Name: Matrikel-Nr.: Studiengang:

Klausur Betriebssysteme und Sicherheit, 21.02.2013

1	2	3	4	5	6	$\sum$
6	6	9	7	7	7	42

Aufgabe 1

6 Punkte

In einem System paralleler Prozesse werden 4 Prozesse A, B, C und D ausgeführt. Für diese Prozesse stehen 6 Betriebsmittel gleichen Typs zur Verfügung. Der Vektor der Betriebsmittel-Anforderungen für die Prozesse sei $ANF = (5, 2, 3, 5)^T$. Ferner sei $ZUW = (1, 1, 1, 2)^T$ der Vektor der aktuellen Betriebsmittel-Zuweisungen.

a) Ist der angegebene Zustand ein sicherer Zustand? Nutzen Sie den Bank-Algorithmus, um Ihre Antwort zu begründen! **5 P**

b) Der Prozess D benötigt nun 6 statt der ursprünglich 5 Betriebsmittel, d.h. die Vektoren ändern sich zu $ANF = (5, 2, 3, 6)^T$, $ZUW = (1, 1, 1, 2)^T$. Wie viele Betriebsmittel werden mindestens benötigt, damit es sich hierbei um einen sicheren Zustand handelt? **1 P**

Name: Matrikel-Nr.: Studiengang:

Aufgabe 2 **6 Punkte**

Gegeben ist eine Menge von Jobs $J = \{A, B, C, D, E, F, G, H\}$ mit folgenden Ausführungszeiten:

A	B	C	D	E	F	G	H
5	2	1	2	6	2	3	1

Zwischen den Jobs aus J bestehen Abhängigkeiten, die durch die folgende Präzedenzrelation gegeben sind: $P = \{(A, E); (B, C); (B, D); (C, F); (D, G); (F, H); (G, H)\}$

Die Jobs sind nicht unterbrechbar.

a) Zeichnen Sie zur gegebenen Job-Menge J den Präzedenzgraphen als Vorgangspfeilnetz und beschriften Sie die Kanten mit den Ausführungszeiten der Jobs! **2 P**

b) Durch Verwendung mehrerer Prozessoren möchten wir die kürzeste mögliche Gesamtbearbeitungszeit der Job-Menge J erreichen. Wie lang ist diese kürzeste Gesamtbearbeitungszeit? Begründen Sie, warum keine weitere Verkürzung möglich ist! **1 P**

c) Geben Sie die kleinste Anzahl an Prozessoren an, mit der diese Gesamtbearbeitungszeit erreicht wird. Geben Sie einen möglichen dabei entstehenden Ablaufplan der Jobs J an. **1 P**

d) Wie ist die durchschnittliche Verweilzeit der Jobs in Ihrem bei c) gefundenen Ablaufplan? **1 P**

e) Gibt es unter den genannten Bedingungen (kürzeste mögliche Gesamtbearbeitungszeit, Einhaltung der Präzedenz) auch Ablaufpläne mit anderer durchschnittlicher Job-Verweilzeit? **1 P**

Name: Matrikel-Nr.: Studiengang:

Aufgabe 3

9 Punkte

Gegeben ist ein 32-Bit System mit virtuellem Speicher, welches eine zweistufige Adressübersetzung nutzt. Ein Seitentabelleneintrag hat dabei, unabhängig von der Stufe, eine Größe von 4 Byte. Das Offset in einer Seite ist mit 12 Bit angegeben. Der Index in das Seitenverzeichnis hat 10 Bit.

a) Wie groß ist eine Seite in diesem System? **0.5 P**

b) Wie viele Einträge stehen in einer Seitentabelle? **0.5 P**

Einem Prozess in diesem System stehen 4 Rahmen physischen Speichers zur Verfügung. Diese werden mittels LRU verwaltet. Zu Beginn ist der Anwendung kein Speicher zugewiesen - alle Rahmen sind frei.

