforderungen gut zu erfüllen (z. B. Effizienz trotz Sicherheit).
- Das Verhalten des Operating System ist dadurch nicht mit vertretbarem Aufwand determinierbar.

Inhaltsübersicht

Motivation

Oftmals muss ein System auf ein Ereignis in gewissen Grenzen temporal determinierbar reagieren. Dies trifft häufig in besonderem Maße auf eingebettete Systeme zu.

Beispiele

- Der Ticketautomat an der Bahnhaltestelle
- Das Anti-Blockier-System im Kraftfahrzeug
- Die Notabschaltung im Kernkraftwerk
- Der Datenbus im Bike

Motivation

Oftmals muss ein System auf ein Ereignis in gewissen Grenzen temporal determinierbar reagieren. Dies trifft häufig in besonderem Maße auf eingebettete Systeme zu.

Beispiele

- Der Ticketautomat an der Bahnhaltestelle
- Das Anti-Blockier-System im Kraftfahrzeug
- Die Notabschaltung im Kernkraftwerk
- Der Drucker im Büro

Motivation

Oftmals muss ein System auf ein Ereignis in gewissen Grenzen temporal determinierbar reagieren. Dies trifft häufig in besonderem Maße auf eingebettete Systeme zu.

Beispiele

- Der Ticketautomat an der Bahnhaltestelle
- Das Anti-Blockier-System im Kraftfahrzeug
- Die Notabschaltung im Kernkraftwerk
- Der Drucker im Büro

Motivation

Oftmals muss ein System auf ein Ereignis in gewissen Grenzen temporal determinierbar reagieren. Dies trifft häufig in besonderem Maße auf eingebettete Systeme zu.

Beispiele

- Der Ticketautomat an der Bahnhaltestelle
- Das Anti-Blockier-System im Kraftfahrzeug
- Die Notabschaltung im Kernkraftwerk
- Der Drucker im Büro

Motivation

Oftmals muss ein System auf ein Ereignis in gewissen Grenzen temporal determinierbar reagieren. Dies trifft häufig in besonderem Maße auf eingebettete Systeme zu.

Beispiele

- Der Ticketautomat an der Bahnhaltestelle
- Das Anti-Blockier-System im Kraftfahrzeug
- Die Notabschaltung im Kernkraftwerk
- Der Drucker im Büro

Definition: Real-Time (Operating) System

Die Korrektheit eines Systems hängt normalerweise ausschließlich von der Korrektheit der ausgeführten Berechnungen ab. Offenbar ist das aber nicht immer ausreichend.

Definition

- Ein Real-Time System ist ein System, dessen Korrektheit nicht nur korrekte Berechnungen verlangt, sondern darüber hinaus zeitliche Vorgaben über das Eintreffen des Ergebnisses macht.
- Ein Real-Time System reagiert auf zeitlich vorhersehbare Weise auf das Eintreten externer Stimuli.

Definition

Definition: Real-Time (Operating) System

Die Korrektheit eines Systems hängt normalerweise ausschließlich von der Korrektheit der ausgeführten Berechnungen ab. Offenbar ist das aber nicht immer ausreichend.

Definition

- Ein Real-Time System ist ein System, dessen Korrektheit nicht nur korrekte Berechnungen verlangt, sondern darüber hinaus zeitliche Vorgaben über das Eintreffen des Ergebnisses macht.
- Ein Real-Time System reagiert auf zeitlich vorhersehbare Weise auf das Eintreten externer Stimuli.

Definition

Ein Real-Time Operating System ist ein Operating System mit den notwendigen Eigenschaften zum Betrieb eines Real-Time System.

Definition: Real-Time (Operating) System

Die Korrektheit eines Systems hängt normalerweise ausschließlich von der Korrektheit der ausgeführten Berechnungen ab. Offenbar ist das aber nicht immer ausreichend.

Definition

- Ein Real-Time System ist ein System, dessen Korrektheit nicht nur korrekte Berechnungen verlangt, sondern darüber hinaus zeitliche Vorgaben über das Eintreffen des Ergebnisses macht.
- Ein Real-Time System reagiert auf zeitlich vorhersehbare Weise auf das Eintreten externer Stimuli.

Definition

Ein Real-Time Operating System ist ein Operating System mit den notwendigen Eigenschaften zum Betrieb eines Real-Time System.

Definition: Real-Time (Operating) System

Die Korrektheit eines Systems hängt normalerweise ausschließlich von der Korrektheit der ausgeführten Berechnungen ab. Offenbar ist das aber nicht immer ausreichend.

Definition

- Ein Real-Time System ist ein System, dessen Korrektheit nicht nur korrekte Berechnungen verlangt, sondern darüber hinaus zeitliche Vorgaben über das Eintreffen des Ergebnisses macht.
- Ein Real-Time System reagiert auf zeitlich vorhersehbare Weise auf das Eintreten externer Stimuli.

Definition

Ein Real-Time Operating System ist ein Operating System mit den notwendigen Eigenschaften zum Betrieb eines Real-Time System.

