Vorlesungsprüfung aus Betriebssysteme 2003-01-22

KNr. MNr. Zuname, Vorname

Ges.)(100) 1.)(25) 2.)(30) 3.)(23) 4.)(22) Zusatzblätter:

Bitte verwenden Sie nur dokumentenechtes Schreibmaterial!

1 Deadlock (25)

Gegeben sind zwei Prozesse P_1, P_2 und zwei Ressourcen R_1, R_2. Die Ressourceallokationen der Prozesse in Abhängigkeit des jeweiligen Prozessfortschritts sind in Tabelle 1 eingetra-

Zeit	P1	P2
t=0		
t=1	P(R1)	P(R1)
t=2	P(R1)	P(R1)
t=3	P(R1)	
t=4		P(R2)
t=5		
t=6	V(R1)	
t=7	V(R1)	V(R1)
t=8		V(R1)
t=9	V(R1)	
t=10		
t=11	P(R2)	P(R1)
t=12	P(R2)	P(R1)
t=13		P(R1)
t=14	V(R2)	V(R1)
t=15	V(R2)	V(R1)
t=16		V(R1)
t=17		
t=18		V(R2)
t=19		
t=20	Termination	Termination

Tabelle 1: Prozess P1 und P2

gen. Benötigt ein Prozess eine vom anderen Prozess belegte Ressource, so wird er auf jeden Fall bis zum Freiwerden der Ressource verzögert. Zu Beginn sind alle Ressourcen verfügbar. Von Ressource R_1 sind 3 Einheiten vorhanden, von R_2 sind 2 Einheiten vorhanden.

1.1 Abarbeitungs Diagramm

Abbildung 1 stellt ein Abarbeitungs Diagramm für die Prozesse P_1 und P_2 dar. Der Fortschritt von P_1 und P_2 bei der (quasi)parallelen Abarbeitung kann als Kantenzug zwischen den Punkten *start* und *end* in der Grafik eingetragen werden (siehe Buch zur Vorlesung: W. Stallings, Operating Systems).

1. Umranden und schraffieren Sie in der Grafik jene Bereiche, durch die ein solcher Kantenzug aufgrund von Ressourcenkonflikten nicht gehen kann.(3P. pro Flaeche, 1P Abzug pro falsches Kästchen)

2. Kennzeichnen Sie auf unterschiedliche Weise die Bereiche, die von einem Kantenzug nicht passiert werden dürfen, wenn eine Abarbeitung von P_1 und P_2 deadlockfrei erfolgen soll. Beschriften Sie diese Bereiche deutlich mit einem "D". (4P. f. oben, 2P. f. unten, 1P Abzug pro falsches Kästchen, alles falsch - keine Punkte)

