AUFGABE 7: ZUGRIFFSKONTROLLE

In dieser letzten Übungsaufgabe 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.

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

Tabelle 1: Aufgabensystem 1 – „Pathfinder“

Aufgabe	**Periode** ms	**Phase** ms	**WCET** ms	**Betriebsmittel**[1]
T_1	20	4	6	$(R_1, 3, 1)$
T_2	50	3	4	
T_3	200	1	9	$(R_1, 1, 7)$

Tabelle 2: Aufgabensystem 2

Aufgabe	**Periode** ms	**Phase** ms	**WCET** ms	**Betriebsmittel**[1]
T_4	20	7	2	$(R_2, 1, 1)$
T_5	50	5	6	$(R_2, 1, 5)$ $(R_3, 3, 2)$
T_6	100	3	6	$(R_3, 1, 5)$ $(R_4, 4, 2)$
T_7	200	1	10	$(R_4, 1, 9)$

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

Tabelle 3: Aufgabensystem 3

Aufgabe	Periode ms	Phase ms	WCET ms	Betriebsmittel[1]
T_8	20	3	6	$(R_5, 1, 5)$ $(R_6, 4, 1)$
T_9	50	8	12	
T_{10}	200	1	6	$(R_6, 1, 5)$ $(R_5, 5, 1)$

Implementierungshinweise:

1. Nutzen Sie die in der Vorgabe enthaltenen Vorlagen und implementieren Sie die o. g. Aufgabensysteme in separaten Dateien. ☞ `aufgabe_{1,2,3}.c`
2. Verändern sie die Datei `CMakeLists.txt` um zwischen den einzelnen Aufgabenimplementierungen umzuschalten.
3. Vergeben Sie die Prioritäten nach dem RMA, simulieren Sie die WCET wie angegeben.
4. Nutzen Sie das von eCos bereitgestellte Mutexkonzept zur Implementierung der Betriebsmittel.
5. Belassen Sie die Lösungen der vorangegangenen Teilaufgaben deaktiviert im Code für die spätere Abgabe.
6. Zeichnen Sie die Abläufe Ihrer Implementierung für die Abgabe auf. ☞ `make trace`

1 Aufgabenstellung

Nach dem erfolgreichen Aufsetzen des `build`*-Verzeichnisses können Sie sich mittels* `make doc` *eine Übersicht über alle in* `libEZS` *bereitgestellten Funktionen inklusive deren Dokumentation erzeugen.*

Denken Sie daran, Ihren Quellcode nach vollständiger Bearbeitung noch vor dem Beginn der Rechnerübung abzugeben. Rufen Sie hierzu in Ihrem `build`*-Verzeichnis* `make submit` *auf.*

1.1 Grundlagen

Aufgabe 1

Welche Bedingungen müssen erfüllt sein, damit es zu einer Verklemmung (engl. Deadlock) kommen kann?

Antwort:

Aufgabe 2

Welcher zusätzliche Parameter muss bei der Laufzeitanalyse einer Aufgabe mit gegenseitigem Ausschluss beachtet werden?

Antwort:

Aufgabe 3

Wie wirkt sich dieser Parameter auf die Laufzeitanalyse von niederprioren/höherprioren Aufgaben im Allgemeinen aus, welche selbst eine Abhängigkeit zu dem kritischen Abschnitt aufweisen?

Antwort:

Aufgabe 4

Wie sieht dies bei niederprioren/höherprioren Aufgaben im Allgemeinen aus, die selbst in **keiner** Abhängigkeit zu dem kritischen Abschnitt stehen?

Antwort:

1.2 Verdrängungssteuerung

Aufgabe 5

Machen Sie sich zunächst mit Stift und Papier (oder einer anderen Darstellungsmöglichkeit Ihrer Wahl) klar, was bei der Koordinierung von Aufgabensystem 1 mit Verdrängungssteuerung (NPCS) bezüglich belegter Betriebsmittel, blockierter Aufgaben, Prioritäten etc. passieren sollte. Zeichnen Sie den entstehenden Ablaufplan.

Aufgabe 6

Welche maximale Blockadezeit (für die höchstpriore Aufgabe) erwarten Sie für dieses Aufgabensystem? Wie lässt sich für NPCS allgemein eine obere Schranke für die Blockadezeit bei beliebigen Aufgabensystemen berechnen (Formel)?

Antwort:

Aufgabe 7

Können Verklemmungen auftreten?

Antwort:

Aufgabe 8

Welche Nachteile erfahren unbeteiligte Aufgaben (die kein Betriebsmittel nutzen)? Hat die Priorität der unbeteiligten Aufgabe einen Einfluss auf diese Beeinträchtigung?

Antwort:

Aufgabe 9

Nutzen Sie nun das Konzept der Verdrängungssteuerung (NPCS) zur Synchronisation von Aufgabensystem 1 in eCos. Messen Sie die Blockadezeit der höchstprioren Aufgabe und archivieren Sie einen Screenshot der zeitlichen Abfolge für die Abgabe.