Inhaltsübersicht

Soft Real-Time System

Definition

Stellt in einem System das gelegentliche Verpassen von Deadlines eine akzeptable Qualitätsminderung dar, so handelt es sich um ein Soft Real-Time System.

Aufgrund dieser relativ moderaten Anforderung an das zeitlich deterministische Verhalten des OS, genügt es oft, bestehende Operating Systems leicht zu modifizieren.

Beispiele

Ticketautomat, Drucker

Soft Real-Time System

Definition

Stellt in einem System das gelegentliche Verpassen von Deadlines eine akzeptable Qualitätsminderung dar, so handelt es sich um ein Soft Real-Time System.

Aufgrund dieser relativ moderaten Anforderung an das zeitlich deterministische Verhalten des OS, genügt es oft, bestehende Operating Systems leicht zu modifizieren.

Beispiele

Ticketautomat, Drucker

Soft Real-Time System

Definition

Stellt in einem System das gelegentliche Verpassen von Deadlines eine akzeptable Qualitätsminderung dar, so handelt es sich um ein Soft Real-Time System.

Aufgrund dieser relativ moderaten Anforderung an das zeitlich deterministische Verhalten des OS, genügt es oft, bestehende Operating Systems leicht zu modifizieren.

Beispiele

Ticketautomat, Drucker

Firm Real-Time System

Definition

Stellt in einem System das Verpassen von Deadlines eine nicht akzeptable Qualitätsminderung dar, so handelt es sich um ein Firm Real-Time System.

Beispiele

Videorekorder, Autonavigationssystem

Firm Real-Time System

Definition

Stellt in einem System das Verpassen von Deadlines eine nicht akzeptable Qualitätsminderung dar, so handelt es sich um ein Firm Real-Time System.

Beispiele

Videorekorder, Autonavigationssystem

Hard Real-Time System

Definition

Wenn das Verpassen einer Deadline katastrophale Folgen nach sich zieht, handelt es sich um ein Hard Real-Time System.

Aufgrund dieser sehr strikten Anforderung ist die Modifikation eines herkömmlichen OS hin zu einem Hard RTOS deutlich schwieriger.

Beispiele

NMD, ABS, AKW

Hard Real-Time System

Definition

Wenn das Verpassen einer Deadline katastrophale Folgen nach sich zieht, handelt es sich um ein Hard Real-Time System.

Aufgrund dieser sehr strikten Anforderung ist die Modifikation eines herkömmlichen OS hin zu einem Hard RTOS deutlich schwieriger.

Beispiele

NMD, ABS, AKW

Hard Real-Time System

Definition

Wenn das Verpassen einer Deadline katastrophale Folgen nach sich zieht, handelt es sich um ein Hard Real-Time System.

Aufgrund dieser sehr strikten Anforderung ist die Modifikation eines herkömmlichen OS hin zu einem Hard RTOS deutlich schwieriger.

Beispiele

NMD, ABS, AKW

Inhaltsübersicht

Worst-Case Execution Time (WCET)

Definition

Die Worst-Case Execution Time (WCET) ist die obere Grenze der Ausführungszeit eines Task.

Zu wissen, wie lange ein Task maximal zur Ausführung benötigt, ist für das Scheduling in einem Real-Time System sehr günstig. Viele Scheduling-Algorithmen nutzen die WCET als Grundlage.

Hinweis

Nicht jeder Task verfügt über eine WCET!

Worst-Case Execution Time (WCET)

Definition

Die Worst-Case Execution Time (WCET) ist die obere Grenze der Ausführungszeit eines Task.

Zu wissen, wie lange ein Task maximal zur Ausführung benötigt, ist für das Scheduling in einem Real-Time System sehr günstig. Viele Scheduling-Algorithmen nutzen die WCET als Grundlage.

Hinweis

Nicht jeder Task verfügt über eine WCET!

Worst-Case Execution Time (WCET)

Definition

Die Worst-Case Execution Time (WCET) ist die obere Grenze der Ausführungszeit eines Task.

Zu wissen, wie lange ein Task maximal zur Ausführung benötigt, ist für das Scheduling in einem Real-Time System sehr günstig. Viele Scheduling-Algorithmen nutzen die WCET als Grundlage.

Hinweis

Nicht jeder Task verfügt über eine WCET!

Periodische und nicht-periodische Tasks

Definition

Ein Task, der alle p Zeiteinheiten ausgeführt wird, heißt periodisch; p ist dabei seine Periode.

Beispiel

AKW-Abschaltung: Temperaturabfrage

Definition

Tasks, die nicht periodisch sind, heißen nicht-periodisch.

Definition

Nicht-periodische Tasks, die zu nicht vorhersehbaren Zeitpunkten eintreffen, heißen sporadisch.

Beispiel

Benutzerschnittstelle am Ticketautomat

Periodische und nicht-periodische Tasks

Definition

Ein Task, der alle p Zeiteinheiten ausgeführt wird, heißt periodisch; p ist dabei seine Periode.

