Name: Matrikel-Nr.: Studiengang:

Klausur Betriebssysteme und Sicherheit, 22.02.2017

1	2	3	4	5	6	$\sum$
8	7	7	7	6	7	42

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

Aufgabe 1 – Seitenersetzung

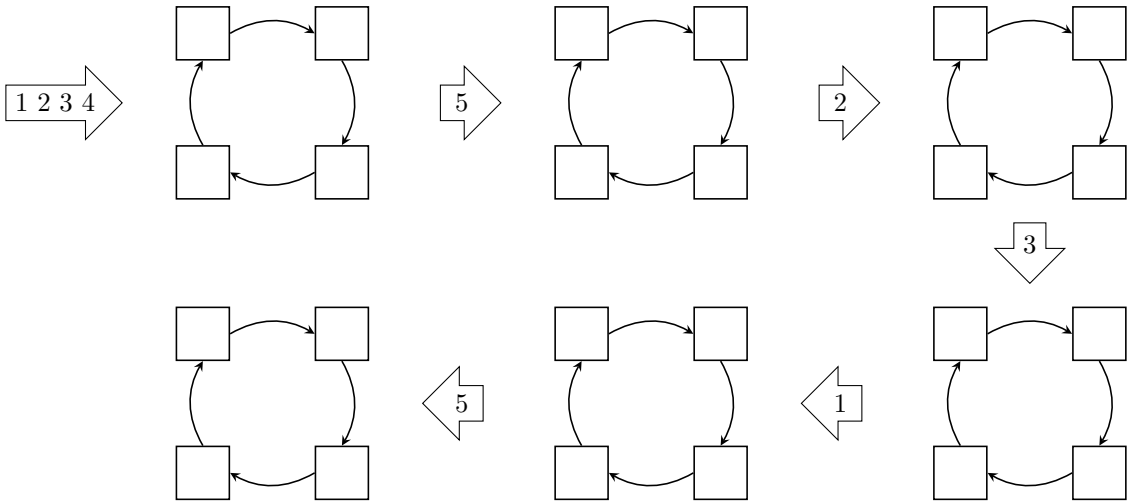
8 Punkte

In einem System mit 4 Rahmen erfolgen Zugriffe auf die Seiten mit den folgenden Seitennummern:

1 2 3 4 5 2 3 1 5

Der Hauptspeicher wird mit der Verdrängungsstrategie *Clock* verwaltet. Zu Beginn sind alle Rahmen leer.

a) Tragen Sie den Zustand der Speicherverwaltung jeweils nach den angegebenen Seitenzugriffen ein. **6 P**



b) Geben Sie an, welche der folgenden Schritte der Speicherverwaltung typischerweise von Hardware und welche von Software ausgeführt werden: **2 P**

1. Auslösen der Seitenfehlerbehandlung
2. Weiterrücken des *Clock*-Zeigers
3. Setzen der Referenzbits
4. Löschen der Referenzbits

Aufgabe 2 – Virtueller Speicher **7 Punkte**

Gegeben seien untenstehende Speicherverwaltungsstrukturen eines 32-bit Systems mit zweistufigen Seitentabellen. Es werden 10 Bit für die Indizierung in das Seitenverzeichnis (SV) verwendet, sowie 10 Bit für die Indizierung in die Seitentabelle (ST). Der Offset beträgt 12 Bit.

SV: `0x1000`

0x0	0x01	w
0x1	0x76	wp
...		
0xBF	0x3F	wp
...		
0x2FC	0x76	wp
...		
0x2FF	0x4F	wp
...		

ST: `0x76000`

0x0	0x0	wp
0x1	0x0	p
...		
0x3FE	0xFFE	wp
0x3FF	0xFFE	wp

ST: `0x3F000`

0x0	0x123	
0x1	0x0	p
...		
0x3FE	0x2A	
0x3FF	0xFFC	wp

ST: `0x4F000`

0x0	0x0	wp
0x1	0x0	p
...		
0x3FE	0x1A	
0x3FF	0xFFD	wp

a) Geben Sie, sofern möglich, die physischen Adressen der lesenden Zugriffe auf folgende virtuelle Adressen an. Falls ein Seitenfehler auftritt, nennen Sie den Grund dafür. **3 P**

- `0x400FFD`:
- `0x760DE`:
- `0xBFFFF002`:
- `0x2FC00000`:

b) Skizzieren Sie in groben Zügen, was passiert, wenn ein schreibender Zugriff auf die virtuelle Adresse `0xBFFFEFF8` erfolgt? Der Stack reicht aktuell von `0xBFFFF000` bis `0xBFFFFFFF` und wächst nach unten. **1 P**

Ein 64-bit System verwendet vierstufige Seitentabellen zur Adressierung des kompletten virtuellen Adressraums. Die Tabellen der verschiedenen Stufen enthalten jeweils 4096 Einträge und füllen jeweils einen vollständigen Rahmen.

c) Wie viele Bits der Adresse bilden den Offset? **1 P**

d) Wie groß ist eine Seite? **1 P**

e) Wie groß ist ein einzelner Seitentabelleneintrag? **1 P**

Aufgabe 3 – Scheduling

7 Punkte

HINWEIS: Am Ende der Seite steht ein karierter Bereich zum Zeichnen von Ablaufplänen *für alle Teilaufgaben* zur Verfügung. Bitte notieren Sie dabei jeweils zu welcher Teilaufgabe die Zeichnung gehört.

