Name: Matrikel-Nr.: Studiengang:

Klausur Betriebssysteme und Sicherheit, 21.02.2018

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

Alle Aussagen sind so ausführlich wie nötig, aber so knapp wie möglich zu begründen.

Aufgabe 1 – Erzeuger/Verbraucher-Problem

4 Punkte

In einem C++-Programm gibt es 2 Threads: Ein Thread erzeugt neue Elemente vom Typ `ele_t` und schreibt sie in einen gemeinsamen 10-elementigen Puffer. Der andere Thread liest diese Elemente und verarbeitet sie weiter.

a) Vervollständigen Sie die Lücken im unten stehenden Programm, so dass die 2 Threads als Erzeuger bzw. Verbraucher auf den gemeinsamen Ringpuffer zugreifen. **3 P**

Hinweis: Die Klasse `sem_t` implementiert einen normalen zählenden Semaphor. Der zur Initialisierung übergebene Wert entspricht dem Anfangswert des internen Zählers. Des Weiteren besitzt die Klasse die Funktionen `down` und `up`, die entsprechend der gängigen Funktionsweise eines Semaphors arbeiten.

Globale Variablen

```
1 ele_t buffer[10];
2 sem_t consumer(    ), producer(    );
3 int next_free = 0, next_full = 0;
```

Erzeuger

```
1 while (1) {
2     
3     ele_t ele = produce();
4     buffer[next_free] = ele;
5     next_free = (next_free + 1) % 10;
6     
7 }
```

Verbraucher

```
1 while (1) {
2     
3     consume(buffer[next_full]);
4     next_full = (next_full + 1) % 10;
5     
6 }
```

b) Zur Verbesserung der Skalierbarkeit des Programms diskutieren die Programmierer mehrere Erzeuger zu nutzen. Nennen Sie die Erweiterungen am oben stehenden Code, die nötig wären, damit zum einen mehrere Erzeuger verwendet werden können und zum anderen eine möglichst hohe Parallelität erreicht wird. **1 P**

Aufgabe 2 – Scheduling

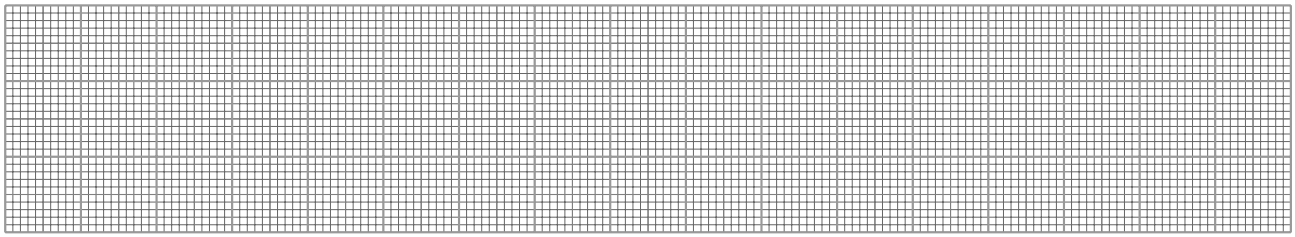
6 Punkte

Ein Fahrassistenzsystem mit einem Prozessor führt periodisch vier Tasks aus: Szenendekomposition A, Objektklassifizierung B, Handlungsplanung und -adaption C sowie die Fahrzeugsteuerung D. Die nebenstehender Tabelle gibt die Perioden p und Ausführungszeiten e der genannten Tasks an. Die relative Deadline d ist dabei gleich der Periodenlänge ($d = p$). Alle Tasks werden als voneinander unabhängig betrachtet und laufende Jobs sind zu beliebiger Zeit unterbrechbar.

	A	B	C	D
p	30	30	50	15
e	6	6	14	3

a) Ist diese Taskmenge mittels statischer Prioritätszuteilung einplanbar? Geben Sie in diesem Fall die Prioritäten der einzelnen Tasks an. Begründen Sie andernfalls, warum es mit statischer Prioriätszuteilung keinen gültigen Ablaufplan geben kann.