Beispiel

AKW-Abschaltung: Temperaturabfrage

Definition

Tasks, die nicht periodisch sind, heißen nicht-periodisch.

Definition

Nicht-periodische Tasks, die zu nicht vorhersehbaren Zeitpunkten eintreffen, heißen sporadisch.

Beispiel

Benutzerschnittstelle am Ticketautomat

Periodische und nicht-periodische Tasks

Definition

Ein Task, der alle p Zeiteinheiten ausgeführt wird, heißt periodisch; p ist dabei seine Periode.

Beispiel

AKW-Abschaltung: Temperaturabfrage

Definition

Tasks, die nicht periodisch sind, heißen nicht-periodisch.

Definition

Nicht-periodische Tasks, die zu nicht vorhersehbaren Zeitpunkten eintreffen, heißen sporadisch.

Beispiel

Benutzerschnittstelle am Ticketautomat

Periodische und nicht-periodische Tasks

Definition

Ein Task, der alle p Zeiteinheiten ausgeführt wird, heißt periodisch; p ist dabei seine Periode.

Beispiel

AKW-Abschaltung: Temperaturabfrage

Definition

Tasks, die nicht periodisch sind, heißen nicht-periodisch.

Definition

Nicht-periodische Tasks, die zu nicht vorhersehbaren Zeitpunkten eintreffen, heißen sporadisch.

Beispiel

Benutzerschnittstelle am Ticketautomat

Periodische und nicht-periodische Tasks

Definition

Ein Task, der alle p Zeiteinheiten ausgeführt wird, heißt periodisch; p ist dabei seine Periode.

Beispiel

AKW-Abschaltung: Temperaturabfrage

Definition

Tasks, die nicht periodisch sind, heißen nicht-periodisch.

Definition

Nicht-periodische Tasks, die zu nicht vorhersehbaren Zeitpunkten eintreffen, heißen sporadisch.

Beispiel

Benutzerschnittstelle am Ticketautomat

Dynamisches und statisches Scheduling

Definition

Werden die Scheduling-Entscheidungen während der Laufzeit getroffen, so handelt es sich um dynamisches Scheduling.

Definition

Werden die Scheduling-Entscheidungen bereits zur Zeit der Entwicklung festgelegt, handelt es sich um statisches Scheduling.

Definition

Systeme mit Zeitgeber, die Tasks ausschließlich nach Start- und Stoppzeiten schedulen, heißen entirely time triggered (TT).

In TT-Systemen findet sich ausschließlich statisches Scheduling.

Dynamisches und statisches Scheduling

Definition

Werden die Scheduling-Entscheidungen während der Laufzeit getroffen, so handelt es sich um dynamisches Scheduling.

Definition

Werden die Scheduling-Entscheidungen bereits zur Zeit der Entwicklung festgelegt, handelt es sich um statisches Scheduling.

Definition

Systeme mit Zeitgeber, die Tasks ausschließlich nach Start- und Stoppzeiten schedulen, heißen entirely time triggered (TT).

In TT-Systemen findet sich ausschließlich statisches Scheduling.

Dynamisches und statisches Scheduling

Definition

Werden die Scheduling-Entscheidungen während der Laufzeit getroffen, so handelt es sich um dynamisches Scheduling.

Definition

Werden die Scheduling-Entscheidungen bereits zur Zeit der Entwicklung festgelegt, handelt es sich um statisches Scheduling.

Definition

Systeme mit Zeitgeber, die Tasks ausschließlich nach Start- und Stoppzeiten schedulen, heißen entirely time triggered (TT).

In TT-Systemen findet sich ausschließlich statisches Scheduling.

Dynamisches und statisches Scheduling

Definition

Werden die Scheduling-Entscheidungen während der Laufzeit getroffen, so handelt es sich um dynamisches Scheduling.

Definition

Werden die Scheduling-Entscheidungen bereits zur Zeit der Entwicklung festgelegt, handelt es sich um statisches Scheduling.

Definition

Systeme mit Zeitgeber, die Tasks ausschließlich nach Start- und Stoppzeiten schedulen, heißen entirely time triggered (TT).

In TT-Systemen findet sich ausschließlich statisches Scheduling.

Kooperatives und präemptives Scheduling

Definition

Nimmt der Scheduler einen Kontextwechsel nur dann vor, wenn der laufende Task dies explizit anbietet oder beendet ist, handelt es sich um kooperatives Scheduling.

Beispiele

Windows≤3.11, Videorekorder

Definition

Nimmt der Scheduler einen Kontextwechsel vor, ohne dass die Bedingungen für kooperatives Scheduling erfüllt sind, handelt es sich um präemptives Scheduling.

Beispiele

Windows≥95, Linux, OS/2, Solaris, BetaNova

Kooperatives und präemptives Scheduling

Definition

Nimmt der Scheduler einen Kontextwechsel nur dann vor, wenn der laufende Task dies explizit anbietet oder beendet ist, handelt es sich um kooperatives Scheduling.