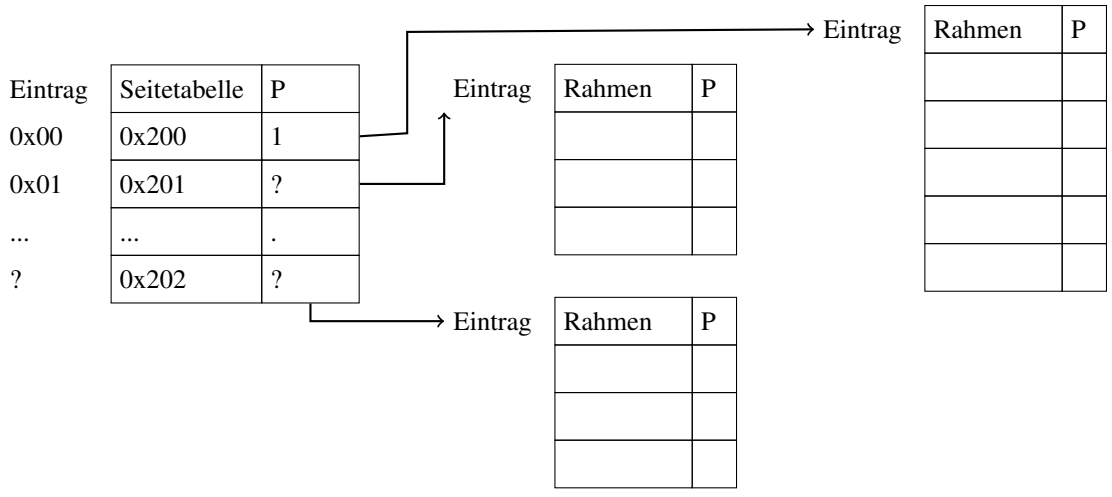
c) Füllen Sie folgendes Schema für Zugriffe auf die im Tabellenkopf angegebenen Adressen aus. Notieren Sie dabei in jeder Spalte, die Nummer (in Hex!) der durch den Zugriff referenzierten Seite in der Zeile des ihr zugeordneten Rahmens. Immer wenn ein Seitenfehler auftritt setzen Sie ein Kreuz in der letzten Zeile. **4 P**

Rahmen	0x88192	0x123	0x112	0x80000FE0	0x88000	0x0434100	0x87000	0x1AAA
0x01								
0x02								
0x03								
0x04								
PF								

Im folgenden sei die vom Betriebssystem generierte Seitentabelle für den Prozess aus c) skizziert. Sie befindet sich im Zustand nach der Ausführung der Zugriffe aus dem vorherigen Aufgabenteil.

d) Füllen Sie an den Stellen die mit einem ? markiert sind die korrekten Daten ein. **1 P**

e) Füllen Sie außerdem die Seitentabellen aus, soweit dies mit Hilfe der obigen Informationen möglich ist. In den Seitentabellen ist das present Bit in der Spalte P einzutragen. Seitentabellen, die nicht present sind bzw. keine gültigen Einträge haben, müssen Sie nicht ausfüllen. **3 P**



Name: Matrikel-Nr.: Studiengang:

Aufgabe 4 **7 Punkte**

Auf einem UNIX-System mit mehreren Benutzern möchte Bob eine Nachricht mit RSA verschlüsseln und an Alice senden. Bobs Verschlüsselungsprogramm benutzt folgenden (Pseudo-)Code, um die zu verschlüsselnde Nachricht im für alle les- und schreibbaren Verzeichnis `/tmp` zwischenzuspeichern:

```
fd = open("/tmp/bob-message", O_CREAT); // Siehe Anmerkung
fchmod(fd, 0600); // Setze Rechte so, dass nur der Besitzer Lesen und
                  // Schreiben kann, jeder andere Zugriff wird verhindert
unlink(fd);
write(fd, message, sizeof(message));
/* ... fd wird benutzt ... */
close(fd);
```

Anmerkung: Wenn `open()` das Flag `O_CREAT` übergeben wird, legt es die gewünschte Datei mit Standardberechtigungen (Besitzer: rw, Gruppe/Andere: r) an, falls sie nicht bereits existiert.

