Prof. Dr. Gerhard Schneider
Dr. Klaus Rechert
Thomas Liebetraut

Institut für Informatik
Technische Fakultät
Universität Freiburg

Freiburg, 16. März 2015

Systeme I - WS 14/15
Abschlussklausur

Herzlich willkommen zur Abschlussklausur Systeme 1!

Die Klausur besteht aus sechs Aufgaben. Bitte überprüfen Sie ihr Exemplar auf Vollständigkeit!

Die Bearbeitungszeit beträgt 90 Minuten.

Bitte lesen Sie alle Fragen genau durch, bevor Sie diese beantworten! Wir gehen davon aus, dass wenn mehrere Aspekte gefragt werden, diese auch beantwortet werden sollten. Dies trifft auch zu, wenn mehrere Fragen zusammen gestellt werden, aber nicht einzeln mit a) - f) gekennzeichnet wurden.

Die zu erreichende Punktzahl ist für jede Frage angegeben. Bitte beantworten Sie alle Fragen nur auf den gehefteten Blättern der Klausur. Verwenden Sie ggf. die Rückseite der Blätter.

Soll eine Lösung nicht berücksichtigt werden, so streichen Sie diese deutlich durch. Ansonsten wird bei der Korrektur immer die erste Lösung berücksichtigt.

Für die Lösung ist immer auch der Rechenweg mit anzugeben!

Hilfsmittel:
Es sind **keine** Hilfsmittel zugelassen! Mobiltelefone sind während der Klausur **ausgeschaltet** wegzulegen.

Viel Erfolg!

Name, Vorname: .....

Matrikelnummer: .....

Unterschrift: .....

Aufgabe	1	2	3	4	5	6	Summe
Punkte	8	8	10	11	6	6	49
Erreicht							

Name: Matrikelnummer:

Aufgabe 1. 1+2+3+2 P.

a) In welchen Situationen kann ein Deadlock auftreten? Wie ist ein Deadlock definiert?

b) Wie kann das Problem von Deadlocks erkannt werden? Skizzieren Sie das entsprechende Verfahren bzw. den Algorithmus kurz.

c) Angenommen vier Prozesse (T_1, T_2, T_3, T_4) greifen auf drei Ressourcenklassen (A, B, C) zu. Der Verfügbarkeitsvektor sei $V = \begin{pmatrix} 3 & 2 & 4 \end{pmatrix}$ und die Maximalanforderungsmatrix sowie die aktuelle Zuweisungsmatrix lauten:

$$M = \begin{pmatrix} 3 & 1 & 1 \\ 1 & 0 & 2 \\ 3 & 2 & 3 \\ 0 & 2 & 4 \end{pmatrix} \qquad E = \begin{pmatrix} 1 & 1 & 0 \\ 0 & 0 & 2 \\ 1 & 0 & 1 \\ 0 & 1 & 1 \end{pmatrix}$$

Ist eine sichere Ausführung aller Prozesse garantiert? Begründen Sie ihre Antwort.

d) Welche Lösungen verhindern ein sogenanntes Busy Waiting (aktives Warten) und wie können diese Lösungen umgesetzt werden?

Aufgabe 2. 2+3+3 P.

Gegeben seien vier Prozesse T_0, T_1, T_2 und T_3, die jeweils zum Zeitpunkt t_i verfügbar werden und eine Ausführungslänge d_i gemäß der folgenden Tabelle besitzen.