3. Zeichnen Sie einen Kantenzug für eine gültige, deadlockfreie Abarbeitung von P_1 und P_2 in der Grafik ein. (1P)

4. Stellt der Punkt P einen Deadlock dar? Begründen Sie Ihre Antwort! (2P, 1P f. Ja/Nein, 1P f. Begruendung)

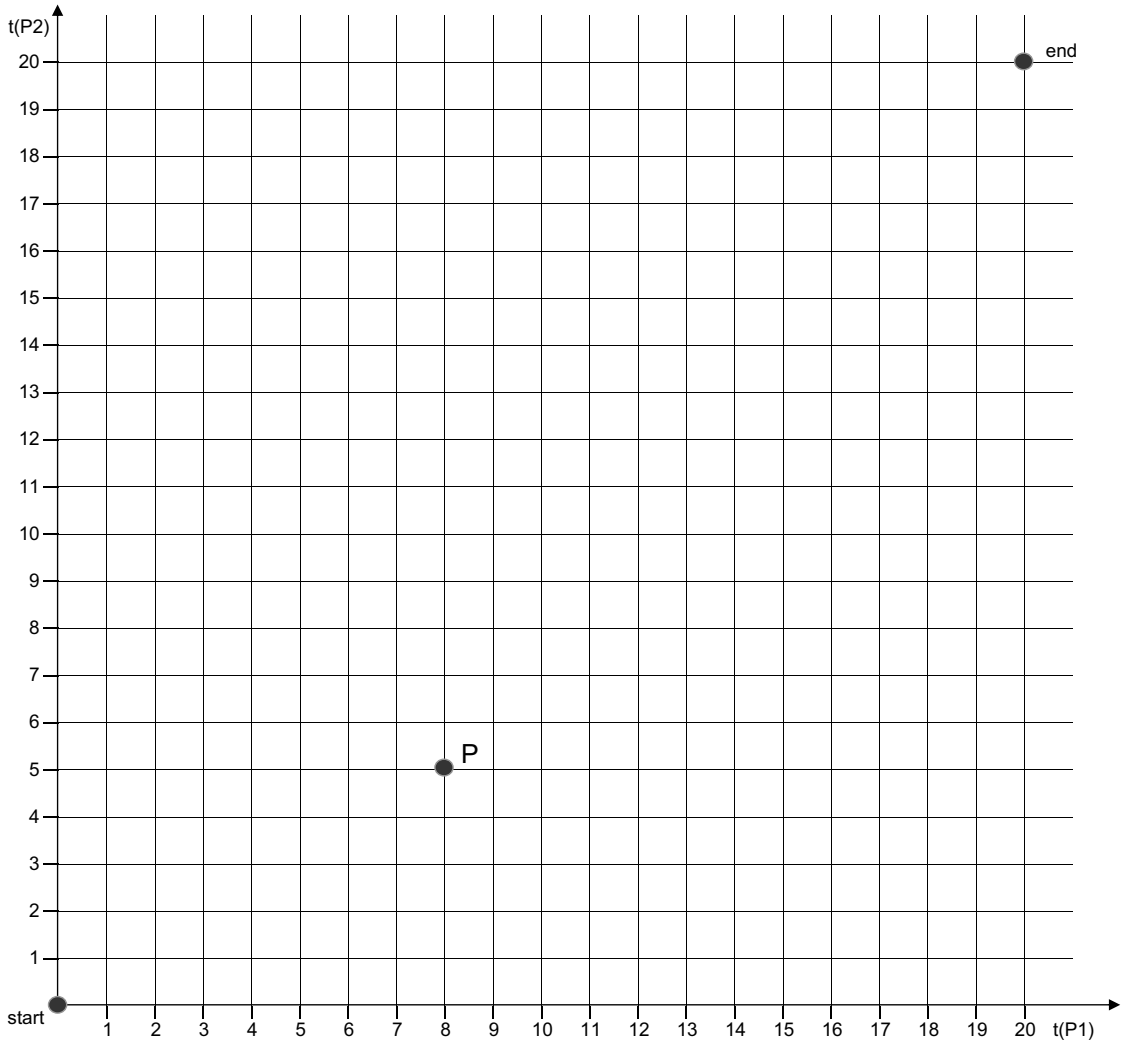


Abbildung 1: Abarbeitungs Diagramm

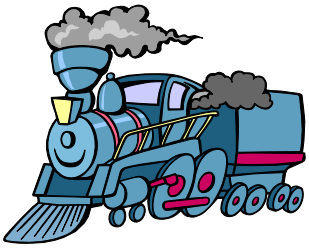
1.2 Deadlock Avoidance

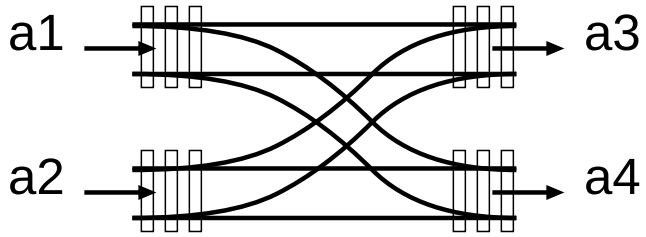
Bestimmen Sie die Claim-Matrix C der Prozesse, sowie die Allokationsmatrix A im Punkt P. (Claim: 2 P, Alloc: 2P)

