

## Übung 5 - Musterlösung

### Aufgabe 1

Gegeben seien 3 periodische Tasks  $t_1, t_2, t_3$  mit Periodendauern  $\Delta t_i$  und maximalen Ausführungszeiten  $\Delta e_i$ :

| Task | $\Delta t_i$ | $\Delta e_i$ |
|------|--------------|--------------|
| 1    | 3            | 1            |
| 2    | 6            | 2            |
| 3    | 18           | 5            |

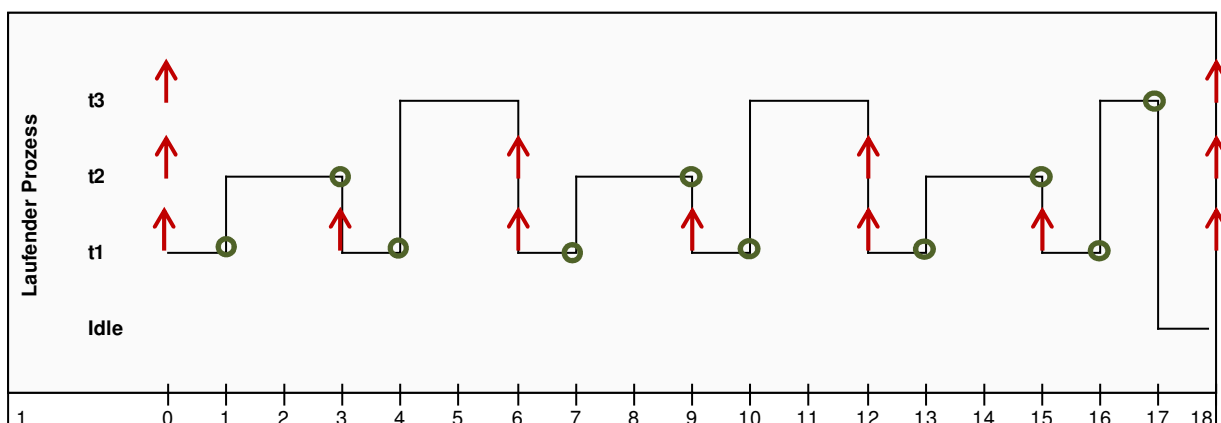
- Entwickeln Sie einen brauchbaren Plan nach monotonen Raten für die Taskmenge
- Entwickeln Sie einen zyklischen Plan für die Taskmenge. (Gehen Sie davon aus, dass jede der Tasks, falls sinnvoll und erforderlich, in kleinere Teiltasks aufgeteilt werden kann.)

Lsg:

a) Ein Plan nach monotonen Raten ( $p_1 = 3, p_2 = 2, p_3 = 1$ ) ist durch folgendes Diagramm (ein UML-Timing-Diagramm) beschrieben. Bei der Interpretation ist folgendes zu beachten:

- Die untere horizontale Skala enthält die diskreten Zeitpunkte
- Die laufenden Tasks sind vertikal angeordnet
- Eine Waagrechte Linie auf Höhe einer Task, bedeutet, dass die Task in diesem Zeitraum den Prozessor nutzt.
- Vertikale Linien stehen für Task-Wechsel
- ↑ = Release-Start
- = Zeitgerechtes Ausführungsende eines Releases

Wenn die Task "Idle" aktiv ist, heißt das, dass gerade keine Task in Ausführung ist.



b)

Die Task 3 wird in 3 Abschnitte eingeteilt, 3a, 3b, 3c (Ausführungszeiten laut Tabelle unten):

Es ergibt sich ein Zyklischer Plan mit  $\Delta mict = 3$  und  $\Delta mjet = 18$  (6 Runden)

| <b>Task</b> | $\Delta t_i$ | $\Delta e_i$ |
|-------------|--------------|--------------|
| 1           | 3            | 1            |
| 2           | 6            | 2            |
| 3a          | 18           | 2            |
| 3b          | 18           | 2            |
| 3c          | 18           | 1            |

Zyklischer Plan (jede Task-Nummer steht für eine Zeiteinheit, – bedeutet "IDLE", d.h. keine Task in Ausführung, | trennt die inneren Zyklen, dargestellt ist ein äußerer Zyklus):

1 2 2 | 1 3a 3a | 1 2 2 | 1 3b 3b | 1 2 2 | 1 3c –

Im Grunde entspricht dieser zyklische Plan exakt dem Plan nach monotonen Raten, wobei die 3 Ausführungsabschnitte der Task 3 in den Zeitintervallen [4,6), [10-12) und [16-17) den Abschnitten 3a, 3b und 3c entsprechen.

