Hochschule RheinMain
Studiengang allegemeine Informatik
Prof. Dr. Robert Kaiser

Klausur Betriebssysteme **M U S T E R L Ö S U N G**

WS 2011/2012

Nachname:Mustermann	Vorname:Hansl
Matr.-Nr.:1234567890	
Unterschrift:	

Sie erhalten eine geheftete Klausur. Bitte lösen Sie die Heftung **nicht**. Bitte geben Sie Ihre persönlichen Daten an, und unterschreiben Sie die Klausur. Die Klausur ist nur mit Unterschrift gültig. Die Klausur muss mit dem Verlassen des Raumes abgegeben werden.

Zum Bestehen der Klausur sind 40 Punkte (50%) notwendig.

Im Falle nicht ausreichenden Platzes benutzen Sie bitte zusätzliche Blätter, die Sie mit Name und Matrikelnummer versehen. Machen Sie bitte eindeutig kenntlich, auf welche Aufgabe sich Ihre Antwort bezieht.

Dauer: 90 min

Hilfsmittel: Taschenrechner für arithm. Operationen,
Bücher, Vorlesungsskript, eigene Aufzeichnungen.

Punkte:

Aufgabe	Soll-Punkte	Ist-Punkte
1	8	
2	11	
3	8	
4	8	
5	8	
6	8	
7	6	
8	9	
9	6	
10	8	
Gesamt	80	

Note:

Aufgabe 1: **(8 Punkte)**

Sind die folgenden Aussagen wahr oder falsch?

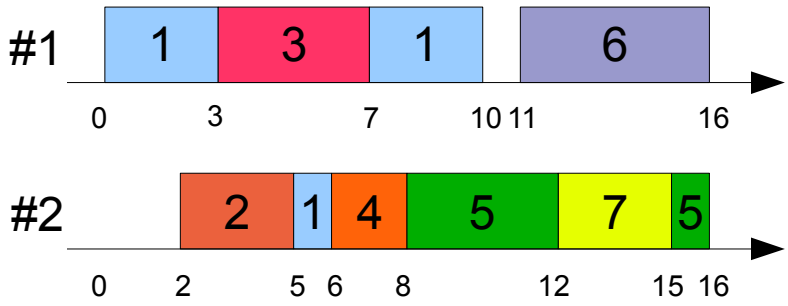
Kreuzen Sie bitte jeweils die richtige Lösung an, höchstens ein Kreuz pro Zeile

Aussage	richtig	falsch
In C wird mit char s[]; ein char-Vektor fester Länge vereinbart, dessen Länge an anderer Stelle definiert werden muss.	**x**	
Der Betriebssystemkern ist der Teil eines Betriebssystems, der im privilegierten Modus arbeitet.	**x**	
Eine Memory Management Unit (MMU) bildet umkehrbar eindeutig virtuelle und physische Adressen aufeinander ab.		**x**
Bei einem RAID1-System steht immer die Hälfte der Plattenkapazität zum Speichern von Nutzdaten zur Verfügung..	**x**	
Der „Peterson-Algorithmus“ dient zur Realisierung des wechselseitigen Ausschlusses zwischen zwei Prozessoren.	**x**	
Der „Working Set“ eines Programms ist die Menge der von diesem Programm augenblicklich benutzten (d.h. geöffneten) Dateien.		**x**
Das „FIFO-“Seitenersetzungsverfahren gehört zu den Stack-Algorithmen		**x**
Bei einem RAID5-System stehen immer Drei Viertel der Plattenkapazität zum Speichern von Nutzdaten zur Verfügung.		**x**
Der „Test-and-Set“ Maschinenbefehl dient zur Realisierung des wechselseitigen Ausschlusses zwischen mehreren Prozessoren.	**x**	
Beim Eintreffen eines Interrupt wechselt ein Prozessor in den privilegierten Modus	**x**	
Die Shell arbeitet als Teil des Betriebssystems immer im privilegierten Modus.		**x**
Wird der physische Speicher eines Rechners vergrößert, so erhöht sich dadurch die Anzahl der Seitenrahmen, nicht aber die der Seiten.	**x**	
Ein Datencache ist ein Beispiel für Tertiärspeicher		**x**
Mehrstufige Seitentabellen erfordern in der Regel weniger Speicherplatz als einstufige, da Programme nur einen kleinen Teil ihres virtuellen Adressraums nutzen.	**x**	
Wird der physische Speicher eines Rechners vergrößert, so führt dies in allen Fällen zu einer Verringerung der Seitenfehlerrate.		**x**
Ein Hard-Link kann nicht über Dateisystemgrenzen hinweg verweisen	**x**	

