Hochschule Ravensburg-Weingarten
Schriftliche Prüfung Betriebssysteme

Prof. Dr. M. Zeller

Datum, Zeit	22. Juli 2009, 08:00 – 09:30 Uhr (90 min)
Aufgabenblätter	12 Seiten (einschl. Deckblatt)
erreichbare Punktzahl	67
zugelassene Hilfsmittel	A (s. Prüfungsplan)

Studiengang	Prf. Nr.	Raum
AI	3618	H142

Name: _____ Matrikelnummer: _____

Vorbemerkung Die Klausur ist ziemlich umfangreich. Lassen Sie sich nicht verunsichern, Sie benötigen nicht alle Punkte für die Note 1,0; Sie benötigen weniger als die Hälfte der Punkte für die Note 4,0.

Hinweise:

- Schreiben Sie bitte Name und Matrikelnummer auf jedes Aufgabenblatt.
- Schreiben Sie Ihre Lösung zu den Aufgaben auf den freien Platz, direkt anschließend an die Fragestellungen. Wenn Sie zusätzliche Blätter verwenden, so schreiben Sie bitte Name und Matrikelnummer auf jedes Blatt.
- Schreiben Sie lesbar!

Vom Prüfer auszufüllen:

Aufgabe	1	2	3	4	5	Summe
Max. Punkte	19	20	12	6	10	67
Punkte						

Aufgabe 1 Virtueller Speicher

Ein Betriebssystem verwendet Paging, um für die verschiedenen Prozesse jeweils einen virtuellen Hauptspeicher zu realisieren. Der virtuelle Speicher wird auf 56 MB Hauptspeicher und 8 MB der Festplatte abgebildet (Swap-Space). Die Gesamtlänge einer Adresse beträgt 28 Bit.

Das Betriebssystem verwendet eine dreistufige Seitentabelle. Die Länge der ersten Seitenadresse (PT1) beträgt 6 Bit; die Länge der zweiten Seitenadresse (PT2) beträgt 5 Bit; die Länge der dritte Seitenadresse (PT3) beträgt 8 Bit; die Länge des Offsets beträgt 9 Bit. Ein Prozess belegt für da Text- und das Heap-Segement stets Adressen von 0 beginnend aufwärts, für das Stack-Segment belegt er Adressen von $2^{27} - 1 = 134\,217\,727$ beginnend abwärts.

6 Bit	5 Bit	8 Bit	9 Bit
PT1	PT2	PT3	Offset

Allg. Hinweis: Schreiben Sie bei den folgenden Aufgaben immer den Rechenweg auf, z. B. "Größe des Speicherbereich XY dividiert durch Anzahl Z".

1.1 (11 Punkte)

Ein Prozess belegt folgende Adressbereiche:

Prog. Teil	Adressbereich	Größe in Byte
Text-Segment	0 - 417 000	417 001
Heap-Segment	417 001 - 30 650 070	30 233 070
Stack-Segment	133 951 109 – 134 217 727	266 619

(1 Punkt) Wie viele Einträge hat die Seitentabelle erster Stufe?

Index PT1 6 Bit: $2^6 = 64$

(1 Punkt) Wie viele Einträge hat eine Seitentabelle dritter Stufe?

Index PT2 8 Bit: $2^8 = 256$

(1 Punkt) Wie groß (in Byte) ist ist eine Seite, wie groß ist eine Kachel?

Offset 9 Bit: $2^9 = 512$

(1 Punkt) Wie viele Seiten belegt das Stack-Segment?

Größe Stack-Segment / Größe einer Seite $266\,619/512 = 520,74\ldots \Rightarrow 521$ Seiten

(3 Punkte) Wie viele Seitentabellen zweiter und dritter Stufe werden für das Text- und das Heap- Segment benötigt?

Anzahl der Seiten: $30\,650\,071/512 = 59\,863,418\ldots \Rightarrow 59\,864$. Anzahl Seiten je Seitentabellen dritter Stufe: 256. Anzahl Seitentabellen dritter Stufe $59\,864/256 = 233,843\ldots \Rightarrow 234$

Es werden 234 Seitentabellen dritter Stufe benötigt.

