Hochschule Ravensburg-Weingarten
Schriftliche Prüfung Betriebssysteme

Prof. Dr. M. Zeller

Datum, Zeit	6. Februar 2007, 14:00 – 15:30 Uhr (90 min)
Aufgabenblätter	10 Seiten (einschl. Deckblatt)
erreichbare Punktzahl	64
zugelassene Hilfsmittel	A (s. Prüfungsplan)

Studiengang	Prf. Nr.	Raum
AI	1825	H061
AI	3618	H061
WI	4021	H061

Name: _____ Matrikelnummer: _____

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!

Falls Sie es wünschen, dass Ihr Prüfungsergebnis auf einer Liste mit Matrikelnummern und Zensuren ausgehängt bzw. per Internet veröffentlicht wird, unterschreiben Sie bitte folgende Erklärung.

Ich bin damit einverstanden, dass mein Klausurergebnis auf diese Weise veröffentlicht wird.

Unterschrift: _____

Bitte haben Sie dafür Verständnis, dass aus Gründen das Datenschutzes keine telefonischen Auskünfte gegeben werden können.

Vom Prüfer auszufüllen:

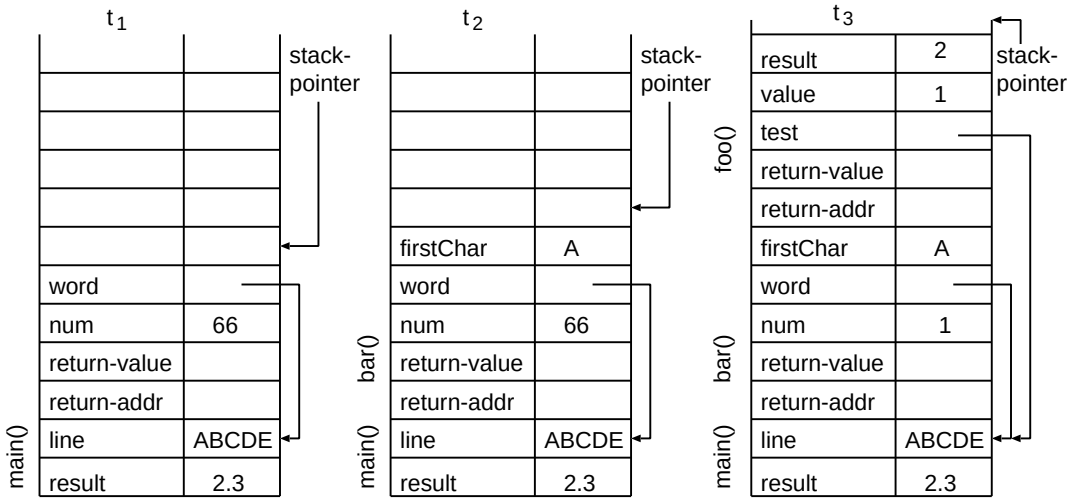
Aufgabe	1	2	3	4	5	6	Summe
Max. Punkte	4	6	9	17	12	16	64
Punkte							

Aufgabe 1 Funktionsaufruf (4 Punkte)

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

```
1  #include <stdio.h>
2
3  int foo (int value, char test[ ]) {
4      int result = 0;
5      if (test[0] == 'A'){
6          result = value + 1;
7      }
8      return result;
9  }
10
11 float bar (char word[ ] , int num){
12     char firstChar = word[0] ;
13     num = num - firstChar;
14     num = foo (num, word);
15     return num;
16 }
17
18 int main (void){
19     char line[ ] = "ABCDE";
20     float result = 2.3;
21     result = bar(line, 66) ;
22     return result;
23 }
```

Ergänzen Sie den Stack zu folgenden Zeitpunkten: t_1 Zeile 21 unmittelbar vor dem Aufruf von `bar()`, t_2 nach der Anweisung in Zeile 12, t_3 nach der Anweisung in Zeile 6.