Ist P ein Safestate? Wenn ja, geben Sie mit dem Banker's Algorithm eine Ableitung an. Wenn nein, begründen Sie warum P kein Safestate ist. (3P: 1P Ja/Nein, 2P Begruendung)

2 Synchronisation (30)

Der Verschubbahnhof *Innovativa* hat eine relativ einfache Gleisstruktur, wie in folgender Abbildung dargestellt ist:





Der Verschubbahnhof hat nur ein einziges Weichensystem, mit dem es möglich ist, von dem einen Gleis auf das andere zu wechseln. Die Anschlusspunkte des Weichensystems sind mit **a1, a2, a3** und **a4** benannt. Wichtig ist, dass entsprechend den eingezeichneten Pfeilen ein Zug nur von **a1** oder **a2** kommend nach **a3** oder **a4** fahren darf.

Für den Verschubbahnhof ist eine Synchronisations-Software zu entwickeln, um Zugkollisionen während des Passierens des Weichensystems zu vermeiden. Das Weichensystem stellt **drei Betriebsarten** zur Verfügung:

KEINE: Keine Züge dürfen passieren.

GERADE: Züge dürfen nur geradeaus passieren.

ALLE: Züge dürfen sowohl geradeaus als auch diagonal passieren.

Es sind folgende Funktionen zu implementieren:

zug_kontrolle(von, nach) zur Kontrolle von Zügen. Diese Funktion steuert einen Zug, fahrend von **von** nach **nach**. Der Zug soll entsprechend der aktuellen Weichenbetriebsart und anderen Zügen synchronisiert werden. Sollte **von**$\notin${**a1,a2**} oder **nach**$\notin${**a3,a4**} gelten (d.h. eine nicht im System vorgesehende Fahrtrichtung), so ist die unten beschriebene Funktion **setze_stop(von)** aufzurufen, wobei **von** die Richtung ist, aus der dieser Zug kommt (der Zug selbst bleibt in diesem Fall einfach stehen).
Wenn das Weichensystem passierbar ist (d.h., kein Konflikt mit der aktuellen Weichenbetriebsart und anderen Zügen), ist die Funktion **stelle_weichen(von, nach)** aufzurufen, um die Weiche korrekt zu stellen. Anschließend ist zum eigentlichen Durchfahren der Weichenanlage die Funktion **passiere()** aufzurufen.

setze_stop(von) zum Sichern der Weichenanlage. Der Parameter **von** bezeichnet die aktuelle Position eines Zuges (hier **von**$\in${**a1, a2, a3, a4**}). Die Funktion soll für die Position **von** die Strecken zu den beiden gegenüberliegenden Weichenanschlusspunkten sperren. Ein Zug in gleicher Fahrtrichtung am Parallelgleis geradeaus fahrend, soll jedoch weiterhin passieren dürfen.

Beispiel: Angenommen, ein Zug kommt fälschlicherweise von (**von=a4**). Somit sind alle weiteren Züge kommend aus **a1** oder **a2** und nach **a4** fahrend, zu blockieren. Ein Zug, kommend von **a1** oder **a2** und nach **a3** fahrend, darf jedoch weiterhin passieren.

weichen_betriebsart_uebernahme() Diese Prozedur ermittelt in einer Endlosschleife durch **p = hole_perm()** die aktuelle Weichenbetriebsart **p ∈ {KEINE, GERADE, ALLE}** und setzt entsprechend die im Programm benötigten Synchronisationskonstrukte.

Implementieren Sie alle Funktionen sowie Initialisierungen derart, sodass **defaultmäßig Weichenbetriebsart GERADE** aktiv ist (z.B.: wegen mechanischem Fehler in der Weichenumschaltung). Das heißt, Züge dürfen defaultmäßig nur geradeaus passieren.

