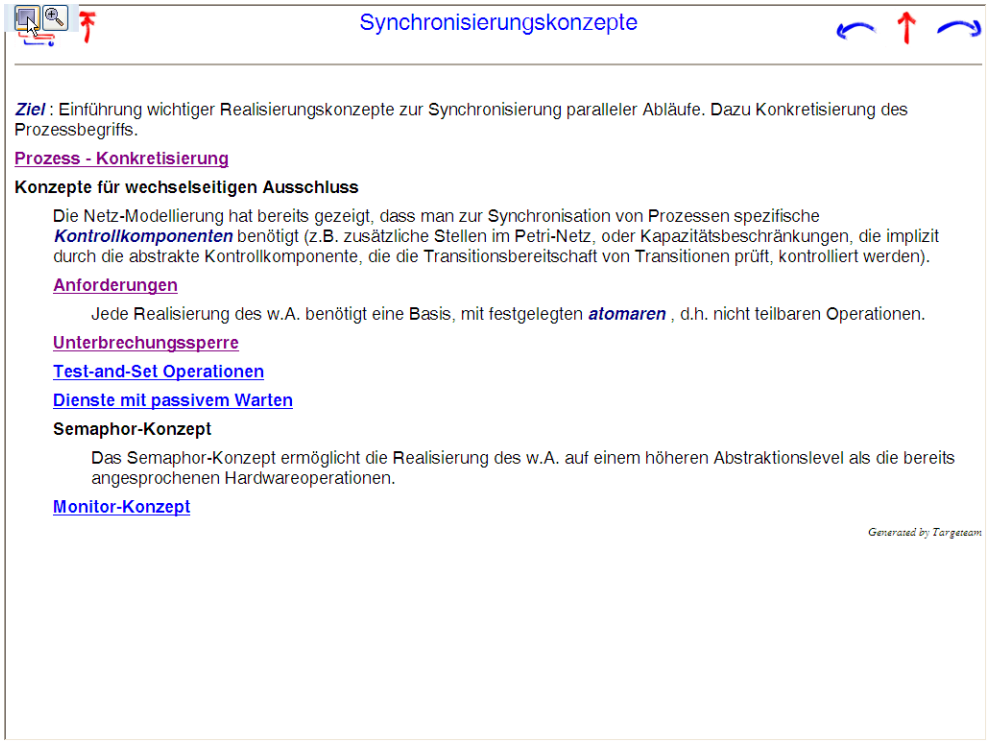
Script **generated by TTT**

Title: Grundlagen_Betriebssysteme (11.11.2015)

Date: Wed Nov 11 13:15:32 CET 2015

Duration: 45:37 min

Pages: 12



Synchronisierungskonzepte

Ziel : Einführung wichtiger Realisierungskonzepte zur Synchronisierung paralleler Abläufe. Dazu Konkretisierung des Prozessbegriffs.

Prozess - Konkretisierung

Konzepte für wechselseitigen Ausschluss

Die Netz-Modellierung hat bereits gezeigt, dass man zur Synchronisation von Prozessen spezifische ***Kontrollkomponenten*** benötigt (z.B. zusätzliche Stellen im Petri-Netz, oder Kapazitätsbeschränkungen, die implizit durch die abstrakte Kontrollkomponente, die die Transitionsbereitschaft von Transitionen prüft, kontrolliert werden).

Anforderungen

Jede Realisierung des w.A. benötigt eine Basis, mit festgelegten ***atomaren*** , d.h. nicht teilbaren Operationen.