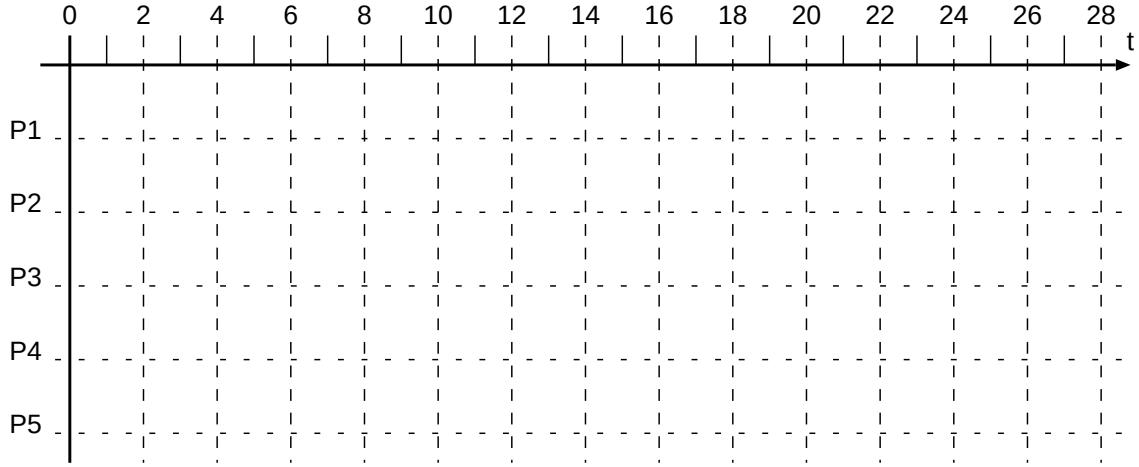
Aufgabe 2 Scheduling

2.1 (4 Punkte)

Ein Betriebssystem bearbeitet fünf Prozesse. Das System verwendet prioritätsbasiertes Scheduling. Wenn zwei Prozesse mit gleicher Priorität bereit sind, darf der Prozess rechnen, der schon mehr Rechenzeit verbraucht hat. Der Scheduler wird immer dann aktiv, wenn ein Prozess gestartet oder beendet wird. Je größer der Zahlenwert der Priorität, desto höher ist die Priorität.

ProzessNr.	Startzeit	Dauer	Priorität
1	0	8	1
2	5	4	2
3	7	3	4
4	10	7	2
5	14	6	3

Tragen Sie die Wartezeiten und die Laufzeiten der Prozesse in das Diagramm ein. Verwenden Sie für Wartezeiten einen Strich: — und für Laufzeiten einen Balken: ▭ oder verwenden Sie unterschiedliche Farben.



2.2 (2 Punkte)

Verwendet man diese Art (s. o.) des Scheduling für interaktive Systeme? (Begründung!)

Nein. Interaktive Systeme sollen dem Nutzer akzeptable Reaktionszeiten garantieren. Sie verwenden daher i. Allg. Round-Robin mit relativ kurzen Zeitscheiben ggf. in Kombination mit Prioritäten. Das obige Verfahren kann zu langen Wartezeiten führen.

Aufgabe 3 Ersetzungsstrategien (9 Punkte)

Das Betriebssystem eines Rechners verwaltet einen Hauptspeicher mit vier Kacheln. Ein Prozess mit sechs Seiten läuft auf dem Rechner. Die Seiten des Prozess werden gemäß der ersten Zeile der folgenden Tabellen referenziert. Als Ersetzungsstrategie kommt LRU und die optimale Strategie zum Einsatz. Ergänzen Sie die Einträge in den Tabellen.

