Name: Matrikel-Nr.: Studiengang:

Klausur Betriebssysteme und Sicherheit, 04. 03. 2020

1	2	3	4	5	6	7	$\sum$
12	11	13	13	13	14	14	90

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

Aufgabe 1 – Seitenersetzung

12 Punkte

a) Was ist das zugrunde liegende Problem, das Seitenersetzungsverfahren lösen? **1 P**

Wir betrachten nun ein System welches über 5 Rahmen verfügt und diese mit Hilfe des Arbeitsmengenmodells verwaltet. Dabei nutzt das System eine Fenstergröße von 2 Referenzen. Im System existieren 3 Prozesse (A,B,C), von denen jeder bisher auf genau eine Seite zugegriffen hat (siehe vorgegebene Spalte „Init“).

b) Vervollständigen Sie die Zuordnung der Seiten zu Rahmen in der folgenden Tabelle. Tragen Sie dazu in jeder Spalte ein, in welchem Rahmen die entsprechende Seite liegt bzw. in welchen sie eingelagert wird. Markieren Sie Seitenfehler durch ein Kreuz in der Zeile *PF*. **9 P**

Rahmen	Init	A:2	A:2	B:2	B:2	A:3	C:1	A:4	C:1	B:4
1	A:1									
2	B:4									
3	C:3									
4										
5										
PF										

c) Ein anderes häufig verwendetes Seitenersetzungsverfahren ist der Clock-Algorithmus. Nennen sie jeweils einen Vor- und Nachteil von Clock gegenüber dem Arbeitsmengenmodell. **2 P**

Aufgabe 2 – Scheduling **11 Punkte**

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

T_1: $p_1 = 6, e_1 = 2$ T_2: $p_2 = 16, e_2 = 2$ T_3: $p_3 = 4, e_3 = 1$

Alle Tasks starten zum Zeitpunkt $t = 0$ und können an beliebiger Stelle unterbrochen werden. Es stehe genau ein Prozessor zur Verfügung und der Scheduling-Overhead werde vernachlässigt.

a) Zunächst werden den Tasks statisch Prioritäten zugeteilt, so dass T_1 die höchste, T_2 eine mittlere und T_3 die niedrigste Priorität erhält. Weisen Sie nach, dass die gegebene Task-Menge unter den genannten Bedingungen nicht einplanbar ist. **3 P**

b) Ist eine Einplanung dieser Task-Menge mit statischen Prioritäten überhaupt möglich? Begründen Sie Ihre Antwort. Geben Sie im Fall einer erfolgreichen Einplanung eine entsprechende Prioritätszuteilung an.

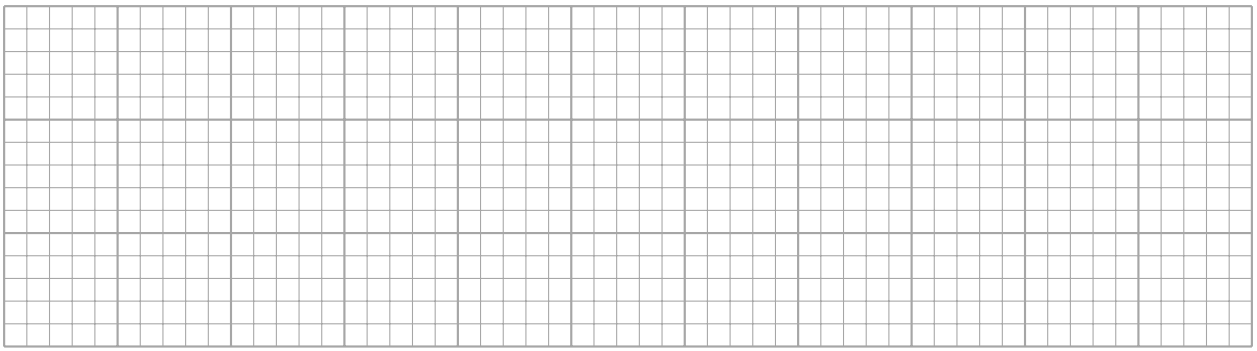
HINWEIS: Sie können folgende Werte annehmen: $\sqrt{2} < 1{,}42$ und $\sqrt[3]{2} < 1{,}26$ **4 P**