Aufgabe 2: **(11 Punkte)**

(a) Die folgenden Aufträge treffen in einem 2-Prozessorsystem zu den angegebenen Zeitpunkten ein. Eine höhere Zahl für die Priorität drücke eine höhere Wichtigkeit aus. Die Aufträge seien reine Rechenaufträge ohne I/O. Die Prozesswechselzeiten werden vernachlässigt. Geben Sie für das Preemptive Prioritäts-Scheduling-Verfahren ein Belegungs-Diagramm für die beiden CPUs für die Abarbeitung der Prozesse an. (7 P.)

Auftrag	Ankunftszeit	Bedienzeit	Priorität	**AbschZt**	**VerwZt**	
1	0	7	3	**10**	**10**	
2	2	3	5	**5**	**3**	
3	3	4	4	**7**	**4**	
4	6	2	5	**8**	**2**	
5	7	5	1	**16**	**9**	
6	11	5	2	**16**	**5**	
7	12	3	3	**15**	**3**	
					36	**5,14**



(b) Bestimmen Sie die mittlere Verweilzeit. (Sie können die freien Spalten im Diagramm verwenden). (2P)

Mittlere Verweilzeit: 5,14

(c) Was versteht man unter Mehrschlangen-Feedback-Scheduling? Erläutern Sie stichwortartig die prinzipielle Vorgehensweise. (2P)

Prozesse werden klassifiziert (priorisiert) als einer bestimmten Gruppe zugehörig (z.B. interaktiv, batch). Diese Priorität kann sich, abhängig vom Verhalten des Prozesses, dynamisch ändern. Alle rechenwilligen Prozesse einer bestimmten Klasse werden in einer eigenen Ready Queue verwaltet. Jede Ready Queue kann ihr eigenes Scheduling-Verfahren haben (z.B. Round-Robin für interaktive Prozesse, FCFS für batch-Prozesse). Zwischen den Ready Queues wird i.d.R. unterbrechendes Prioritäts-Scheduling angewendet, d.h.: jede Ready Queue besitzt eine feste Priorität im Verhältnis zu den anderen; wird ein Prozess höherer Priorität rechenwillig, wird der laufende Prozess unterbrochen (preemption).

Aufgabe 3: **(8 Punkte)**

Betrachten Sie das Leser-Schreiber-Problem: mehrere, gleichzeitig lesende Prozesse sind zugelassen, mehrere, gleichzeitig schreibende Prozesse oder gleichzeitiges Lesen und Schreiben jedoch nicht. Geben Sie eine Lösung für die Synchronisation an, basierend auf Semaphoren, die als Typ `semaphore` mit den üblichen Operationen `void P(semaphore * sem)` und `void V(semaphore * sem)` sowie einer Initialisierungsoperation `init(semaphore * sem, int initvalue)` und der Operation `stand(semaphore *sem)` zum Erfragen des Zählerstandes gegeben seien. Ergänzen Sie das im folgenden angegebene Programmskelett um die Aufrufe zur Initialisierung der Semaphoren sowie zur korrekten Synchronisation im Leser-Programm (reader) und im Schreiber-Programm (writer).

```
/* Definition der Semaphoren                                          */

semaphore mutex, nLeser, db;
int nLeser;




/* Initialisierung der Semaphoren                                     */
init(&mutex, 1);
nLeser = 0;
init(&db, 1);









void writer(void) {              /* Schreiber                         */
   int item;
   while(TRUE) {




      produce_data(&item);          /* erzeuge Datum                  */




      P(&db);

      write_item(item);           /* schreibe Datum in den Puffer */

      V(&db);
```

```
   }
}

void reader(void) {                  /* Leser                    */
   int item;
   while(TRUE) {


      P(&mutex);
      ++nLeser;
      if(nLeser == 1)
          P(&db);
      V(&mutex);

      read_item(&item);             /* lies Eintrag aus Puffer   */

      P(&mutex);
      nLeser--;
      if(nLeser == 0)
          V(&db);
      V(&mutex);


      use_data(item);              /* verarbeite Eintrag          */









   }
}
```