Auslagerung nach LRU

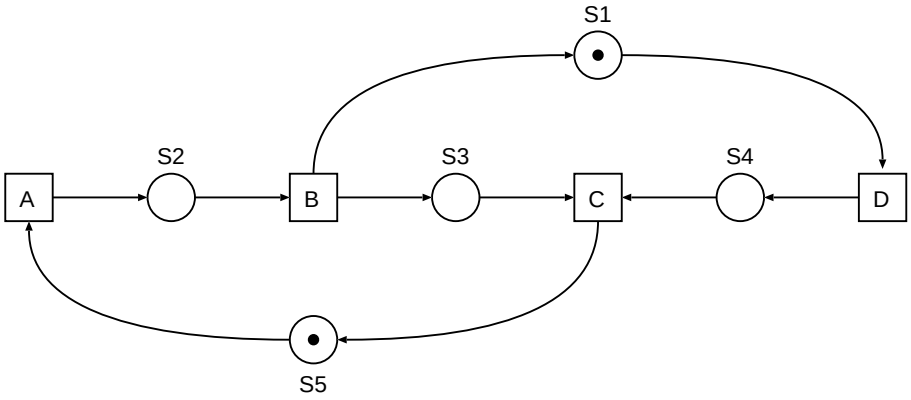
SeitenNr.	0	1	3	2	5	1	4	1	5	2	3	4	5
K1	0	0	0	0	5	5	5	5	5	5	5	5	5
K2	–	1	1	1	1	1	1	1	1	1	1	4	4
K3	–	–	3	3	3	3	4	4	4	4	3	3	3
K4	–	–	–	2	2	2	2	2	2	2	2	2	2

Auslagerung nach optimaler Strategie

SeitenNr.	0	1	3	2	5	1	4	1	5	2	3	4	5
K1	0	0	0	0	5	5	5	5	5	x	x	x	x
K2	–	1	1	1	1	1	1	1	1	x	x	x	x
K3	–	–	3	3	3	3	4	4	4	x	x	x	x
K4	–	–	–	2	2	2	2	2	2	x	x	x	x

Aufgabe 4 Synchronisation

Das folgende PetriNetze zeigt die Synchronisation von 3 Prozessen *P_AB*, *P_C*, *P_D*. Es handelt sich um ein BedingungsEreignisNetz. Die Transitionen *A* und *B* gehören zu Prozess *P_AB*, die Transition *C* gehört zu Prozess *P_C*, die Transition *D* gehört zu Prozess *P_D*.



4.1 (2 Punkte)

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

S1, S3, S4, S5

4.2 (4 Punkte)

Geben Sie den QuellCode für die Prozesse *P_AB*, *P_C* und *P_D* an. Sie können dazu PseudoPascal verwenden (s. Skript von Frau Keller) oder (Pseudo)Java.

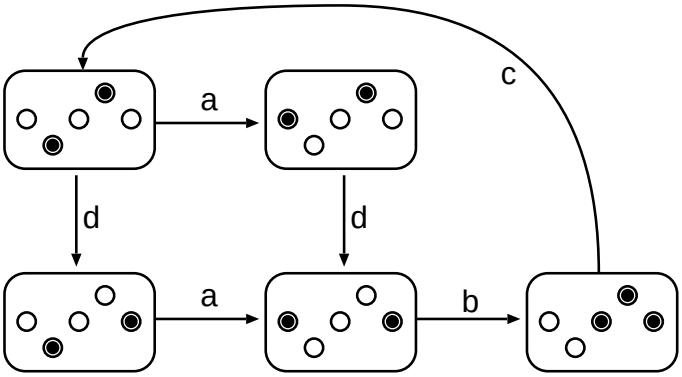
```
Prozess P_AB{
 while(true){
    S5.down();
    A();
    B();
    S1.up();
    S3.up();
  }
}
```

```
Prozess P_C{
 while(true){
    S3.down();
    S4.down();
    C();
    S5.up();
  }
}
```

```
Prozess P_D{
 while(true){
    S1.down();
    D();
    S4.up();
  }
}
```

