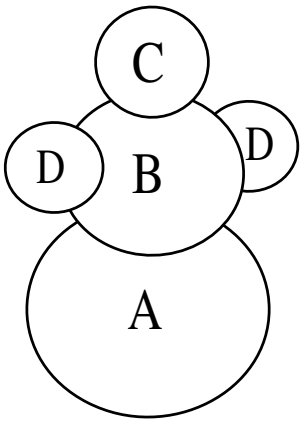
Vorlesungsprüfung aus Systemprogrammierung 2002-01-23

KNr. MNr. Zuname, Vorname

Ges.)(100)	1.)(35)	2.)(20)	3.)(20)	4.)(25)	Zusatzblätter:

Bitte verwenden Sie nur dokumentenechtes Schreibmaterial!

1 Synchronisation (35)



Die Informatik-Studenten feiern das kommende Semesterende, indem sie gemeinsam Schneemänner bauen. Die einzelnen Teile eines Schneemannes werden gemäß obigen Bauplanes mit *A* bis *D* benannt.

Sie wollen Algorithmen zum Bauen des Schneemannes entwickeln:

- Es sind die Funktionen **setze_A()**, **setze_B()**, **setze_C()** und **setze_D()** zu entwickeln, die jeweils das bezeichnete Teil am nächst möglichen Schneemann anbringen. Jeder Student kann *mehrmals* beliebig eine dieser Funktionen verwenden, auch *gleichzeitig* mit anderen Studenten. Die jeweilige Funktion soll solange blockieren, bis ein Schneemann frei ist, an dem das Teil montiert werden kann.

Die erforderten vorhandenen Teile für das Anbringen eines neuen Teiles sind aus folgender Tabelle auszulesen:

Anzubringendes Teil	Benötigte(s) Teil(e)
A	-
B	A
C	A B
D	A B

- Es steht ihnen dazu Funktion **setze(x)** zum Plazieren eines Teiles zur Verfügung, wobei `x` ein Teil aus `{A,B,C,D}` bezeichnet. Die Funktion fügt den Teil automatisch am ersten Schneemann an, der diesen noch nicht vollständig enthält.

a) Die "parallele Klasse" (18)

Die Studenten der "parallelen Klasse" haben folgende Regel festgelegt:

- Es darf an maximal **MAX** Schneemännern gleichzeitig gebaut werden. Ein weiterer Schneemann darf erst angefangen werden, wenn ein anderer fertig geworden ist.

Die Synchronisation ist mit (*möglichst wenig!*) Semaphoren durchzuführen. 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.

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

a1) Initialisierungen (3)

Geben Sie hier die notwendigen Initialisierungen von Semaphoren an.

a2) (5)

Programm zum Platzieren einer Kugel A:
`setze_A()`

```
BEGIN
```



```
END
```

a3) (4)

Program zum Platzieren der Kugel B:
`setze_B()`

```
BEGIN
```



```
END
```

a4) (3)

Program zum Platzieren der Kugel C:
`setze_C()`

```
BEGIN
```



```
END
```

a5) (3)

Programm zum Platzieren eines Armes D:
`setze_D()`

```
BEGIN
```



```
END
```

b) Die "sequentielle Klasse" (17)

Die Studenten der "sequentiellen Klasse" haben folgende Regel festgelegt:

- Es darf an maximal **einem** Schneemann gleichzeitig gebaut werden. Ein weiterer Schneeman darf erst angefangen werden, wenn der vorherige fertig geworden ist.

Die Synchronisation ist mit (*möglichst wenig!*) Sequencern und Eventcountern durchzuführen. Auf Sequencer kann die Funktion **sticket(s)** angewendet werden. Auf Eventcounter können die Funktionen **eawait(e,v)** und **eadvance(e)** angewendet werden.