Unterbrechungssperre

Test-and-Set Operationen

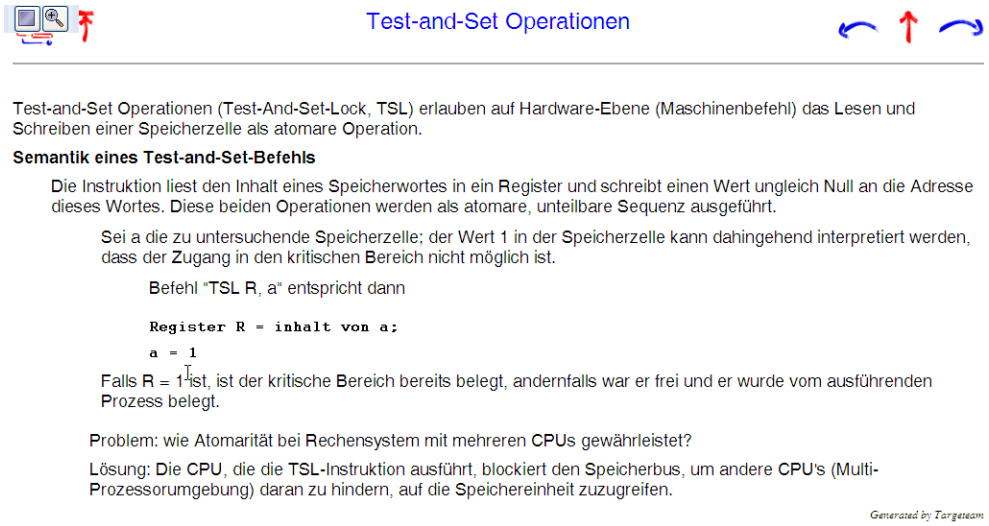
Dienste mit passivem Warten

Semaphor-Konzept

Das Semaphor-Konzept ermöglicht die Realisierung des w.A. auf einem höheren Abstraktionslevel als die bereits angesprochenen Hardwareoperationen.

Monitor-Konzept

Generated by Targeteam



Test-and-Set Operationen

Test-and-Set Operationen (Test-And-Set-Lock, TSL) erlauben auf Hardware-Ebene (Maschinenbefehl) das Lesen und Schreiben einer Speicherzelle als atomare Operation.

Semantik eines Test-and-Set-Befehls

Die Instruktion liest den Inhalt eines Speicherwortes in ein Register und schreibt einen Wert ungleich Null an die Adresse dieses Wortes. Diese beiden Operationen werden als atomare, unteilbare Sequenz ausgeführt.

Sei a die zu untersuchende Speicherzelle; der Wert 1 in der Speicherzelle kann dahingehend interpretiert werden, dass der Zugang in den kritischen Bereich nicht möglich ist.

Befehl "TSL R, a" entspricht dann

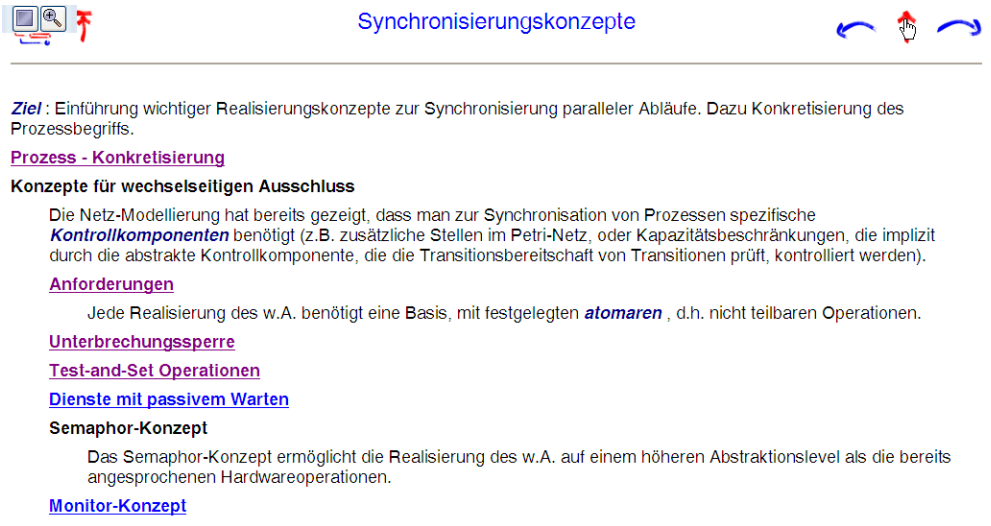
```
Register R = inhalt von a;
a = 1
```

Falls R = 1 ist, ist der kritische Bereich bereits belegt, andernfalls war er frei und er wurde vom ausführenden Prozess belegt.

Problem: wie Atomarität bei Rechensystem mit mehreren CPUs gewährleistet?

Lösung: Die CPU, die die TSL-Instruktion ausführt, blockiert den Speicherbus, um andere CPU's (Multi-Prozessorumgebung) daran zu hindern, auf die Speichereinheit zuzugreifen.