Anzahl Seiten je Seitentabellen zweiter Stufe: 32. Anzahl Seitentabellen zweiter Stufe $234/32 = 7,31\ldots \Rightarrow 8$

Es werden 8 Seitentabellen zweiter Stufe benötigt.

(2 Punkte) Benötigt das System eine (oder mehrere) weitere Seitentabelle(n) zweiter Stufe für das Stack-Segment oder kann der Stack über die vorhandenen Seitentabellen zweiter Stufe adressiert werden (Begründung)?

> Es ist eine weitere Seitentabellen zweiter Stufe notwendig, da der Stack in einem anderen Adressbereich liegt und sich daher (in der gegebenen Konstellation) keine Seitentabelle mit dem Text-/Heap-Segment teilen kann. Die Adressen auf dem Stack fangen mit einer Sequenz von gesetzten Bits an, so dass auch in den Seitentabellen zweiter und dritter Stufe zunächst der oberste Eintrag genutzt wird. Die Adressen des Text-/Heap-Segments wachsen "von unten", so dass auch in den Seitentabellen zunächst die unteren Einträge genutzt werden.

(1 Punkt) Wie viele Kacheln verwaltet das Betriebssystem?

> (Größe Hauptspeicher + Größe Swap) / Größe Kachel: $64 * 2^{20}/2^9 = 64 * 2^{11}$
> Kacheln: 128 K (131 072).

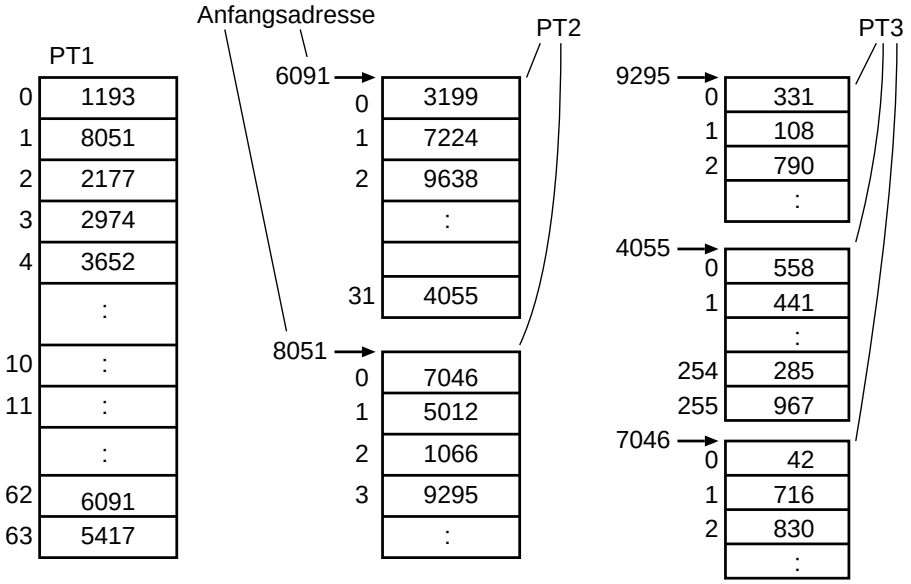
(1 Punkt) Auf welche Größe kann der physische Speicher maximal ausgebaut werden (Hauptspeicher plus Swap-Space)?

> Adress-Breite: 28 Bit d. h. es können maximal 2^{28} Byte (Wörter) adressiert werden (256 MB). Der virtuelle Speicher kann also auf 256 MB ausgebaut werden.

1.2 (8 Punkte)

Im Weiteren soll eine virtuelle Adresse durch vier Dezimalzahlen für PT1, PT2, PT3 und Offset dargestellt werden. Beispiel: Die dezimalen Werte (33, 13, 56, 116) stehen für die virtuelle Adresse 100001 01101 00111000 001110100.

Die folgende Abbildung zeigt einen Ausschnitt aus der Seitentabelle erster Stufe und einige Ausschnitte aus den Seitentabellen zweiter und dritter Stufe. Achtung: In den Seitentabellen dritter Stufe stehen nur die signifikanten Bits, so dass der Offset lediglich angehängt werden muss!



