Nebenläufigkeit

Reader-Writer-Problem

Reader/Writer-Problem

[Wikipedia, 10.04.2007]

- "... situations in which many threads must access the same shared memory at one time, some reading and some writing, with the natural constraint that no process may access the share for reading or writing while another process is in the act of writing to it. (In particular, it is allowed for two readers to access the share at the same time.)

Beispiele

- Datenbank-Zugriff
- Datei-Zugriff

Start – Probleme/Lösungsansatz

Race condition

- Gemeinsamer Zugriff auf das Array *values* in der Klasse *Daten*

Probleme: Korrupte Daten

- Writer schreibt values[0]-values[5] und wird unterbrochen. Ein Reader liest inkonsistente Werte.
- Mehrere Writer überschreiben die Werte gegeneinander

Kritischer Abschnitt

- Schreiben und Lesen auf das Array *values*

Lösung: synchronized

Step 1: Probleme/Lösungsansatz

Problem

- Lesender Prozess kann nicht lesen, während bereits ein anderer am lesen ist.

Ansatz

- Anzahl schreibender und lesender Threads festhalten

HTW Chur
Hochschule für Technik und Wirtschaft

Fachhochschule Ostschweiz
University of Applied Sciences

Step 2: Probleme/Lösungsansatz

Problem:

- Kritischer Abschnitt

```
while(data.noReaders != 0 || data.noWriters != 0) {
        //
}
data.noWriters++;
```

Lösung:

- synchronized

Step 3: Probleme/Lösungsansatz

Probleme:

- Wenn ein Writer im kritischen Abschnitt wartet, kommt kein Reader zum Zug
- Unnötiger CPU-Verbrauch

Spinlock

- "A spinlock is a lock where the thread simply waits in a loop ("spins") repeatedly checking until the lock becomes available. As the thread remains active but isn't performing a useful task, the use of such a lock is a kind of busy waiting." [Wikipedia, 10.04.2007]

Lösung

- Synchronisation mittels wait/notify/notifyAll

Step 4: Probleme/Lösungsansatz

Probleme:

- Mit *notifyAll* werden zu viele Threads aufgeweckt. Eigentlich sollte man sagen können:
  - benachrichtige genau einen Writer oder
  - benachrichtige alle Reader

Lösung

- Ab Java 5.0 die Locks mit mehreren Bedingungen (Condition) benutzen (http://java.sun.com/j2se/1.5.0/docs/api/java/util/concurrent/locks/Condition.html)
- Ab Java 5.0 direkt den ReadWriteLock verwenden

Step 5: Eigenschaften des ReadWriteLock

Deadlook / Fairness

- "When the write lock is released either the longest-waiting single writer will be assigned the write lock, or if there is a reader waiting longer than any writer, the set of readers will be assigned the read lock. […] if readers are active and a writer enters the lock then no subsequent readers will be granted the read lock until after that writer has acquired and released the write lock."

Lock Downgrading/Upgrading

- "[…] allows downgrading from the write lock to a read lock, by acquiring the write lock, then the read lock and then releasing the write lock. However, upgrading from a read lock to the write lock is not possible."

(http://java.sun.com/j2se/1.5.0/docs/api/java/util/concurrent/locks/ReentrantReadWriteLock.html)