AUFGABE 6: ZUGRIFFSKONTROLLE IN ECOS

In dieser letzten Übung werden Sie sich mit gegenseitigem Ausschluss und Zugriffskontrolle befassen. Diese Übungsaufgabe zielt auf die Probleme der Prioritätsumkehr und der Verklemmung ab und dient dazu die in der Vorlesung vorgestellten echtzeitfähigen Synchronisationsprotokolle (vgl. VII 11 ff.) praktisch anzuwenden.

Auch in dieser Übung greifen wir der Einfachheit halber auf die folgenden synthetischen Aufgabensysteme zurück:

Aufgabensystem 1: Prioritätsumkehr

Aufgabe	**Periode** ms	**Phase** ms	**WCET** ms	**Betriebsmittel**[1]
T_1	20	4	6	(BM_1,3,1)
T_2	50	3	4	
T_3	200	1	9	(BM_1,1,7)

Aufgabensystem 2: Transitive Blockung

Aufgabe	**Periode** ms	**Phase** ms	**WCET** ms	**Betriebsmittel**[1]
T_4	20	7	2	(BM_2,1,1)
T_5	50	5	6	(BM_2,1,5) (BM_3,3,2)
T_6	100	3	6	(BM_3,1,5) (BM_4,4,2)
T_7	200	1	10	(BM_4,1,9)

Aufgabensystem 3: Verklemmung

Aufgabe	**Periode** ms	**Phase** ms	**WCET** ms	**Betriebsmittel**[1]
T_8	20	3	6	(BM_5,1,5) (BM_6,4,1)
T_9	50	8	12	
T_{10}	200	1	6	(BM_6,1,5) (BM_5,5,1)

[1] Notation Betriebsmittel: (Betriebsmittel, Anforderungszeitpunkt, Haltezeit)

Implementierungshinweise:

A. Nutzen Sie die in der Vorgabe enthaltenen Vorlagen und implementieren Sie die o. g. Aufgabensysteme in separaten Dateien.

B. Vergeben Sie die Prioritäten nach dem RMA, simulieren Sie die WCET wie angegeben.

C. Nutzen Sie das von eCos bereitgestellte Mutexkonzept zur Implementierung der Betriebsmittel.

D. Implementieren Sie anschließend die verschiedenen Varianten zur Synchronisation, nutzen Sie hierfür die in er Tafelübung vorgestellten Mechanismen von eCos.

E. Belassen Sie die Lösungen der vorangegangenen Teilaufgaben deaktiviert im Code für die spätere Abgabe.

Zentrale Fragen: Wie so häufig haben die verschiedenen Techniken ihre Vor- und Nachteile. Die zentrale Fragestellung für alle folgenden Aufgaben ist daher:

A. *Welche maximale Blockadezeit (für die höchstpriore Aufgabe) erwarten Sie und wie lässt sie sich messen?*

B. *Können Deadlocks auftreten?*

C. *Welche Nachteile erfahren unbeteiligte Aufgaben (die kein Betriebsmittel nutzen)?*

Machen Sie sich bei den nachfolgenden Aufgaben zunächst mit Stift und Papier (oder einer anderen Darstellungsmöglichkeit Ihrer Wahl) klar, was bezüglich belegter Betriebsmittel, blockierter Aufgaben, Prioritäten etc. passieren sollte. Implementieren Sie die Aufgabe anschließend wie beschrieben.

Aufgabenstellung

A. *Verdrängungssteuerung:* Nutzen Sie das Konzept der Verdrängungssteuerung (NPCS) zur Synchronisation von Aufgabensystem 1. *Beantworten Sie Frage A bis C (s. o.).*

☞ cyg_scheduler_lock()
☞ ...unlock()

B. *Prioritätsvererbung:* Verwenden Sie als nächstes Prioritätsvererbung (PI) um Aufgabensystem 2 umzusetzen. Wählen Sie hierfür bei der Initialisierung der Mutexobjekte `CYG_MUTEX_INHERIT` als Protokoll aus (`cyg_mutex_set_protocol()`). *Beantworten Sie Frage A bis C (s. o.).*