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

b1) Deklarationen (3)

Geben Sie hier die Deklarationen von Sequenzern/Eventcountern an. Verwenden Sie hierzu die Notation `Sequencer X;` bzw. `Eventcounter Y;`.

b2) (4)

Programm zum Platzieren einer Kugel A:
`setze_A()`

```
BEGIN
```



```
END
```

b3) (3)

Program zum Platzieren der Kugel B:
`setze_B()`

```
BEGIN
```



```
END
```

b4) (4)

Program zum Platzieren der Kugel C:
`setze_C()`

```
BEGIN
```



```
END
```

b5) (3)

Programm zum Platzieren eines Armes D:
`setze_D()`

```
BEGIN
```



```
END
```

2 Speicherverwaltung (20)

a) Kombination aus Segmentierung und Paging (13)

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	0xFFFFFFFF	0x001
0xABC	0xBC050010	0x002
0xB00	0xBC050012	0x0B5
0xEB0	0xA0010010	0x004
0xFFF	0x00000000	0x005

Seitentabelle	
Address	Frame#
0x00000000	0x123456
0x00000001	0x152451
0x00000002	0x152452
...	...
0xA0010010	0x761010
0xA0010011	0x761011
0xA0010012	0x761012
0xA0010013	0x761013
0xA0010014	0x761014
...	...
0xBC050010	0x666012
0xBC050011	0x666011
0xBC050012	0x666010
0xBC050013	0x66600F
...	...
0xBC0500C6	0x666000
0xBC0500C7	0x666001
0xBC0500C8	0x666002
...	...
0xFFFFFFFF	0x000001

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):

Virtuelle Adresse	Physikalische Adresse (zu ermitteln)
0x000001FF	
0xABC00100	
0xFFF002AB	
0x000000EA	
0xEB000000	
0xB000B4FF	
0xABC001FF	
0xB000B500	
0xEB0002AA	

b) Verständnisfragen (7)

Kreuzen Sie bitte die richtigen Antworten an! Achtung! Falsche Antworten werden negativ gewertet!

- Eine Austauschstrategie, die auf alle Seiten des Hauptspeichers angewandt wird, nennt man
  - ◯ lokale Ersetzungsstrategie
  - ◯ globale Ersetzungsstrategie
- 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
- Beim *Fixed Partitioning* können sich keine Partitionen im Hauptspeicher überlappen.
  - ◯ richtig
  - ◯ falsch
- Zu welchen Effekten(mehrere möglich) kann es bei Segmentierung kommen?
  - ◯ Unterschiedliche Segmentlängen
  - ◯ Internal Fragmentation
  - ◯ External Fragmentation
- Bei einem Buddy System sind die anforderbaren Speicherblockgrößen immer größer als die größte bisher angeforderte Speicherblockgröße.
  - ◯ richtig
  - ◯ falsch

3 Security (20)

a) (4)

Was versteht man unter dem Schlagwort *Least Priviledge*?

Geben Sie dazu ein Beispiel an:

b) (6)

Nennen Sie bitte drei Arten von Security Threats und geben Sie jeweils ein Beispiel dazu an:

c) (10)

Ist es möglich mit einem symmetrischen Verschlüsselungsalgorithmus eine fälschungssichere elektronische Unterschrift zu realisieren?

◯ Ja ◯ Nein

Wenn ja, geben Sie bitte an wie so ein Algorithmus funktioniert. Im Fall, dass Sie die vorherige Frage mit nein beantwortet haben, begründen Sie bitte, warum das nicht möglich ist.

Welchen Vorteil hat die symmetrische gegenüber der asymmetrischen Verschlüsselung?

4 Deadlock (25)

Gegeben sind zwei Prozesse P_1, P_2 und die Ressourcen R_1, R_2 und R_3. Die Anzahl der vorhandenen Ressourcen im System können Sie aus dem Ressourcen-Vektor R ablesen. Aus der Claim-Matrix C ist die maximal notwendige Anzahl von Ressource pro Prozess ersichtlich. 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.

$$\mathbf{R} = \begin{pmatrix} 2 \\ 1 \\ 1 \end{pmatrix} \qquad \mathbf{C} = \begin{pmatrix} 2 & 1 \\ 1 & 0 \\ 1 & 1 \end{pmatrix}$$

Das Diagramm in Abbildung 1 zeigt die Ressource-Anforderungen der beiden Prozesse. Die Anforderungen von Prozess P_1 sind entlang der x-Achse, die Anforderungen von Prozess P_2 entlang der y-Achse aufgetragen.

Der Fortschritt von P_1 und P_2 bei der (quasi)parallelen Abarbeitung kann als Kantenzug zwischen den Punkten *0* 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.

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".

3. Zeichnen Sie einen Kantenzug für eine gültige, deadlockfreie Abarbeitung von P_1 und P_2 in der Grafik ein. Dieser Kantenzug soll durch möglichst viele Punkte (S_1-S_6) führen.