Hinweis: Der Fall, dass hinter einem wartenden Zug ein neuer Zug nachkommt, braucht nicht berücksichtigt zu werden.

Die Verwendung von globalen Variablen bzw. Busy-Waiting zur Synchronisation ist verboten!

a) Initialisierungen (4)

Die Synchronisation ist mit (*möglichst wenig!*) Semaphoren durchzuführen wobei unnötige Einschränkungen der Parallelität zu vermeiden sind. Verwenden Sie zur Initialisierung der Semaphoren die Funktion **init(s,v)**, welche als ersten Parameter den Semaphor und als zweiten Parameter den entsprechenden Initialisierungswert erhält. Danach können die Funktionen **P(s)** und **V(s)** auf den Semaphor angewendet werden.

Geben Sie hier die notwendigen Initialisierungen von Semaphoren an.

b) Setzen einer neuen Weichenbetriebsart (8)

Programm zum Setzen der Weichenbetriebsart für Wartungsarbeiten.
`weichen_betriebsart_uebernahme()`

```
BEGIN

  current = GERADE;
```



```
END
```

c) Zugkontrolle (14)

Programm zur Kontrolle eines Zuges:
`zug_kontrolle(von, nach)`

```
BEGIN
```



```
END
```

d) Stopsignal setzen (4)

Programm zum Stoppen der für einen Zug stehend auf `von` kritischen entgegenkommenden Züge:
`setze_stop(von)`

```
BEGIN
```



```
END
```

3 Speicherverwaltung (23)

a) Kombination aus Segmentierung und Paging

Es werden folgende Begriffe (englische Notation) aus dem Buch zur Vorlesung verwendet:

Base	Basisadresse Seitentabelle des Segmentes
Length	Länge des Segmentes (Anzahl der Seiten des Segmentes)
Virt.Addr.	Virtuelle Adresse
Frame#	Seitenrahmennummer (im physischen Speicher)
Page#	Seitennummer (im virtuellen Speicher)
Seg#	Segmentnummer

Das im Folgenden betrachtete Speicherverwaltungssystem verwendet zur Adressierung 32-bit Adressen (virtuell und physikalisch). Für das Paging sind alle Seitenrahmen 256 Bytes groß. Das verwendete Adressformat ist folgendes:

Seg# (12 bit)	Page# (12 bit)	Offset (8 bit)

Hierbei wird assoziativer Zugriff (associative mapping) auf die Segmenttabelle und direkter Zugriff (direct mapping) auf die Seitentabelle verwendet.

Verwenden Sie für die Adressumsetzung folgende Segmenttabelle und Seitentabelle (alle Werte sind als Hexadezimalzahlen angegeben):

Segmenttabelle		
Seg#	Base	Length
0x000	0x34500234	0x0FF
0x39A	0x2A2AA342	0x005
0x723	0x2EE23323	0x002
0xCD3	0xB0010010	0x001

Seitentabelle	
Address	Frame#
0x2A2AA342	0x456D43
0x2A2AA343	0x123499
0x2A2AA344	0x19D453
0x2A2AA345	0x4F5D1D
...	...
0x2EE23323	0x453AA1
0x2EE23324	0x434DAA
0x2EE23325	0x421111
...	...
0x345002E7	0x12432A
0x345002E8	0xABDDAD
0x345002E9	0xDF45F4
0x345002EA	0x45DAEE
...	...
0xB0010010	0x761010
0xB0010011	0x859631
0xB0010012	0x5A64D2
0xB0010013	0x5A42DF
0xB0010014	0x4AF5DA
...	...