## Aufgabe 2

Gegeben seien 3 periodische Tasks  $t_1, t_2, t_3$  mit Periodendauern  $\Delta t_i$ , maximalen Ausführungszeiten  $\Delta e_i$  und von der Anwendung vorgegeben Prioritäten (nach "Wichtigkeit")  $p_i$ :

| <b>Task</b> | $\Delta t_i$ | $\Delta e_i$ | $p_i$ |
|-------------|--------------|--------------|-------|
| 1           | 100          | 30           | 3     |
| 2           | 6            | 1            | 2     |
| 3           | 25           | 5            | 1     |

- Beweisen Sie, dass es für die Taskmenge keinen brauchbaren Plan nach festen Prioritäten gibt.
- Ändern Sie die Prioritäten so, dass die Taskmenge einen brauchbaren Plan hat und beweisen Sie dies (ohne den Plan aufzustellen)
- Entwickeln Sie einen brauchbaren Plan für die Taskmenge mit den Prioritäten nach (b)

Lsg:

a) Task 1 hat die höchste Priorität und startet deshalb als erstes und wird vollständig ausgeführt, also 30 Zeiteinheiten. Sie kann nicht durch andere Tasks, die alle niedrigere Priorität haben, verdrängt werden. – Bis dahin gibt es aber schon mehrere Fristverletzungen für Task 2 (erste Deadline ist bei 6!) und eine für Task 3 (erste Deadline bei 25).

b) Die Prioritäten nach monotonen Raten sind wie folgt:  $p_1 = 1, p_2 = 3, p_3 = 2$ .

Der LL-Test für diese Taskmenge ergibt folgendes (vgl. Skript, S. 6-17):

$$\begin{aligned}
 U(T) &= \sum_{i=1}^3 \frac{\Delta e_i}{\Delta t_i} = \frac{30}{100} + \frac{1}{6} + \frac{5}{25} \\
 &= \frac{3}{10} + \frac{1}{6} + \frac{1}{5} = \frac{9 + 5 + 6}{30} = \frac{20}{30} = 0,66
 \end{aligned}$$

Die Auslastungsgrenze für 3 Tasks ist ca. 0,78. Damit ist  $U(T)$  kleiner als die Auslastungsgrenze und die Taskmenge ist planbar.

c) Die folgende Tabelle zeigt den Plan wie folgt: in der ersten Spalte steht die Task, die den Prozessor belegt, in der zweiten Spalte, wie lange die Task den Prozessor belegt und in der dritten Spalte die Gesamt-Rechenzeit, wenn die Task den Prozessor wieder freigibt. Ist für eine Task ein Release-Ende erreicht, ist Tasknummer und Rechenzeit **fett und rot** gedruckt. (D.h., wenn die Task wieder rechnet, entspricht das dem Start eines neuen Releases.

| Task     | Ausführungszeit | Gesamtausführungszeit |
|----------|-----------------|-----------------------|
| <b>2</b> | <b>1</b>        | 1                     |
| <b>3</b> | <b>5</b>        | 6                     |
| <b>2</b> | <b>1</b>        | 7                     |
| 1        | 5               | 12                    |
| <b>2</b> | <b>1</b>        | 13                    |
| 1        | 5               | 18                    |
| <b>2</b> | <b>1</b>        | 19                    |
| 1        | 5               | 24                    |
| <b>2</b> | <b>1</b>        | 25                    |
| <b>3</b> | <b>5</b>        | 30                    |
| <b>2</b> | <b>1</b>        | 31                    |
| 1        | 5               | 36                    |
| <b>2</b> | <b>1</b>        | 37                    |
| 1        | 5               | 42                    |
| <b>2</b> | <b>1</b>        | 43                    |
| <b>1</b> | <b>5</b>        | 48                    |
| <b>2</b> | <b>1</b>        | 49                    |
| IDLE     | 1               | 50                    |