Entscheiden Sie für jeden der 6 Punkte (S_1-S_6) im Diagramm, ob die Punkte erreicht werden können, oder nicht und kreuzen Sie dementsprechend in der untenstehenden Tabelle an.

Punkt	erreichbar	nicht erreichbar
S_1	◯	◯
S_2	◯	◯
S_3	◯	◯
S_4	◯	◯
S_5	◯	◯
S_6	◯	◯

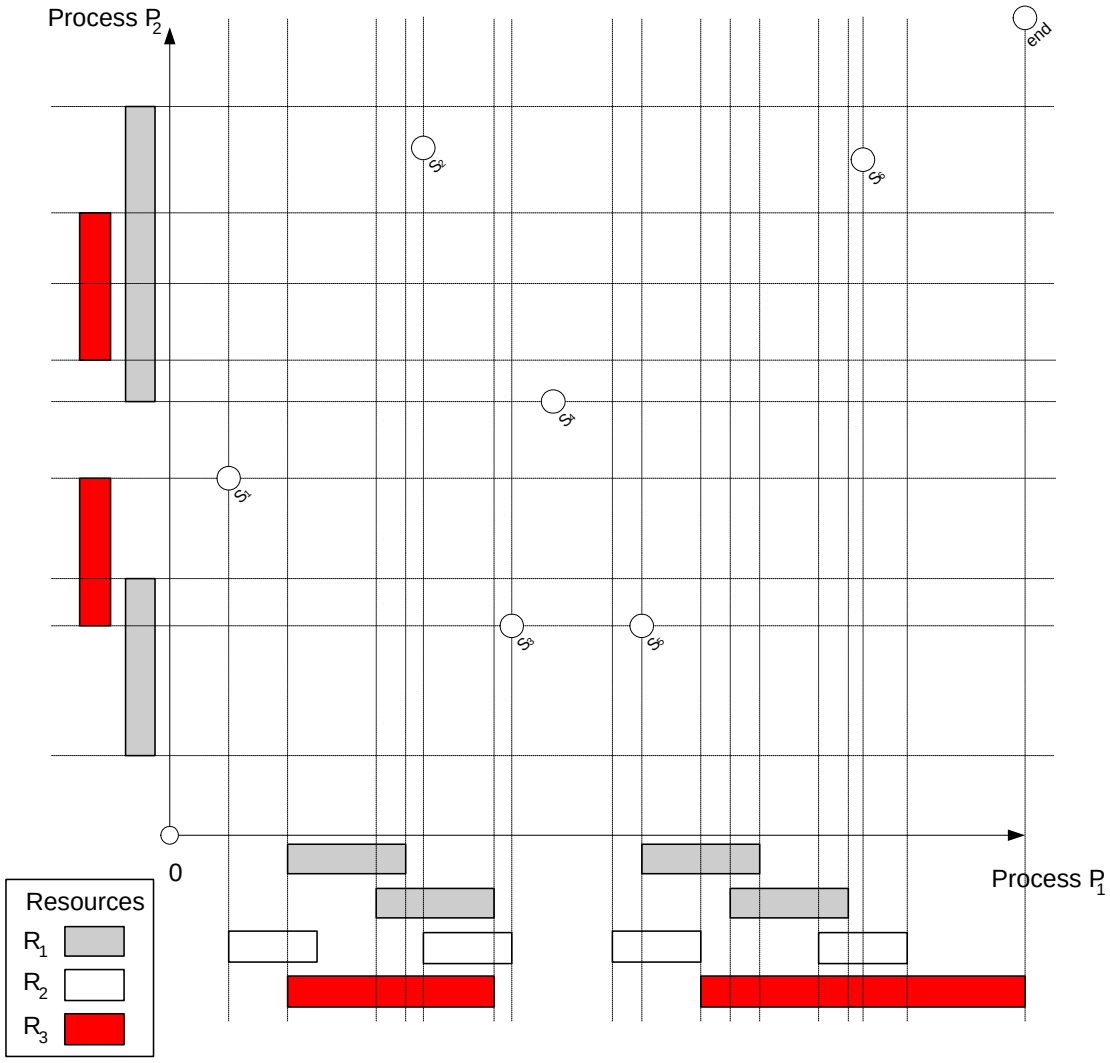


Abbildung 1: Abarbeitungs Diagramm