Die physische Adresse soll in Form einer Dezimalzahl dargestellt werden.

Ergänzen Sie die fehlenden Werte in der Tabelle soweit möglich. Wenn Sie einen Wert nicht eintragen können, so begründen Sie dies bitte stichwortartig:

virt. Adresse				phys. Adresse
1	0	2	491	425 451
0	18	244	418	
62	31	254	81	
1	3	0	375	
				366 717
				114 000
				495 367

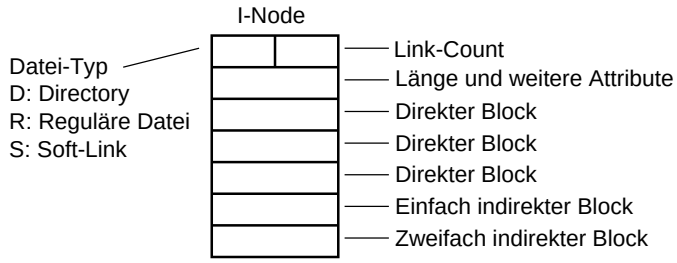
virt. Adresse				phys. Adresse
1	0	2	491	425 451
0	18	244	418	X
62	31	254	81	146 001
1	3	0	375	169 847
1	0	1	125	366 717
		Y		114 000
62	31	255	263	495 367

X: Für die Adresse in PT1[0] (d. h. für 1193) ist keine PT2 angegeben.
Y: Für die Kachel-Nr. 222 gibt es keinen Eintrag in einer PT3.

Aufgabe 2 Datei System mit I-Nodes

Ein Dateisystem verwendet I-Nodes für die Verwaltung von Dateien. Für die Freispeicherverwaltung von I-Nodes und Blöcken verwendet das System je eine Bitmap.

Ein I-Node des Systems besitzt folgendes Format:



Außer den angegebenen Attributen Die Daten sind also über drei direkte Blöcke, einen einfach indirekten Block und einen zweifach indirekten Block erreichbar. Ein Block enthält 4096 Byte, ein Zeiger auf einen Block enthält 16 Byte. I-Nodes enthalten nie selbst Daten einer Datei.

2.1 (3 Punkte)

Wie groß kann eine Datei in diesem Dateisystem maximal sein? Bitte geben Sie alle Rechenschritte an.

2^{12}Byte pro Block/ 16 Byte pro Zeiger = 256 Zeiger pro Block.
Anzahl Blöcke: $1 + 1 + 1 + 256 + 256^2 = 65\,795$ Blöcke = 269 496 320 Byte (ca. 257 MB)

2.2 (2 Punkte)

Wie groß kann das Dateisystem maximal sein (Begründung)?

16 Byte pro Zeiger →Es können max. 2^{128} Blöcke adressiert werden.
2^{128} Blöcke * 2^{12} Byte/Block = 2^{140} Byte.

2.3 (3 Punkte)

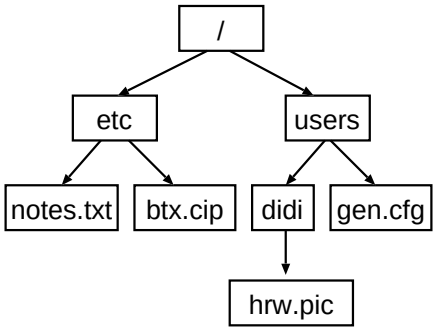
Wie viele Blöcke belegt eine Datei, die 140 MB Daten enthält. Berücksichtigen Sie *nicht* den Platz, der im Datei-Verzeichnis (Directory) belegt wird und ebenfalls *nicht* den Platz, der durch den I-Node belegt wird.

Dateigröße/Blockgröße: $140 * 2^{20}/2^{12} = 140 * 2^8$ (35 840)Blöcke für Daten. Drei direkte Blöcke, 256 Blöcke über den einfach indirekten Block, 35 581 über den zweifach indirekten Block. Ein Block enthält bis zu 256 Zeiger, d. h. es werden $35\,581/512 = 138,98\ldots \Rightarrow 139$ weitere einfach indirekte Blöcke benötigt.