☞ `cyg_scheduler_lock()`
☞ `...unlock()`
☞ make trace

1.3 Prioritätsvererbung

Aufgabe 10

Machen Sie sich zunächst mit Stift und Papier (oder einer anderen Darstellungsmöglichkeit Ihrer Wahl) klar, was bei der Koordinierung von Aufgabensystem 2 mithilfe der Prioritätsvererbung, bezüglich belegter Betriebsmittel, blockierter Aufgaben, Prioritäten etc. passieren sollte. Zeichnen Sie den entstehenden Ablaufplan.

Aufgabe 11

Welche maximale Blockadezeit (für die höchstpriore Aufgabe) erwarten Sie für dieses Aufgabensystem? Wie lässt sich für Prioritätsvererbung allgemein eine obere Schranke für die Blockadezeit bei beliebigen Aufgabensystemen berechnen (Formel)?

Antwort:

Aufgabe 12

Können Verklemmungen auftreten?

Antwort:

Aufgabe 13

Welche Nachteile erfahren unbeteiligte Aufgaben (die kein Betriebsmittel nutzen)? Hat die Priorität der unbeteiligten Aufgabe einen Einfluss auf diese Beeinträchtigung?

Aufgabe 14

Verwenden Sie Prioritätsvererbung (PI) um Aufgabensystem 2 in eCos umzusetzen. Wählen Sie hierfür bei der Initialisierung der Mutexobjekte `CYG_MUTEX_INHERIT` als Protokoll aus. Messen Sie die Blockadezeit der höchstprioren Aufgabe und archivieren Sie einen Screenshot Ihres der zeitlichen Abfolge der Aufgaben für die Abgabe.

☞ `cyg_mutex_set_protocol()`

☞ make trace

1.4 Prioritätsobergrenzen

Aufgabe 15

Machen Sie sich zunächst mit Stift und Papier (oder einer anderen Darstellungsmöglichkeit Ihrer Wahl) klar, was bei der Koordinierung von Aufgabensystem 3 mithilfe von Prioritätsobergrenzen, bezüglich belegter Betriebsmittel, blockierter Aufgaben, Prioritäten etc. passieren sollte. Zeichnen Sie den entstehenden Ablaufplan.

Aufgabe 16

Welche maximale Blockadezeit (für die höchstpriore Aufgabe) erwarten Sie für dieses Aufgabensystem und wie lässt sie sich messen? Wie lässt sich für PCP allgemein eine obere Schranke für die Blockadezeit bei beliebigen Aufgabensystemen berechnen (Formel)?

Antwort:

Aufgabe 17

Können Verklemmungen auftreten?

Antwort:

Aufgabe 18

Welche Nachteile erfahren unbeteiligte Aufgaben (die kein Betriebsmittel nutzen)? Hat die Priorität der unbeteiligten Aufgabe einen Einfluss auf diese Beeinträchtigung?

Antwort:

Aufgabe 19

Setzen Sie nun noch die Prioritätsobergrenzen (PCP) ein um Aufgabensystem 3 in eCos zu realisieren. Wählen Sie hierfür bei der Initialisierung der Mutexobjekte `CYG_MUTEX_CEILING` als Protokoll und konfigurieren Sie die Prioritätsobergrenzen der Betriebsmittel passend – beachten Sie hierbei, dass die Priorität des Betriebsmittels um eins höher sein muss als die des höchstprioren verwendenden Fadens und nicht, wie eigentlich üblich, gleich der Fadenpriorität. Messen Sie die Blockadezeit der höchstprioren Aufgabe und archivieren Sie einen Screenshot Ihres der zeitlichen Abfolge der Aufgaben für die Abgabe.

☞ `cyg_mutex_set_protocol()`

☞ make trace

Aufgabe 20

Zeichnen Sie den Ablauf mittels Tracer auf und überlagern Sie diese Aufzeichnung mit der *vermuteten* Systemobergrenze. Welche Variante von PCP ist in eCos implementiert? Welchem anderen Synchronisationsverfahren ähnelt diese? Worin liegt der entscheidende Unterschied zur anderen, in der Vorlesung vorgestellten PCP-Variante?

Antwort:

Aufgabe 21

Wieso kommt es zu keiner Verklemmung?

Antwort:

2 Erweiterte Aufgabe

Die Erweiterten Übungsaufgaben sind nur für Teilnehmer verpflichtend, die das 7,5-ECTS-Modul belegen. Wir werden Sie natürlich auch dann bei der Bearbeitung unterstützen, wenn Sie diese Teilaufgaben freiwillig bearbeiten.

Aufgabe 22

Synchronisieren Sie Aufgabensystem 2 nun auch sowohl mit Prioritätsobergrenzen als auch Verdrängungssteuerung. Messen Sie wiederum die Blockadezeiten und sichern Sie Ihre Ablaufpläne. Vergleichen Sie das Verhalten des Aufgabensystems in Bezug auf die drei Synchronisationsmethoden. Was stellen Sie fest?

Antwort:

Aufgabe 23

Implementieren Sie Aufgabensystem 3 mithilfe von Prioritätsvererbung. Welches Verhalten stellen Sie fest? Sichern Sie auch hierfür Ihren Ablaufplan.

Antwort:

Hinweise

- Bearbeitung: Gruppe mit je drei Teilnehmern.
- Abgabefrist: 25.07.2022 (vor der ersten Rechnerübung) ☞ `make submit`
- Fragen bitte an `i4ezs@lists.cs.fau.de`