

Script generated by TTT

Title: Baumgarten: GBS (08.11.2013)

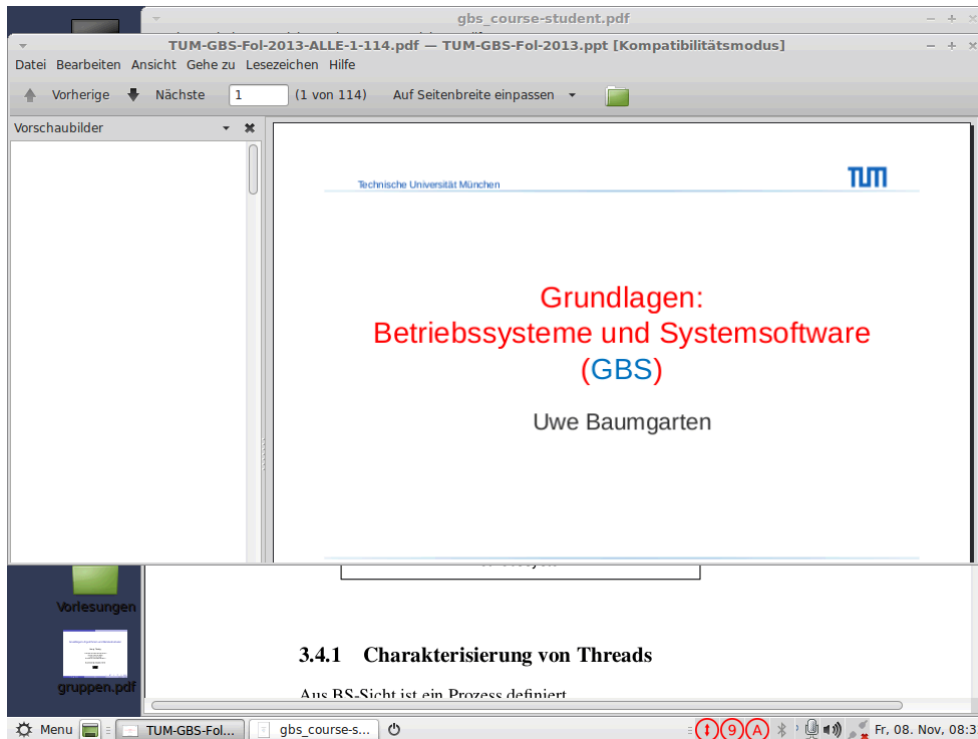
Date: Fri Nov 08 08:31:36 CET 2013

Duration: 86:30 min

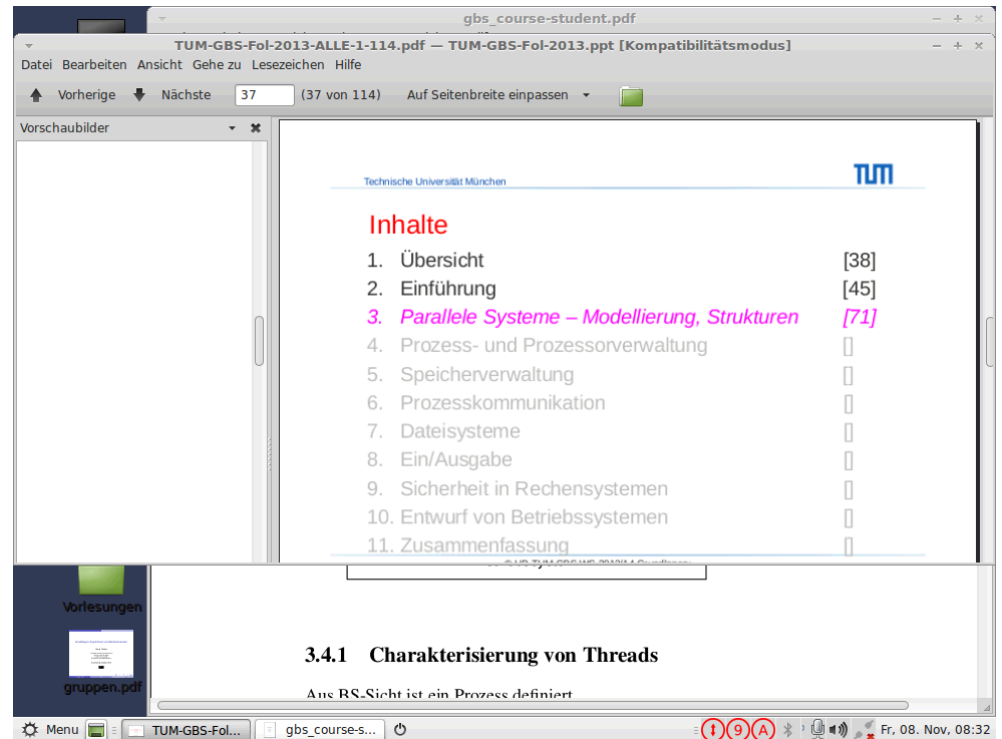
Pages: 46

Grundlagen: Betriebssysteme und Systemsoftware (GBS)