Insgesamt: 35 840 Blöcke für Daten, 139 + 1 einfach indirekte Blöcke, ein zweifach indirekter Block = 35 981 Blöcke.

2.4 (8 Punkte)

Verzeichnisse sind Dateien, die zu jeder verwalteten Datei einen Eintrag enthalten. Ein Eintrag besteht aus dem Namen und einem Verweis auf den I-Node der Datei. Die nebenstehende Skizze zeigt ein Dateisystem; es gibt in diesem Dateisystem keine anderen Dateien.



Die Dateien `/` (Wurzel-Verzeichnis), `etc`, `users` und `didi` sind Verzeichnisse. Die Datei `btx.cip` ist ein Hard-Link, der auf `hrw.pic` verweist. Die Datei `notes.txt` ist ein Soft-Link, der auf `gen.cfg` verweist. Die Datei `gen.cfg` ist 5 KB groß, sie beginnt mit "Do not change ..." und endet mit " ... to refresh.".

Abb. 1 zeigt alle vom Dateisystem verwendeten I-Nodes und Blöcke sowie einen Ausschnitt der Freispeicherverwaltung. Ergänzen Sie die Skizze an den mit ◯ gekennzeichneten Stellen. An Stellen, die keinen definierten Wert besitzen tragen Sie bitte ein Kreuz ein.

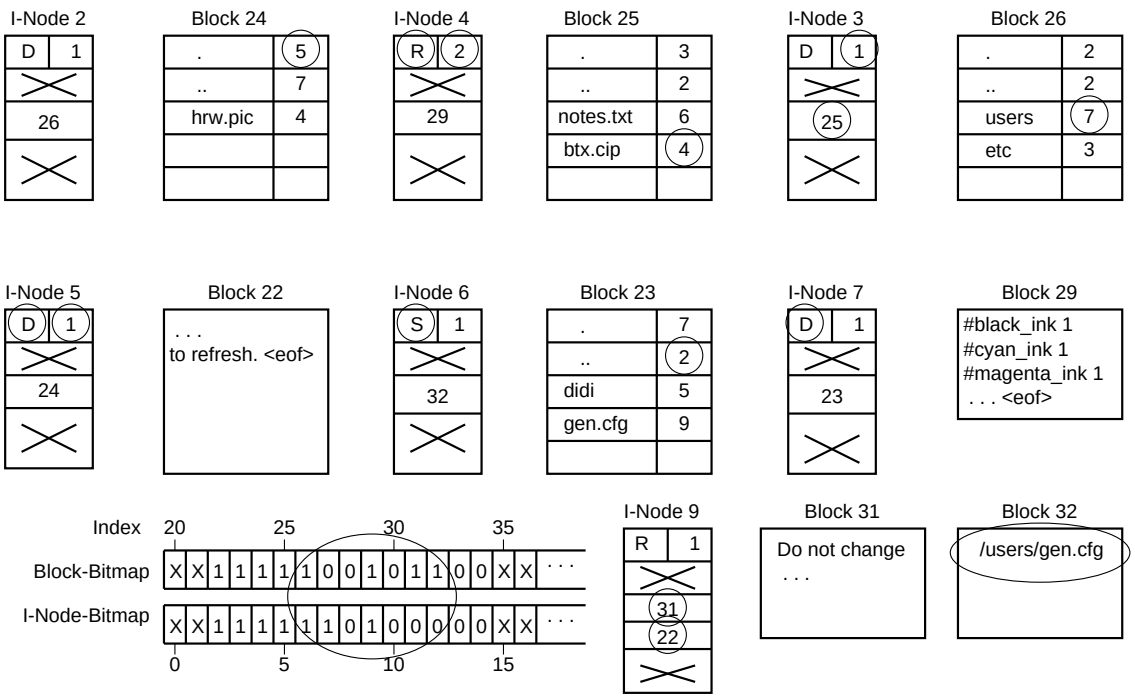


Abbildung 1: Skizze des Dateisystems

Der Eintrag `<EOF>` bedeutet: Gemäß Längen-Eintrag im I-Node endet die Datei an dieser Stelle. Der erste Eintrag der Freispeicher-Bitmaps bezeichnet den Block 0 bzw. den I-Node 0. Der Wert 1 bedeutet, dass der entsprechende Block bzw. I-Node belegt ist, der Wert 0 bedeutet, dass der entsprechende Block bzw. I-Node frei ist.

