Fadensynchronisation und -kommunikation

1 Domain Definition

Die Domäne *Fadensychronisation und -kommunikation* beschäftigt sich mit der Bereitstellung von Mechnismen zur Sychronisation von *Threads* und der Kommunikation zwischen ihnen. Des Weiteren wird die Unterstützung von Signalen und das Blockieren von Threads angeboten.

Features, die von dieser Domäne nicht abgedeckt werden, sind die Bereitstellung von Threads (starten, auswählen, beenden), Speicher-Management, Listen und Pong.

2 Konzept Modell

2.1 Grundlegende Konzepte

2.1.1 Gegenseitiger Ausschluss

Gegenseitiger Ausschluss garantiert, dass ausgewählte Aktionen immer nur von einem Thread gleichzeitig ausgeführt werden können. Wie zum Beispiel das Betreten bestimmter Programmabschnitte, Zugriffe auf Datenstrukturen usw. Um gegenseitigen Ausschluss zu erreichen, bietet diese Domäne verschiedene Sychronisationsmechanismen:

- Semaphore
- Monitore

2.1.2 Signale

Signale sind asynchrone Nachrichten, die einem Thread zugestellt werden können. Der Thread kann dann mit einem sogenannten *Signal Handler* auf ein ankommendes Signal angemessen reagieren. Falls ein Thread keine eigenen Signal-Handler implementiert, so ist es auch möglich, einen Standard-Handler für ein Signal zu spezifizieren. So könnte z.B. der Standard-Handler eines KILL-Signals das Beenden des Threads bewirken.

2.1.3 Blockieren von Threads

Das Blockieren von Threads ist eine Methode, den Scheduler zu veranlassen, diesen Thread nicht mehr in den Ablauf mit einzuplanen. Dies kann durch Ent-

nahme des Threads aus der Bereit-Liste des Schedulers erfolgen oder durch einfaches markieren dieses Threads als *blockiert*. Die Methoden zur Modifikation der Bereit-Liste oder der Thread-Struktur werden von anderen Domänen bereitgestellt.

2.1.4 Nachrichtenaustausch

Um Kommunikation zwischen verschiedenen Threads bereitzustellen, bietet diese Domäne verschiedene Möglichkeiten:

- Mailbox
- Port
- Rendevous
- Shared Memory

Davon können die meisten blockierend und nicht blockierend verwendet werden, was jedoch nicht als Extra-Feature in das Feature-Modell eingeht. Stattdessen werden entsprechende Funktionen bei den einzelnen Features vorhanden sein, die durch das Function-Level-Linking ausgewählt werden.

2.1.5 Verklemmungen

Diese Domäne bietet drei Möglichkeiten, Verklemmungen zu begegnen:

- Deadlock Detection & Resolution
- Deadlock Avoidance
- Deadlock Prevention

2.2 Zusammenhänge

2.2.1 interne Zusammenhänge

Wenn das *Blockieren von Threads* unterstützt wird, müssen alle Mechnismen, die ein Nicht-Fortsezten des Threads erzwingen, so umprogrammiert werden, dass sie den Thread tatsächlich blockieren und nicht nur beim Scheduler einen Threadwechsel beantragen.

2.2.2 externe Zusammenhänge

Falls Threads mit einer Priorität versehen sind, müssen Sychronisations- und Kommunikationsmechnismen so angepasst werden, dass sie abhängig von der Priorität Threads deblockieren.

3 Domain Lexicon

Blocking Ein *blockierter* Thread wird vom Scheduler ignoriert. Meist wird er aus der ready-Queue ausgehängt.

Critical Section Codebereich, der synchronisiert werden muss.

Deadlock Blockadesituation zwischen mehreren(?) Threads, die sich selbst nicht auflöst.

Deadlock Avoidance Restriktionen auf Lock-Vergabe, die einen zyklischen Wartegraphen verhindern.

Deadlock Detection & Resolution Erkennen eines zyklischen Wartegraphen und daraus folgende Aktionen mit dem Ziel, den Zyklus aufzulösen.

Deadlock Prevention Restriktionen auf Lock-Vergabe, die eine oder mehrere notwendige Deadlock-Bedingungen verhindern.

Inter Thread Communication (ITC) IPC-artige Mechanismen zur Synchronisation und Kommunikation zwischen Threads.

Mailbox Nachrichten-Queue mit mehreren Sendern und Empfängern

Monitor Gekapselter Codebereich, in dem sich zu einem Zeitpunkt maximal ein Aktivitätsträger aufhalten kann.

MultiWait `poll()`-ähnliches, gleichzeitiges Warten auf mehrere Ereignisse in den Bereichen Synchronisation und Kommunikation.

Port Nachrichten-Queue mit mehreren Sendern und einem Empfänger

Priority Inversion Blockierung eines hochprioren Prozesses durch ein Lock eines niedrigprioren Prozesses.

Priority Problem Solver Anhebung der Priorität des Lock-haltenden Prozesses um einen festen oder dynamischen Wert im Priority Inversion Fall.

Rendevous Nachrichten-Queue mit einem Sender und einem Empfänger. Blockiert beide Seiten, bis die Nachricht übertragen wurde.

Resource Counting Semaphore Semaphore für mehrere gleichartige Ressourcen.

Semaphore Synchronisationsprimitive, siehe Dijkstra.

Shared Memory Von mehreren Prozess gemeinsam nutzbarer Speicherbereich.

Signal Asynchron eintreffende typisierte Nachricht ohne Inhalt.

Signal Handler Vom Thread bereitgestellte Behandlungsroutine für ausgewählte Signale.