HINWEIS: Sie können $\sqrt[4]{2} < 1{,}2$ annehmen. **3 P**



b) Geben Sie sowohl die Berechnungsvorschrift der Hyperperiode an, als auch die konkrete Hyperperiode für die genannte Taskmenge. Beschreiben Sie, was die Hyperperiode angibt und wozu diese benötigt wird. **1 P**

Die Objektklassifizierung B des oben beschriebenen Systems sei unterteilt in Fahrzeugerkennung F, Hinderniserkennung H, Personenerkennung P, Verkehrszeichen- und -signalerkennung V. Um genügend Leistungsreserven zu haben, soll die Objektklassifizierung auf einen Beschleuniger mit zwei Verarbeitungseinheiten ausgelagert werden. Der Nachteil ist, dass auf dem Beschleuniger keine Kontextwechsel stattfinden können; Jobs also nicht mehr unterbrechbar sind. Die Jobs benötigen folgende Bearbeitungszeiten: F $\rightarrow$ 5, H $\rightarrow$ 2, P $\rightarrow$ 2, V $\rightarrow$ 1

c) Um die eingangs genannte Ausführungszeit von 6 Zeiteinheiten einzuhalten, soll die Gesamtbearbeitungszeit für Task B minimiert werden. Welcher aus der Vorlesung bekannte Algorithmus eignet sich hierfür? Erreicht der Algorithmus dieses Ziel im vorliegenden Fall? Ist das auch für beliebige Taskmengen der Fall? **2 P**

Aufgabe 3 – Kryptografie **7 Punkte**

Kevin, Stuart und Bob möchten jeweils paarweise vertraulich miteinander kommunizieren. Das heißt beispielsweise, Stuart soll einen Nachrichtenaustausch zwischen Bob und Kevin nicht mitlesen können.

a) Nennen Sie (ohne Begründung) ein geeignetes kryptographisches Verfahren. **1 P**

b) Wir betrachten nun die Schlüssel, die für die Verwendung des von Ihnen in a) gewählten Verfahrens benötigt werden: Wer erzeugt welche(n) Schlüssel? Ist beim Schlüsselaustausch untereinander etwas zu beachten? Welcher Benutzer besitzt am Ende welche(n) Schlüssel? **3 P**

c) Beschreiben Sie kurz einen möglichen Angreifer, gegen den das von Ihnen in a) gewählte Verfahren die Vertraulichkeit der Kommunikation *nicht* schützt. **1 P**

Zufällig stößt Bob auf den Dienst *WhosGotMail* und schlägt vor, diesen zu nutzen. Der Anbieter des Dienstes stellt ein ausführbares Programm bereit und wirbt damit, übertragene Nachrichten mittels eines „neuartigen, eigens entwickelten Verschlüsselungsverfahrens“ zu schützen. Die Kommunikation erfolgt stets über den „Hochsicherheitsserver“ des Anbieters, der die Nachrichten auch auf gefährliche Inhalte (bspw. Adressen von Webseiten, die bekanntermaßen Schadprogramme verbreiten) überprüft.

d) Kevin ist skeptisch und möchte den Dienst *WhosGotMail* nicht verwenden. Hat er Recht? Dann zeigen Sie zwei Probleme des beschriebenen Dienstes auf. Andernfalls, nennen Sie zwei Gründe, warum die drei *WhosGotMail* für ihre vertrauliche Kommunikation nutzen sollten. **2 P**

Aufgabe 4 – Unix

7 Punkte

Scarlet, die Nutzerin eines Unix-Systems, hat ein Programm `movefile` zum Verschieben von Dateien geschrieben. Hauptbestandteil des Programms ist diese Funktion:

```
1  void movefile(const char *source, const char *target)
2  {
3     char buffer[8];
4     int bytes;
5
6     // öffnet Datei "source" zum Lesen
7     int fd1 = open(source, O_RDONLY);
8
9     // erstellt Datei "target" mit Rechten rw-r--r-- und öffnet sie zum Schreiben
10    int fd2 = open(target, O_WRONLY | O_CREAT, 0644);
11
12    unlink(source);
13
14    do {
15       bytes = read(fd1, buffer, 8);  // liest 8 Byte aus fd1 in den Puffer
16       write(fd2, buffer, bytes);     // schreibt den Puffer in fd2
17    } while (bytes > 0);              // Abbruch, wenn fd1 vollständig gelesen
18
19    close(fd1);
20    close(fd2);
21 }
```