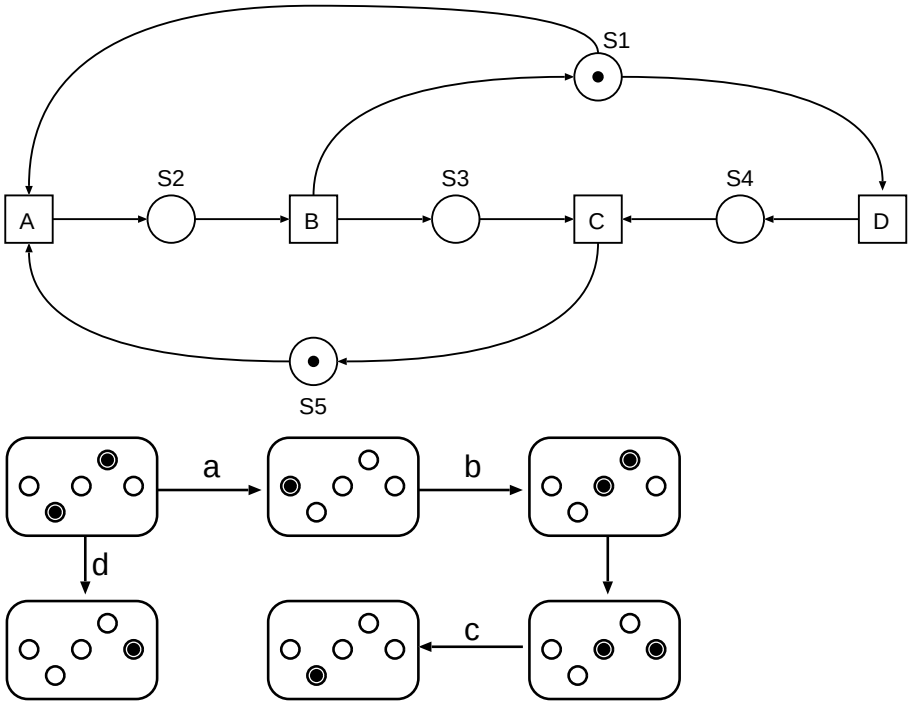
4.3 (3 Punkte)

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.



4.4 (3 Punkte)

Das PetriNetz wird leicht modifiziert (s. u.). Zeichnen Sie den Ereignisgrafen des modifizierten PetriNetzes. Sie können die Vorlage unten verwenden oder eine eigene Skizze anfertigen. Geben Sie zu jedem Übergang die Transition, die ihn auslöste, an.



4.5 (3 Punkte)

Auf einem Rechner laufen zwei Prozesse P_1 und P_2. Beide verwenden zeitweilig die beiden Ressourcen R_1 und R_2. Unter welchen Umständen kann es zu einem Deadlock kommen;

d. h. keiner der beiden Prozesse kann weiterlaufen? Hinweis: Evtl. hilft es, wenn Sie sich erst die nächste Frage anschauen.

Folgende Bedingungen müssen gelten:

Exclusive Nutzung der Ressourcen, Nachfordern von Ressourcen, zyklisches Warten.

4.6 (2 Punkte)

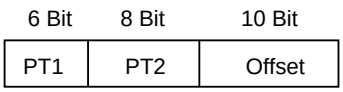
Nehmen Sie an, die eine Ressource aus Aufgabe 4.5 sei ein CDLaufwerk, die andere Ressource der Hauptspeicher (Anforderung mit `malloc()`). Kann es unter diesen Umständen zu einem Deadlock kommen? (Begründung!)

Die dritte Bedingung "zyklisches Warten"ist nicht erfüllt. Zumindest die Anforderung von Hauptspeicher mit `malloc()` wartet nicht sondern kehrt stets sofort zurück. Es kann nur dann zu einem Deadlock kommen, wenn mindestens einer der Prozesse selbst einen wartendene Zugriff auf den Hauptspeicher implementiert; z. B. indem er den Aufruf an `malloc()` in ein Schleife wiederholt, so lange der Rückgabewert `NULL` ist.

