Hochschule Ravensburg-Weingarten
Schriftliche Prüfung Betriebssysteme

Prof. Dr. M. Zeller

Datum, Zeit	20. Februar 2009, 08:00 – 09:30 Uhr (90 min)
Aufgabenblätter	12 Seiten (einschl. Deckblatt)
erreichbare Punktzahl	69
zugelassene Hilfsmittel	A (s. Prüfungsplan)

Studiengang	Prf. Nr.	Raum
AI	3618	H061

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!

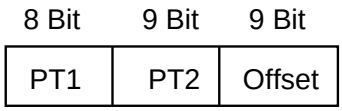
Vom Prüfer auszufüllen:

Aufgabe	1	2	3	4	5	6	Summe
Max. Punkte	17	7	21	8	6	10	69
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 48 MB Hauptspeicher und 8 MB der Festplatte abgebildet (Swap-Space). Die Gesamtlänge einer Adresse beträgt 26 Bit.

Das Betriebssystem verwendet eine zweistufige Seitentabelle. Die Länge der ersten Seitenadresse (PT1) beträgt 8 Bit; die Länge der zweiten Seitenadresse (PT2) beträgt 9 Bit; die Länge des Offsets beträgt 9 Bit.



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

1.1 (8 Punkte)

Ein Prozess belegt folgende Adressbereiche:

Prog. Teil	Adressbereich	Größe in Byte
TextSegment	0 - 390 000	390 001
HeapSegment	390 001 - 22 403 070	22 013 070
Stack Segment	268 291 437 – 268 435 456	144 020

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

Index PT1 8 Bit: $2^8 = 256$

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

Index PT2 9 Bit: $2^9 = 512$

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

Offset 9 Bit: $2^9 = 512$

(1 Punkt) Wie viele Seiten belegt das StackSegment?

Größe Stack-Segment / Größe einer Seite $144\,020/512 = 281,289\ldots \Rightarrow 282$ Seiten

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

Anzahl der Seiten: $22\,403\,071/512 = 43\,755,99\ldots \Rightarrow 43\,756$. Anzahl Seiten je Seitentabellen zweiter Stufe: 512. Anzahl Seitentabellen zweiter Stufe $43\,756/512 = 85,46\ldots \Rightarrow 86$

Es werden 86 Seitentabellen zweiter Stufe benötigt.

(1 Punkt) Wie viele Kacheln verwaltet das Betriebssystem?

Größe Hauptspeicher + Größe Swap / Größe Kachel: $56 * 2^{20}/2^9 = 56 * 2^{11}$ Kacheln: 112 K (114 688).

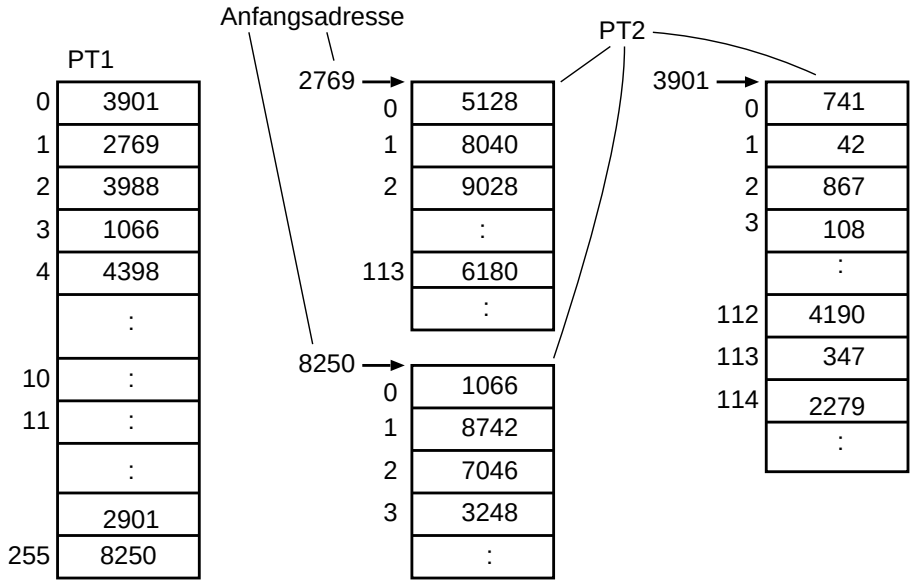
(1 Punkt) Auf welche Größe kann der virtuelle Speicher maximal ausgebaut werden (Hauptspeicher plus Swap-Space)