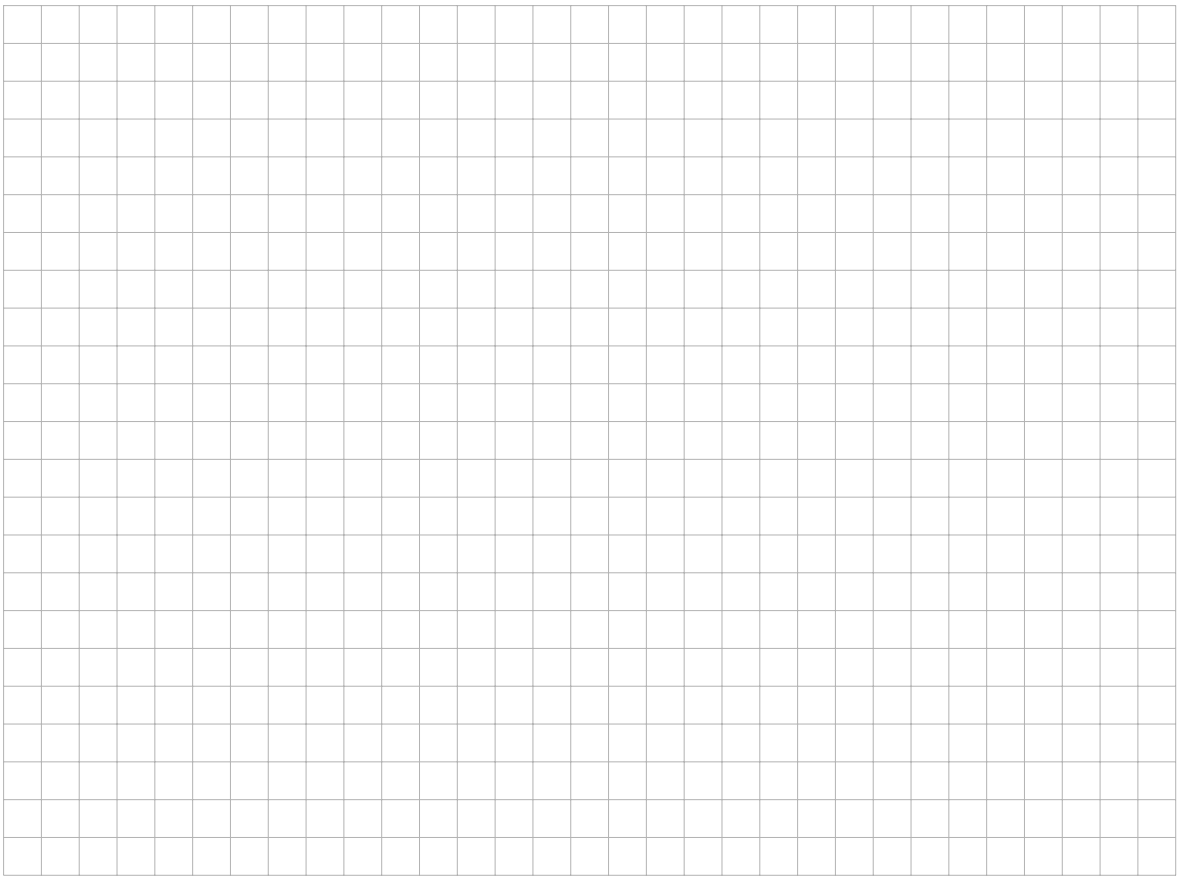
Task T_i	Verfügbarkeit t_i	Länge d_i
T_0	0	3
T_1	0	4
T_2	0	1
T_3	7	2

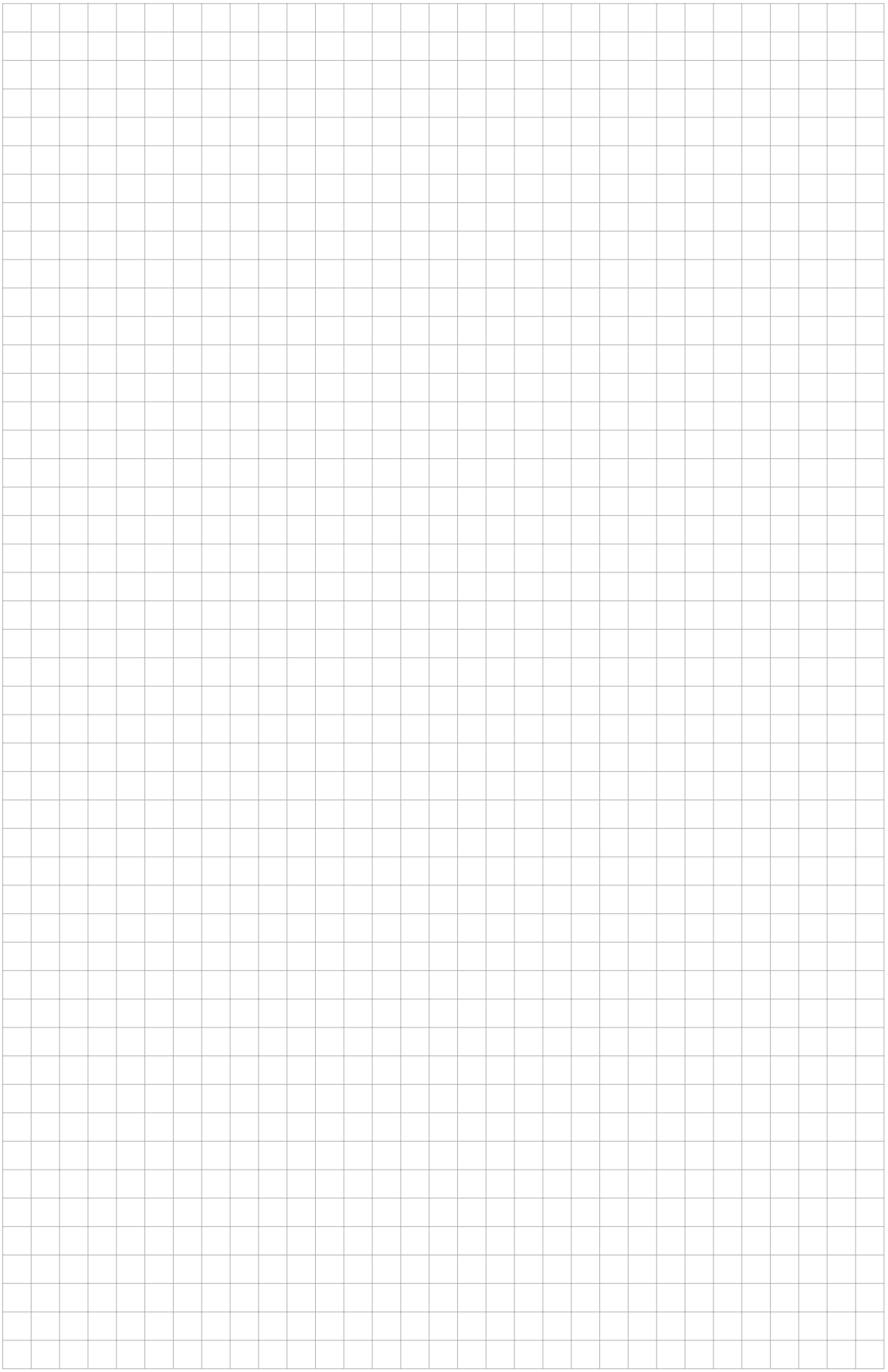
a) Geben Sie an, welche Abarbeitungsreihenfolge nach dem Schedulingverfahren Shortest-Job-First entsteht.