C. *Prioritätsobergrenzen:* Setzen Sie als letzten noch die Prioritätsobergrenzen (PCP) ein um Aufgabensystem 3 zu realisieren. Wählen Sie hierfür bei der Initialisierung der Mutexobjekte `CYG_MUTEX_CEILING` als Protokoll. *Beantworten Sie Frage A bis C (s. o.). Zeichnen Sie den Ablauf mittels Tracer auf und überlagern Sie diese Aufzeichnung mit der vermuteten Systemobergrenze. Wieso kommt es zu keiner Verklemmung?*

Erweiterte Aufgabe

A. *Konzeptvergleich:* Implementieren nun noch die anderen Kombinationen (jedes Aufgabensystem mit jeder Synchronisationstechnik). *Messen Sie wiederum die Blockadezeit. Was fällt Ihnen insbesondere beim Aufgabensystem 2 und 3 auf?*

Bonusaufgabe

Die Bearbeitung dieser Aufgabe ist freiwillig. Bei erfolgreicher Bearbeitung (textuell oder Implementierung) erhalten Sie fünf Euro Guthaben für die I4-Kaffeekasse – das entspricht 20 Tassen Kaffee.

Bei dieser Aufgabe handelt es sich um ein aktuelles, bisher nicht zufriedenstellend gelöstes Problem bei der Umsetzung des I4Copters.

Hintergrund Die Belegung eines Betriebsmittels ist ein kritischer Abschnitt. Normalerweise macht es keinen Sinn, dass eine Aufgabe in einem kritischen Abschnitt die Kontrolle über die CPU abgibt,

da sie den Abschnitt ja so schnell wie möglich wieder verlassen möchte bzw. sollte. Die vorgestellten Synchronisationsmechanismen unterstützen die Aufgabe bei diesem Vorhaben indem sie beispielsweise ihre Priorität anheben.

Problembeschreibung Der I4Copter besitzt einen seriellen SPI-Bus (Betriebsmittel) über den drei Aufgaben (T_A, T_B, T_C) unterschiedlicher Priorität Nachrichten versenden. Senden und empfangen erfolgt beim SPI gleichzeitig, der Bus ist also synchron. Die Hardware übernimmt den eigentlichen Versand und signalisiert den Empfang mittels eines Interrupts. Die Aufgaben müssen also lediglich eine Bustransaktion aufsetzen, die Nachricht an eine beliebige Stelle im Speicher legen und bis zum Empfangsinterrupt warten. Sie benötigt die CPU also nicht, es muss aber verhindert werden, dass eine andere Aufgabe eine neue Transaktion aufsetzt – dies würde zum Abbruch der aktuellen Transaktion führen.

Forderungen

A. Der Faden gibt Kontrolle über die CPU während der Übertragung ab. Aktives Warten findet nicht statt.

B. Unbeteiligte, d. h. niederpriore Fäden sollen laufen dürfen.

C. Nachrichten sollen mit der Fadenpriorität versendet werden.

D. Kein extra Faden für die Buskoordinierung.

Hinweise Die klassischen Verfahren versagen hier. Obwohl sie die Nachrichtenpriorität berücksichtigen, ist das Problem, dass sie niederpriore Fäden ausschließen und die CPU dadurch u. U. untätig bleiben würde.

Hinweise

- Erforderliche Dateien: wie vorgegeben
- Bearbeitung: Gruppe mit je zwei/drei Teilnehmern.
- Abgabezeit: 07.02.2013

- Fragen bitte an i4ezs@lists.cs.fau.de
- Die Funktionsreferenz der *eCos*-API findet sich unter
  http://ecos.sourceware.org/docs-latest/ref/kernel.html
- Eigene Quelldateien müssen Sie in die CMakeLists.txt eintragen. Bei jeder Änderung müssen Sie im Build-Verzeichnis cmake erneut aufrufen.
- Bei nachträglichen Änderungen an der Vorgabe (z. B. zur Fehlerkorrektur) *muss* der Inhalt des Build-Verzeichnisses gelöscht und cmake erneut aufgerufen werden. Dieses Vorgehen empfiehlt sich auch bei nicht nachvollziehbaren Fehlern.