Beispiele

Windows$\leq$3.11, Videorekorder

Definition

Nimmt der Scheduler einen Kontextwechsel vor, ohne dass die Bedingungen für kooperatives Scheduling erfüllt sind, handelt es sich um präemptives Scheduling.

Beispiele

Windows$\geq$95, Linux, OS/2, Solaris, BetaNova

Kooperatives und präemptives Scheduling

Definition

Nimmt der Scheduler einen Kontextwechsel nur dann vor, wenn der laufende Task dies explizit anbietet oder beendet ist, handelt es sich um kooperatives Scheduling.

Beispiele

Windows$\leq$3.11, Videorekorder

Definition

Nimmt der Scheduler einen Kontextwechsel vor, ohne dass die Bedingungen für kooperatives Scheduling erfüllt sind, handelt es sich um präemptives Scheduling.

Beispiele

Windows$\geq$95, Linux, OS/2, Solaris, BetaNova

Kooperatives und präemptives Scheduling

Definition

Nimmt der Scheduler einen Kontextwechsel nur dann vor, wenn der laufende Task dies explizit anbietet oder beendet ist, handelt es sich um kooperatives Scheduling.

Beispiele

Windows$\leq$3.11, Videorekorder

Definition

Nimmt der Scheduler einen Kontextwechsel vor, ohne dass die Bedingungen für kooperatives Scheduling erfüllt sind, handelt es sich um präemptives Scheduling.

Beispiele

Windows$\geq$95, Linux, OS/2, Solaris, BetaNova

Weitere Merkmale

- Ein- oder Mehrprozessor-Scheduling
- Online- oder Offline-Scheduling
- Scheduling mit abhängigen oder unabhängigen Tasks

Weitere Merkmale

- Ein- oder Mehrprozessor-Scheduling
- Online- oder Offline-Scheduling
- Scheduling mit abhängigen oder unabhängigen Tasks

Weitere Merkmale

- Ein- oder Mehrprozessor-Scheduling
- Online- oder Offline-Scheduling
- Scheduling mit abhängigen oder unabhängigen Tasks

Weitere Definitionen

Definition

Die Differenz zwischen der Ausführungszeit eines Tasks und dem Deadline-Intervall heißt Laxity.

Definition

Die Differenz zwischen der Fertigstellungszeit und der Deadline – maximiert über alle Tasks – heißt Maximum Lateness. Wenn kein Task seine Deadline überschreitet, ist die Maximum Lateness negativ.

Definition

Ein Menge von Tasks heißt schedulable, wenn ein Schedule der Form existiert, dass jeder Task seine Deadline erfüllt.

Weitere Definitionen

Definition

Die Differenz zwischen der Ausführungszeit eines Tasks und dem Deadline-Intervall heißt Laxity.

Definition

Die Differenz zwischen der Fertigstellungszeit und der Deadline – maximiert über alle Tasks – heißt Maximum Lateness. Wenn kein Task seine Deadline überschreitet, ist die Maximum Lateness negativ.

Definition

Ein Menge von Tasks heißt schedulable, wenn ein Schedule der Form existiert, dass jeder Task seine Deadline erfüllt.

Weitere Definitionen

Definition

Die Differenz zwischen der Ausführungszeit eines Tasks und dem Deadline-Intervall heißt Laxity.

Definition

Die Differenz zwischen der Fertigstellungszeit und der Deadline – maximiert über alle Tasks – heißt Maximum Lateness. Wenn kein Task seine Deadline überschreitet, ist die Maximum Lateness negativ.

Definition

Ein Menge von Tasks heißt schedulable, wenn ein Schedule der Form existiert, dass jeder Task seine Deadline erfüllt.

Inhaltsübersicht

Hinweis

Hinweis

Die im folgenden besprochenen Algorithmen sind alle optimal hinsichtlich der Schedulability, einige darüber hinaus optimal in Hinblick auf die Maximum Lateness.

Earliest Due Date (EDD)

Voraussetzungen nicht-periodische, voneinander unabhängige, gleichzeitig eintreffende Tasks; Einprozessorsystem

Merkmale kooperativ; kann statisch implementiert werden, wenn Ausführungszeiten aller Tasks bekannt

Vorgehensweise

1. Sortiere alle Tasks nach Deadline aufsteigend.
2. Führe die Tasks gemäß dieser Sortierreihenfolge aus.

Earliest Due Date (EDD)

Voraussetzungen nicht-periodische, voneinander unabhängige, gleichzeitig eintreffende Tasks; Einprozessorsystem

Merkmale kooperativ; kann statisch implementiert werden, wenn Ausführungszeiten aller Tasks bekannt

Vorgehensweise

1. Sortiere alle Tasks nach Deadline aufsteigend.
2. Führe die Tasks gemäß dieser Sortierreihenfolge aus.

Earliest Due Date (EDD)

Voraussetzungen nicht-periodische, voneinander unabhängige, gleichzeitig eintreffende Tasks; Einprozessorsystem

Merkmale kooperativ; kann statisch implementiert werden, wenn Ausführungszeiten aller Tasks bekannt