### Aufgabe 3

Gegeben seien 3 periodische Tasks  $t_1, t_2, t_3$  mit Periodendauern  $\Delta t_i$  und maximalen Ausführungszeiten  $\Delta e_i$  und Prioritäten  $p_i$  nach monotonen Raten:

| Task | $\Delta t_i$ | $\Delta e_i$ | $p_i$ |
|------|--------------|--------------|-------|
| 1    | 19           | 3            | 1     |
| 2    | 9            | 2            | 2     |
| 3    | 4            | 2            | 3     |

Führen Sie für diese Taskmenge (a) den LL-Test, (b) den HB-Test und (c) eine Antwortzeitanalyse durch.

a) Der LL-Test ist negativ:

$$U(T) = \sum_{i=1}^3 \frac{\Delta e_i}{\Delta t_i} = \frac{3}{19} + \frac{2}{9} + \frac{2}{4} = 0.16 + 0.23 + 0.05 = 0.89 > \mathbf{0.78}$$

b) Der HB-Test ist ebenfalls negativ:

$$\prod_{i=1}^3 \left( \frac{\Delta e_i}{\Delta t_i} + 1 \right) = 1.16 * 1.23 * 1.5 = 2.1402 > \mathbf{2}$$

c) Antwortzeitanalyse: Wir lösen die Rekurrenz-Relation  $w_i^{n+1} = \Delta e_i + \sum_{j \in hp(i)} \left\lceil \frac{w_i^n}{\Delta t_j} \right\rceil \Delta e_j$  mit dem Lösungsalgorithmus (Skript S. 6-24):

$$w_1^0 = 3$$

$$w_1^1 = 3 + \left\lceil \frac{3}{9} \right\rceil * 2 + \left\lceil \frac{3}{4} \right\rceil * 2 = 3 + 2 + 2 = 7$$

$$w_1^2 = 3 + \left\lceil \frac{7}{9} \right\rceil * 2 + \left\lceil \frac{7}{4} \right\rceil * 2 = 3 + 2 + 4 = 9$$

$$w_1^3 = 3 + \left\lceil \frac{9}{9} \right\rceil * 2 + \left\lceil \frac{9}{4} \right\rceil * 2 = 3 + 2 + 6 = 11$$

$$w_1^4 = 3 + \left\lceil \frac{11}{9} \right\rceil * 2 + \left\lceil \frac{11}{4} \right\rceil * 2 = 3 + 4 + 6 = 13$$

$$w_1^5 = 3 + \left\lceil \frac{13}{9} \right\rceil * 2 + \left\lceil \frac{13}{4} \right\rceil * 2 = 3 + 4 + 8 = \mathbf{15}$$

$$w_1^6 = 3 + \left\lceil \frac{15}{9} \right\rceil * 2 + \left\lceil \frac{15}{4} \right\rceil * 2 = 3 + 4 + 8 = \mathbf{15} = \Delta r_1 < \mathbf{19} = \Delta t_1$$

$$w_2^0 = 2$$

$$w_2^1 = 2 + \left\lceil \frac{2}{4} \right\rceil * 2 = 3 + 2 = 5$$

$$w_2^2 = 2 + \left\lceil \frac{5}{4} \right\rceil * 2 = 3 + 4 = \mathbf{7}$$

$$w_2^3 = 2 + \left\lceil \frac{7}{4} \right\rceil * 2 = 3 + 4 = \mathbf{7} = \Delta r_2 < \mathbf{9} = \Delta t_2$$

$$w_3^0 = 2 = \Delta r_3 < \mathbf{4} = \Delta t_3$$

Da für alle Tasks  $\Delta r_i < \Delta d_i = \Delta t_i$  gilt, ist die Taskmenge planbar.