Uwe Baumgarten



The screenshot shows a PDF viewer window titled 'gbs_course-student.pdf'. The main content area displays the title slide with the text: 'Technische Universität München', 'Grundlagen: Betriebssysteme und Systemsoftware (GBS)', and 'Uwe Baumgarten'. The navigation bar at the top indicates '1 (1 von 114)'. The bottom status bar shows the current slide title '3.4.1 Charakterisierung von Threads' and the text 'Ans RS-Sicht ist ein Prozess definiert'.



The screenshot shows a PDF viewer window titled 'gbs_course-student.pdf'. The main content area displays a table of contents titled 'Inhalte'. The navigation bar at the top indicates '37 (37 von 114)'. The bottom status bar shows the current slide title '3.4.1 Charakterisierung von Threads' and the text 'Ans RS-Sicht ist ein Prozess definiert'.

Inhalte	
1. Übersicht	[38]
2. Einführung	[45]
3. <i>Parallele Systeme – Modellierung, Strukturen</i>	[71]
4. Prozess- und Prozessorverwaltung	[]
5. Speicherverwaltung	[]
6. Prozesskommunikation	[]
7. Dateisysteme	[]
8. Ein/Ausgabe	[]
9. Sicherheit in Rechensystemen	[]
10. Entwurf von Betriebssystemen	[]
11. Zusammenfassung	[]

gbs_course-student.pdf

TUM-GBS-Fol-2013-ALLE-1-114.pdf — TUM-GBS-Fol-2013.ppt [Kompatibilitätsmodus]

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe

Vorherige Nächste 71 (71 von 114) Auf Seitenbreite einpassen

Vorschaubilder

Technische Universität München **TUM**

3. Parallele Systeme – Modellierung etc.

- Fragestellungen
- Grundlagen
- **Modellierung paralleler Systeme** [77]
 - Petri-Netze [86]
- Thread-Konzept [95]
- Synchronisation [101]
- Verklemmungen [109]

Quelle: [JS12] Kap. 3

Vorlesungen

3.4.1 Charakterisierung von Threads

Ans RS: Sicht ist ein Prozess definiert

gruppen.pdf

Menu TUM-GBS-Fol... gbs_course-s... Fr, 08. Nov, 08:33

gbs_course-student.pdf

TUM-GBS-Fol-2013-ALLE-1-114.pdf — TUM-GBS-Fol-2013.ppt [Kompatibilitätsmodus]

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe

Vorherige Nächste 95 (95 von 114) Auf Seitenbreite einpassen

Vorschaubilder

Technische Universität München **TUM**

Thread-Konzept

- Ziele
 - Modellierung und Realisierung von nebenläufigen Aktivitäten
- Was sind Threads?
 - Threads == Kontrollflüsse, die Aktivitäten in einem Rechnerystem
- Traditionell
 - EIN Prozess mit EINEM Kontrollfluss (Thread)
- Heute
 - In der Regel: Multithreading
 - [JS12, Kap. 3, p. 60]
- Charakterisierung von Threads
- Threads in Java

3.4. Parallele Systeme – Threads

Quelle: [JS12] Kap. 3

Vorlesungen

3.4.1 Charakterisierung von Threads

Ans RS: Sicht ist ein Prozess definiert

gruppen.pdf

Menu TUM-GBS-Fol... gbs_course-s... Fr, 08. Nov, 08:35

Technische Universität München **TUM**

Charakterisierung von Threads

- Prozessbestandteile
 - Ein Adressraum (i.d.R. virtueller Adressraum (VA))
 - Eine Handlungsvorschrift (Programmcode)
 - Ein oder mehrere Aktivitätsträger (Threads)
- Motivation
 - Thread als Abstraktion eines physischen Prozessors
 - Parallelität innerhalb eines Prozesses mit einfacher Interaktion der Threads über gemeinsamen Speicher
 - Geringer Aufwand für Erzeugen und Löschen
 - Thread = leichtgewichtiger Prozess
 - Normale Prozesse = schwergewichtige Prozesse
 - Verbesserung der Performanz
 - Sowohl CPU- als auch E/A-intensive Applikationen
 - Echte Parallelität in Multiprozessor/Multicore-Systemen