Vorgehensweise

1. Sortiere alle Tasks nach Deadline aufsteigend.
2. Führe die Tasks gemäß dieser Sortierreihenfolge aus.

Earliest Due Date (EDD)

Voraussetzungen nicht-periodische, voneinander unabhängige, gleichzeitig eintreffende Tasks; Einprozessorsystem

Merkmale kooperativ; kann statisch implementiert werden, wenn Ausführungszeiten aller Tasks bekannt

Vorgehensweise

1. Sortiere alle Tasks nach Deadline aufsteigend.
2. Führe die Tasks gemäß dieser Sortierreihenfolge aus.

Earliest Deadline First (EDF)

Voraussetzungen nicht-periodische, voneinander unabhängige Tasks; Einprozessorsystem

Merkmale präemptiv; dynamisch

Vorgehensweise

1. Führe immer den Task mit der am nächsten liegenden Deadline aus (ähnlich EDD)
2. Bei Neuankunft eines Tasks mit früherer Deadline, wechsele zu diesem.

Earliest Deadline First (EDF)

Voraussetzungen nicht-periodische, voneinander unabhängige Tasks; Einprozessorsystem

Merkmale präemptiv; dynamisch

Vorgehensweise

1. Führe immer den Task mit der am nächsten liegenden Deadline aus (ähnlich EDD).
2. Bei Neuankunft eines Tasks mit früherer Deadline, wechsele zu diesem.

Earliest Deadline First (EDF)

Voraussetzungen nicht-periodische, voneinander unabhängige Tasks; Einprozessorsystem

Merkmale präemptiv; dynamisch

Vorgehensweise

1. Führe immer den Task mit der am nächsten liegenden Deadline aus (ähnlich EDD).
2. Bei Neuankunft eines Tasks mit früherer Deadline, wechsele zu diesem.

Earliest Deadline First (EDF)

Voraussetzungen nicht-periodische, voneinander unabhängige Tasks; Einprozessorsystem

Merkmale präemptiv; dynamisch

Vorgehensweise

1. Führe immer den Task mit der am nächsten liegenden Deadline aus (ähnlich EDD).
2. Bei Neuankunft eines Tasks mit früherer Deadline, wechsele zu diesem.

Least Laxity (LL)

Voraussetzungen nicht-periodische, voneinander unabhängige Tasks, von denen die Länge ihrer Ausführungszeit im Voraus bekannt ist; Einprozessorsystem

Merkmale präemptiv; dynamisch

Vorgehensweise

Führe immer den Task mit der geringsten Laxity aus.

Hinweis

Führt zu dynamischen Prioritäten, die in einem Standard-OS meist nicht implementiert sind.

Least Laxity (LL)

Voraussetzungen nicht-periodische, voneinander unabhängige Tasks, von denen die Länge ihrer Ausführungszeit im Voraus bekannt ist; Einprozessorsystem

Merkmale präemptiv; dynamisch

Vorgehensweise

Führe immer den Task mit der geringsten Laxity aus.

Hinweis

Führt zu dynamischen Prioritäten, die in einem Standard-OS meist nicht implementiert sind.

Least Laxity (LL)

Voraussetzungen nicht-periodische, voneinander unabhängige Tasks, von denen die Länge ihrer Ausführungszeit im Voraus bekannt ist; Einprozessorsystem

Merkmale präemptiv; dynamisch

Vorgehensweise

Führe immer den Task mit der geringsten Laxity aus.

Hinweis

Führt zu dynamischen Prioritäten, die in einem Standard-OS meist nicht implementiert sind.

Least Laxity (LL)

Voraussetzungen	nicht-periodische, voneinander unabhängige Tasks, von denen die Länge ihrer Ausführungszeit im Voraus bekannt ist; Einprozessorsystem
Merkmale	präemptiv; dynamisch

Vorgehensweise

Führe immer den Task mit der geringsten Laxity aus.

Hinweis

Führt zu dynamischen Prioritäten, die in einem Standard-OS meist nicht implementiert sind.

Latest Deadline First (LDF)

Voraussetzungen nicht-periodische, voneinander abhängige, gleichzeitig eintreffende Tasks, von denen die Länge ihrer Ausführungszeit im Voraus bekannt ist; Abhängigkeiten liegen als Halb- oder vollständige Ordnung vor; Einprozessorsystem

Merkmale kooperativ; dynamisch

Vorgehensweise

1. Liegen die Abhängigkeiten als Halbordnung vor, erzeuge mittels topologischer Sortierung eine vollständige Ordnung.
2. Führe die Tasks in der Reihenfolge der vollständigen Ordnung aus.

Latest Deadline First (LDF)

Voraussetzungen nicht-periodische, voneinander abhängige, gleichzeitig eintreffende Tasks, von denen die Länge ihrer Ausführungszeit im Voraus bekannt ist; Abhängigkeiten liegen als Halb- oder vollständige Ordnung vor; Einprozessorsystem

Merkmale kooperativ; dynamisch

Vorgehensweise

1. Liegen die Abhängigkeiten als Halbordnung vor, erzeuge mittels topologischer Sortierung eine vollständige Ordnung.
2. Führe die Tasks in der Reihenfolge der vollständigen Ordnung aus.