Aufgabe 4: **(8 Punkte)**

Die unten stehende Tabelle stelle einen Aussschnitt aus dem 32-bit breit organisierten RAM-Speicher eines „Little Endian“ Rechners ohne Memory Management Unit dar. Tragen Sie in die einzelnen Felder bitte jeweils in Form zweistelliger Hexadezimalzahlen ein, wie der Inhalt dieses Speichers nach der Ausführung des angegebenen C-Programms aussieht.

```
main()
{
     char *p = (char*)0x1000;
     short *s = (short*)0x1004;
     int   *l = (int*)0x1008;
     char  *str = { 0x01, 0x02, 0x03, 0x04, '\0'};

     *p = 0x5a;
     p[1] = 0xa5;
     *s = 0x1234;
     s[1] = 0x5678;
     *l = 0x1234;
     strcpy(&l[1], str);
     if(l[0] == l[1])
          l[2] = -1;
     else
          l[2] = 0;
}
```

Adresse/Byte	0	1	2	3
0x1000	**0x5a**	**0xa5**		
0x1004	**0x34**	**0x12**	**0x78**	**0x56**
0x1008	**0x34**	**0x12**	**0x00**	**0x00**
0x100C	**0x01**	**0x02**	**0x03**	**0x04**
0x1010	**0x00**	**0x00**	**0x00**	**0x00**
0x1014				

Aufgabe 5: **(8 Punkte)**

Ein inode-basiertes Dateisystem hat eine Datenblockgröße von 1024 Bytes und hält in jedem Inode bis zu 4 direkte Verweise auf Datenblöcke und jeweils **zwei** Verweise auf einfach und doppelt indirekte Blöcke. Alle indirekten Blöcke enthalten ihrerseits wieder 4 Verweise auf untergeordnete Blöcke (indirekte Blöcke bzw. Datenblöcke).

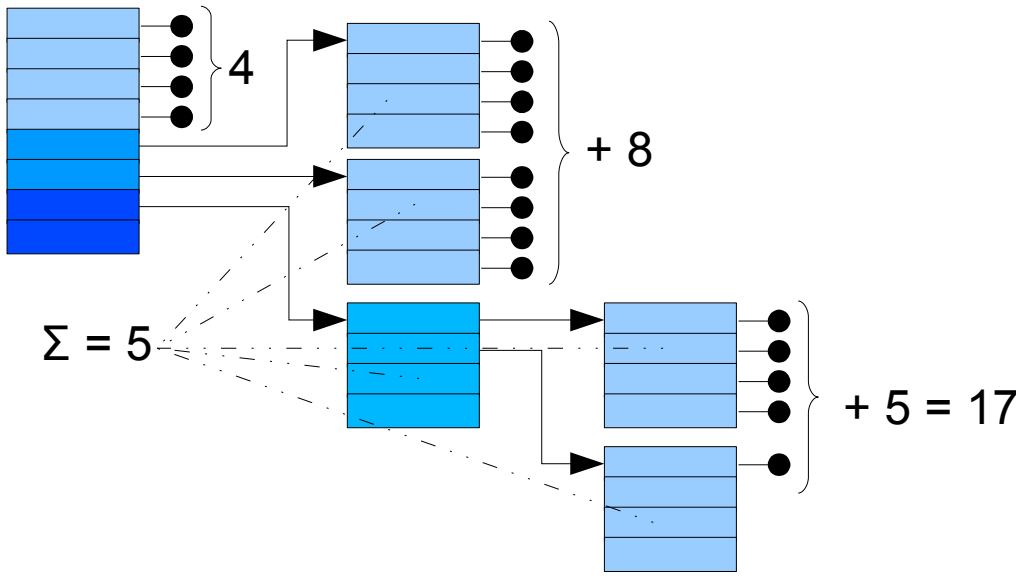
a) Wie groß kann eine Datei maximal werden? (Begründung?)(3P)

(4 + 2 * 4 + 2 * 4 * 4)) * 1.024 = 45.056

b) Wie viele indirekte Blöcke werden für eine Datei der Größe 17171 Bytes benötigt? (Begründung und Skizze?)(5P)

Es werden insgesamt 17 Datenblöcke benötigt: 17*1024=17408 > 17171

Um diese zu adressieren werden 5 indirekte Blöcke benötigt.