In einem Echtzeitsystem sind vier Tasks einzuplanen. Sie sind gegeben durch ihre Periodenlänge p und Ausführungszeit e in der Form $T : (p, e)$. Die relative Deadline d ist dabei gleich der Periodenlänge ($d = p$).

$$A : (6, 1) \qquad B : (8, 2) \qquad C : (4, 1) \qquad D : (12, 3)$$

Zur Ausführung steht genau ein Prozessor zur Verfügung. Alle Tasks sind zum Zeitpunkt $t = 0$ bereit, voneinander unabhängig und jederzeit unterbrechbar.

a) Welcher Task ist zum Zeitpunkt 484,8 aktiv, wenn die Einplanung mittels EDF erfolgt? **2 P**

Gegeben sei die Präzedenzrelation $\{(A,C), (A,E), (B,E), (B,D), (C,G), (D,H), (E,F), (F,G), (F,H)\}$ zwischen acht, jeweils einmalig auszuführenden Jobs $A,\ldots,H$. Die Jobs sind *nicht* unterbrechbar und erfordern folgende Bedienungszeiten:

A	B	C	D	E	F	G	H
1	2	4	3	2	4	1	2

Zur Bearbeitung stehen zwei gleichartige Prozessoren zur Verfügung.

b) Zeichnen Sie den Präzedenzgraphen als Vorgangspfeilnetz. **2 P**

c) Stellen Sie einen Ablaufplan (Gantt-Diagramm) gemäß LPT auf. **1 P**

d) Ist der LPT-Ablaufplan unter den gegebenen Voraussetzungen bereits optimal bezüglich der Gesamtbearbeitungszeit? Begründen Sie in diesem Fall, warum keine weitere Verkürzung möglich ist. Andernfalls, geben Sie die geringstmögliche Gesamtbearbeitungszeit sowie einen zugehörigen Ablaufplan an. **2 P**

Aufgabe 4 – Sicherheit **7 Punkte**

In einem System existieren die fünf Benutzer *Alice*, *Bob*, *Carol*, *Dave* und *Oskar*. Alice und Bob möchten auf einfache Weise miteinander kommunizieren und nutzen für diesen Zweck eine Datei `postfach`. Möchte bspw. Alice eine Nachricht für Bob hinterlassen, legt sie diese in der Datei ab. Zu einem späteren Zeitpunkt liest Bob die Nachricht und löscht sie bzw. ersetzt sie durch seine Antwort. Um ihre Kommunikation vertraulich zu halten soll die Datei `postfach` dabei *ausschließlich* für Alice und Bob zugreifbar sein.

a) Geben Sie für die Datei `postfach` eine geeignete Rechtezuweisung mittels Zugriffssteuerlisten (ACL) an. **1 P**

b) Geben Sie eine mögliche Rechtezuweisung für `postfach` im Rahmen des klassischen Unix-Rechtemodells an. Welche zusätzlichen Voraussetzungen sind dabei erforderlich? **3 P**

c) Die Datei `postfach` wurde in einem Verzeichnis `public_dir` angelegt, das folgende Rechtezuweisung besitzt

```
rwx rwx rwx   root   root   public_dir
```

Wie könnte *Oskar* diese Situation ausnutzen, um die Kommunikation zwischen *Alice* und *Bob* in Zukunft mitzulesen?

HINWEIS: Wir gehen davon aus, dass *Alice* und *Bob* arglos sind und nicht mit einem Angriff rechnen. **1 P**

Ein zentraler Systemdienst hat aufgrund eines Programmierfehlers eine Datei *system.d* mit den folgenden Unix-Rechten angelegt

```
rws rws rwx   root   root   system.d
```

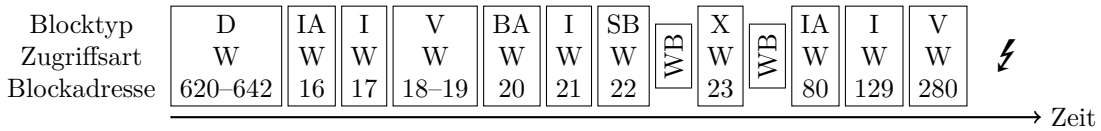
d) Erläutern Sie, warum dies ein Sicherheitsproblem darstellt. **2 P**