Adress-Breite: 26 Bit d. h. es können maximal 2^{26} Byte (Wörter) adressiert werden (64 MB). Der virtuelle Speicher kann also auf 64 MB ausgebaut werden.

1.2 (9 Punkte)

Im Weiteren soll eine virtuelle Adresse durch vier Dezimalzahlen für PT1, PT2, und Offset dargestellt werden. Beispiel: Die dezimalen Werte (33, 13, 116) stehen für die virtuelle Adresse 00100001 000001101 001110100.

Die folgende Abbildung zeigt einen Ausschnitt aus der Seitentabelle erster Stufe und einige Ausschnitte aus den Seitentabellen zweiter Stufe. Achtung: In den Seitentabellen zweiter 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	113	47	3 164 207
1	2	491	
0	1	21	
			56 496
			545 870
255	3	176	
			177 989
3	2	501	

X: die Kachelnummer 110 taucht in keiner der angegebenen PT2 auf.
Y: Es ist keine PT2 an Adresse 1066 angegeben.

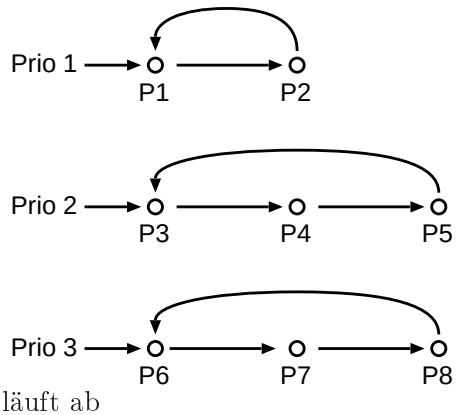
virt. Adresse			phys. Adresse
1	113	47	3 164 207
1	2	491	4 622 827
0	1	21	21 525
	X		56 496
255	0	78	545 870
255	3	176	1 663 152
0	113	325	177 989
3	2	501	Y

Aufgabe 2 Scheduling

Ein Betriebssystem verwendet präemptive Multitasking mit einer Kombination aus Round-Robin und prioritätsbasiertem Scheduling. Es gibt drei Prioritätsstufen: 1, 2 und 3, wobei die Stufe 1 die höchste Priorität darstellt und die Stufe 3 die niedrigste. Der Scheduler wird aktiv, wenn eine Zeitscheibe abläuft oder wenn ein Prozess blockiert wird. Das System verwaltet 8 Prozesse (P1 ... P8). Ein Prozess ist rechnend (RE), einige Prozesse sind bereit (BR) einige sind blockiert (BL).

2.1 (7 Punkte)

Die folgende Tabelle soll die Zustände der Prozesse zu verschiedenen Zeitpunkten darstellen. Der Zustand unmittelbar vor dem Zeitpunkt t_0 ist gegeben. Es treten nun der Reihe nach Ereignisse auf. Mit dem Ausdruck "Prozess P7 wird deblockiert" ist gemeint, dass die Ursache für die Blockade aufgehoben ist. Z. B. weil eine Aus- oder Eingabe des Prozess beendet wurde. Die Zeitpunkte t_{-2} und t_{-1} fassen jeweils zwei Ereignisse zusammen.



Ergänzen Sie die Tabelle gemäß den auftretenden Ereignissen.