Generated by Targeteam



Synchronisierungskonzepte

Ziel : Einführung wichtiger Realisierungskonzepte zur Synchronisierung paralleler Abläufe. Dazu Konkretisierung des Prozessbegriffs.

Prozess - Konkretisierung

Konzepte für wechselseitigen Ausschluss

Die Netz-Modellierung hat bereits gezeigt, dass man zur Synchronisation von Prozessen spezifische ***Kontrollkomponenten*** benötigt (z.B. zusätzliche Stellen im Petri-Netz, oder Kapazitätsbeschränkungen, die implizit durch die abstrakte Kontrollkomponente, die die Transitionsbereitschaft von Transitionen prüft, kontrolliert werden).

Anforderungen

Jede Realisierung des w.A. benötigt eine Basis, mit festgelegten ***atomaren*** , d.h. nicht teilbaren Operationen.

Unterbrechungssperre

Test-and-Set Operationen

Dienste mit passivem Warten

Semaphor-Konzept

Das Semaphor-Konzept ermöglicht die Realisierung des w.A. auf einem höheren Abstraktionslevel als die bereits angesprochenen Hardwareoperationen.

Monitor-Konzept

Generated by Targeteam

Dienste mit passivem Warten

Unterscheidung zwischen

aktivem Warten : Prozess muss periodisch selber prüfen, ob die Voraussetzungen zum Weiterarbeiten erfüllt sind.

passivem Warten : Prozess wird in Warte-Zustand versetzt und aufgeweckt, wenn das Ereignis, auf das er wartet, eingetreten ist.

Methoden oder Dienste, so dass unter Mithilfe des Betriebssystems ein Prozess in den Zustand ***wartend*** übergeführt wird.

beim Eintreten eines passenden "Weckereignisses" wird der Prozess vom Betriebssystem vom Zustand ***wartend*** in den Zustand ***rechenbereit*** übergeführt.

Beispiel: Java wait und notify.

Synchronisierungskonzepte

Ziel : Einführung wichtiger Realisierungskonzepte zur Synchronisierung paralleler Abläufe. Dazu Konkretisierung des Prozessbegriffs.

Prozess - Konkretisierung

Konzepte für wechselseitigen Ausschluss

Die Netz-Modellierung hat bereits gezeigt, dass man zur Synchronisation von Prozessen spezifische ***Kontrollkomponenten*** benötigt (z.B. zusätzliche Stellen im Petri-Netz, oder Kapazitätsbeschränkungen, die implizit durch die abstrakte Kontrollkomponente, die die Transitionsbereitschaft von Transitionen prüft, kontrolliert werden).

Anforderungen

Jede Realisierung des w.A. benötigt eine Basis, mit festgelegten ***atomaren*** , d.h. nicht teilbaren Operationen.

Unterbrechungssperre

Test-and-Set Operationen

Dienste mit passivem Warten

Semaphor-Konzept

Das Semaphor-Konzept ermöglicht die Realisierung des w.A. auf einem höheren Abstraktionslevel als die bereits angesprochenen Hardwareoperationen.

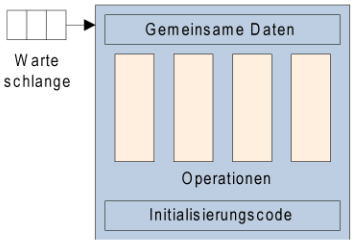
Monitor-Konzept

Monitor-Konzept

Das Monitor-Konzept basiert auf der Idee, die in einem kritischen Bereich bearbeiteten Daten zusammen mit den darauf definierten Zugriffsoperationen in einer sprachlichen Einheit - dem Monitor - zusammenzufassen.

soll Probleme, wie sie bei der Programmierung von Semaphoren auftreten, vermeiden.

zu einem Zeitpunkt höchstens ein Prozess innerhalb des Monitors aktiv.



Definition von Bedingungsvariablen zur Spezifikation anwendungsspezifischer Bedingungen.

Operationen auf Bedingungsvariable `cond`

`cond.wait()`: Prozess wird wartend gesetzt.

`cond.signal()`: ein wartender Prozess wird aktiviert.

Synchronisation

Eine wichtige Systemeigenschaft betrifft die Synchronisation paralleler Ereignisse. Mehrere Prozesse konkurrieren um eine gemeinsame Ressource (CPU), oder greifen auf gemeinsame Daten zu.

