Lernzettel

Synchronisation und Nebenläufigkeit: Mutex, Semaphoren, Bedingungen, Deadlocks

Universität: Technische Universität Berlin
Kurs/Modul: Technische Grundlagen der Informatik
Erstellungsdatum: September 19, 2025

Lernzettel: Synchronisation und Nebenläufigkeit: Mutex, Semaphoren, Bedingungen, Deadlocks

(1) Mutex. Ein Mutex (Mutual Exclusion) ist ein Lock, der den gleichzeitigen Zugriff auf eine kritische Sektion sicherstellt. Nur der Thread, der den Mutex hält, darf die geschützte Ressource verwenden. Die Inhaberbeziehung verhindert Race Conditions.

Beispiel (Pseudocode):

$$\text{Lock}(M)$$

Kritische Sektion

$$\text{Unlock}(M)$$

(2) Semaphore. Ein Semaphore ist ein Zähler, der den Zugriff auf eine Ressource steuert. Es gibt zwei Arten: Binär-Semaphoren (wie ein Mutex) und Zähler-Semaphoren. Die Operationen heißen typischerweise Wait (P) und Signal (V).

Wichtige Operationen:

$$Wait(S): \quad S \leftarrow S - 1 \quad \text{und blockiere, falls } S < 0$$

$$Signal(S): \quad S \leftarrow S + 1$$

Beispiel (Pseudocode):

$$\text{Wait}(S)$$

Critical section guarded by S

$$\text{Signal}(S)$$

(3) Bedingungen (Condition Variables). Bedingungsvariablen werden together mit einem Mutex verwendet, um auf eine bestimmte Bedingung zu warten und anschließend fortzufahren, wenn diese Bedingung erfüllt ist.

Typisches Muster:

$$\text{Lock}(M)$$

$$\text{while not } cond: \quad \text{Wait}(C, M)$$

... kritischer Abschnitt ...

$$\text{Unlock}(M)$$

Was geschieht bei Wait(C, M):

$\text{Wait}(C, M)$: M wird freigegeben; C wird blockiert; nach Signalisierung Re-Aktivierung von M

(4) Deadlocks. Ein Deadlock tritt auf, wenn vier Bedingungen gleichzeitig erfüllt sind. Vermeide oder erkenne sie, um Deadlocks zu verhindern.

- Gegenseitiger Ausschluss (Mutual Exclusion) für Ressource.

- Halten und Warten (Hold and Wait) auf weitere Ressourcen.
- Keine Vorabbefreiung (No Preemption) von Ressourcen.
- Zirkuläres Warten (Circular Wait) von Threads/Ressourcen.

Vermeidung und Erkennung (Kurzüberblick):

- Ressourcen nach einer festen Ordnung zuweisen (Locking-Order).
- Versuchen, mit Timeout zu arbeiten (Try-Lock).
- Vermeiden von Hold-and-Wait durch Anfordern aller Ressourcen auf einmal.
- Deadlock-Erkennung und Neustart bzw. Ressourcenfreigabe bei Detektion.