Ermitteln Sie unter Benützung obiger Tabellen die physikalischen Adressen zu folgenden virtuellen Adressen (ergibt sich bei der Umwandlung eine ungültige Adresse, so schreiben Sie bitte **ungültig** in das entsprechende Feld) (13P, pro Fehler -2):

Virtuelle Adresse	Physikalische Adresse (zu ermitteln)
0x39A00123	
0xCD30016A	
0x39A002A2	
0x72300023	
0x39A00023	
0x72300100	
0xCD300101	
0x0000B502	
0x0000B4FF	

b) Verständnisfragen

Kreuzen Sie bitte die richtigen Antworten an. Bzw., geben Sie an, ob die Aussage richtig oder falsch ist oder beantworten Sie die Frage. (Pro Frage 1P(ausser 7.frage), 7. Frage: pro kreuz ein punkt - falsches Kreuz -1 Punkt

- Eine Austauschstrategie, die auf alle Seiten des Hauptspeichers angewandt wird, nennt man
  - ◯ lokale Ersetzungsstrategie
  - ◯ globale Ersetzungsstrategie
- Beim *Fixed Partitioning* sind die Partionen *immer* von gleicher Größe.
  - ◯ richtig
  - ◯ falsch
- Beim *Fixed Partitioning* können sich Partitionen im Hauptspeicher überlappen.
  - ◯ richtig
  - ◯ falsch
- Bei einem Buddy System sind die Blockgrößen das Produkt einer Zweierpotenz mit der kleinsten Blockgröße.
  - ◯ richtig
  - ◯ falsch
- Beim *Fixed Partitioning* ist die Hauptspeichernutzung extrem effizient.
  - ◯ richtig
  - ◯ falsch
- Welche Replacement-Policy ist am einfachsten zu implementieren?
  - ◯ Least recently used
  - ◯ First in - first out
  - ◯ Optimale Strategie
- Zu welchen Effekten (mehrere möglich) kann es bei Segmentierung kommen?
  - ◯ Unterschiedliche Segmentlängen
  - ◯ Internal Fragmentation
  - ◯ External Fragmentation
- Was ist ein *Working Set* im Speichermanagement?

- Wenn alle Seiten eines Working Sets im Hauptspeicher sind, kann ein Prozess effizient abgearbeitet werden.
  - ◯ richtig
  - ◯ falsch

4 Scheduling (22)

4.1 Single Processor Scheduling (13)

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.

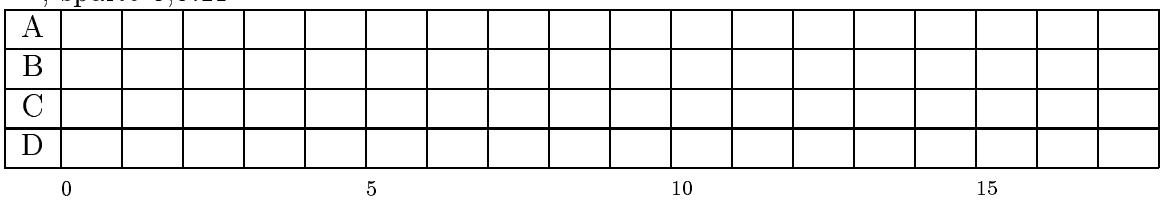
Task	Ausführungszeit	Periodendauer
A	2	8
B	1	4
C	2	7
D	1	5

Ermitteln Sie für dieses Taskset die *notwendige* und die *hinreichende* Bedingung für das *Rate Monotonic Scheduling* (RMS) Verfahren. Berechnen Sie die Zahlenwerte auf mindestens eine Kommastelle genau.