3.4. Parallele Systeme – Threads

Quelle: [JS12] Kap. 3

© UB TUM GBS WS 2013/14 Grundlagen: Betriebssysteme und Systemsoftware (IN0009) 96

Technische Universität München **TUM**

Abgrenzung: Threads vs. Prozesse

- Zielsetzung
 - Prozesse gruppieren Ressourcen
 - Threads sind Aktivitätsträger
- Threadspezifische Informationen
 - Befehlszähler (PC)
 - Registerwerte
 - Keller/Stack
 - Threadzustand
- Prozessspezifische Informationen
 - Adressraum
 - Globale Variablen
 - Offene Dateien
 - Kindprozesse
 - Eintreffenden Alarme bzw. Interrupts
 - Verwaltungsinformationen

3.4. Parallele Systeme – Threads

Quelle: [JS12] Kap. 3

© UB TUM GBS WS 2013/14 Grundlagen: Betriebssysteme und Systemsoftware (IN0009) 97

Beispiel: Threads in Web-Servern

- Verteiler-Thread
- Server-Thread
- [JS12, Kap. 3, p. 62]

3.4. Parallele Systeme – Threads

Quelle: [JS12] Kap. 3

Schlischer, TU München 3.4. THREAD-KONZEPT

jeder Prozess genau einen Thread, der den Kontrollfluss des Prozess repräsentiert, während in Multithreaded-Umgebungen ein Prozess mehrere Threads besitzen kann.

3.4.1 Charakterisierung von Threads

Aus RS-Sicht ist ein Prozess definiert

EXKURS: Threads vs. Prozesse

- Aktivitätsträger
 - Threads (als primäre Träger)
 - Prozesse (als sekundäre Träger) (synonym: Task)
 - Jobs und Agenten (als mögliche weitere Träger)
- Arten von Prozessen (nach Aufgabe)
 - Rechenprozesse und E/A-Prozesse
 - Benutzerprozesse und Systemprozesse
- Hardwareunterstützung
 - Registersätze
 - Hyperthreading

3.4. Parallele Systeme – Threads

Threads in Java

- Definition
 - [JS12, Kap. 3, p. 63]
 - Asynchroner Ablauf der Threads
- ErgebnISRückgabe
 - Direkter Ansatz
[JS12, Kap. 3, p. 63-64]
 - Callback Ansatz
Rückgabe über Aufruf von Methoden
[JS12, Kap. 3, p. 64-65]

3.4. Parallele Systeme – Threads

Quelle: [JS12] Kap. 3

The screenshot shows a PDF viewer window titled 'gbs_course-student.pdf'. The main content is a slide titled 'Threads in Java' with the same bullet points as the first image. Below the slide is a diagram with three boxes: 'Anforderung' (top), 'Betriebssystem' (middle), and 'Netzwerk-Verbindung' (bottom). A curved arrow points from 'Anforderung' to 'Betriebssystem', and a straight arrow points from 'Betriebssystem' to 'Netzwerk-Verbindung'. The viewer interface includes a sidebar with thumbnails of slides 97, 98, 99, and 100 (selected), and a bottom status bar with system icons and the date 'Fr, 08. Nov, 08:52'.

The screenshot shows a PDF viewer window titled 'gbs_course-student.pdf'. The main content is a slide titled 'Threads in Java' with the same bullet points as the first image. Below the slide is a text block: 'Die Anweisung t.start() ruft die run-Methode der Klasse CallbackDigest auf.' followed by the section header 'ErgebnISRückgabe'. The viewer interface is similar to the previous screenshot, showing slide thumbnails and a bottom status bar with the date 'Fr, 08. Nov, 08:56'.

