

Betriebssysteme

Studiengang Informatik / SAT

Dipl.-Inf., Dipl.-Ing. (FH) Michael Wilhelm
Hochschule Harz
FB Automatisierung und Informatik
mwilhelm@hs-harz.de
Raum 2.202
Tel. 03943 / 659 338

•Gliederung

1. Einführung
2. Prozesse und Threads
3. Speicherverwaltung
4. Dateiverwaltung
5. JNI
6. STL
- 7. Deadlocks**

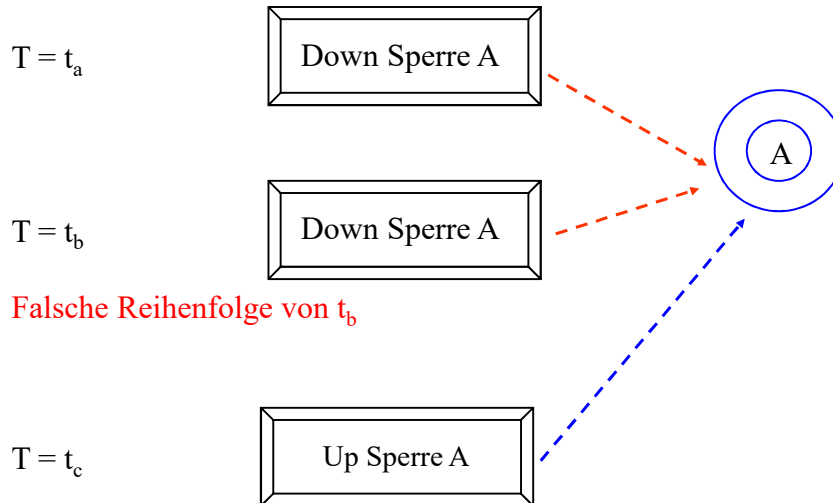
Deadlock

- Ein Deadlock oder Verklemmung, wird ausgelöst, wenn man mindestens zwei Threads und Semaphore oder ähnliches benutzt.
- Zwei "aktive" Elemente warten auf ein gemeinsames Ereignis
- Arten von Deadlocks
 - Self-Deadlock, Verklemmung (ein Prozess beteiligt)
 - Rekursives Deadlock (ein Prozess beteiligt)
 - Lock-Ordering-Deadlock (zwei Threads oder Prozesse beteiligt)
 - Heavily Contended Locks, Ursache (fair, Prioritäten)

Bedingungen für Deadlocks (Alle müssen erfüllt sein):

1. Bedingung des wechselseitigen Ausschlusses:
Jedes Betriebsmittel ist genau von einem Prozess belegt oder frei
2. Die Belegungs- und Wartebedingung:
Ein Prozess der bereits Betriebsmittel belegt, kann weitere Betriebsmittel anfordern
3. Die Ununterbrechbarkeitsbedingung:
Die Betriebsmittel, die von einem Prozess belegt werden, können nicht entzogen werden, sondern müssen freigegeben werden
4. Die zyklische Wartebedingung:
Eine Kette von zwei oder mehr Prozessen wartet auf die Freigabe eines Betriebsmittels vom nächsten Prozess

Self-Deadlocks

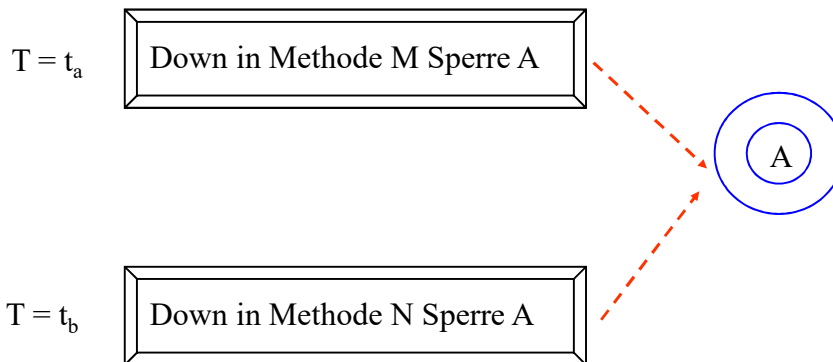


Self-Deadlocks

- Eine Thread belegt beim Zeitpunkt t_a eine Sperre.
- Geplant ist die Freigabe der Sperre beim Zeitpunkt t_c
- Beim Zeitpunkt t_b , welcher $> t_a$ und kleiner t_c ist, wird erneut die Sperre angefordert.
- Fehler beim Programmieren bzw. beim Entwurf
 - Die zweite Anforderung (down) darf erst nach der Freigabe (up) erfolgen

Ein Thread bzw. ein Prozess

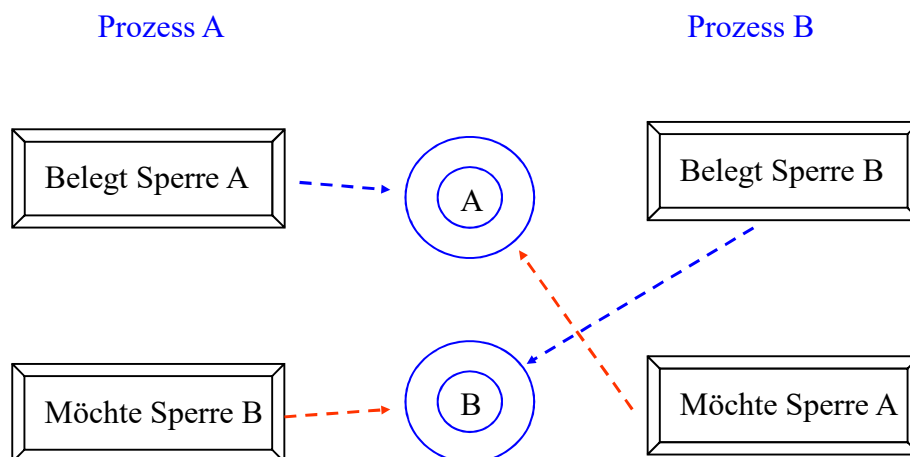
Rekursives Deadlock:



Die zweite, rekursive, Anforderung (down) darf erst nach der Freigabe (up) erfolgen

Ein Thread bzw. ein Prozess

Allgemeines oder Lock-Ordering Deadlock



Zwei Thread bzw. zwei Prozesse

Heavily Contended Locks

- Ein Prozess **P1** belegt beim Zeitpunkt t_a eine Sperre A
- Der Prozess hat eine niedrige Priorität
- Beim Zeitpunkt t_b startet ein weiterer Prozess **P2** mit einer höheren Priorität
- Dieser fordert auch die Sperre A
- Da er aber eine höhere Priorität als P1 hat, setzt er P2 in eine Warteschleife
- Als Scheduler-Modell wird das prioritätengesteuertes **faire** Verfahren gewählt
- **Folge: ?**
- Der Prozess **P1** kann nicht starten, da er eine zu kleine Priorität hat
- Der Prozess **P2** hat eine höhere Priorität, aber kann die Sperre A nicht erlangen

Heavily Contended Locks

Lösung:

- Der Prozess **P1** erhält durch den Prozess P2, **vererbt**, eine höhere Priorität
- Er kann so abgearbeitet werden
- Der Prozess **P2** wartet bis P1 fertig ist
- Danach kann er die Sperre A erhalten
- Die Lösung ist also die "Vererbung", bzw. Übertragung, der eigenen Priorität, um alle wartende Prozesse / Threads zu ermöglichen, die Sperren freizugeben
- Das Problem ist auch als **Pathfinder**-Problem bekannt

Engpässe mit Sperren bzw. Semaphoren

Beispiele:

- Bounded Buffer Problem
 - Die n-Threads benötigen im kritischen Bereich ein Semaphore
 - Starker Engpass
 - Lösung:
 - Man splittet die Tabelle
 - Ein Schlüssel pro Thread bestimmt welche "Tabelle" bzw. welches Semaphor benötigt wird.
- HashTable
 - Hier ist auch der Zugriff der Engpass
 - Lösung siehe oben
- Lese- Schreibproblem
 - Unterscheidung zwischen lesen und schreiben
- Feinkörniges Sperren

Deadlock:

Analog auch in:

- Sperren je eines Datensatzes in einer Datenbank durch verschiedene Prozesse

Betriebsmittel:

Resource die zu jedem Zeitpunkt nur von einem einzigen Prozess benutzt werden kann

- unterbrechbare Betriebsmittel :Speicher
- ununterbrechbare Betriebsmittel: Drucker

Reihenfolge (vgl. API)

- Anfordern des Betriebsmittels
- Benutzen des Betriebsmittels
- Freigeben des Betriebsmittels

Beispiele für beschränkte Betriebsmittel:

- Einträge in die Prozesstabelle
- Größe der I-Node Tabelle (geöffnete Dateien)
- Größe der Auslagerungsdatei
- Hauptspeicher

Deadlock Behebung mit:

- Keine Verhinderung
- Mittels Unterbrechung und Restart
Prozessen werden Betriebsmittel entzogen. Müssen wieder angefordert werden.
- Restart ab letzten gesicherten Statusbackup
Administrator speichert an Checkpunkten den Systemzustand. Bei Deadlock wird der vorherige Zustand restauriert. „Deadlock-Prozess“ wird später gestartet. CPU-Zeit wird „verschwendet“.
- Prozessabbruch
Alle beteiligten Prozesse werden abgebrochen. Restart und Hoffnung !
Ein unbeteiligter Prozesse wird abgebrochen. Sorgfältige Auswahl (Compiler vs. Datenbank).

Threads

■ Anzahl der Threads

- Beliebig ?
- Je mehr Threads, desto häufiger wird umgeschaltet
- Performance-Einbußen

■ Threads müssen auf die Hardware-Threads abgebildet werden

- Das BS stellt Funktionen zur Ermittlung zur Verfügung
- OpenMP: `omp_get_max_threads();`
- C#: `public static void GetMaxThreads( out int workerThreads, out int completionPortThreads )`

<http://msdn.microsoft.com/de-de/library/system.threading.threadpool.getmaxthreads.aspx>

Threads

■ Anzahl der Threads

- Obergrenze ist die Anzahl der Hardware-Thread

■ Limitierende Faktoren

- Hauptspeicher pro Thread
 - Ein- und Auslagern des Hauptspeichers
- Cache-Speicher
 - Cache-Speicher ist sehr klein, ca. 12 MB
 - Dieser muss bei einem Wechsel immer "mit ausgetauscht" werden
 - Cacheinhalt ist "ungültig"
 - Zugriff deshalb auf den langsamen Hauptspeicher