Aufgabe 5 Virtueller Speicher

Ein Betriebssystem verwendet Paging, um für die verschiedenen Prozesse jeweils einen virtuellen Hauptspeicher zu realisieren. Der virtuelle Speicher wird auf 8 MB Hauptspeicher und 4 MB der Festplatte abgebildet (Swap-Space). Die Gesamtlänge einer Adresse beträgt 24 Bit.

Das Betriebssystem verwendet eine zweistufige Seitentabelle. Die Länge der ersten Seitenadresse (PT1) beträgt 6 Bit; die Länge der zweiten Seitenadresse (PT2) beträgt 8 Bit; die Länge des Offsets beträgt 10 Bit.



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

5.1 (8 Punkte)

Ein Prozess belegt folgende Adressbereiche:

Prog. Teil	Adressbereich		Größe in Byte
TextSegment	0	203 000	203 001
HeapSegment	203 001	864 070	661 010
Stack Segment	16 777 215	16 590 998	186 218

(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 zweiter Stufe?

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

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

Offset 10 Bit: $2^{10} = 1024$

(1 Punkt) Wie viele Seiten belegt das StackSegment?

Größe Stack-Segment / Größe einer Seite $186\,218/1024 = 181,85 \Rightarrow 182$ Seiten

(1 Punkt) Wie viele Seitentabellen zweiter Stufe werden für das StackSegment benötigt?

Anzahl Seiten / Anzahl Einträge in einer Tabelle 2. Stufe $182/256 = 0,7\ldots$ Es wird *eine* Tabelle 2. Stufe benötigt.

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

Anzahl Seiten: $864\,071/1024 = 843,819\ldots \Rightarrow 844$ Seiten. Anzahl Seiten / Anzahl Einträge in einer Tabelle 2. Stufe $844/256 = 3,29\ldots$ 4 Tabellen 2. Stufe

(1 Punkt) Wie viele Kacheln verwaltet das Betriebssystem?

Größe Hauptspeicher / Größe Kachel: $8 * 2^{30}/2^{10} = 8192$ Kacheln

5.2 (4 Punkte)

Ein Betriebssystem verwendet Segmentierung, um für die verschiedenen Prozesse einen virtuellen Hauptspeicher zu realisieren. Die Gesamtlänge einer Adresse beträgt 20 Bit. Die SegmentNr. ist 4 Bit breit, der Offset verwendet 16 Bit. Alle Größenangaben sind in Byte gegeben.

Es befinden sich zwei Prozesse im System: Prozess 1 verwendet 2 Segmente, Prozess 2 verwendet 3 Segmente.

Prozess 1

SegmentNr.	BasisAdresse	Größe
0	327 680	51 360
1	196 608	46 908

Prozess 2

SegmentNr.	BasisAdresse	Größe
0	393 216	26 112
1	420 896	46 908
2	245 880	38 766

Die folgende Tabelle soll die Zuordnung von logischen und physischen Adressen zeigen. Berechnen Sie die fehlenden Größen soweit möglich.

ProzessNr.	SegmentNr.	Offset	physische Adresse
1	1	21 084	217 692
1	0	798	328 478
1	1	57 410	Fehler
2	0	1 556	394 772
Fehler			291 036
2	2	15 904	261 784

5.3 (4 Punkte)

Ein weiteres Programm mit zunächst einem Segment soll gestartet werden. Das Segment hat die Größe 31 892 Byte. Der Lader des Betriebssystem muss also eine neue Basis-Adresse für diese Segment vergeben. Welche der folgenden Adressen kommen dafür in Frage bzw. nicht in Frage?

Adresse	geeignet	nicht geeignet	Begründung
288 538	X		nicht belegt, keine Überlappung
409 274		X	Überlappung mit Segment 1 von Prozess 2
102 318	X		nicht belegt, keine Überlappung
172 976		X	Überlappung mit Segment 1 von Prozess 1

Aufgabe 6 Datei System

Diese Aufgabe wird noch nachgeliefert ...