Aufgabe 6: **(8 Punkte)**

Gegeben sei ein Rechner mit 64 Byte physischem Speicher. Die Seitengröße des Rechners betrage 16 Byte. Die Seitentabelle habe folgenden Inhalt:

Index	Inhalt	R-Flag
0	3	0
1	1	0
2	2	0
3	2	0

a) Tragen Sie in der unten angegebenen, tabellarischen Darstellung des physischen Speichers ein, welche Inhalte durch die Ausführung des folgenden Programms in welche Speicherzellen geschrieben werden.(6P)

```
main()
{
     char *string = (char*)10;
     char *s1     = (char*)32;
     char *s2     = (char*)57;

     strcpy(string, "Hallo Welt");
     strcpy(s1, "Tschuess ");
     strcpy(s2, "Welt");
}
```

Adresse	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0																
16	**W**	**e**	**l**	**t**	**\0**											
32	**T**	**s**	**c**	**h**	**u**	**e**	**s**	**s**	**' '**	**W**	**e**	**l**	**t**	**\0**		
48											**H**	**a**	**l**	**l**	**o**	**' '**

b) Tragen Sie bitte in der nachfolgenden Tabelle die Werte der "referenziert"-Flags (R-Flags) nach der Ausführung des angegebenen Programms ein.(2P)

Index	Inhalt	R-Flag
0	3	**R**
1	1	**R**
2	2	**R**
3	2	**R**

Virtueller Speicher

Adresse	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0											H	a	l	l	o	' '
16	W	e	l	t	\0											
32	T	s	c	h	u	e	s	s	' '	\0						
48										W	e	l	t	\0		

Aufgabe 7: **(6 Punkte)**

Beim Einsatz des „Aging“ Seitenersetzungsverfahrens ergebe sich die unten angegebene Liste eingelagerter Seiten. Es sind nur 4 Seitenrahmen verfügbar. Die Spalte „Zähler“ gibt die den Seiten jeweils zugeordneten Zähler des Aging-Verfahrens als Dualzahl an.

Seitennummer	Zähler
9	00010000
32	01101011
17	00111111
21	00001111

Es wird nun eine neue Seite mit der Nummer 42 angefordert.

a) Welche Seite wird ausgelagert, um für Seite 42 Platz zu machen?(2P)

Seite 21 wird ausgelagert

b) Während der nun folgenden 20 Millisekunden wird jede der eingelagerten Seiten mindestens ein mal referenziert. Anschließend wird der Aging-Algorithmus aufgerufen, um die Zählerstände zu aktualisieren. Tragen Sie bitte in der folgenden Tabelle die Nummern der nun eingelagerten Seiten und die zugehörigen, aktualisierten Zählerstände ein.(4P)

Seitennummer	Zähler
9	**10001000**
32	**10110101**
17	**10011111**
42	**10000000**

Aufgabe 8: **(9 Punkte)**

Es liege ein Paging-System mit 5 Seitenrahmen und 10 Seiten (0..9) vor. Ferner sei der Reference String 1 4 2 4 3 2 5 4 7 6 7 4 2 1 1 2 6 7 6 6 gegeben.

(a) Wieviele Seitenfehler treten im Falle von LRU als Seitenersetzungsalgorithmus auf, wenn alle fünf Rahmen anfangs leer sind? (4 P.) **8 Seitenfehler!**

(b) Sagen Sie unter Verwendung der Distanzkette die Anzahl der Seitenfehler voraus, wenn vier oder sechs Seitenrahmen zugeordnet würden. (4 P.)

1	4	2	4	3	2	5	4	7	6	7	4	2	1	1	2	6	7	6	6
1	4	2	4	3	2	5	4	7	6	7	4	2	1	1	2	6	7	6	6
	1	4	2	4	3	2	5	4	7	6	7	4	2	2	1	2	6	7	7
		1	1	2	4	3	2	5	4	4	6	7	4	4	4	1	2	2	2
				1	1	4	3	2	5	5	5	6	7	7	7	4	1	1	1
						1	1	3	2	2	2	5	6	6	6	7	4	4	4
								1	3	3	3	3	5	5	5	5	5	5	5
									1	1	1	1	3	3	3	3	3	3	3
P	P	P		P		P		P	P				P						
∞	∞	∞	2	∞	3	∞	4	∞	∞	2	3	5	7	1	2	5	5	2	1

(c) Was versteht man unter Belady's Anomalie(1P)

Eine Erhöhung der Anzahl der Seitenrahmen führt je nach Seitenersetzungsverfahren u.U. zu mehr Seitenfehlern. Dieses unerwartete Verhalten wurde erstmals von Belady demonstriert.