Threads in Java

- Definition
 - [JS12, Kap. 3, p. 63]
 - Asynchroner Ablauf der Threads
- ErgebnISRückgabe
 - Direkter Ansatz
[JS12, Kap. 3, p. 63-64]
 - Callback Ansatz
Rückgabe über Aufruf von Methoden
[JS12, Kap. 3, p. 64-65]

3.4. Parallele Systeme – Threads

Quelle: [JS12] Kap. 3

Synchronisation

3.5. Parallele Systeme – Synchronisation

- Aufgabe der Synchronisation
 - Abstimmung des zeitlichen Verhaltens nebenläufiger Aktivitäten untereinander
- Beispiele
 - Einfache nebenläufige Prozesse
 - Nichtdeterministisch
 - [JS12, Kap. 3, p. 66]
 - Lösung durch Verfahren zum wechselseitigen Ausschluss
 - Erzeuger-Verbraucher-Problem mit WA
 - [JS12, Kap. 3, p. 53]
- Synchronisation in Java (3.5.6)

Quelle: [JS12] Kap. 3

TUM-GBS-Fol-2013-ALLE-1-114.pdf — TUM-GBS-Fol-2013.ppt [Kompatibilitätsmodus]

101 (101 von 114)

Synchronisation

- Aufgabe der Synchronisation
 - Abstimmung des zeitlichen Verhaltens nebenläufiger Aktivitäten untereinander
- Beispiele
 - Einfache nebenläufige Prozesse
 - Nichtdeterministisch
 - [JS12, Kap. 3, p. 66]
 - Lösung durch Verfahren zum wechselseitigen Ausschluss
 - Erzeuger-Verbraucher-Problem mit WA
 - [JS12, Kap. 3, p. 53]
- Synchronisation in Java (3.5.6)

Die Anweisung `t.start()` ruft die `run`-Methode der Klasse `CallbackDigest` auf.

Ergebnisrückgabe

int x; x = 0

Prozess P1
A: $x = x + 1$

Prozess P2
B: $x = x + 2$

Zeitpunkt Z

- Das Ergebnis ist vom zeitlichen Ablauf abhängig. Es sind folgende Fälle möglich:
 - **Fall 1**
P1 liest $x = 0$, erhöht, speichert $x = 1$;
P2 liest $x = 1$, erhöht, speichert $x = 3$; => Wert von $x = 3$
 - **Fall 2**
P2 liest $x = 0$, erhöht, speichert $x = 2$;
P1 liest $x = 2$, erhöht, speichert $x = 3$; => Wert von $x = 3$
 - **Fall 3**
P1 und P2 lesen $x = 0$;
P1 erhöht, speichert $x = 1$;
P2 erhöht, speichert $x = 2$; => Wert von $x = 2$

B: $x = x + 2$

Zeitpunkt Z

- Das Ergebnis ist vom zeitlichen Ablauf abhängig. Es sind folgende Fälle möglich:
 - **Fall 1**
P1 liest $x = 0$, erhöht, speichert $x = 1$;
P2 liest $x = 1$, erhöht, speichert $x = 3$; => Wert von $x = 3$ 1 < 2
 - **Fall 2**
P2 liest $x = 0$, erhöht, speichert $x = 2$;
P1 liest $x = 2$, erhöht, speichert $x = 3$; => Wert von $x = 3$ 2 < 1
 - **Fall 3**
P1 und P2 lesen $x = 0$;
P1 erhöht, speichert $x = 1$;
P2 erhöht, speichert $x = 2$; => Wert von $x = 2$
 - **Fall 4**
P1 und P2 lesen $x = 0$;
P2 erhöht, speichert $x = 2$;
P1 erhöht, speichert $x = 1$; => Wert von $x = 1$
- Verhinderung des Nichtdeterminismus nur dadurch möglich, dass man

Synch: Wechselseitiger Ausschluss

- Gegeben:
 - Petri-Netz mit Anfangsmarkierung M0
 - Wenn zwei Transitionen t1 und t2 **wechselseitig ausgeschlossen** sind, dann ist keine Markierung M' erreichbar, so dass t1 und t2 gleichzeitig schaltbereit sind.
- Derartige Transitionen heißen „kritische Abschnitte“ (critical region).
- Mutual exclusion
- Nutzung der XBM in kritischen Abschnitten

3.5. Parallele Systeme – Synchronisation

Quelle: [JS12] Kap. 3

Synch – WA: Modellierung

- Abläufe (als Transitionen):
 - Ausführen unkritischer Abschnitte
 - Betreten des **kritischen** Abschnitts
 - Ausführen des **kritischen** Abschnitts
 - Verlassen des kritischen Abschnitts
- [JS12, Kap. 3, p. 68]
- Leser-Schreiber-Problem
 - Lese-Aktionen parallel (z.B. beschränkt auf 3)
 - Lese- und Schreib-Aktionen sind WA
 - Schreib-Aktionen untereinander sind WA
- [JS12, Kap. 3, p. 69]

P
E

3.5. Parallele Systeme – Synchronisation

Quelle: [JS12] Kap. 3

gbs_course-student.pdf

Technische Universität München

3.5.3 Modellierung

Modelliert man parallele Einheiten, die kritische Abschnitte besitzen, durch Petri-Netze, so sind vier Phasen dieser parallelen Aktivitäten von Interesse:

1. Ausführen unkritischer Abschnitte/Transaktionen
2. Betreten eines kritischen Abschnitts
3. Ausführen der Transaktion(en) des kritischen Abschnitts
4. Verlassen des kritischen Abschnitts.

