

Grundlegende Übungen

Aufgabenblatt 3 - ereignisgesteuerte Ablaufplanung

Peter Ulbrich, Martin Hoffmann

29. November 2010

Ausgangspunkt für diese Aufgabenstellung ist ein ereignisgesteuerter Ablaufplaner. Der Ablaufplaner hat dabei die Eigenschaften, wie sie in den Tafelübungen zu Aufgabe 3 besprochen wurden.

Aufgabe 1 Gegeben seien die vier Aufgaben in der folgenden Tabelle:

Aufgabe	p/a	Periode	min. Zwischenankunftszeit	WCET
T1	p	20 ms	NA	4 ms
T2	p	20 ms	NA	2 ms
T3	p	40 ms	NA	5 ms

Auch in ereignisgesteuerten Systemen trifft man häufig eine Menge von periodischen Aufgaben an. In der Regel wird dies erreicht, indem ein Zeitgeber zu bestimmten Zeitpunkten die entsprechenden Aufgaben aktiviert.

Hinweis: Sie müssen bzw. sollen hier keine Rücksicht auf die WCETs der einzelnen Aufgaben nehmen. Treffen mehrere Aufgaben zum selben Zeitpunkt zur Bearbeitung ein, sollen diese auch zeitgleich ausgelöst werden.

- Implementieren Sie die Aufgaben T1 - T3 und aktivieren Sie diese in den angegebenen Zeitabständen, vergeben Sie die Prioritäten der einzelnen Aufgaben anhand das *Rate Monotonic Algorithm* (siehe Kapitel 4 "Ablaufsteuerung", Folie 4-27 und 4-28).
- Nehmen Sie an, T1 berechnet ein Ergebnis, das von T2 weiterverarbeitet wird. Stellen Sie in ihrer obigen Implementierung sicher, dass T2 erst dann ausgeführt wird, wenn T1 das benötigte Ergebnis bereitgestellt hat. Nehmen Sie weiterhin an, dass Sie keine Annahme über die Reihenfolge der Aktivierung von T1 und T2 machen können.
- Nehmen Sie an, die WCET der Abarbeitung von Aufgabe T1 verlängert sich aufgrund transienter Fehler manchmal von 4 ms auf 35 ms. Welche Auswirkungen hat dieser Fehler auf das vorliegende System? Vergleichen Sie dieses Verhalten mit dem eines zeitgesteuerten Systems. Welche Unterschiede stellen Sie fest?

Aufgabe 2 Gegeben seien die drei Aufgaben in der folgenden Tabelle:

Aufgabe	p/a	Periode	min. Zwischenankunftszeit	WCET
T1	p	5 ms	NA	0.5 ms
T2	p	10 ms	NA	1 ms
T3	p	20 ms	NA	5 ms

Hier besteht nun ein Produzenten-Konsumenten-Verhältnis zwischen T1 und T3! Weil T1 diese Zwischenergebnisse schneller erzeugt, als T3 sie abarbeitet, müssen sie in einem Ringpuffer zwischengespeichert werden. Bei diesem Puffer handelt es sich um eine gemeinsame Datenstruktur, der Zugriff hierauf muss also synchronisiert werden. In Ermangelung entsprechender Mechanismen, können Sie hierfür die globale Unterbrechungssynchronisation mit `guard.enter()` und `guard.leave()` verwenden. **Hinweis:** Sollte das eingesetzte Betriebssystem Mechanismen zur Fadensynchronisation zur Verfügung stellen, sollten diese unbedingt anstelle von `guard.enter()` und `guard.leave()` verwendet werden.

- Implementieren Sie die Aufgaben T1, T2 und T3 wie in Teilaufgabe 1, verwenden Sie einen Ringpuffer für den Datenaustausch zwischen T1 und T3. Vergeben Sie auch hier die Prioritäten der einzelnen Aufgaben gemäß des *Rate Monotonic Algorithm*.
- Nehmen Sie an, dass Aufgrund eines transienten Fehlers die Aufgabe T2 vorübergehend alle 2 ms ausgelöst wird. Welche Folgen hat dies für die Aufgabe T3? Welche Folgen hat dies mittelbar für die Aufgabe T1? **Hinweis:** Wie verfährt die Aufgabe T1, wenn der gemeinsame Puffer voll ist?
- Nehmen sie nun an, die Aufgabe T1 trete nun für einen gewissen Zeitraum alle 2.5 ms statt alle 5 ms ein. Welche Folgen hat dies für die Aufgabe T3? In welchen Szenarien könnte dieses Problem besonders problematisch sein?

Allgemeine Hinweise

- Entpacken Sie die Vorgabe für Aufgabe 3 im Unterverzeichnis `aufgabe3`. Erstellen sie im Unterverzeichnis `tests` für jede der Aufgaben eine Datei `aufgabeX.cc` X=1,2,3, in der Sie die einzelnen Aufgaben implementieren.
- Beantworten Sie die Zusatzfragen innerhalb der Dateien `aufgabeX.cpp`. Geben Sie jeweils deutlich an, auf welche Zusatzaufgabe sich Ihre Erläuterungen beziehen.
- Für etwaige Zeitmessungen verwenden Sie die Klasse `Timer` (siehe EZ-Stubs Einführung, Folien 31ff).
- In manchen Fällen sind Zufallszahlen hilfreich - in `debug/random.h` findet ihr die Schnittstelle eines Generators für Pseudozufallszahlen (Mersenne-Twister). **Wichtig:** Vor der Verwendung mit `random_init()` initialisieren.