C(1)	2		F(1)	18
C(2)	4		F(2)	14
C(3)	2		F(3)	12
C(4)	1		F(4)	11
C(5)	3		F(5)	8
C(6)	0		F(6)	8
C(7)	1		F(7)	7
C(8)	0		F(8)	7
C(9)	0		F(9)	7
C(10)	0		F10)	7
C(∞)	7		F(∞)	7

6 Seitenrahmen --> 8 Seitenfehler, 4 Seiten -> 11 Seitenfehler

Aufgabe 9: **(6 Punkte)**

a) Was versteht man unter einer „kritischen Abschnitt“? (2P)

Ein kritischer Abschnitt ist ein Code-Bereich, dessen Ausführung nicht unterbrochen werden darf, da es sonst zu Dateninkonsistenzen kommen kann.

b) Kennzeichnen Sie bitte in dem folgenden C-Codefragment Anfang und Ende des kritischen Abschnittes (2P):

```
#define GROESSE 4711

static int datenpuffer[GROESSE];

static int datenzahler = 0;

void speichere_datum(int datum)
{
<------------- Anfang ------------------>
    datenpuffer[datenzaehler] = datum;
    ++datenzaehler;
<-------------- Ende ------------------->

}
```

c) Das Programm werde nun wie folgt geändert:

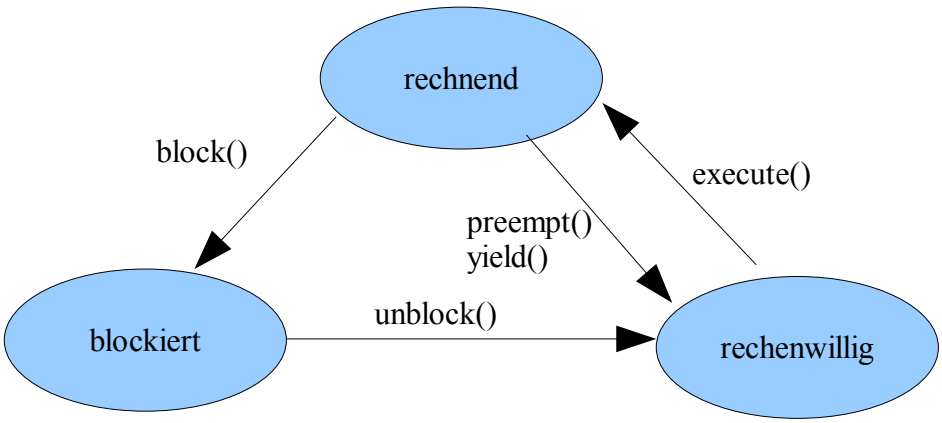
```
void speichere_datum(int datum)
{
    datenpuffer[datenzaehler++] = datum;
}
```

Lässt sich so der kritische Abschnitt beseitigen (Begründung) ? (2P)

Nein: Auch wenn der Abschnitt in C nur noch eine Zeile umfasst, so ist er auf Maschinenebene doch nicht atomar: Es werden mehrere Maschineninstruktionen verwendet, zwischen denen jeweils eine Unterbrechung erfolgen kann.

Aufgabe 10: **(8 Punkte)**

a) Welche drei Prozesszustände werden typischerweise unterschieden? Geben Sie das Zustandsübergangsdiagramm an, und erläutern Sie bitte die vier möglichen Zustandsübergänge (3P).



block(): Prozess wird blockiert (z.B.: Warten auf ein Ereignis)

unblock(): Prozess wird rechenwillig (z.B.: erwartetes Ereignis ist eingetreten)

execute(): Prozess wird rechenend (durch Scheduler aktiviert)

preempt()(/yield()): Prozessor wird entzogen (preempt()) oder abgegeben (yield())

b) Grenzen Sie die Begriffe „Prozess“ und „Thread“ gegeneinander ab.(3P)

Prozess: Hat eigenen Adressraum/eigene Daten/eigene offene Dateien, etc.

Thread: Adressraum, statische Variablen, offene Dateien gemeinsam. Nur stack und evtl Thread Local Storage sind privat

c) Erläutern Sie den Unterschied zwischen kooperativem und preemptivem Multitasking.(2P)

Kooperativ: Prozesse geben CPU nur freiwillig ab (yield())

Preemptiv: CPU kann auch ohne Einverständnis des Prozesses entzogen werden. .