Latest Deadline First (LDF)

Voraussetzungen nicht-periodische, voneinander abhängige, gleichzeitig eintreffende Tasks, von denen die Länge ihrer Ausführungszeit im Voraus bekannt ist; Abhängigkeiten liegen als Halb- oder vollständige Ordnung vor; Einprozessorsystem

Merkmale kooperativ; dynamisch

Vorgehensweise

1. Liegen die Abhängigkeiten als Halbordnung vor, erzeuge mittels topologischer Sortierung eine vollständige Ordnung.
2. Führe die Tasks in der Reihenfolge der vollständigen Ordnung aus.

Latest Deadline First (LDF)

Voraussetzungen	nicht-periodische, voneinander abhängige, gleichzeitig eintreffende Tasks, von denen die Länge ihrer Ausführungszeit im Voraus bekannt ist; Abhängigkeiten liegen als Halb- oder vollständige Ordnung vor; Einprozessorsystem
Merkmale	kooperativ; dynamisch

Vorgehensweise

1. Liegen die Abhängigkeiten als Halbordnung vor, erzeuge mittels topologischer Sortierung eine vollständige Ordnung.
2. Führe die Tasks in der Reihenfolge der vollständigen Ordnung aus.

Rate Monotonic Scheduling (RM)

Voraussetzungen

- Periodische, voneinander unabhängige Tasks
- Deadline und Periodendauer bei jedem einzelnen Task gleich lang
- Länge alle Ausführungszeiten im Voraus bekannt
- Es gilt $\sum_{i=1}^{n} \frac{c_i}{p_i} \leq n(2^{\frac{1}{n}} - 1)$, wobei $i \in \{\text{Tasks}\}$, c_i Ausführungszeit, d_i Deadline-Intervall, p_i Periodendauer und n Anzahl der Tasks

Vorgehensweise

Vergib Prioritäten nach Periodenlänge: Je kürzer p_i desto höher die Priorität.

Rate Monotonic Scheduling (RM)

Voraussetzungen

- Periodische, voneinander unabhängige Tasks
- Deadline und Periodendauer bei jedem einzelnen Task gleich lang
- Länge alle Ausführungszeiten im Voraus bekannt
- Es gilt $\sum_{i=1}^{n} \frac{c_i}{p_i} \leq n(2^{\frac{1}{n}} - 1)$, wobei $i \in \{\text{Tasks}\}$, c_i Ausführungszeit, d_i Deadline-Intervall, p_i Periodendauer und n Anzahl der Tasks

Vorgehensweise

Vergib Prioritäten nach Periodenlänge: Je kürzer p_i desto höher die Priorität.

Rate Monotonic Scheduling (RM)

Voraussetzungen

- Periodische, voneinander unabhängige Tasks
- Deadline und Periodendauer bei jedem einzelnen Task gleich lang
- Länge alle Ausführungszeiten im Voraus bekannt
- Es gilt $\sum_{i=1}^{n} \frac{c_i}{p_i} \leq n(2^{\frac{1}{n}} - 1)$, wobei $i \in \{\text{Tasks}\}$, c_i Ausführungszeit, d_i Deadline-Intervall, p_i Periodendauer und n Anzahl der Tasks

Vorgehensweise

Vergib Prioritäten nach Periodenlänge: Je kürzer p_i desto höher die Priorität.

Rate Monotonic Scheduling (RM)

Voraussetzungen

- Periodische, voneinander unabhängige Tasks
- Deadline und Periodendauer bei jedem einzelnen Task gleich lang
- Länge alle Ausführungszeiten im Voraus bekannt
- Es gilt $\sum_{i=1}^{n} \frac{c_i}{p_i} \leq n(2^{\frac{1}{n}} - 1)$, wobei $i \in \{\text{Tasks}\}$, c_i Ausführungszeit, d_i Deadline-Intervall, p_i Periodendauer und n Anzahl der Tasks

Vorgehensweise

Vergib Prioritäten nach Periodenlänge: Je kürzer p_i desto höher die Priorität.

Rate Monotonic Scheduling (RM)

Voraussetzungen

- Periodische, voneinander unabhängige Tasks
- Deadline und Periodendauer bei jedem einzelnen Task gleich lang
- Länge alle Ausführungszeiten im Voraus bekannt
- Es gilt $\sum_{i=1}^{n} \frac{c_i}{p_i} \leq n(2^{\frac{1}{n}} - 1)$, wobei $i \in \{\text{Tasks}\}$, c_i Ausführungszeit, d_i Deadline-Intervall, p_i Periodendauer und n Anzahl der Tasks

Vorgehensweise

Vergib Prioritäten nach Periodenlänge: Je kürzer p_i desto höher die Priorität.