Modellierung jeder Phase durch eine Transition - Koordination des wechselseitigen Ausschluss durch Kontrollstelle s.

Prozess 1: t1 (Eintritt), t2 (k.A.), t3 (Ausgang), t4 (Austritt)

Prozess 2: t1 (Eintritt), t2 (k.A.), t3 (Ausgang), t4 (Austritt)

Legende:
 t1: Phase 1: unkritische Transition
 t2: Phase 2: Eintritt in kritischen Abschnitt
 t3: Phase 3: Ausführung des kritischen Abschnitts
 t4: Phase 4: Verlassen des kritischen Abschnitts
 s: Kontrollstelle

gbs_course-student.pdf

Technische Universität München

Synchronisationskonzepte

- Konkretisierung von Prozessen
 - Ein Prozess ist ein eindeutig identifizierbarer Ablauf eines Programms.
- Zustände von Prozessen (ggf. auch Threads)
 - ERZEUGT
 - RECHNEND
 - RECHENWILLIG
 - WARTEND
 - TERMINIERT
- [JS12, Kap. 3, p. 70]
 - Alternative: „start“ überführt in RECHENWILLIG

Quelle: [JS12] Kap. 3

Synchronisationskonzepte

- Konkretisierung von Prozessen
 - Ein Prozess ist ein eindeutig identifizierbarer Ablauf eines Programms.
- Zustände von Prozessen (ggf. auch Threads)
 - ERZEUGT
 - RECHNEND
 - RECHENWILLIG
 - WARTEND
 - TERMINIERT
- [JS12, Kap. 3, p. 70]
 - Alternative: „start“ überführt in RECHENWILLIG

3.5. Parallele Systeme – Synchronisation

Quelle: [JS12] Kap. 3

Synchronisationskonzepte

- Konkretisierung von Prozessen
 - Ein Prozess ist ein eindeutig identifizierbarer Ablauf eines Programms.
- Zustände von Prozessen (ggf. auch Threads)
 - ERZEUGT
 - RECHNEND
 - RECHENWILLIG
 - WARTEND
 - TERMINIERT
- [JS12, Kap. 3, p. 70]
 - Alternative: „start“ überführt in RECHENWILLIG

3.5. Parallele Systeme – Synchronisation

Quelle: [JS12] Kap. 3

Anforderungen an Lösungen des WA

3.5. Parallele Systeme – Synchronisation

- Die kritischen Abschnitte sind wechselseitig ausgeschlossen.
- Die Realisierung darf nicht von der Reihenfolge der Ausführung der kritischen Abschnitte abhängen.
- Die Realisierung darf nicht von Annahmen über die Ausführungszeit der Prozesse abhängen.
- Kein Prozess darf unendlich lange daran gehindert werden, seinen kritischen Abschnitt auszuführen.
- Grundlage für Lösungen:
 - Atomare (nicht teilbare) Operationen

Quelle: [JS12] Kap. 3

Lösungen des WA

3.5. Parallele Systeme – Synchronisation

- Unterbrechungssperren
 - Disable Interrupt in Ein-Prozessorsystemen
- Test-and-Set Operationen
 - Atomarer Austausch von Werten einer Speicherzelle
- Dienste mit passivem Warten
 - Bei Bedarf Überführen in Zustand „WARTEND“
 - Beim Wecken Überführen in Zustand „RECHENBEREIT“
 - Alternative: aktives Warten
- Semaphor-Konzept
 - „Signal“ für FREI und BELEGT
- Monitor-Konzept
 - Objektorientiert mit WA-Methoden

Quelle: [JS12] Kap. 3

Anforderungen an Lösungen des WA

3.5. Parallele Systeme – Synchronisation

- Die kritischen Abschnitte sind wechselseitig ausgeschlossen.
- Die Realisierung darf nicht von der Reihenfolge der Ausführung der kritischen Abschnitte abhängen.
- Die Realisierung darf nicht von Annahmen über die Ausführungszeit der Prozesse abhängen.
- Kein Prozess darf unendlich lange daran gehindert werden, seinen kritischen Abschnitt auszuführen.
- Grundlage für Lösungen:
 - Atomare (nicht teilbare) Operationen