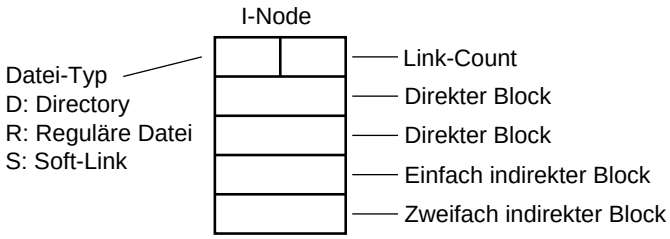
t_0: Ein Zeitscheibe für das präemptive Multitasking läuft ab
t_1: Prozess P1 wird blockiert
t_2: Prozess P2 wird blockiert
t_3: Prozess P5 wird blockiert
t_4: Prozess P4 wird blockiert
t_5: Prozess P5 wird deblockiert
t_6: Ein Zeitscheibe für das präemptive Multitasking läuft ab
t_7: Ein Zeitscheibe für das präemptive Multitasking läuft ab

	t_{-2}	t_{-1}	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
P1	BL	BL	BL	BR	RE	BL	BL	BL	BL	BL	BL	BL
P2	BL	BL	RE	RE	BR	RE	BL	BL	BL	BL	BL	BL
P3	BL	BL	BL	BL	BL	BL	BL	BL	BL	BL	BL	BL
P4	BR	RE	BR	BR	BR	BR	BR	RE	BL	BL	BL	BL
P5	BL	BL	BL	BR	BR	BR	RE	BL	BL	BR	RE	RE
P6	BR	BR	BR	BR	BR	BR	BR	BR	RE	RE	BR	BR
P7	RE	BR	BR	BR	BR	BR	BR	BR	BR	BR	BR	BR
P8	BL	BL	BL	BL	BL	BL	BL	BL	BL	BL	BL	BL

Aufgabe 3 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:



Die Daten sind also über zwei direkte Blöcke, einen einfach indirekten Block und einen zweifach indirekten Block erreichbar. Ein Block enthält 4096 Byte, ein Zeiger auf einen Block enthält 8 Byte. I-Nodes enthalten nie selbst Daten einer Datei.

3.1 (3 Punkte)

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

2^{12}Byte pro Block/ 8 Byte pro Zeiger = 512 Zeiger pro Block.
Anzahl Blöcke: $1 + 1 + 512 + 512^2 = 262\,658$ Blöcke = 1 075 847 168 Byte (ca. 1 GB)

3.2 (2 Punkte)

Wie groß kann das Dateisystem maximal sein (Begründung)?

8 Byte pro Zeiger →Es können max. 2^{64} Blöcke adressiert werden.
2^{64} Blöcke * 2^{12} Byte/Block = 2^{76} Byte.

3.3 (3 Punkte)

Wie viele Blöcke belegt eine Datei, die 100 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: $100 * 2^{20}/2^{12} = 100 * 2^8$ (25 600) Blöcke für Daten. Zwei direkte Blöcke, 512 Blöcke über den einfach indirekten Block, 25 086 über den zweifach indirekten Block. Ein Block enthält bis zu 512 Zeiger, d. h. es werden $25\,086/512 = 48,99\ldots \Rightarrow 49$ weitere einfach indirekte Blöcke benötigt.

Insgesamt: 25 600 Blöcke für Daten, 49 + 1 einfach indirekte Blöcke, ein zweifach indirekter Block = 25 651 Blöcke.

3.4 (7 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. Folgende Skizze zeigt ein Dateisystem; es gibt in diesem Dateisystem keine anderen Dateien.

Die Dateien `/` (Wurzel-Verzeichnis), `etc` und `users` sind Verzeichnisse. Die Datei `hbc.conf` ist ein Hard-Link, der auf `prog.conf` verweist. Die Datei `gen.conf` ist ein Soft-Link, der auf `prog.conf` verweist. Die Datei `text.bla` ist 5 KB groß, sie beginnt mit "Der Bundesrat ..." und endet mit " ... CO2-Emissionen.".