Inhaltsübersicht

Tauglichkeit von Linux als RTOS

Linux als Betriebssystem für Soft Real-Time Systems:

- Linux unterstützt die Vergabe von statischen Prioritäten.
- Kernelpatches mit alternativen Schedulern sind verfügbar.
- Dank offener Quellen ist die Entwicklung eines Schedulers mit dynamischen Prioritäten denkbar.

Linux als Betriebssystem für Hard Real-Time Systems:

- Mangelnde Scheduling-Fähigkeiten
- Ungeeignetes Interrupt-Handling
- Optimierung auf Effizienz, nicht auf Einhaltung von Deadlines

Tauglichkeit von Linux als RTOS

Linux als Betriebssystem für Soft Real-Time Systems:

- Linux unterstützt die Vergabe von statischen Prioritäten.
- Kernelpatches mit alternativen Schedulern sind verfügbar.
- Dank offener Quellen ist die Entwicklung eines Schedulers mit dynamischen Prioritäten denkbar.

Linux als Betriebssystem für Hard Real-Time Systems:

- Mangelnde Scheduling-Fähigkeiten
- Ungeeignetes Interrupt-Handling
- Optimierung auf Effizienz, nicht auf Einhaltung von Deadlines

Tauglichkeit von Linux als RTOS

Linux als Betriebssystem für Soft Real-Time Systems:

- Linux unterstützt die Vergabe von statischen Prioritäten.
- Kernelpatches mit alternativen Schedulern sind verfügbar.
- Dank offener Quellen ist die Entwicklung eines Schedulers mit dynamischen Prioritäten denkbar.

Linux als Betriebssystem für Hard Real-Time Systems:

- Mangelnde Scheduling-Fähigkeiten
- Ungeeignetes Interrupt-Handling
- Optimierung auf Effizienz, nicht auf Einhaltung von Deadlines

Tauglichkeit von Linux als RTOS

Linux als Betriebssystem für Soft Real-Time Systems:

- Linux unterstützt die Vergabe von statischen Prioritäten.
- Kernelpatches mit alternativen Schedulern sind verfügbar.
- Dank offener Quellen ist die Entwicklung eines Schedulers mit dynamischen Prioritäten denkbar.

Linux als Betriebssystem für Hard Real-Time Systems:

- Mangelnde Scheduling-Fähigkeiten
- Ungeeignetes Interrupt-Handling
- Optimierung auf Effizienz, nicht auf Einhaltung von Deadlines

Tauglichkeit von Linux als RTOS

Linux als Betriebssystem für Soft Real-Time Systems:

- Linux unterstützt die Vergabe von statischen Prioritäten.
- Kernelpatches mit alternativen Schedulern sind verfügbar.
- Dank offener Quellen ist die Entwicklung eines Schedulers mit dynamischen Prioritäten denkbar.

Linux als Betriebssystem für Hard Real-Time Systems:

- Mangelnde Scheduling-Fähigkeiten
- Ungeeignetes Interrupt-Handling
- Optimierung auf Effizienz, nicht auf Einhaltung von Deadlines

Tauglichkeit von Linux als RTOS

Linux als Betriebssystem für Soft Real-Time Systems:

- Linux unterstützt die Vergabe von statischen Prioritäten.
- Kernelpatches mit alternativen Schedulern sind verfügbar.
- Dank offener Quellen ist die Entwicklung eines Schedulers mit dynamischen Prioritäten denkbar.

Linux als Betriebssystem für Hard Real-Time Systems:

- Mangelnde Scheduling-Fähigkeiten
- Ungeeignetes Interrupt-Handling
- Optimierung auf Effizienz, nicht auf Einhaltung von Deadlines

Tauglichkeit von Linux als RTOS

Linux als Betriebssystem für Soft Real-Time Systems:

- Linux unterstützt die Vergabe von statischen Prioritäten.
- Kernelpatches mit alternativen Schedulern sind verfügbar.
- Dank offener Quellen ist die Entwicklung eines Schedulers mit dynamischen Prioritäten denkbar.

Linux als Betriebssystem für Hard Real-Time Systems:

- Mangelnde Scheduling-Fähigkeiten
- Ungeeignetes Interrupt-Handling
- Optimierung auf Effizienz, nicht auf Einhaltung von Deadlines

Tauglichkeit von Linux als RTOS

Linux als Betriebssystem für Soft Real-Time Systems:

- Linux unterstützt die Vergabe von statischen Prioritäten.
- Kernelpatches mit alternativen Schedulern sind verfügbar.
- Dank offener Quellen ist die Entwicklung eines Schedulers mit dynamischen Prioritäten denkbar.

Linux als Betriebssystem für Hard Real-Time Systems:

- Mangelnde Scheduling-Fähigkeiten
- Ungeeignetes Interrupt-Handling
- Optimierung auf Effizienz, nicht auf Einhaltung von Deadlines

Inhaltsübersicht

Implementierung von Hard Real-Time

- Hardware- und Interruptverwaltung durch Real-Time Kernel
- Linux-Kernel läuft als Idle-Task, wenn Real-Time Kernel unbeschäftigt.
- System erhält zum eigentlichen Kernel- und User-Space einen Realtime-Space.
- Interrupt-Anforderungen des Linux-Kernel werden vom Real-Time Kernel abgefangen.