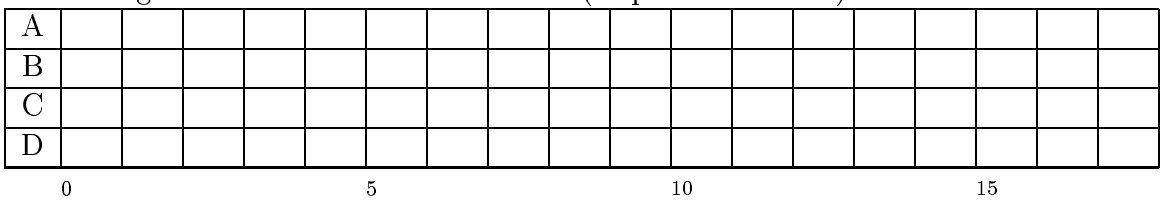
Ist die notwendige Bedingung erfüllt? 1P ◯ Ja ◯ Nein
Ist die hinreichende Bedingung erfüllt? 1P ◯ Ja ◯ Nein

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. Eine Vorlage dient als Ersatz, streichen Sie gegebenenfalls eine falsch ausgefüllte Vorlage deutlich durch.

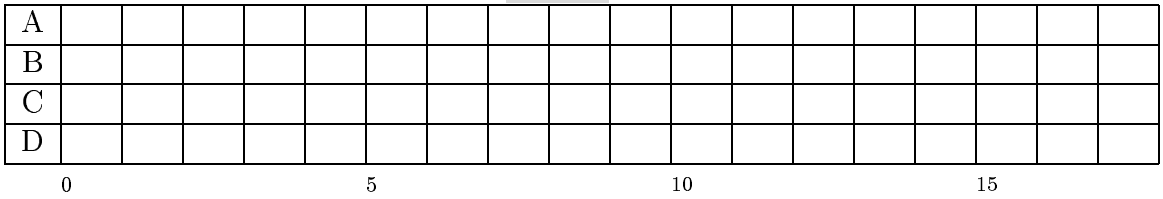
Scheduling nach dem **RMS**-Verfahren: ersten 3 spalten: 1 Punkt, spalte 4,5:1P; spalte 6,7: 1P; spalte 8,9:1P



Scheduling nach dem **EDF**-Verfahren: 5x (3 spalten: 1Punkt)



Ersatzvorlage: Scheduling nach dem -Verfahren:



4.2 Verständnisfragen (9)

Pro falsche oder fehlende Antwort -1P. Welche Scheduling-Strategie ähnelt einer Round Robin-Strategie mit sehr langen Zeitscheiben?

Beurteilen Sie die folgenden Aussagen! Falsche Antworten werden negativ gewertet!

◯ Ja ◯ Nein Earliest Deadline First Scheduling ist optimal in Single-Prozessor Systemen.

◯ Ja ◯ Nein Earliest Deadline First Scheduling ist optimal in Multi-Prozessor Systemen.

◯ Ja ◯ Nein Earliest Deadline First Scheduling findet in Single-Prozessor Systemen immer eine Lösung.

◯ Ja ◯ Nein Earliest Deadline First Scheduling findet in Single-Prozessor Systemen immer eine Lösung, wenn eine solche existiert.

◯ Ja ◯ Nein Earliest Deadline First Scheduling findet in Multi-Prozessor Systemen immer eine Lösung.

◯ Ja ◯ Nein Earliest Deadline First Scheduling findet in Multi-Prozessor Systemen immer eine Lösung, wenn eine solche existiert.

◯ Ja ◯ Nein Bei Round Robin Scheduling kann das Problem der Starvation nicht auftreten.

◯ Ja ◯ Nein Beim Scheduling nach dem Round-Robin-Verfahren werden I/O-intensive Prozesse benachteiligt.

◯ Ja ◯ Nein Wenn in einem Single-Prozessor-System bei allen Tasks die Deadline gleich ihrer Periode ist, dann liefert RMS Scheduling dasselbe Ergebnis wie EDF Scheduling.

◯ Ja ◯ Nein Die Prioritäten beim RMS Scheduling ergeben sich durch die *Processor Utilization* der Tasks.

◯ Ja ◯ Nein Eine für das RMS Scheduling hinreichende Bedingung stellt auch eine hinreichende Bedingung für das EDF Scheduling dar.