VO 182.022 22. Oktober 2008

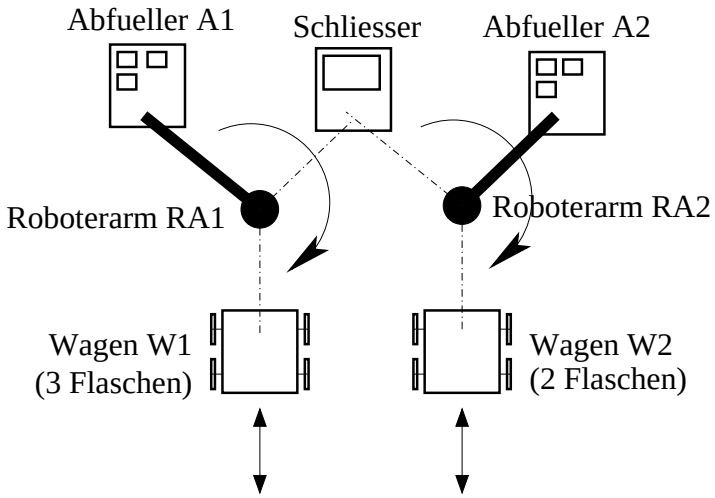
Prüfung Betriebssysteme

KNr. MNr. Zuname, Vorname

Ges.)(100)	1.)(30)	2.)(20)	3.)(20)	4.)(30)	Zusatzblätter:

Bitte verwenden Sie nur dokumentenechtes Schreibmaterial!

1 Synchronisation (30)



Die Molkerei *Lolatte* besitzt zwei Abfüllanlagen sowie eine gemeinsame Schließanlage für die Flaschen nach dem Befüllen. Je ein Roboterarm nimmt fertig gefüllte Flaschen von der jeweiligen Abfüllanlage und bringt sie zum Schließer, wo die Flasche luftdicht abgeschlossen wird. Anschließend wird die Flasche in einen Wagen gestellt, welcher regelmäßig ausgelehrt wird. Der Wagen "W1" kann drei Flaschen aufnehmen, bevor er geleert werden muss. Der Wagen "W2" kann nur zwei Flaschen aufnehmen, bevor er wieder geleert werden muss. Die Roboterarme haben die Eigenheit, dass sie sich beide **nur im Uhrzeigersinn drehen** lassen, wie auch in obiger Abbildung dargestellt.

Das Beladen der beiden Wagen läuft nach folgenden Regeln ab:

- Jeder Roboterarm kann direkt ohne Zeitverzögerung von seiner Abfüllanlage eine Flasche holen.
- Wenn ein Roboterarm sich gerade zum Schließer hinbewegt oder bereits wieder wegbewegt, wäre die Gefahr gegeben, dass er mit dem zweiten Roboterarm kollidiert. Daher muss dieser Abschnitt entsprechend gesichert werden.
- Der *Schließer* wartet bis ein Roboterarm direkt vor ihm eine Flasche platziert hat und verschließt diese dann.

- Der Drehwinkel für den Roboterarm ist zwischen dem Schließer und der Abfüllanlage bzw. dem Wagen genau 120 Grad.
- Wenn ein Wagen fertig beladen ist, soll er sofort mit seiner Auslieferung beginnen.
- Mit dem Anfüllen eines Wagens kann natürlich erst dann wieder begonnen werden wenn der Wagen zurückgekehrt ist. Der Roboterarm kann allerdings schon die nächste Flache verschließen bevor der Wagen zurueck ist.

Synchronisieren Sie den Arbeitsablauf der beiden Roboterarme, des Schließers und der beiden Wagen mittels **Semaphoren**. Achten Sie auf maximale Parallelität. Verwenden Sie möglichst wenige Synchronisationskonstrukte. Die Verwendung von globalen Variablen ist verboten.

Allgemeine zu verwendende Funktionen:

`initS(` ***Semaphor,init*** `)` Legt einen Semaphor mit dem angegebenen Namen *Semaphor* an und initialisiert ihn mit der Zahl *init*. Danach können die Funktionen **P(*Semaphor*)** und **V(*Semaphor*)** auf den Semaphor angewendet werden.

Zu verwendende Funktionen für den Schließer:

`close()` Mit dieser Funktion wird eine Flasche verschlossen. Funktioniert nur, wenn einer der Roboterarme zum Abfüller ausgerichtet ist und eine Flasche häelt. Die Flasche braucht zum Schließen vom Roboterarm nicht losgelassen werden.