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.

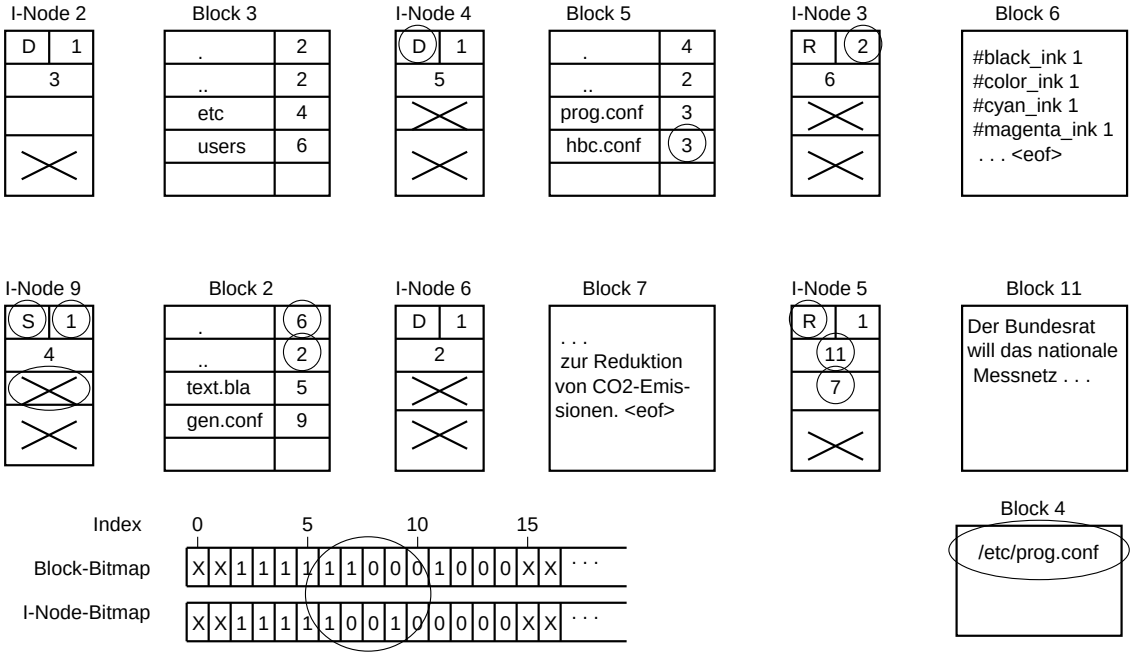


Abbildung 1: Skizze des Dateisystems

Der Eintrag `<EOF>` bedeutet: Gemäß Längen-Eintrag im I-Node endet die Datei an dieser Stelle. Der erste Eintrag der Freispeicher-Bitmap 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.

3.5 (4 Punkte)

Die Datei `text.bla` wird um 4 KB vergrößert. Welche Änderungen ergeben sich in dem gegebenen Dateisystem? Wenn das System zusätzliche Blöcke verwenden, so sind dies Block 12, 13, 14 ....Wenn das System zusätzliche I-Nodes verwenden, so sind dies I-Node 10, 11, 12 ....

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

Neu: Ein Block (z. B. 16) für die zusätzlichen Daten und ein indirekter Block (z. B. 15), der einen Verweis auf den Datenblock (in diesem Fall 16) enthält.

Änderungen: Im I-Node 5 Verweis auf indirekten Block eintragen (in diesem Fall 15), Länge der Datei anpassen (+ 4096). In der Block-Bitmap die beiden neu belegten Blöcke als belegt markieren (in diesem Fall Stelle 15 und 16 auf 1 setzen).

3.6 FAT Datei-System (2 Punkte)

Eine Partition der Größe 2 GB soll durch ein FAT-Datei-System verwaltet werden. Die Größe eines Blocks beträgt 4 KB, ein Zeiger auf einen Block besteht aus 4 Byte

Wie groß ist die FAT für diese Dateisystem?

