VO 182.022 14. Jänner 2009

Prüfung Betriebssysteme

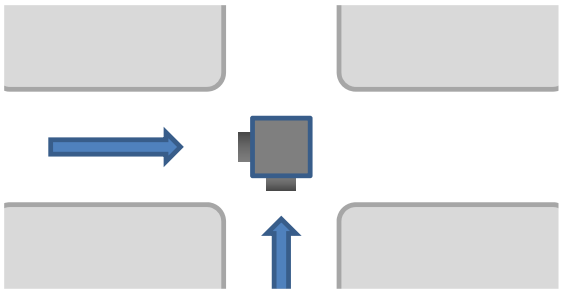
KNr. MNr. Zuname, Vorname

Ges.)(100)	1.)(36)	2.)(21)	3.)(18)	4.)(25)	Zusatzblätter:

Bitte verwenden Sie nur dokumentenechtes Schreibmaterial!

1 Synchronisation (36)

Der Autoverkehr einer Kreuzung von zwei Einbahnstraßen wird mit einer einfachen Ampel geregelt (siehe Skizze). Die Ampel hat auf jeder der beiden relevanten Seiten eine rote und eine grüne Lampe, um anzuzeigen, dass die Autos anhalten müssen (rot) bzw. passieren dürfen (grün). Die Autos der beiden Straßen erhalten jeweils abwechselnd eine Grünphase von 45 Sekunden. Am Anfang bzw. Ende jeder Grünphase wird ohne Verzögerung zwischen rotem und grünem Licht (bzw. umgekehrt) umgeschaltet.



Schreiben Sie drei Prozess-Templates zur Simulation des Verkehrs an der Ampel. Der Prozess *Ampel* soll die Ampel simulieren. Das Prozess-Template *Auto-W* simuliert ein von Westen und das Template *Auto-S* ein von Süden über die Kreuzung fahrendes Auto.

Die Mutual Exclusion und das korrekte Passieren der Ampel bei grün soll durch Semaphore geregelt werden. Der zyklische Prozess *Ampel* gibt die Kreuzung abwechselnd für die Autos der beiden Straßen frei und wartet zwischen den Umschaltvorgängen mit dem Funktionsaufruf *warte(45)* für 45 Sekunden.

Das Template *Auto-W* simuliert die Fahrt eines von Westen kommenden Autos, das die Kreuzung passiert. Mehrere von Westen kommende Autos werden durch das Starten mehrerer Kopien von *Auto-W* simuliert. Entsprechendes gilt für das Template *Auto-S*, das ein von Süden kommendes Auto simuliert.

Zum Durchfahren der Kreuzung rufen Auto-Prozesse die Funktion *kreuzung_passieren()* auf. Dabei gilt: Ein Auto darf nur in die Kreuzung einfahren, wenn diese frei ist. Die Synchronisation muss sicherstellen, dass ein Auto nur bei grünem Ampelsignal in die Kreuzung einfährt und dass das Umschalten der Ampel nicht durch ankommende Autos verzögert wird. Schaltet die Ampel um, darf das gerade in der Kreuzung befindliche Auto die Kreu-

zung noch verlassen, dann müssen die Autos der anderen Fahrtrichtung in die Kreuzung einfahren können.

Schreiben Sie Codestücke für *Ampel*, *Auto-W* und *Auto-S* und geben Sie Initialisierungen für die benötigten Semaphore an.

Initialisierungen

Ampel

```
forever{
```



```
}
```

Auto-W

Auto-S

2 Security (21)

2.1 Begriffe (6)

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

- Confidentiality:

- Integrity:

- Availability:

2.2 Security Threats (4)

Füllen Sie folgende Tabelle derart aus, dass Sie jede angegebene Art der Bedrohung einem der Begriffe *Confidentiality (Secrecy)*, *Integrity* und *Availability* zuordnen:

Art der Bedrohung	**bedroht**
Interruption	
Interception	
Modification	
Fabrication	

2.3 Design Principles for Securiy (7)

Nennen Sie *sieben* Designprinzipien für Security!

2.4 Bedrohungen durch Malware (4)

Erläutern Sie die folgenden Bedrohungen: *Logic Bomb*, *Trojan Horse*, *Virus* und *Worm*.

- Logic Bomb:

- Trojan Horse:

- Virus:

- Worm:

3 Deadlock (18)

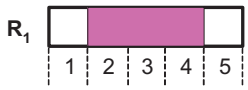
Gegeben sind zwei Prozesse, P_1 und P_2, die jeweils die Ressourcen R_1 und R_2 benötigen. Jede der zwei Ressourcen ist zweimal vorhanden. Benötigt ein Prozess eine vom anderen Prozess belegte Ressource, so wird er auf jeden Fall bis zum Freiwerden der Ressource verzögert.

Der Fortschritt von P_1 und P_2 bei der (quasi)parallelen Abarbeitung kann als Kantenzug zwischen den Punkten *begin* und *end* in der Grafik eingetragen werden. Die Achsenbeschriftung entspricht dabei der Zeilennummer des gerade auszuführenden Befehls.