Zu verwendende Funktionen für die Roboterarme:

`turn(` ***Winkel*** `)` Bewegt den Roboterarm im Uhrzeigersinn um *Winkel* Grad weiter (Winkel ist ein Wert im Bereich von 0-360 Grad). Beachten Sie, dass Roboterarm RA2 komplett baugleich wie RA1 ist und sich daher ebenfalls ausschließlich im Uhrzeigersinn dreht!

`get()` Nimmt von der Abfüllanlage eine Flasche. Funktioniert nur, wenn der Roboterarm zur Abfüllanlage ausgerichtet ist.

`put()` Stellt die Flasche in den Wagen. Funktioniert nur, wenn der Roboterarm zum jeweiligen Wagen ausgerichtet ist.

Zu verwendende Funktionen für die Wagen:

`deliver()` Mit dieser Funktion fährt der Wagen in die Vertriebshalle und kommt anschließend wieder leer zurück.

a) Initialisierungen (4)

Initialisieren Sie die notwendigen Semaphore. Der **Anfangszustand** des Systems entspricht dem obigem Bild. Beide Wagen sind unbeladen und beide Roboterarme sind zur jeweiligen Abfüllanlage platziert.

b) (4)

Entwerfen Sie einen Prozess für *Schliesser*.

```
Prozess Schliesser:

do forever() {

                                                    }
```

c) (16)

Entwerfen Sie je einen Prozess für *Roboterarm RA1* und *Roboterarm RA2*.

Prozess *Roboterarm RA1*:

```
do forever() {
}
```

Prozess *Roboterarm RA2*:

```
do forever() {
}
```

d) (6)

Entwerfen Sie die Prozesse für *Wagen W1* und *Wagen W2*:

Prozess *Wagen W1*:

```
do forever() {

}
```

Prozess *Wagen W2*:

```
do forever() {

}
```

2 Security (20)

2.1 Security by Obscurity vs. Open Design (4)

Beschreiben Sie den Unterschied zwischen *Security by Obscurity* und *Open Design*. Geben Sie die Vor- und Nachteile an.

2.2 Verständnisfragen (4)

Beurteilen Sie die folgenden Aussagen! Fehlende Antworten werden negativ, falsche Antworten werden doppelt negativ gewertet!

◯ Ja ◯ Nein *Threshold detection* ist ein statistisches Verfahren zur Intrusion Detection.

◯ Ja ◯ Nein Das *Least Privilege* Prinzip besagt, dass jeder Benutzer alle Rechte per default hat.

◯ Ja ◯ Nein Bei *Masquerading* wird der Inhalt einer Message verändert.

◯ Ja ◯ Nein Eine *denial-of-service* Attacke ist eine passive Attacke.

2.3 Begriffe (8)

Was versteht man unter *Confidentiality (Secrecy)*, *Integrity* und *Availability*? Erklären Sie die drei Begriffe.

- Confidentiality:

- Integrity:

- Availability:

2.4 Wie funktioniert Public Key Encryption? (4)

3 Uniprocessor Scheduling (20)

Gegeben ist nebenstehendes Taskset. Alle Tasks sind periodisch, wobei die Deadlines mit dem Ende der jeweiligen Periode gleichzusetzen sind. Der Overhead für den Taskwechsel ist vernachlässigbar.

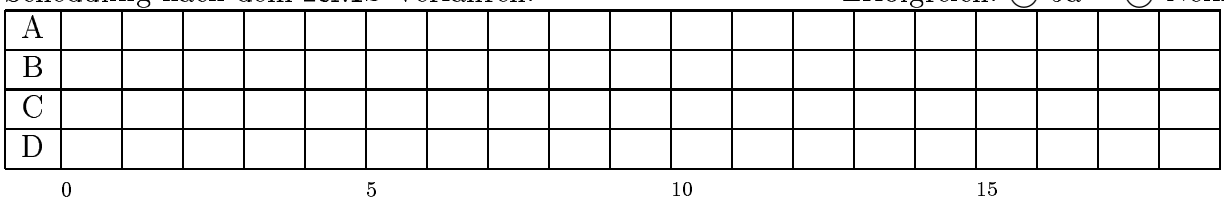
Task	Ausführungszeit	Periodendauer
A	1	7
B	2	6
C	2	8
D	1	9

Ermitteln Sie für dieses Taskset die *notwendige* und die *hinreichende* Bedingung für das *Rate Monotonic Scheduling* (RMS) Verfahren. Berechnen Sie die **Zahlenwerte** überschlagsmäßig ($\sqrt[2]{2} \approx 1,41 \quad \sqrt[3]{2} \approx 1,26 \quad \sqrt[4]{2} \approx 1,19 \quad \sqrt[5]{2} \approx 1,15 \quad \sqrt[6]{2} \approx 1,12$).