Anzahl Blöcke d. h. Anzahl Einträge: $2 * 2^{30}/4 * 2^{10} \rightarrow 2^{19}$. Pro Eintrag 4 Byte $4 * 2^{19}$ d. h. 2 MB.

Aufgabe 4 Ersetzungsstrategien (8 Punkte)

Das Betriebssystem eines Rechners verwaltet einen Hauptspeicher mit 4 Kacheln. Das Betriebssystem verwendet den Aging-Algorithmus mit einem 3-Bit Zähler. Auf dem System läuft ein Prozess mit insgesamt 6 Seiten. Die Seiten der Prozesse werden gemäß der ersten Zeile der folgenden Tabellen referenziert. Neben der Seitennummer steht in Klammern der Zählerstand der Seite. Der Eintrag ZS bedeutet, dass jeder Zählerstand durch den Aging-Algorithmus halbiert wird. In der Spalte unter einem Zugriff sollen die Folgen dieses Zugriffs dagestellt werden. So wird z. B. in der ersten Spalte durch den Zugriff auf Seite 1 diese Seite in Kachel 3 eingelagert. Die Seiten 0, 4, und 3 wurden schon vorher eingelagert.

Ergänzen Sie die Tabelle.

	Seitenreferenz								
	1	ZS	2	0	ZS	3	5	2	4
K0	0(2)	0(1)	0(1)	0(5)	0(2)	0(2)	0(2)	0(2)	4(4)
K1	4(3)	4(1)	4(1)	4(1)	4(0)	3(4)	3(4)	3(4)	3(4)
K2	3(1)	3(0)	2(4)	2(4)	2(2)	2(2)	2(2)	2(6)	2(6)
K3	1(4)	1(2)	1(2)	1(2)	1(1)	1(1)	5(4)	5(4)	5(4)

Aufgabe 5 Funktionsaufruf (6 Punkte)

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

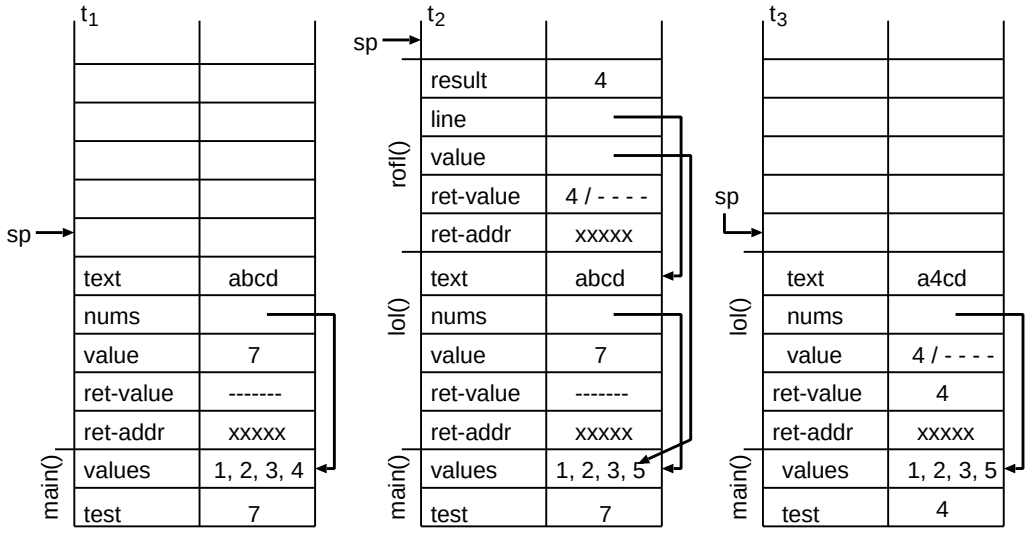
```
1  int rofl (int *value, char line[]) {
2      int result = *value;
3      if (line[0] < 'x'){
4          *value = result + 1;
5      }
6      return result;
7  }
8
9  int lol (int value, int nums[]){
10     char text[] = "abcd";
11     value = rofl(&nums[3], text);
12     text[1] = value + '0';
13     return value;
14 }
15
16 int main (void){
17     int values[ ] = {1, 2, 3, 4};
18     int test = 7;
19     test = lol(test, values) ;
20     return test;
21 }
```