Unterhalb bzw. links der Diagrammachsen sind Balken vorgesehen, in denen die Anforderungen von Ressourcen für P_1 bzw. P_2 eingetragen werden.

1. Tragen Sie die Anforderungen von Ressourcen für P_1 bzw. P_2 ein. Dabei ist anzunehmen, dass eine Ressource bereits ab Start der Anweisung `get()` als belegt gilt und erst nach Beendigung der Anweisung `free()` als wieder freigegeben gilt:

   ```
   2: get(R1)
   3: ...
   4: free(R1)
   ```

   Sind mehrere Instanzen einer Ressource belegt, so geben Sie die zuletzt belegte Instanz frei.

2. Umranden und schraffieren Sie in der Grafik jene Bereiche, durch die der Kantenzug einer (quasi)parallelen Abarbeitung aufgrund von Ressourcenkonflikten nicht möglich ist.

3. 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.

4. Zeichnen Sie einen Kantenzug für eine gültige, deadlockfreie Abarbeitung von P_1 und P_2 in der Grafik ein.

5. Beschriften Sie einen Punkt im Koordinatensystem (d.h. schreiben Sie den jeweiligen Buchstaben im Kästchen links unterhalb) ...

   ... mit 'A', von welchem aus der Punkt *end* bzw. welcher vom Punkt *begin* erreichbar ist

   ... mit 'B', welcher unweigerlich zu einen Deadlock führt, sofern ein solcher Punkt vorhanden ist

   ... mit 'C', welcher einen nicht erlaubten Zustand darstellt, sofern eine solcher Punkt vorhanden ist

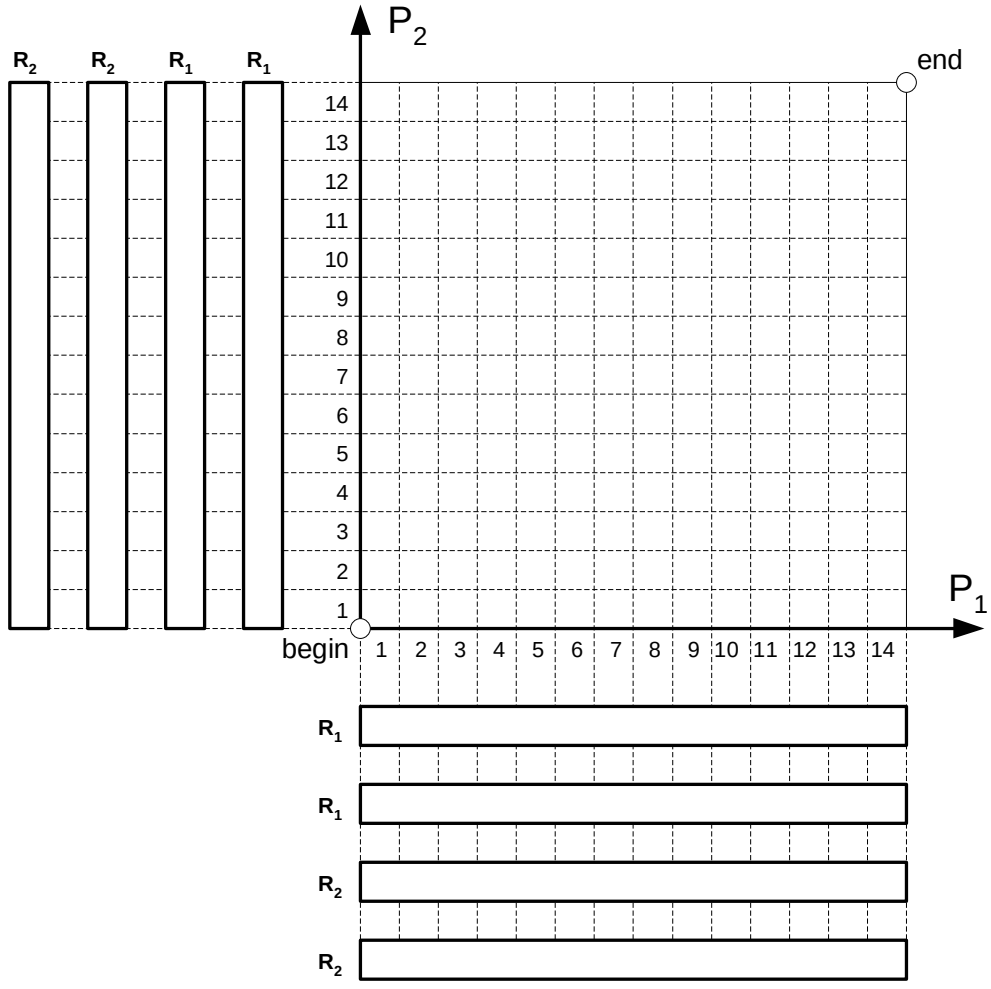
Anmerkung: Achten Sie bitte darauf, dass alle Lösungen gut erkennbar und die Lösungen zu den Teilaufgaben 2 und 3 *deutlich unterscheidbar* sind.