a) Warum ist die Datei `source` nach Zeile 12 noch lesbar? Wann wird der Inhalt der Datei tatsächlich gelöscht? **2 P**

b) Welches Problem kann auftreten, wenn ein Benutzer das Programm `movefile` während der Ausführung abbricht? **1 P**

c) Das gegebene Programm arbeitet ineffizient. Schlagen Sie eine Änderung zur Verbesserung der Effizienz vor. **1 P**

d) Scarlet führt das Programm `movefile` aus, um eine Datei mit folgenden Rechten zu verschieben:

```
--- r-- r--   scarlet   users
```

Scarlet ist dabei Mitglied der Gruppe `users`. Zu welchem Ergebnis wird die Ausführung von `movefile` führen? **2 P**

e) Der Systemaufruf `read` zum Lesen aus der Datei `source` wird im Betriebssystem an den Festplattentreiber weitergereicht, welcher wiederum die Festplatte anweist, die entsprechenden Datenblöcke asynchron zu lesen. Welche Hardwarefunktionalität nutzt die Festplatte, um dem Treiber die Fertigstellung einer Leseoperation zu signalisieren? **1 P**

Aufgabe 5 – Dateisysteme **7 Punkte**

Eine Raumsonde ist erfolgreich auf dem Jupiter-Mond *Europa* gelandet. Um die Instrumente für die Erforschung der Oberfläche in Betrieb zu nehmen, muss die derzeit installierte Missionssoftware `moon-lander` durch die neue Version `ice-explorer` ersetzt werden. Dazu wird die neue Software über die Empfangsantenne heruntergeladen, im Dateisystem des Steuercomputers abgelegt und der Steuercomputer anschließend neu gestartet.

a) Aufgrund kosmischer Strahlung kann der Steuercomputer jederzeit abstürzen. In diesem Fall muss die installierte Missionssoftware neu gestartet werden. Erläutern Sie, warum die folgende Update-Routine in `moon-lander` den Erfolg der Mission gefährdet. **1 P**

```
1 // öffnet Programmdatei "installed-mission-software" zum Schreiben
2 int fd = open("installed-mission-software", O_WRONLY, 0644);
3 // download_buffer enthält vom Kontrollzentrum gesendetes Update "ice-explorer"
4 write(fd, download_buffer, download_size);
5 // Setzt Größe der Programmdatei auf Größe des Downloads
6 ftruncate(fd, download_size);
7 close(fd);
8 reboot();
```

b) Aus Sicherheitsgründen haben die Programmierer der Sonde entschieden, dass die neue Missionssoftware `ice-explorer` als separate Programmdatei neben `moon-lander` abgelegt wird. Aktuell ist nur `moon-lander` installiert. Ergänzen Sie im nachfolgendem Schema die Metadaten für die 23.000 Blöcke große Missionssoftware `ice-explorer` *nachdem* diese erfolgreich geschrieben wurde.

HINWEIS: Es handelt sich um ein einfaches Unix-Dateisystem. Metadaten werden ausschließlich in den Blöcke 0 – 3 gemäß dem vorgegebenen Schema abgespeichert. Nicht genannte Bits der Allokations-Bitmaps in Block 0 und Block 2 sind `0`. Datenblöcke für Dateiinhalte werden pro Datei am Stück allokiert. Damit genügt in jedem Inode ein einziger Blockzeiger auf den ersten Block und die Anzahl insgesamt verwendeter Blöcke. Es gibt nur ein Verzeichnis und auf einen Superblock wurde verzichtet. **3 P**