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

- t_1 Zeile 10 unmittelbar nach der Zuweisung zu `text`
- t_2 Zeile 6 unmittelbar vor der Anweisung `return`
- t_3 Zeile 13 unmittelbar vor der Anweisung `return`

Verwenden Sie dabei folgende Symbole:

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


t1
sp
text
abcd
nums
value
7
ret-value
----
ret-addr
xxxxx
values
1, 2, 3, 4
test
7
main()
t2
sp
result
4
line
value
4 / - - - -
ret-addr
xxxxx
text
abcd
nums
value
7
ret-value
----
ret-addr
xxxxx
values
1, 2, 3, 5
test
7
rofl()
lol()
main()
t3
sp
text
a4cd
nums
value
4 / - - - -
ret-value
4
ret-addr
xxxxx
values
1, 2, 3, 5
test
4
lol()
main()

Aufgabe 6 Synchronisation

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

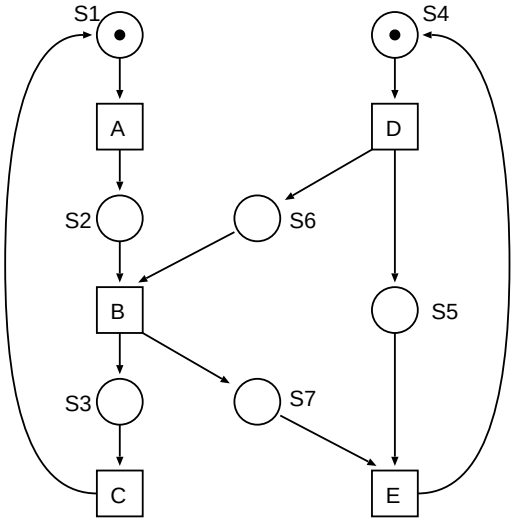


Abbildung 2: Ein paralleles System in Form eines Petri-Netzes

6.1 (2 Punkte)

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

S1, S3, S6 und S7

6.2 (3 Punkte)

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

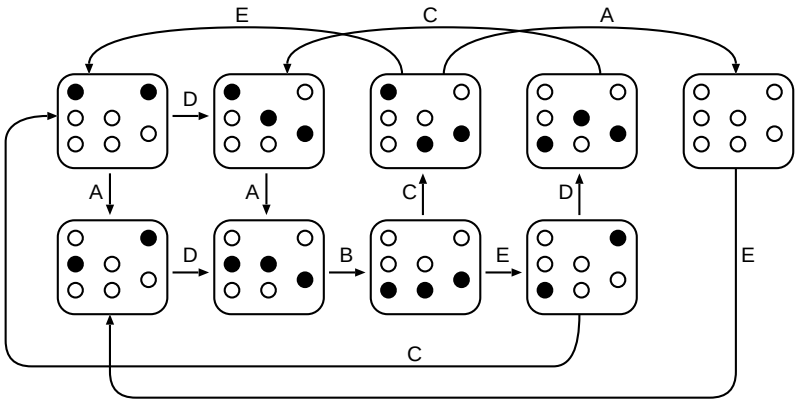
```
Prozess P_AB{
 while(true){
     S1.down()
     A();
     S6.down()
     B();
     S3.up();
     S7.up();
   }
}
```

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

```
Prozess P_DE{
 while(true){
     D():
     S6.up();
     S7.down();
     E();
   }
}
```

6.3 (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.



6.4 (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.