2.5 (2 Punkte)

Die Datei `gen.cfg` wird um 4 KB vergrößert. Welche Änderungen ergeben sich in dem gegebenen Dateisystem? Wenn das System zusätzliche Blöcke verwendet, so sind dies die Blöcke 33, 34, 35 .... Wenn das System zusätzliche I-Nodes verwendet, so sind dies die I-Nodes 10, 11, 12 ....

Was wird angelegt, welche Werte werden wo eingetragen bzw. verändert?

> Neu: Ein Block (z. B. 33) für die zusätzlichen Daten.
>
> Änderungen: Im I-Node 9 Verweis auf neuen Block eintragen (in diesem Fall 33), Länge der Datei anpassen (+ 4096). In der Block-Bitmap den neu belegten Block als belegt markieren.

2.6 FAT Datei-System (2 Punkte)

Eine Partition der Größe 1 GB soll durch ein FAT-Datei-System verwaltet werden. Die Größe eines Blocks beträgt 2 KB, ein Zeiger auf einen Block besteht aus 3 Byte

Wie groß ist die FAT für diese Dateisystem?

> Anzahl Blöcke d. h. Anzahl Einträge: $1 * 2^{30}/2 * 2^{10} \rightarrow 2^{19}$. Pro Eintrag 3 Byte $3 * 2^{19}$ d. h. 1,5 MB.

Aufgabe 3 Ersetzungsstrategien

Das Betriebssystem eines Rechners verwaltet einen Hauptspeicher mit 5 Kacheln. Auf dem System laufen Prozesse mit insgesamt 7 Seiten. Die Seiten der Prozesse werden gemäß der ersten Zeile der folgenden Tabellen referenziert. Wird eine Seite referenziert, so soll die Wirkung in der zugehörigen Spalte dargestellt werden.

3.1 Clock-Algorithmus (7 Punkte)

Das Betriebssystem verwendet den Clock-Algorithmus. Bsp.: In der Spalte, in der die Seite 0 referenziert wird, ist dargestellt, dass die Seite 0 in den Hauptspeicher eingelagert wurde. Zahl in Klammern gibt den Wert des R-Bits an, der * Bezeichnet den Zeiger des Clock-Algorithmus.

SeitenNr.	0	1	2	3	4	5	6	3	1	7	4
K1	0(1)	0(1)	0(1)	0(1)	0(1)*	5(1)	5(1)	5(1)	5(1)	5(1)*	5(0)
K2	– *	1(1)	1(1)	1(1)	1(1)	1(0)*	6(1)	6(1)	6(1)	6(1)	6(0)
K3	–	– *	2(1)	2(1)	2(1)	2(0)	2(0)*	2(0)*	1(1)	1(1)	1(0)
K4	–	–	– *	3(1)	3(1)	3(0)	3(0)	3(1)	3(1)*	3(0)	4(1)
K5	–	–	–	– *	4(1)	4(0)	4(0)	4(0)	4(0)	7(1)	7(1)*

3.2 Optimale Strategie (5 Punkte)

Wie würde die Ersetzung gemäß der optimale Strategie aussehen? Füllen Sie nur die Felder ohne 'X' aus.