Block 0 (IA)	Block 1 (I)	Block 2 (BA)	Block 3 (V)	Blöcke 4 . . .
Inode-Bitmap	Inode-Block	Block-Bitmap	Verzeichnis-Block	(Daten bzw. frei)
0,1: 1	$I_0 \rightarrow 3{,}1$ $I_1 \rightarrow 4{,}500$	0 – 3 : 1 4 – 503 : 1	`"moon-lander"` $\rightarrow I_1$	

c) Als Speichermedium für das Dateisystem wurde im Steuercomputer Flash-Speicher verbaut, der nur begrenzt oft beschrieben werden kann. Welches in der Vorlesung besprochene Konsistenzverfahren für Dateisysteme kann das in der Missionssoftware enthaltene Betriebssystem einsetzen, damit die Abnutzung des Flash-Speichers minimiert wird, wenn zukünftig regelmäßig Messdaten der Instrumente zwischengespeichert werden sollen? Nennen Sie ein weiteres Verfahren, welches sich schlechter eignet und begründen Sie warum das so ist. **3 P**

Aufgabe 6 – Speicherverwaltung **9 Punkte**

Gegeben sei eine 32-Bit Architektur mit virtuellem, byteadressierbarem Speicher und zweistufigen Seitentabellen. Das Seitenverzeichnis (SV) und die Seitentabellen (ST) enthalten jeweils 1024 Einträge, welche jeweils 4 Byte groß sind. Der Offset beträgt 12 Bit.

a) Wie groß ist der virtuelle Adressraum? Wie groß ist eine Seite? **1 P**

Im folgenden sei eine Seitentabellenstruktur gegeben. Alle nicht genannten oder frei gelassenen Einträge werden als ungültig angenommen.

SV: `0x1000`

`0x0`	`0x8007`
...	
`0x300`	`0x4007`
...	

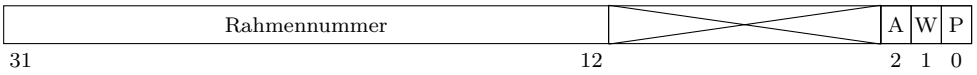
ST: `0x4000`

...	
`0x100`	`0xFFFF0007`
...	
`0x2FF`	`0xB1F006`
...	

ST: `0x8000`

...	
`0x20`	`0xC0100004`
`0x21`	`0xA005`
...	
`0x2FF`	`0xB1F007`
...	
`0x3FF`	`0x7AC003`

Seitentabellen- und Seitenverzeichnisseinträge haben dabei folgenden Aufbau:



P . . . Present-Bit
W . . . Writable-Bit
A . . . Accessed-Bit

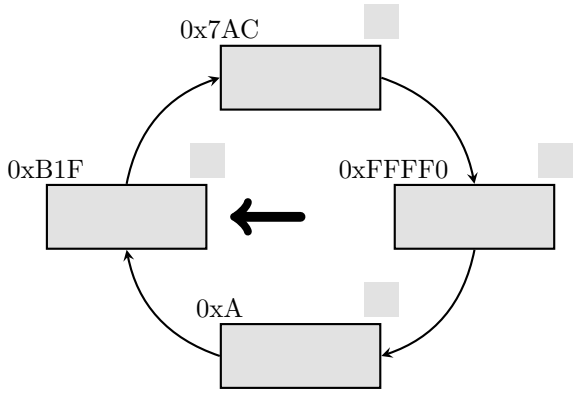
b) Nennen Sie die *virtuellen Adressen*, die laut obiger Seitentabellen auf folgende *physische Adressen* abgebildet werden: **4 P**

- `0xAFFE`:
- `0xFFFF0124`:
- `0x7ACE01`:
- `0xB1FABC`:

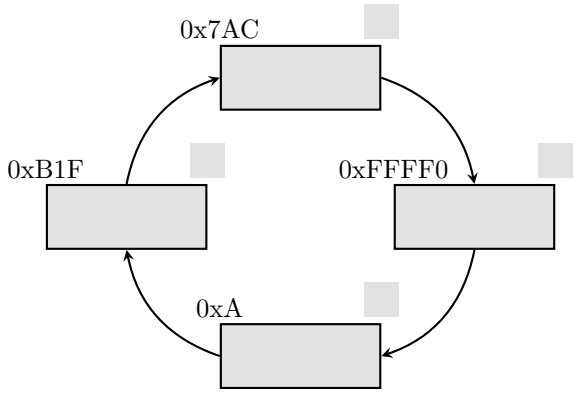
c) Für die in b) verwendeten vier Rahmen kommt der Clock-Algorithmus als Seitenersetzungsstrategie zum Einsatz. Ergänzen Sie in den unten stehenden Schemata jeweils die Seiten, die den genannten Rahmen zugeordnet sind, die Accessed-Bits sowie den Clock-Zeiger, so dass die folgenden Situationen korrekt wiedergegeben werden.