b) Geben Sie an, welche Abarbeitungsreihenfolge nach dem Schedulingverfahren Round Robin entsteht (das Quantum beträgt 2 Zeiteinheiten).

c) Was bedeutet der Begriff *präemptives Multitasking*? Wie unterscheidet sich präemptives von nicht-präemptivem Multitasking? Welche Art von Multitasking liegt bei Teilaufgabe a) bzw. b) vor?

Hinweis: Gantt-Diagramme oder eine Tabelle sind gut geeignete Darstellungsformen für die Abarbeitungsreihenfolge.





Aufgabe 3. 10 P.

Betrachten Sie die folgenden drei Codefragmente. Sie implementieren jeweils drei Prozesse, die folgende Randbedingungen erfüllen sollen:

- Ein Prozess P1 errechnet ein Ergebnis, das von Prozess P2 weiterbenutzt werden soll.
- Prozess P2 darf nicht ausgeführt werden, bevor das Ergebnis von P1 vorliegt.
- P1 wiederum muss warten bis P2 seine Berechnungen fertiggestellt hat, bevor er ein neues Ergebnis an P2 liefert.
- Prozess P2 liefert sein Ergebnis an Prozess P3 weiter. Dieser besitzt einen Puffer mit N freien Plätzen.

Ergänzen Sie die drei Codefragmente so, dass diese Bedingungen erfüllt werden. Verwenden Sie dazu Semaphoren mit den Operationen `semaphore.down()` und `semaphore.up()`.

Deklarieren Sie dazu geeignete Semaphoren, verwenden Sie sinnvolle Namen und initialisieren Sie die Semaphoren sinnvoll.

Prozess P1:

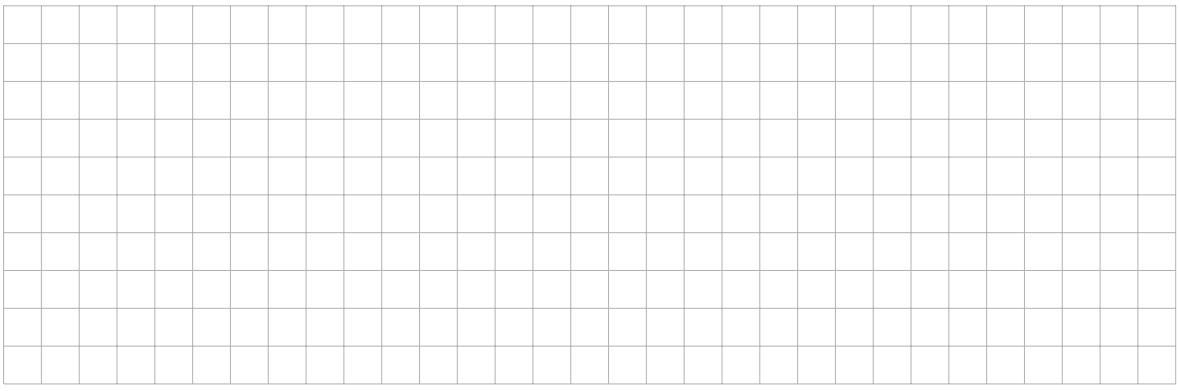
```
while True:
  data = produce()
  put(data, x)
```

Prozess P2:

```
while True:
  data = get(x)
  b = process(d)
  put(b, buf)
```

Prozess P3:

```
while True:
  data = get(buf)
  consume(data)
```



Aufgabe 4. 1+3+2+3+2 P.