## Aufgabe 4

Gegeben seien 4 periodische Tasks  $t_1, t_2, t_3, t_4$  mit Startzeiten  $s_i$ , maximalen Ausführungszeiten  $\Delta e_i$  und Prioritäten  $p_i$ . Die Tasks benutzen zwei kritische Bereiche die durch zwei Semaphore A und B geschützt sind. Die ununterbrochene Ausführung der Tasks ist beschrieben durch entsprechende Ausführungsfolgen. Jeder Großbuchstabe in einer Anweisungsfolge entspricht einer Zeiteinheit (A=rechnet im kritischen Bereich A, B= rechnet im kritischen Bereich B, E=rechnet außerhalb des kritischen Bereichs, vgl. Skript S. 6-28ff).

Das jeweils erste Auftreten einer Aktion im kritischen Bereich wird eingeleitet durch eine *wait*-Operation (wenn der Semaphor nicht frei ist, wird die Task blockiert). Diese ist in der Ausführungsfolge explizit dargestellt durch einen Kleinbuchstaben (*a* bzw. *b*). Nach der letzten Aktion in einem kritischen Bereich findet eine *signal*-Operation statt. Diese ist in der Ausführungsfolge explizit dargestellt durch einen "gestrichenen" Kleinbuchstaben (*a'* bzw. *b'*). Eventuell auf den Semaphor wartende Tasks werden dann wieder rechenbereit. Sowohl *wait*- als auch *signal*-Operationen haben aber keine signifikante Zeitdauer.



| Task | $p_i$ | $s_i$ | $\Delta e_i$ | Ausführungsfolge |
|------|-------|-------|--------------|------------------|
| 1    | 4     | 7     | 4            | EaAAa'bBb'       |
| 2    | 3     | 2     | 5            | EbBaAa'Bb'E      |
| 3    | 2     | 5     | 4            | EEEE             |
| 4    | 1     | 0     | 5            | EaAAAa'E         |

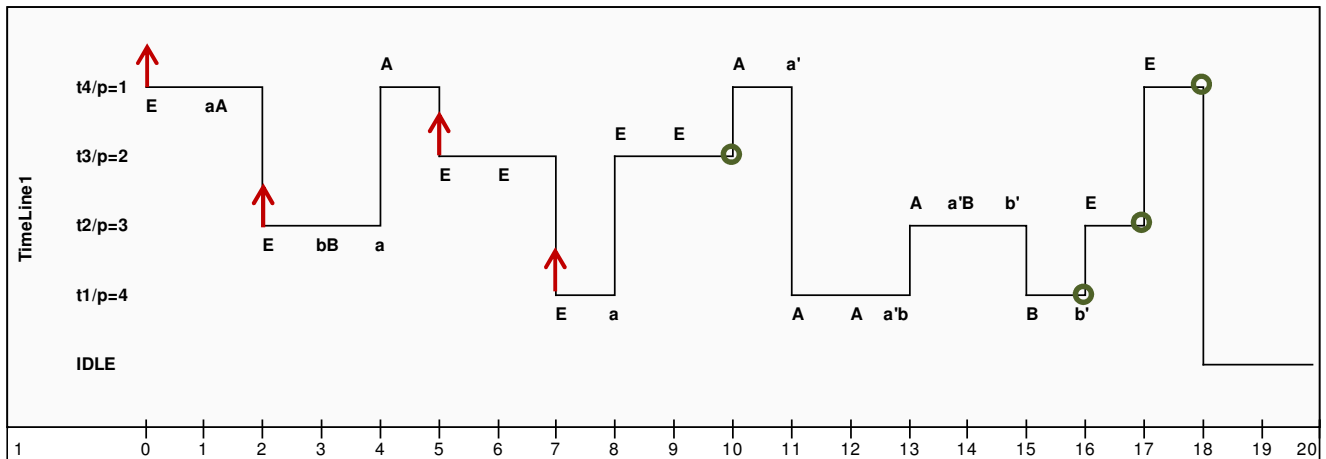
- Beschreiben Sie die Ausführung der drei Tasks auf einem Prozessor beim Planen nach monotonen Raten mit Verdrängen (*preemptive Multitasking*).
- Beschreiben Sie, wie sich das Verhalten unter Anwendung von *Prioritätsvererbung* ändert.