HINWEIS: Sie können die Seitennummer, also die virtuellen Adressen ohne den Offset, anstelle der kompletten virtuellen Adresse angeben. Alle auftretenden Speicherzugriffe sind zulässig. **4 P**

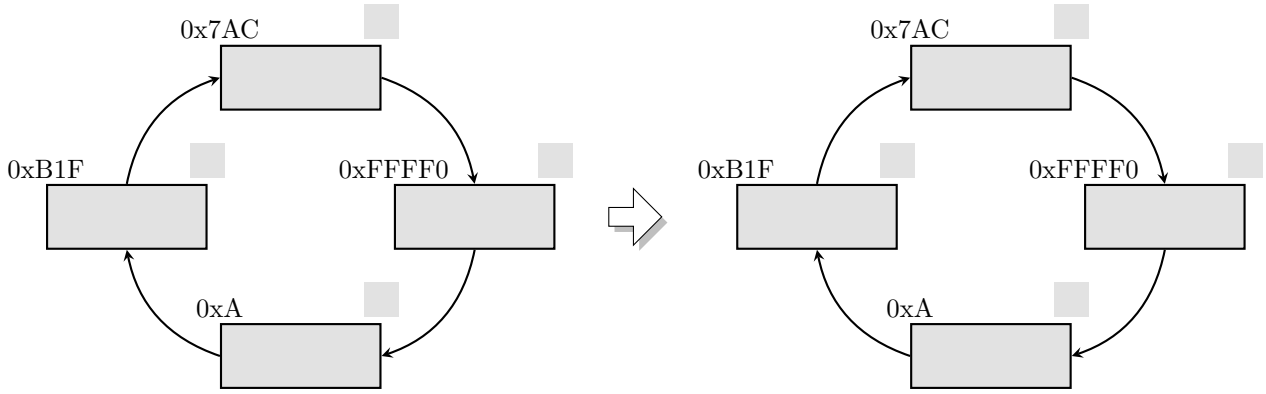
I. Tragen Sie zunächst die in b) vorliegende Ausgangssituation ein.



II. Nun erfolgt ein schreibender Zugriff auf die virtuelle Adresse `0x40042`. Tragen Sie den Zustand des Clock-Algorithmus nach diesem Zugriff ein.



III. Nun findet ein lesender Zugriff auf die virtuelle Adresse `0x2FF123` statt, gefolgt von einem lesenden Zugriff auf die virtuelle Adresse `0x3FF137`. Ergänzen Sie wiederum die Darstellung des Clock-Algorithmus.



Aufgabe 7 – Wechselseitiger Ausschluss **2 Punkte**

In einem Programm mit 2 Threads werden die unten folgenden Codeteile zur Absicherung eines kritischen Abschnitts verwendet. Dabei bezeichnet `/* kA */` den kritischen Abschnitt, für dessen Ausführung die Kriterien des wechselseitigen Ausschlusses garantiert werden soll.

Globale Variablen

```
1 int next = 1;
2 int t1 = 0, t2 = 0;
```

Thread 1

```
1 t1 = 1;
2 next = 2;
3 while(t2 == 1 && next == 1) {}
4 /*
5    kA
6 */
7 t1 = 0;
```

Thread 2

```
1 t2 = 1;
2 next = 1;
3 while (t1 == 1 && next == 2) {}
4 /*
5    kA
6 */
7 t2 = 0;
```

a) Bei einem Codereview wird festgestellt, dass der verwendete Code wechselseitigen Ausschluss nicht gewährleistet. Begründen Sie wieso der Reviewer Recht hat. **1 P**

b) Schlagen Sie eine Änderung des oben stehenden Codes vor, so dass das gefundene Problem behoben wird und der Code die Anforderungen an wechselseitigen Ausschluss erfüllt. **1 P**