Alice hat den folgenden öffentlichen RSA-Schlüssel: $n_a = 15, c_a = 7$. Bob hat den folgenden öffentlichen RSA-Schlüssel: $n_b = 35, c_b = 5$.

a) Nennen Sie je einen Vorteil und einen Nachteil von asymmetrischer verglichen mit symmetrischer Verschlüsselung! **2 P**

b) Welche Auswirkungen hat es, wenn die temporäre Datei gelöscht (unlink()) wird, während Bobs Programm sie noch benutzt und warum? **2 P**

c) Zeigen sie eine Schwachstelle in der Implementierung von Bobs Programm auf, die es dem Benutzer Eve erlaubt, an die unverschlüsselte Nachricht zu gelangen. **1 P**

d) Alice hat von Bob die Nachricht y = 8 bekommen. Geben Sie den Schlüssel an, der zum Entschlüsseln benötigt wird und entschlüsseln Sie die Nachricht! **2 P**

Name: Matrikel-Nr.: Studiengang:

Aufgabe 5

7 Punkte

Gegeben ist folgendes UNIX-Programm:

```
int main() {
    int i    = 0;
    int pid1 = fork();
    if(pid1 == 0) {
        i = i + 1;
        int pid2 = fork();
    }
    else {
        waitpid(pid1, NULL, 0);
    }
    printf("%d ",i);
    return 0;
}
```

a) Tragen Sie in die folgende Tabelle für einen möglichen Ablauf des Programms die Abarbeitungsschritte der beteiligten Prozesse und den jeweiligen Wert der benutzten Variablen ein. **3 P**

Prozess A	Prozess B	Prozess C

b) Vervollständigen Sie das in a) gegebene Programm, so dass seine Ausführung *immer* die Ausgabe „2 1 0 “ erzeugt. Lassen Sie dabei den vorgegebenen Code wie er ist und ergänzen Sie ihn nur an der dafür vorgesehenen Stelle. **4 P**

```
int main() {
    int i    = 0;
    int pid1 = fork();
    if(pid1 == 0) {
        i = i + 1;
        int pid2 = fork();









    }
    else {
        waitpid(pid1, NULL, 0);
    }
    printf("%d ",i);
    return 0;
}
```

Name: Matrikel-Nr.: Studiengang:

Aufgabe 6 **7 Punkte**

Der Mars Rover Curiosity habe zur Aufnahme und Verarbeitung von Bodenproben folgendes System: Zwei einfache Prozessoren, auf denen jeweils einer der beiden Threads P und C ausgeführt werden, tauschen über einen ein-elementigen Puffer Daten miteinander aus. Dabei liest der erste Thread P wiederholt Messwerte von seinen Sensoren und schreibt diese in den Puffer, während der zweite Thread C diese Daten aufbereitet und sendet. Die zur Verfügung stehenden CPUs besitzen keine Hardware-Unterstützung für `test-and-set` oder `exchange`-Operationen, sodass Sie auf atomare Load- und Store-Befehle angewiesen sind. Verwenden Sie daher folgende zwei Funktionen, wenn Sie atomar auf den Speicher zugreifen müssen:

`load(variable)` – liefert als Rückgabewert den atomar gelesenen Wert der Variablen

`store(value, variable)` – weist den übergebenen Wert atomar der Variablen zu

a) Geben Sie in C-ähnlichem Pseudocode eine einfache Lösung an, bei der die beiden Threads P und C im strengen Wechsel exklusiv auf den Puffer zugreifen. Verwenden Sie zur Synchronisation ausschließlich Load- und Store-Befehle und so wenige Ressourcen wie möglich. **3 P**

Um das knappe Energie-Budget zu erfüllen, überlegt sich der Entwickler des Systems, statt 2 CPUs nur eine einzusetzen und beide Threads darauf auszuführen. Dabei sind ihm die Verarbeitung und Übertragung der Daten wichtiger als deren Erfassung und er entscheidet sich, Thread C eine höhere Priorität zu geben als P, so dass immer wenn C ausführungsbereit ist, C ausgeführt wird.

b) Welches Problem tritt dabei auf? **1 P**

c) Schlagen Sie eine Lösung mit Hilfe von Semaphoren vor! Verwenden Sie dazu die Primitive `sem_down()` und `sem_up()` und wiederum möglichst wenige Ressourcen. Gehen Sie zur Vereinfachung davon aus, dass Thread C, der eine höhere Priorität hat, zuerst gestartet wird und P erst läuft, wenn dieser blockiert. **3 P**