Program P_1:

```
 1: a=1;
 2: b=3;
 3: get(R1);
 4: get(R1);
 5: get(R2);
 6: get(R2);
 7: c=a+b;
 8: free(R2);
 9: a=b*4;
10: free(R1);
11: b=9;
12: free(R2);
13: free(R1);
14: return;
```

Program P_2:

```
 1: a=12;
 2: get(R2);
 3: b=3*a;
 4: c=a+b;
 5: get(R2);
 6: get(R1);
 7: a=b*c;
 8: free(R1);
 9: free(R2);
10: d=a+b;
11: free(R2);
12: a=b*5;
13: a=b+5;
14: return;
```



4 Memory Management (25)

Das betrachtete Speicherverwaltungssystem verwendet zur Adressierung 16-Bit Adressen. Für die angegebenen virtuellen Speicheradressen sind, in Abhängigkeit von der Adressierungsart, die entsprechenden physikalischen Adressen zu ermitteln. Von den angebenen Adressen sind die niederwertigen 8 Bit der Offset der Adresse und die höherwertigen 8 Bit die Segmentnummer bzw. die Seitennummer. Bei Paging sind alle Seiten 256 Bytes (Hexadezimal 0x100) lang. Alle Werte mit führendem **0x** sind als Hexadezimalzahl angegeben, Werte mit abschließendem **b** sind als Binärzahl zu interpretieren. Ergibt sich bei der Umwandlung eine ungültige Adresse, so schreiben Sie bitte **ungültig** in das entsprechende Feld.

Es werden folgende Begriffe (englische Notation) verwendet:

Base	Basisadresse des Segmentes
Entry	Eintrag in der Adressübersetzungstabelle
Frame#	Seitenrahmennummer (im physischen Speicher)
Length	Länge des Segmentes
Page#	Seitennummer (im virtuellen Speicher)

a) Paging — Assoziativer Zugriff (associative mapping)

Adressübersetzungstabelle:

Page#	Frame#
00010110b	0x41
00001000b	0x5A
01011110b	0x13
00100100b	0xAF
00010001b	0x20

Zu berechnende Adressen:

Virtuelle Adresse	Physikalische Adresse
0x5E41	
0x24AB	
0x108A	
0x16B2	

b) Paging — Direkter Zugriff (direct mapping)

Adressübersetzungstabelle:

Entry	Frame#
0	0x24
1	0xB1
2	0x77
3	0x86
4	0xF4
5	0xCC
6	0x01

Zu berechnende Adressen:

Virtuelle Adresse	Physikalische Adresse
0x0311	
0x14AC	
0x0512	
0x08A8	

c) Segmentierung — Direkter Zugriff (direct mapping)

Adressübersetzungstabelle:

Entry	Base	Length
0	0x2840	0x004
1	0x0563	0x0FF
2	0x72AB	0x100
3	0x0300	0x010
4	0x11FD	0x0F0
5	0x4444	0x030
6	0x3112	0x060
7	0x4220	0x010

Zu berechnende Adressen:

Virtuelle Adresse	Physikalische Adresse
0x01F2	
0x0305	
0x0010	
0x04A0	
0x4222	

Bewertung: 1 Pluspunkt pro richtiger Lösung, 1 Punkt Abzug pro falscher Lösung.

d) Verständnisfragen (12)

Kreuzen Sie bitte die richtigen Antworten an. Achtung! Falsche Antworten werden negativ gewertet. (Bewertung: 2 Pluspunkte pro richtiger Antwort, 2 Punkte Abzug pro falscher Antwort.)

- Bei folgender Speichverwaltungstechnik tritt sowohl *interne* als auch *externe* Fragmentierung auf.
  - ◯ fixed partitioning
  - ◯ simple segmentation
  - ◯ virtual memory with combination of paging and segmentation
  - ◯ virtual memory paging
  - ◯ dynamic partitioning
  - ◯ simple paging
- Zu welchen Effekten kann es bei *Segmentierung* kommen?
  - ◯ Internal Fragmentation
  - ◯ External Fragmentation
- In einem fehlerfreien System können zwei oder mehrere virtuelle Adressen auf eine physikalische Adresse abgebildet sein.
  - ◯ richtig
  - ◯ falsch
- Eine virtuelle Adresse verweist immer auf eine Seite auf dem Sekundärspeicher.
  - ◯ richtig
  - ◯ falsch
- Um die externe Fragmentierung zu reduzieren, muss man die *Page Size* vergrößern.
  - ◯ richtig
  - ◯ falsch
- Ein virtueller Speicher kann sowohl mit Paging als auch mit Segmentierung implementiert werden.
  - ◯ richtig
  - ◯ falsch