Name: Matrikel-Nr.: Studiengang:

Klausur Betriebssysteme und Sicherheit, 24.07.2013

1	2	3	4	5	6	7	$\sum$
7	10	6	6	4	5	4	42

Aufgabe 1

7 Punkte

Die Firma *Make Money Inc.* hat einen Unix-Rechner, um Aktien zu handeln. Transaktionen müssen der Börse signiert übermittelt werden. Dazu wird das RSA-Verfahren verwendet. Nur Nutzer in der Gruppe `trading` sollen in der Lage sein Transaktionen zu signieren.

Ein Programm `/usr/bin/trade` soll benutzt werden, um Aktien-Transaktionen zu erstellen, zu signieren und abzuschicken. Das Programm soll hierzu den Schlüssel in der Datei `/etc/trading-key` verwenden. Der Schlüssel soll von den Nutzern nicht direkt zugreifbar sein, damit sie nicht von anderen Rechnern signierte Transaktionen abschicken können.

a) Welchen Teil des Schlüssels hat die Börse und welcher liegt auf dem Rechner? **1 P**

b) Was stellt die Börse fest, wenn die Signatur einer übermittelten Transaktion gültig ist? **1 P**

c) Welche der klassischen 3 Schutzziele werden **nicht** gewährleistet? **2 P**

d) Geben sie UNIX-Rechte für das Programm `/usr/bin/trade` und die Datei `/etc/trading-key` an, so dass die genannten Sicherheitseigenschaften erfüllt sind! **3 P**

Name: Matrikel-Nr.: Studiengang:

Aufgabe 2 **10 Punkte**

Ein Programmierer schützt sein aus mehreren Threads bestehendes Programm vor Wettlaufsituationen, indem er jeden Zugriff auf eine geteilte Ressource durch Verwendung eines zugehörigen binären Semaphors schützt. In einer konkreten Situation führen nun 2 Threads folgenden Code aus:

Globale Variablen:

```
sem_t sem_a = 1;
sem_t sem_b = 1;
```

Thread A

```
down(sem_a);
a = a + 10;
down(sem_b);
b = 20;
a = a + b;
up(sem_b);
up(sem_a);
```

Thread B

```
down(sem_b);
b = b + 20;
down(sem_a);
a = 10;
b = a + b;
up(sem_a);
up(sem_b);
```

a) Kann es zwischen den beiden Threads zu einer Verklemmung kommen? Begründen Sie Ihre Antwort unter Verwendung der hinreichenden und notwendigen Bedingungen für das Auftreten von Verklemmungen! **8 P**

Ein Kollege schlägt dem Programmierer vor, Semaphore mit statischen Prioritäten zu versehen. Die Funktion zum Anfordern eines Semaphors (`down(S)`) wird in diesem Fall nur dann den Semaphor belegen, wenn der Semaphor frei ist und wenn der aufrufende Thread nicht bereits einen Semaphor mit höherer Priorität belegt hat. Andernfalls blockiert der aufrufende Thread bis die höher-prioren Semaphore freigegeben wurden.

b) Verhindert dieses Vorgehen das Auftreten einer Verklemmung? Begründen Sie Ihre Antwort! **2 P**

Name: Matrikel-Nr.: Studiengang:

Aufgabe 3

6 Punkte

In einem Echtzeitsystem ist eine Menge von drei periodischen Tasks so einzuplanen, dass deren Jobs in jeder Periode erfolgreich beendet werden. Das Periodenende entspricht der Deadline. Die Parameter der Tasks beschreiben jeweils die Periode p und die konstante Bearbeitungszeit e der Jobs:

T_1	T_2	T_3
$p_1 = 12 \quad e_1 = 4$	$p_2 = 3 \quad e_2 = 1$	$p_3 = 8 \quad e_3 = 1$

Die Tasks sind unabhängig voneinander und an beliebiger Stelle unterbrechbar. Der Scheduling-Overhead werde vernachlässigt. Es stehe genau ein Prozessor zur Verfügung.

a) Geben Sie einen Ablaufplan nach dem Scheduling-Verfahren RMS an! Berechnen Sie die durchschnittliche Wartezeit der Jobs in diesem Ablaufplan! **2 P**

$\overline{t_w} =$

b) Gibt es einen gültigen Ablaufplan (nicht notwendigerweise nach RMS) mit längerer durchschnittlicher Wartezeit? Begründen Sie Ihre Antwort! **2 P**

c) Es soll eine weitere Task T_4 mit Parametern p_4 und e_4 eingeplant werden. Geben Sie ein Beispiel für T_4 an, so dass die gesamte Taskmenge weiterhin einplanbar ist! Dabei soll gelten $e_4 > 0$. **1 P**

$p_4 =$ $e_4 =$

d) Geben Sie ein Beispiel für T_4 an, so dass die gesamte Taskmenge mit keinem Scheduling-Verfahren mehr eingeplant werden kann! Es soll $p_4 > e_4$ gelten. **1 P**

$p_4 =$ $e_4 =$

Name: Matrikel-Nr.: Studiengang:

Aufgabe 4 — 6 Punkte

Gegeben sind folgende UNIX-Programme:

Programm A

```
int main()
{
    int arg = get_arg ();
    if (arg == 0)
        exit (0);

    pid_t pid = fork ();
    if (pid > 0 ) {
        waitpid (pid, NULL, 0);
        printf ("%d ", arg);
    }
    else if (pid == 0) {
        exec ("B", arg - 1);
    }
    exit (0);
}
```

Programm B

```
int main()
{
    int arg = get_arg ();
    if (arg == 0)
        exit (0);

    pid_t pid = fork ();
    if (pid > 0 ) {
        exec ("A", arg - 1);

    }



    exit (0);
}
```