Beispiele

RT-Linux, RTAI

Implementierung von Hard Real-Time

- Hardware- und Interruptverwaltung durch Real-Time Kernel
- Linux-Kernel läuft als Idle-Task, wenn Real-Time Kernel unbeschäftigt.
- System erhält zum eigentlichen Kernel- und User-Space einen Realtime-Space.
- Interrupt-Anforderungen des Linux-Kernel werden vom Real-Time Kernel abgefangen.

Beispiele

RT-Linux, RTAI

Implementierung von Hard Real-Time

- Hardware- und Interruptverwaltung durch Real-Time Kernel
- Linux-Kernel läuft als Idle-Task, wenn Real-Time Kernel unbeschäftigt.
- System erhält zum eigentlichen Kernel- und User-Space einen Realtime-Space.
- Interrupt-Anforderungen des Linux-Kernel werden vom Real-Time Kernel abgefangen.

Beispiele

RT-Linux, RTAI

Implementierung von Hard Real-Time

- Hardware- und Interruptverwaltung durch Real-Time Kernel
- Linux-Kernel läuft als Idle-Task, wenn Real-Time Kernel unbeschäftigt.
- System erhält zum eigentlichen Kernel- und User-Space einen Realtime-Space.
- Interrupt-Anforderungen des Linux-Kernel werden vom Real-Time Kernel abgefangen.

Beispiele

RT-Linux, RTAI

Implementierung von Hard Real-Time

- Hardware- und Interruptverwaltung durch Real-Time Kernel
- Linux-Kernel läuft als Idle-Task, wenn Real-Time Kernel unbeschäftigt.
- System erhält zum eigentlichen Kernel- und User-Space einen Realtime-Space.
- Interrupt-Anforderungen des Linux-Kernel werden vom Real-Time Kernel abgefangen.

Beispiele

RT-Linux, RTAI

Real-Time Space

Real-Time Tasks im Real-Time Space völlig autonom vom Rest des Systems, d. h.

- kein Schutz durch Linux-Kernel
- kein Einsatz konventioneller Debugger
- keine Interprozess-Kommunikation ohne weitere Maßnahmen, wie z. B.
  - FIFOs
  - Shared Memory
  - Mailboxes
  - Messages

Real-Time Space

Real-Time Tasks im Real-Time Space völlig autonom vom Rest des Systems, d. h.

- kein Schutz durch Linux-Kernel
- kein Einsatz konventioneller Debugger
- keine Interprozess-Kommunikation ohne weitere Maßnahmen, wie z. B.
  - FIFOs
  - Shared Memory
  - Mailboxes
  - Messages

Real-Time Space

Real-Time Tasks im Real-Time Space völlig autonom vom Rest des Systems, d. h.

- kein Schutz durch Linux-Kernel
- kein Einsatz konventioneller Debugger
- keine Interprozess-Kommunikation ohne weitere Maßnahmen, wie z. B.
  - FIFOs
  - Shared Memory
  - Mailboxes
  - Messages

Real-Time Space

Real-Time Tasks im Real-Time Space völlig autonom vom Rest des Systems, d. h.

- kein Schutz durch Linux-Kernel
- kein Einsatz konventioneller Debugger
- keine Interprozess-Kommunikation ohne weitere Maßnahmen, wie z. B.
  - FIFOs
  - Shared Memory
  - Mailboxes
  - Messages

Real-Time Space

Real-Time Tasks im Real-Time Space völlig autonom vom Rest des Systems, d. h.

- kein Schutz durch Linux-Kernel
- kein Einsatz konventioneller Debugger
- keine Interprozess-Kommunikation ohne weitere Maßnahmen, wie z. B.
  - FIFOs
  - Shared Memory
  - Mailboxes
  - Messages

Real-Time Space

Real-Time Tasks im Real-Time Space völlig autonom vom Rest des Systems, d. h.

- kein Schutz durch Linux-Kernel
- kein Einsatz konventioneller Debugger
- keine Interprozess-Kommunikation ohne weitere Maßnahmen, wie z. B.
  - FIFOs
  - Shared Memory
  - Mailboxes
  - Messages

Real-Time Space

Real-Time Tasks im Real-Time Space völlig autonom vom Rest des Systems, d. h.

- kein Schutz durch Linux-Kernel
- kein Einsatz konventioneller Debugger
- keine Interprozess-Kommunikation ohne weitere Maßnahmen, wie z. B.
  - FIFOs
  - Shared Memory
  - Mailboxes
  - Messages

Real-Time Space

Real-Time Tasks im Real-Time Space völlig autonom vom Rest des Systems, d. h.

- kein Schutz durch Linux-Kernel
- kein Einsatz konventioneller Debugger
- keine Interprozess-Kommunikation ohne weitere Maßnahmen, wie z. B.
  - FIFOs
  - Shared Memory
  - Mailboxes
  - Messages