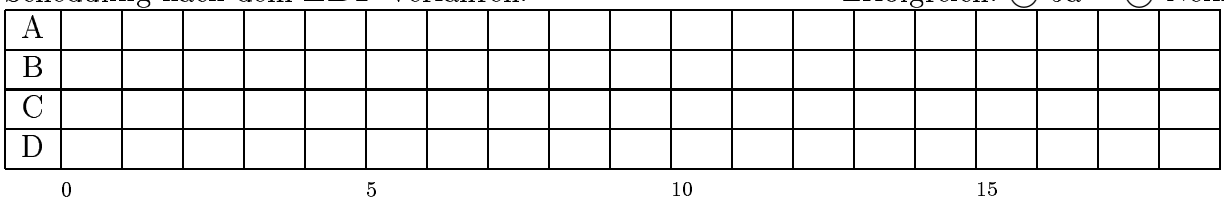
Ist die notwendige Bedingung erfüllt? ◯ Ja ◯ Nein
Ist die hinreichende Bedingung erfüllt? ◯ Ja ◯ Nein

Versuchen Sie das Taskset einmal mit dem RMS und einmal mit dem *Earliest-Deadline-First* (EDF) Verfahren zu schedulen. Verwenden Sie dazu die nachstehenden Vorlagen. Tragen Sie bei jeder Vorlage die aktiven Taskzeiten ein und bezeichnen Sie deutlich eventuelle Deadlineverletzungen. Kreuzen Sie jeweils an, ob das Scheduling erfolgreich war. Eine Vorlage dient als Ersatz, streichen Sie gegebenenfalls eine falsch ausgefüllte Vorlage deutlich durch.

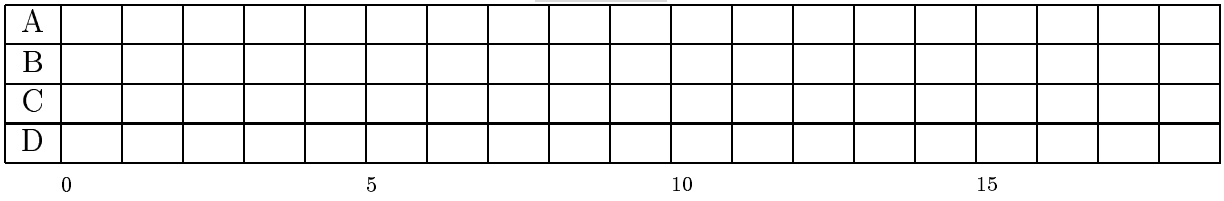
Scheduling nach dem **RMS**-Verfahren: Erfolgreich: ◯ Ja ◯ Nein



Scheduling nach dem **EDF**-Verfahren: Erfolgreich: ◯ Ja ◯ Nein



Ersatzvorlage: Scheduling nach dem -Verfahren: Erfolgreich: ◯ Ja ◯ Nein



4 BS-Abstraktionen und Prozesse (30)

Nennen Sie drei Abstraktionen, die ein Betriebssystem zur Verfügung stellt. Geben Sie für jede dieser Abstraktionen einerseits die Benutzerwahrnehmung andererseits die zur Realisierung der Abstraktion notwendigen Betriebssystemmechanismen an.

Welche Schritte muss ein Betriebssystem bei einem Process Switch (Context Switch) durchführen?

Nennen Sie die prozessspezifischen Informationen, die ein Betriebssystem im Process Control Block verwaltet.

Erklären Sie den Unterschied zwischen den Begriffen *Prozess* und *Thread*. In welcher Beziehung stehen diese beiden Konzepte?

In der Vorlesung wurden zwei Möglichkeiten der Implementierung von Threads genannt. Um welche Implementierungen handelt es sich dabei? Wie machen sich die Unterschiede in der Implementierung für Applikationen bemerkbar?

Erklären Sie, was man unter (a) einem *monolithischen Kernel* und (b) einem *Microkernel* versteht.