Dabei führt `exec(program, zahl)` das angegebene Programm aus und übergibt `<zahl>` als erstes Argument. Die Funktion `get_arg()` gibt dieses erste Argument als Integer-Wert zurück. Gehen Sie davon aus, dass alle Systemaufrufe fehlerfrei zurückkehren.

a) Stellen Sie den zeitlichen Ablauf des Aufrufs von „A 3“ als Baum dar und tragen Sie dabei jeden Funktionsaufruf ein! **4 P**

b) Geben Sie die möglichen Ausgaben des Aufrufs „A 6“ an! **2 P**

Name: Matrikel-Nr.: Studiengang:

Aufgabe 5 **4 Punkte**

In einer Anwendung existieren zwei globale Variablen `i` und `lock`:

```
int i    = 3;
int lock = 0;
```

Weiterhin ist folgender Programmabschnitt gegeben:

```
while (xchg(&lock, 0, 1) == 1)
{ /* do nothing */ }

i = i + 1;
do_long_work(i);

lock = 0;
```

Die verwendete Funktion `xchg()` prüft hierbei atomar, ob der Wert der Variable `lock` gleich 0 ist und setzt ihn nur in diesem Fall auf den Wert 1. Der Rückgabewert der Funktion ist der vorherige Wert der Variable `lock`.

a) Vier Threads führen das oben gegeben Programm-Fragment parallel je einmal aus. Welche Werte kann die Variable `i` nach vollständiger Ausführung der vier Threads haben? **1 P**

b) Welches Problem hat der oben dargestellte Code-Abschnitt? **1 P**

c) Geben Sie eine Implementierung (in Pseudo-Code) an, die das Problem behebt! **2 P**

Name: Matrikel-Nr.: Studiengang:

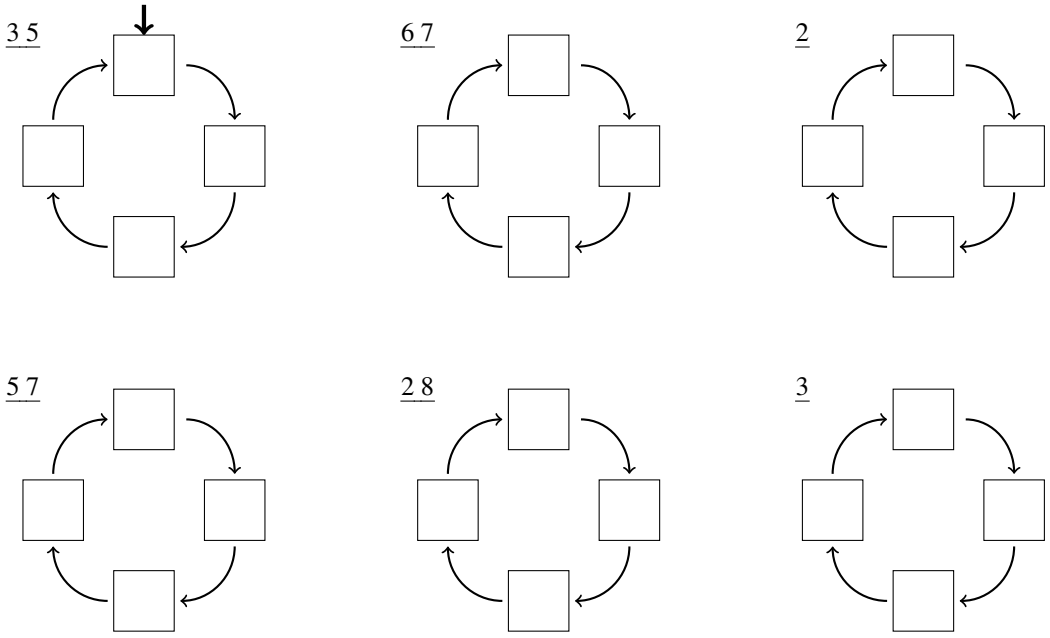
Aufgabe 6

5 Punkte

In einem Betriebssystem ist die Seitenersetzung mittels Clock-Algorithmus implementiert. Für einen Prozess stehen hierbei 4 Rahmen zur Verfügung. Die Seitenreferenz-Folge ist:

3 5 6 7 2 5 7 2 8 3

a) Tragen Sie den Zustand des Systems nach den jeweils oben links stehenden Seitenreferenzen in die Schemata ein. Notieren Sie ebenfalls die Referenzbits, sowie den Zeiger des Clock-Algorithmus! **4 P**



b) Wie oft wurde der Clock-Algorithmus insgesamt ausgeführt? **1 P**

Name: Matrikel-Nr.: Studiengang:

Aufgabe 7 **4 Punkte**

Ein 32-bit-System verwendet eine Seitengröße von 16 Kilobyte. Zur Speicherverwaltung wird eine zweistufige Seitentabelle verwendet. Die Tabellen auf beiden Stufen enthalten die selbe Anzahl an Einträgen. Diese Einträge passen jeweils exakt in eine Seite. Das System verfügt über 128 MB physischen Speicher.

a) Wieviele Bit einer Adresse werden für das Offset verwendet? **0.5 P**

b) Wieviele Einträge hat eine Seitentabelle? **1 P**

c) Wieviele Rahmen verwaltet das System? **0.5 P**

d) Wieviele Rahmen werden mindestens für die Speicherung der Seitentabellen-Hierarchie benötigt, falls der komplette physische Speicher in den virtuellen Adressraum eingeblendet werden soll? **1 P**

e) Wieviele Rahmen werden dafür maximal benötigt? **1 P**