Quelle: [JS12] Kap. 3

Lösungen des WA

3.5. Parallele Systeme – Synchronisation

- Unterbrechungssperren
 - Disable Interrupt in Ein-Prozessorsystemen
- Test-and-Set Operationen
 - Atomarer Austausch von Werten einer Speicherzelle
- Dienste mit passivem Warten
 - Bei Bedarf Überführen in Zustand „WARTEND“
 - Beim Wecken Überführen in Zustand „RECHENBEREIT“
 - Alternative: aktives Warten
- Semaphor-Konzept
 - „Signal“ für FREI und BELEGT
- Monitor-Konzept
 - Objektorientiert mit WA-Methoden

Quelle: [JS12] Kap. 3

Synchronisationskonzepte

3.5. Parallele Systeme – Synchronisation

- Konkretisierung von Prozessen
 - Ein Prozess ist ein eindeutig identifizierbarer Ablauf eines Programms.
- Zustände von Prozessen (ggf. auch Threads)
 - ERZEUGT
 - RECHNEND
 - RECHENWILLIG
 - WARTEND
 - TERMINIERT
- [JS12, Kap. 3, p. 70]
 - Alternative: „start“ überführt in RECHENWILLIG

Quelle: [JS12] Kap. 3

Lösungen des WA

3.5. Parallele Systeme – Synchronisation

- Unterbrechungssperren
 - Disable Interrupt in Ein-Prozessorsystemen
- Test-and-Set Operationen
 - Atomarer Austausch von Werten einer Speicherzelle
- Dienste mit passivem Warten
 - Bei Bedarf Überführen in Zustand „WARTEND“
 - Beim Wecken Überführen in Zustand „RECHENBEREIT“
 - Alternative: aktives Warten
- Semaphor-Konzept
 - „Signal“ für FREI und BELEGT
- Monitor-Konzept
 - Objektorientiert mit WA-Methoden

Mono

Mutex

Quelle: [JS12] Kap. 3

Lösungen des WA: Semaphore

3.5. Parallele Systeme – Synchronisation

- Dijkstra 1968
- Operationen
 - Initialisierung
 - Prolog P (protekt)
 - Epilog V (vrej)
- Atomarität von P und V
- Bedeutung
 - [JS12, Kap. 3, p. 74]
- Weitere Eigenschaften
 - Binäre Semaphore (Mutex) mit Werten „0“ und „1“
 - Zählende Semaphore
 - Semaphore in POSIX-Schnittstelle
 - pthread_mutex_init(mutex), ..._lock(mutex), ..._unlock(mutex)

Quelle: [JS12] Kap. 3

The screenshot shows a PDF viewer window titled 'gbs_course-student.pdf'. The main content is a slide from a presentation, which is identical to the slide shown in the previous image (slide 107). The slide title is 'Lösungen des WA: Semaphore' and it lists various synchronization concepts and their implementations. The slide is displayed in a 'Kompatibilitätsmodus' (compatibility mode) window. The viewer interface includes a menu bar with options like 'Datei', 'Bearbeiten', 'Ansicht', 'Gehe zu', 'Lesezeichen', and 'Hilfe'. A navigation bar shows 'Vorherige', 'Nächste', and page numbers '107' and '(107 von 114)'. A 'Vorschaubilder' (preview images) pane is visible on the left. The system tray at the bottom shows the date and time: 'Fr. 08. Nov. 09:33'.

Lösungen des WA: Semaphore

3.5. Parallele Systeme – Synchronisation

- Dijkstra 1968
- Operationen
 - Initialisierung
 - Prolog P (protekt)
 - Epilog V (vrej)
- Atomarität von P und V
- Bedeutung
 - [JS12, Kap. 3, p. 74]
- Weitere Eigenschaften
 - Binäre Semaphore (Mutex) mit Werten „0“ und „1“
 - Zählende Semaphore
 - Semaphore in POSIX-Schnittstelle
 - pthread_mutex_init(mutex), ..._lock(mutex), ..._unlock(mutex)

Quelle: [JS12] Kap. 3

gbs_course-student.pdf

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe

Vorherige Nächste 74 (81 von 228) 150%

Vorschaubilder

104

105

106

107

Vorlesungen

gruppen.pdf

Schlichter, TU München 35. SYNCHRONISATION

• **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 Semaphore:** die Kontrollvariable s nimmt nur boolesche Werte an. man spricht auch von Mutex.