Betrachten Sie folgende FAT-Tabelle (Der Eintrag -1 heißt hier „Ende der Kette“):

Block 0	3
Block 1	8
Block 2	
Block 3	1
Block 4	
Block 5	14
Block 6	11
Block 7	-1
Block 8	9
Block 9	-1
Block 10	
Block 11	-1
Block 12	
Block 13	
Block 14	7
Block 15	
Block 16	6
…	…

a) Wieviele Dateien enthält diese FAT?

b) Gehen Sie von einer Blockgröße von 1000 Bytes aus. Geben Sie für jede Datei in der FAT an, wie groß sie mindestens sein muss und höchstens sein kann (vgl. Aufgabenteil a)).

c) Beschreiben Sie, wie der Zugriff auf das 31337-te Byte in einem FAT-Dateisystem stattfindet. Gehen Sie dabei weiterhin von einer Blockgröße von 1000 Bytes aus.

d) Beschreiben Sie, wie der Zugriff auf das 31337-te Byte in einem Inode-basierten Dateisystem stattfindet. Verwenden Sie dazu Inodes mit 10 direkten, und jeweils 1 einfach, doppelt und dreifach indirekten Zeigern. Jeder Zeiger auf einen Festplattenblock soll 10 Byte groß sein, die Datenblöcke selbst sind weiterhin 1000 Bytes groß.

e) Vergleichen Sie die beiden Szenarien aus Aufgabenteil c) und d) bezüglich der Zugriffszeit. Dabei dauert es 50ms, einen beliebigen Festplattenblock zu lesen.

Aufgabe 5. 2+2+2 P.

a) Was versteht man unter virtuellem Speicher? Was ist der Unterschied zwischen logischen und physikalischen Adressen?

b) Erläutern Sie das Konzept von Paging anhand von **vier**stufigen Seitentabellen in einem 64bit-System. Verwenden Sie sinnvolle Größen für die einzelnen Parameter. Verdeutlichen Sie Ihre Erklärung mit Hilfe einer Skizze.

c) Beschreiben Sie im Detail, wie im Schema von Teilaufgabe b) der Zugriff auf eine gegebene logische Speicheradresse stattfindet.

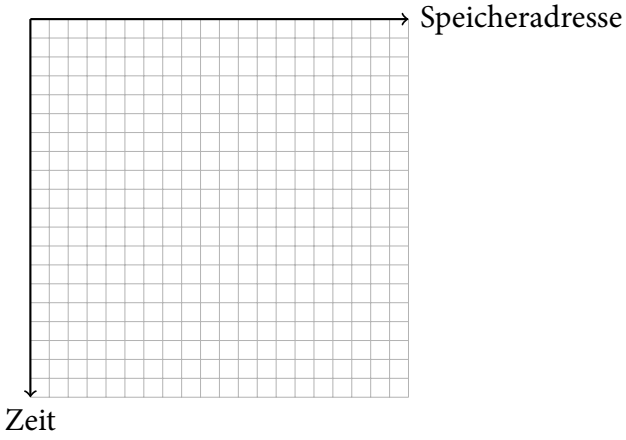
Aufgabe 6. 2+2+2 P.

a) Beschreiben Sie das Prinzip der (Speicher-)Lokalität. Welche Auswirkungen hat dieses Prinzip auf Adressen, insbesondere in Hinsicht auf logische Adressen in Verbindung mit Segmentierung und/oder Paging?

Eine Methode, Lokalität graphisch darzustellen, sind Speicherspurdiagramme. In diesen Diagrammen wird für jeden Speicherzugriff zu einem bestimmten Zeitpunkt ein Punkt eingetragen. Auf diese Weise erhält man über die Zeit hinweg einen Weg über verschiedene Speicherstellen.

b) Welches Zugriffsmuster wird durch folgenden Quellcode erzeugt?

```
l = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
for i in range(0, 5):
    sum = 0
    for j in range(0, 10):
        sum = sum + l[j]
```



c) Welches Zugriffsmuster wird durch folgenden Quellcode erzeugt?

```
l = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
sum = 0
for i in range(0, 5):
    sum = sum + l[i] + l[9-i]
```