Lsg.:

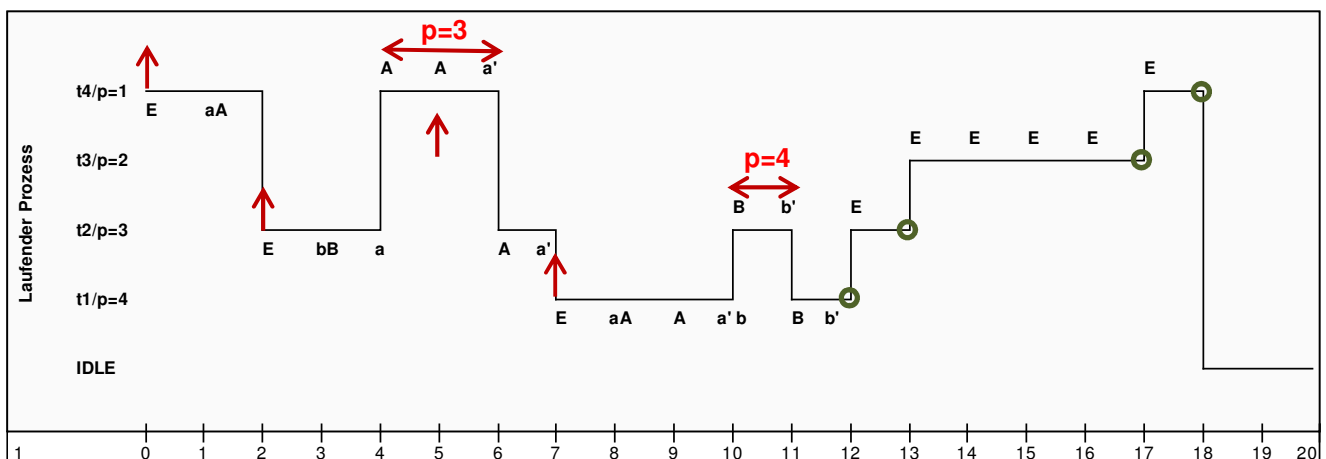
a)

Bei der Interpretation des folgenden Diagramms (ein UML-Timing-Diagramm) ist folgendes zu beachten:

- Die untere horizontale Skala enthält die diskreten Zeitpunkte
- Die laufenden Tasks sind vertikal angeordnet
- Eine Waagrechte Linie auf Höhe einer Task, bedeutet, dass die Task in diesem Zeitraum den Prozessor nutzt.
- Vertikale Linien stehen für Task-Wechsel
- Die Beschriftung an den vertikalen Linien bezeichnen die Durchgeführten Aktionen der Task
- Wenn bei einer Task  $t$  vor einem Taskwechsel  $a$  oder  $b$  erfolgt, heißt das, dass die Task  $t$  in diesem Moment blockiert wird, weil sie den kritischen Bereich A bzw. B betreten will, dieser aber belegt ist
- Ansonsten finden Taskwechsel immer zu der Task statt, die die höchste Priorität hat und rechenbereit (also nicht blockiert) ist.
-  = Release-Start
-  = Release-Ende



b) Mit Prioritätsvererbung (*priority inheritance protocol*) – Eine Task im kritischen Bereich übernimmt die Priorität der Task, die sie gerade verdrängt.



Man sieht, dass das Release-Ende der Tasks hier der Reihenfolge in den Prioritäten entspricht. Insbesondere sind die Tasks mit hoher Priorität (t1 und t2) deutlich früher fertig als ohne Prioritätsvererbung.

Beim *Priority Ceiling Protocol* übernimmt eine Task zur Laufzeit beim Betreten eines kritischen Bereichs die Prioritätsobergrenze aller kritischen Bereiche, die sie belegt (vgl. Skript S.6-35).

Die Prioritätsobergrenzen sind hier wie folgt:

$$p_{\max}(A) = p_{\max}(B) = 4.$$

D.h. Task t1 läuft im Intervall [4,7) mit Priorität 4 statt mit Priorität 3. An der Ausführungsreihenfolge ändert sich in diesem Beispiel dadurch nichts.