Aufgabe 5 – Dateisysteme

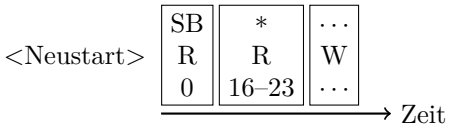
6 Punkte

Ein experimentelles Betriebssystem stellt zur Speicherung persistenter Daten ein Unix-artiges Dateisystem bereit, welches alle Dateiinhalte und Verwaltungsstrukturen in Blöcken auf dem Speichermedium organisiert. Jeder Block enthält entweder Daten (D) oder eine der üblichen Arten von Metadaten: Inodes (I), Verzeichniseinträge (V), Allokations-Bitmap für Inodes bzw. Blöcke (IA bzw. BA), Superblock (SB). Zusätzlich existiert noch eine weitere Art von dateisystemspezifischen Metadaten (X).

Das Betriebssystem führt die nachfolgend angegebene Sequenz von Schreiboperation (Zugriffsart W) für die angegebenen Blockadressen und einige Write-Barriers (Zugriffsart WB) aus. Unmittelbar danach kommt es zu einem Stromausfall (↯).



Nach dem Wiederherstellen der Stromversorgung startet das Betriebssystem neu und führt beim Wiedereinhängen des Dateisystems die nebenstehende Sequenz von Lesezugriffen (Zugriffsart R) aus, gefolgt von einigen Schreibzugriffen:



a) Die ausgeführten Schreiboperationen und Write-Barriers sowie die Lesezugriffe unmittelbar nach dem Neustart sind charakteristisch für Dateisysteme mit einem bestimmten, in der Vorlesung besprochenen Konsistenzverfahren. Um welches Verfahren handelt es sich? Wozu dient dabei der Metadaten-Block vom Typ X sowie die beiden Write-Barriers? **3 P**

Die konkrete Implementierung des Konsistenzverfahrens aus Teilaufgabe a) speichert in einem Block vom Typ X auch immer eine Liste, die Paare von Blockadressen enthält. Für die oben angegebene Sequenz von Schreiboperationen sei der Inhalt dieser Liste wie folgt:

(16, 80), (17, 129), (18, 280), (19, 281), (20, 66), (21, 130), (22, 0)

b) Welche wichtige Information kodiert diese Liste? Ergänzen Sie im nachfolgenden Schema die Schreiboperationen, die das Betriebssystem nach dem Neustart ausführen muss, um eventuelle Inkonsistenzen im Dateisystem zu beseitigen. **2 P**

c) Nennen Sie jeweils einen Vor- und einen Nachteil von „Soft Updates“ gegenüber anderen Verfahren zur Sicherstellung der Konsistenz von Dateisystemen. **1 P**

Aufgabe 6 – Synchronisation **7 Punkte**

Ein Programm mit zwei Threads T_1 und T_2 verwendet ein Lock um Zugriffe auf gemeinsam genutzte Daten zu serialisieren. Dazu werden zwei globale Variablen, mit der nebenstehenden initialen Belegung, verwendet.

```
int is_locked = 0;
int lock = 0;
```

Thread 1 schützt seine kritischen Abschnitte mit Hilfe der folgenden Funkionen `lock_T1` und `unlock`. Dabei bezeichnet `cmpxchg(&var, alt, neu)` eine atomare Compare-and-Exchange-Operation, d. h. `var` und `alt` werden verglichen, und nur wenn sie gleich sind, wird der Wert von `neu` an `var` zugewiesen. In jedem Fall gibt die Funktion den alten Wert von `var` zurück.

```
1 void lock_T1() {
2   is_locked = 1;
3   while (true) {
4     if (cmpxchg(&lock, 0, 1) == 0) {
5       break;
6     }
7   }
8 }
```

```
1 void unlock() {
2   lock = 0;
3   is_locked = 0;
4 }
```

Thread 2 verwendet in jedem Fall die obenstehende Funktion `unlock`. Für `lock_T2` sind drei Varianten verfügbar.

a) Welche der angegebenen Implementierungen sollte für Thread T_2 verwendet werden? Beschreiben Sie für die beiden anderen Implementierungen kurz welches Problem sie aufweisen und skizzieren Sie einen Ablauf, der dieses Problem zeigt. **4 P**

```
1 void lock_T2_V1() {
2   while (lock == 1)
3     {}
4   cmpxchg(&lock, 0, 1);
5   is_locked = 1;
6 }
```

Variante 1

```
1 void lock_T2_V2() {
2   while (cmpxchg(&lock, 0, 1) == 1)
3     {}
4   is_locked = 1;
5 }
```

Variante 2

```
1 void lock_T2_V3() {
2   while (cmpxchg(&lock, 0, 1) == 1)
3     {}
4   while (is_locked == 1)
5     {}
6   is_locked = 1;
7 }
```

Variante 3

b) Nennen Sie zwei Nachteile (der korrekten Implementierung) des vorgestellten Locks. Wie könnte man diese lösen? **3 P**