Beispiele

Die beiden Beispiele basieren auf der speicherbasierten Prozessinteraktion, d.h. Prozesse (oder auch Threads) interagieren über gemeinsam zugreifbare Speicherzellen.

Beispiel: gemeinsame Daten

Erzeuger-Verbraucher-Problem

Definition: Wechselseitiger Ausschluss

Modellierung

Synchronisierungskonzepte

Semaphore

Synchronisierung von Java Threads

Operationen

Die Operationen P und V sind ***atomar*** ; sie sind unteilbar und sie werden wechselseitig ausgeschlossen ausgeführt.

Sei s die Koordinierungsvariable, dann sind die P und V Operationen wie nachfolgend definiert.

Mögliche Realisierung auf Hardware-Ebene mittels des Test-and-Set Maschinenbefehls.

Informelle Charakterisierung

```
public void P (int s) {
    s = s - 1;
    if ( s < 0 ) { Prozess in die Menge der bzgl. s wartenden Prozesse einreihen }
}

public void V (int s) {
    s = s + 1;
    if ( s ≤ 0 ) { führe genau einen der bzgl. s wartenden Prozesse in den Zustand rechenwillig
    über }
}
```

Binäres Semaphor : die Kontrollvariable s nimmt nur boolesche Werte an.

man spricht auch von Mutex.

Mutex in Posix

Mutex in Posix

einfache Sperre zur Absicherung gemeinsamer Ressourcen

pthread_mutex_init(mutex) ⇒ initialisiert und konfiguriert ein Mutex.

pthread_mutex_lock(mutex) ⇒ entspricht der P-Operation; sperrt ein Mutex.

pthread_mutex_unlock(mutex) ⇒ entspricht der V-Operation; gibt ein Mutex frei.

Mutex Objekte und Semaphore mit Integer Werten stehen auch in Windows zur Verfügung.

Operationen

Die Operationen P und V sind ***atomar*** ; sie sind unteilbar und sie werden wechselseitig ausgeschlossen ausgeführt.

Sei s die Koordinierungsvariable, dann sind die P und V Operationen wie nachfolgend definiert.

Mögliche Realisierung auf Hardware-Ebene mittels des Test-and-Set Maschinenbefehls.

Informelle Charakterisierung

```
public void P (int s) {
    s = s - 1;
    if ( s < 0 ) { Prozess in die Menge der bzgl. s wartenden Prozesse einreihen }
}

public void V (int s) {
    s = s + 1;
    if ( s ≤ 0 ) { führe genau einen der bzgl. s wartenden Prozesse in den Zustand rechenwillig
    über }
}
```

Binäres Semaphor : die Kontrollvariable s nimmt nur boolesche Werte an.

man spricht auch von Mutex.

Mutex in Posix

Generated by Targeteam

Einsatz von Semaphoren

Notation: vordefinierter Typ `semaphor(int s)` . ***Semaphor-Objekt*** ist Instanz des Typs `semaphor` ; Semaphor wird mit Parameter s initialisiert.

Zugang zu kritischen Abschnitten

Realisierung der kritischen Abschnitte von Prozessen, in denen auf eine exklusiv benutzbare Ressource X zugegriffen wird:

1. Definition eines Semaphor-Objekts wa: semaphor(1), d.h. Initialisierung der Kontrollvariable des Semaphor-Objekts wa mit 1.
2. Klammern der kritischen Abschnitte, in denen auf die Ressource X zugegriffen wird, mit P und V Operationen:

   wa.P

   Code mit Zugriffen auf X

   wa.V

Semaphor-Konzept erfüllt w.A. Lösungsanforderungen

Wechselseitiger Ausschluss für alle kritischen Abschnitte.

keine Reihenfolge-Annahmen.

keine Ausführungszeit-Annahmen.

kein Verhungern.

Semaphore wurden 1968 von Dijkstra eingeführt. Ein ***Semaphor*** (Signalmast) ist eine ganzzahlige Koordinierungsvariable s, auf der nur die drei vordefinierten Operationen (Methoden) zulässig sind:

Initialisierung,

Prolog P (kommt von protekt),

Epilog V (kommt von vrej).

Operationen

Einsatz von Semaphoren

Beispiel Erzeuger-Verbraucher

Beispiel Philosophenproblem