– **Mutex in Posix**
Ein Posix Mutex ist als binäres Semaphore eine einfache Sperre, die die Nutzung gemeinsamer Ressourcen absichert.
pthread_mutex_init(mutex) ⇒ initialisiert und konfiguriert ein Mutex.

Handwritten annotations: $= 1$, $= 0$, $= W$, A , B , \uparrow , \downarrow , $<$, $>$

Menu TUM-GBS-Fol... gbs_course-s... Fr. 08. Nov. 09:42

Lösungen des WA: Semaphore

3.5. Parallele Systeme – Synchronisation

- Dijkstra 1968
- Operationen
 - Initialisierung
 - Prolog P (protekt)
 - Epilog V (vrej)
- Atomarität von P und V
- Bedeutung
 - [JS12, Kap. 3, p. 74]
- Weitere Eigenschaften
 - Binäre Semaphore (Mutex) mit Werten „0“ und „1“
 - Zählende Semaphore
 - Semaphore in POSIX-Schnittstelle
 - pthread_mutex_init(mutex), ..._lock(mutex), ..._unlock(mutex)

Quelle: [JS12] Kap. 3

Lösungen des WA: Semaphor-Beispiele

3.5. Parallele Systeme – Synchronisation

- Kritische Abschnitte
 - [JS12, Kap. 3, p. 75]
- Erzeuger-Verbraucher-Problem mit beschränktem Puffer
 - Variante 1 nur mit Wechselseitigem Ausschluss
 - [JS12, Kap. 3, p. 75]
 - Variante 2 zuzüglich der Absicherung von LEER durch „voll“
 - [JS12, Kap. 3, p. 76]
 - Variante 3 zuzüglich der Absicherung von VOLL durch „leer“
 - [JS12, Kap. 3, p. 76]

Quelle: [JS12] Kap. 3

gbs_course-student.pdf

TUM-GBS-Fol-2013-ALLE-1-114.pdf — TUM-GBS-Fol-2013.ppt [Kompatibilitätsmodus]

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe

Vorherige Nächste 108 (108 von 114) Auf Seitenbreite einpassen

Vorschaubilder

Technische Universität München

Lösungen des WA: Semaphor-Beispiele

- Kritische Abschnitte
 - [JS12, Kap. 3, p. 75]
- Erzeuger-Verbraucher-Problem mit beschränktem Puffer
 - Variante 1 nur mit Wechselseitigem Ausschluss
 - [JS12, Kap. 3, p. 75]
 - Variante 2 zuzüglich der Absicherung von LEER durch „voll“
 - [JS12, Kap. 3, p. 76]
 - Variante 3 zuzüglich der Absicherung von VOLL durch „leer“
 - [JS12, Kap. 3, p. 76]

3.5. Parallele Systeme – Synchronisation

Quelle: [JS12] Kap. 3

man spricht auch von Mutex.

- **Mutex in Posix**

Ein Posix Mutex ist als binäres Semaphor eine einfache Sperre, die die Nutzung gemeinsamer Ressourcen absichert.

pthread_mutex_init(mutex) => initialisiert und konfiguriert ein Mutex.

Vorlesungen

gruppen.pdf

Fr, 08. Nov, 09:49

gbs_course-student.pdf

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe

Vorherige Nächste 76 (83 von 228) 150%

Vorschaubilder

Schlichter, TU München

3.5. SYNCHRONISATION

Problem: es kann eine Verklemmung auftreten, wenn der Verbraucher wa.P ausführt und warten muss, weil der Puffer kein Element enthält.

- **Variante 2**

Einführen eines zusätzlichen Semaphors voll: semaphor(0), das die Datenelemente im Puffer zählt:

```

Erzeuger E:
while (true) {
  produziere
  wa.P
  schreibe nach W
  wa.V
  voll.V
}

Verbraucher V:
while (true) {
  voll.P
  wa.P
  entnimm aus W
  wa.V
  verarbeite
}
  
```

Für den Erzeuger ergibt sich natürlich ein analoges Problem, falls der Puffer W nur eine beschränkte Kapazität besitzt.

Vorlesungen

gruppen.pdf

Fr, 08. Nov, 09:51

gbs_course-student.pdf

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe

Vorherige Nächste 76 (83 von 228) 150%

Vorschaubilder

Schlichter, TU München

3.5. SYNCHRONISATION

Problem: es kann eine Verklemmung auftreten, wenn der Verbraucher wa.P ausführt und warten muss, weil der Puffer kein Element enthält.