SeitenNr.	0	1	2	3	4	5	6	3	7	1	3	6	4	5	2	0
K1	0	0	0	0	0	5	5	5	7	7	X	X	X	X	X	X
K2	–	1	1	1	1	1	1	1	1	1	X	X	X	X	X	X
K3	–	–	2	2	2	2	6	6	6	6	X	X	X	X	X	X
K4	–	–	–	3	3	3	3	3	3	3	X	X	X	X	X	X
K5	–	–	–	–	4	4	4	4	4	4	X	X	X	X	X	X

Aufgabe 4 Funktionsaufruf (6 Punkte)

Ein Compiler verwendet nur den Stack, um Daten zwischen verschiedenen Funktionen eines Programms auszutauschen. Folgendes Programm ist gegeben:

```
1  int foxi(int *, char[]);
2
3  int fix (char word[], int value) {
4      int result = 1;
5      result = foxi(&value, word);
6      return result * 2;
7  }
8
9  int foxi(int *result, char line[]){
10     *result = line[0];
11     line[0] = 68;
12     return 42;
13 }
14
15 int main (void){
16     short result = 0;
17     char line[] = "Lupo";
18     result = fix(line, 10);
19     printf("\n line: %s\n\n", line);
20     return result;
21 }
```

Ergänzen Sie die Skizze des Stacks auf der nächsten Seite zu folgenden Zeitpunkten:

- t_1 nach Zeile 17 unmittelbar vor dem Funktionsaufruf `fix(...)`
- t_2 nach Zeile 10 unmittelbar vor der Anweisung `line[0] = 68;`
- t_3 in Zeile 6 unmittelbar vor dem Rücksprung zur aufrufenden Funktion

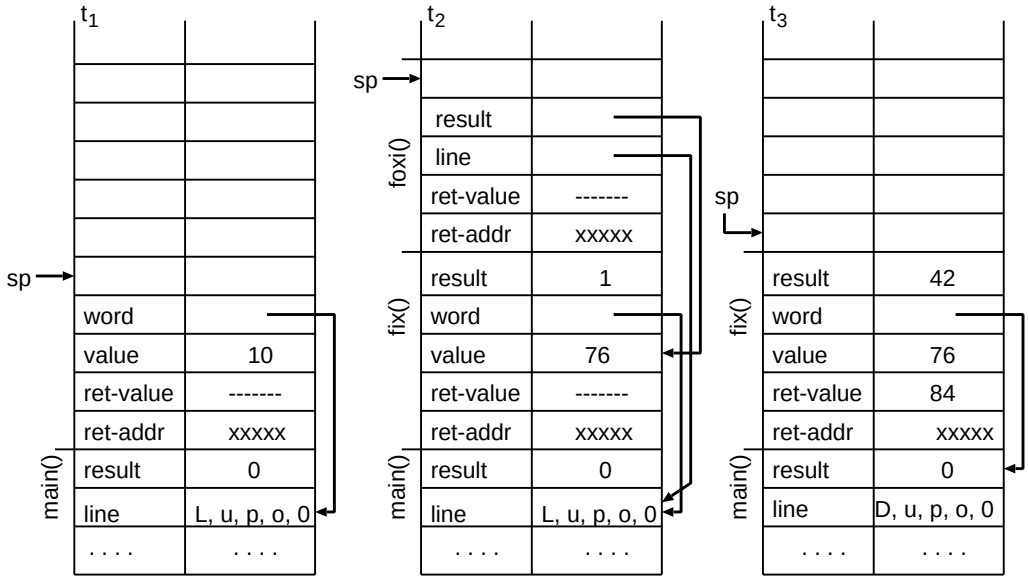
Verwenden Sie dabei folgende Symbole:

$\longrightarrow$	Pointer
– – –	Variable/Speicherstelle angelegt aber nicht initialisiert
xxx	Variable/Speicherstelle besitzt einen unbekannten Wert
sp $\rightarrow$	Stelle, auf die der Stackpointer zeigt

Ausschnitt aus der ASCII-Code-Tabelle:

D	L	o	p	u
68	76	111	112	117

Hinweis: Es sind verschiedene Varianten möglich. Achten Sie aber bitte darauf, dass Ihre Lösung in sich konsistent ist.


t1
sp
word
value
10
ret-value
----
ret-addr
xxxxx
main()
result
0
line
L, u, p, o, 0
. . . .
. . . .
t2
sp
foxi()
result
line
ret-value
----
ret-addr
xxxxx
fix()
result
1
word
value
76
ret-value
----
ret-addr
xxxxx
main()
result
0
line
L, u, p, o, 0
. . . .
. . . .
t3
sp
fix()
result
42
word
value
76
ret-value
84
ret-addr
xxxxx
main()
result
0
line
D, u, p, o, 0
. . . .
. . . .