- **Variante 2**

Einführen eines zusätzlichen Semaphors voll: semaphor(0), das die Datenelemente im Puffer zählt:

```

Erzeuger E:
while (true) {
  produziere
  wa.P
  schreibe nach W
  wa.V
  voll.V
}

Verbraucher V:
while (true) {
  voll.P
  wa.P
  entnimm aus W
  wa.V
  verarbeite
}
  
```

Für den Erzeuger ergibt sich natürlich ein analoges Problem, falls der Puffer W nur eine beschränkte Kapazität besitzt.

- **Variante 3**

Einführen eines zusätzlichen Semaphors leer: semaphor(n), das die Anzahl der freien Elemente im Puffer zählt:

Vorlesungen

gruppen.pdf

Fr, 08. Nov, 09:51

gbs_course-student.pdf

Fenstermenü

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe

Vorherige Nächste 76 (83 von 228) 150%

Vorschaubilder

Schlichter, TU München

3.5. SYNCHRONISATION

Problem: es kann eine Verklemmung auftreten, wenn der Verbraucher wa.P ausführt und warten muss, weil der Puffer kein Element enthält.

- **Variante 2**

Einführen eines zusätzlichen Semaphors voll: semaphor(0), das die Datenelemente im Puffer zählt:

```

Erzeuger E:
while (true) {
  produziere
  wa.P
  schreibe nach W
  wa.V
  voll.V
}

Verbraucher V:
while (true) {
  voll.P
  wa.P
  entnimm aus W
  wa.V
  verarbeite
}
  
```

Für den Erzeuger ergibt sich natürlich ein analoges Problem, falls der Puffer W nur eine beschränkte Kapazität besitzt.

- **Variante 3**

Einführen eines zusätzlichen Semaphors leer: semaphor(n), das die Anzahl der freien Elemente im Puffer zählt:

Vorlesungen

gruppen.pdf

Fr, 08. Nov, 09:51

gbs_course-student.pdf

TUM-GBS-Fol-2013-ALLE-1-114.pdf — TUM-GBS-Fol-2013.ppt [Kompatibilitätsmodus]

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe

Vorherige Nächste 108 (108 von 114) Auf Seitenbreite einpassen

Vorschaubilder

105

106

107

108

Technische Universität München

Lösungen des WA: Semaphor-Beispiele

- Kritische Abschnitte [JS12, Kap. 3, p. 75]
- Erzeuger-Verbraucher-Problem mit beschränktem Puffer
 - Variante 1 nur mit Wechselseitigem Ausschluss
 - Variante 2 zuzüglich der Absicherung von LEER durch „voll“
 - [JS12, Kap. 3, p. 76]
 - Variante 3 zuzüglich der Absicherung von VOLL durch „leer“
 - [JS12, Kap. 3, p. 76]

3.5. Parallele Systeme – Synchronisation

Quelle: [JS12] Kap. 3

Vorlesungen

gruppen.pdf

Menu TUM-GBS-Fol... gbs_course-s... Fr, 08. Nov, 09:55

gbs_course-student.pdf

TUM-GBS-Fol-2013-ALLE-1-114.pdf — TUM-GBS-Fol-2013.ppt [Kompatibilitätsmodus]

Datei Bearbeiten Ansicht Gehe zu Lesezeichen Hilfe

Vorherige Nächste 111 (111 von 114) Auf Seitenbreite einpassen

Vorschaubilder

108

109

110

111

Technische Universität München

Verklemmungen

- Definition
 - Eine Verklemmung ist ein Zustand, in dem die beteiligten Prozesse wechselseitig auf den Eintritt von Bedingungen warten, die nur durch andere Prozesse in dieser Gruppe selbst hergestellt werden können.
- Notwendige Bedingungen
 - BM sind exklusiv (XBM)
 - BM sind nicht entziehbar (DBM)
 - „Haben-und-Anfordern“
 - Zyklische Wartebedingung

3.6. Parallele Systeme – Verklemmungen

Quelle: [JS12] Kap. 3

Vorlesungen

gruppen.pdf

Menu TUM-GBS-Fol... gbs_course-s... Fr, 08. Nov, 09:58

Verklemmungen

- Definition
 - Eine Verklemmung ist ein Zustand, in dem die beteiligten Prozesse wechselseitig auf den Eintritt von Bedingungen warten, die nur durch andere Prozesse in dieser Gruppe selbst hergestellt werden können.
- Notwendige Bedingungen
 - BM sind exklusiv (XBM)
 - BM sind nicht entziehbar (DBM)
 - „Haben-und-Anfordern“
 - Zyklische Wartebedingung

3.6. Parallele Systeme – Verklemmungen

Quelle: [JS12] Kap. 3