Aufgabe 5 Synchronisation

Das folgende Petri-Netz zeigt die Synchronisation von drei Prozessen P_AC, P_BD und P_E. Es handelt sich um ein Bedingungs-Ereignis-Netz. Die Transitionen A, C gehören zu Prozess P_AC, die Transitionen B, D gehören zum Prozess P_BD, die Transition E gehört zu Prozess P_E.

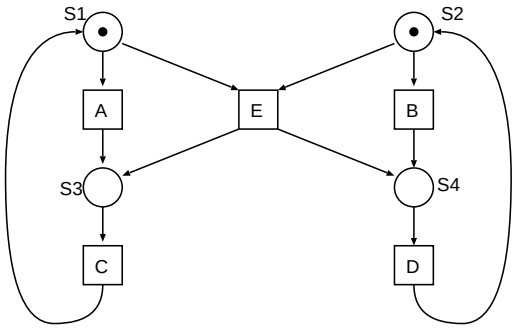


Abbildung 2: Ein paralleles System in Form eines Petri-Netzes

5.1 (2 Punkte)

Welche Stellen müssen Sie als Semaphor realisieren, um die drei Prozesse gemäß dem obigen Petri-Netz zu synchronisieren?

Alle Stellen: S1, S2, S3 und S4

5.2 (3 Punkte)

Geben Sie den Quell-Code für die Prozesse P_AC, P_BD und P_E an. Sie können dazu PseudoPascal verwenden (s. Skript von Frau Keller) oder (Pseudo)Java.

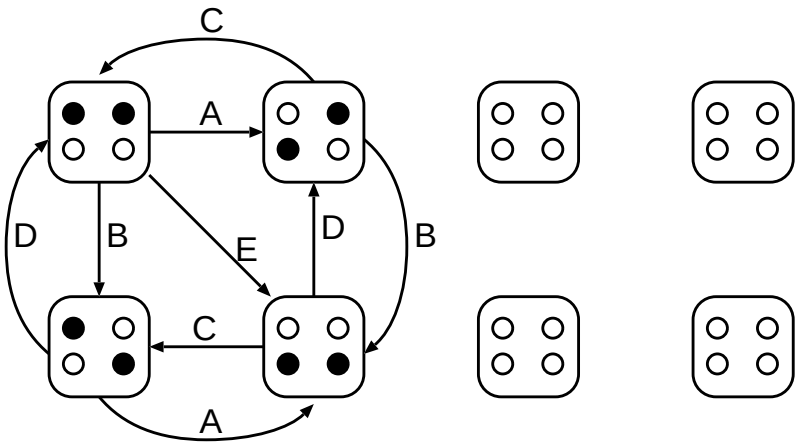
```
Prozess P_AB{
 while(true){
    S1.down()
    A();
    S3.up()
    S3.down();
    C();
    S1.up();
  }
}
```

```
Prozess P_BD{
 while(true){
    S2.down();
    B();
    S4.up();
    S4.down();
    D();
    S2.up();
  }
}
```

```
Prozess P_E{
 while(true){
     S1.down();
     S2.down();
     E();
     S3.up();
     S4.up();
   }
}
```

5.3 (3 Punkte)

Zeichnen Sie den Ereignisgrafen des Petri-Netzes. Sie können die Vorlage unten verwenden oder eine eigene Skizze anfertigen. Geben Sie zu jedem Übergang die Transition, die ihn auslöste, an.



5.4 (2 Punkte)

Kann das System, das in dem oben angegebenen Petri-Netz (s. Abb. 2) dargestellt ist, in einen Deadlock geraten; wenn ja, wie; wenn nein, warum nicht?

Es kann nicht in einen Deadlock geraten; aus jedem Zustand gibt es einen Übergang in einen anderen Zustand.