c) Die Task T_1 soll häufiger ausgeführt werden. Bis zu welcher Periode p_1 ist eine Einplanung mit *dynamischen Prioritäten* möglich? **3 P**

d) Eine der folgenden Aussagen trifft zu. Kreuzen Sie sie an. **1 P**

☐ Es gibt Taskmengen, die mit statischen Prioritäten einplanbar sind, aber nicht mit dynamischen.

☐ Es gibt Taskmengen, die mit dynamischen Prioritäten einplanbar sind, aber nicht mit statischen.



Aufgabe 3 – Virtueller Speicher **13 Punkte**

Gegeben sei ein 32-Bit-System mit zweistufigen Seitentabellen. Die erste Stufe wird Seitenverzeichnis (SV) genannt und umfasst 2048 Einträge. Seitentabellen der zweiten Stufe (ST), haben 512 Einträge. Das Seitenoffset beträgt 12 Bit.

a) Wie viele Bits werden für die Indizierung des Seitenverzeichnisses bzw. der Seitentabelle jeweils benötigt? **2 P**

Ein Prozess verwendet die folgenden Seitentabellen. Alle nicht aufgeführten Einträge werden dabei als ungültig („not present“) angenommen. Der Einsprungpunkt für die Adressübersetzung, also das Seitenverzeichnis, liegt an Adresse `0xFFE8 0000`.

Addr.: `0xFFE8 0000`

0x006	0xFFEC3	pw
0x60D	0xFFECA	pw
0x6F5	0xFFED8	p

Addr.: `0xFFEC 3000`

0x00F	0x00000	
0x01A	0x02902	w
0x0A1	0x02902	pw

Addr.: `0xFFEC A000`

0x002	0x0CAFE	p
0x011	0x0C0CE	pw
0x123	0x0C0C5	p

Addr.: `0xFFED 8000`

0x0DB	0x0BADB	p
0x0CA	0xCACA0	w
0x1FF	0xFFFFF	

b) Welches Ergebnis haben die folgenden Zugriffe? Dokumentieren Sie die Zwischenschritte der Adressumsetzung, indem Sie die von Ihnen verwendeten Tabellenindizes (SV und ST) notieren. **9 P**

- Lesezugriff auf `0xDEAD BABE` ⇒ SV: ST: Ergebnis:

- Schreibzugriff auf `0x00C0 FFEE` ⇒ SV: ST: Ergebnis:

- Schreibzugriff auf `0x00C1 A115` ⇒ SV: ST: Ergebnis:

c) Nennen Sie je einen Vor- und Nachteil invertierter Seitentabellen gegenüber hierarchischen Seitentabellen. **2 P**

Aufgabe 4 – Synchronisation

13 Punkte

In einem Programm mit zwei Threads werden die zwei nebenstehenden Funktionen `foo` und `bar` jeweils einmal aufgerufen. Die globale Variable `x` ist dabei definiert mit `int x = 1`. Weitere schreibende Zugriffe auf `x` gibt es im Programm nicht.

```
int x = 1;
```

```
1 void foo() {
2     x = x * 2;
3 }
```

```
1 void bar() {
2     x = 3;
3 }
```

a) Nennen Sie *alle* möglichen Ergebnisse bei sequentieller Ausführung. **1 P**

b) Beschreiben Sie eine (parallele) Ausführung, die zum Ergebnis `x == 2` führt. **4 P**

Die Aufrufe von `foo` und `bar` sollen synchronisiert werden, um das unerwünschte Ergebnis `x == 2` auszuschließen.

c) Als mögliche Optionen werden Spin-Locks und Semaphore diskutiert. Nennen Sie für beide jeweils einen Vorteil gegenüber dem anderen. Welchen der Mechanismen würde Sie im vorliegenden Beispiel einsetzen und warum? **4 P**

Die nebenstehende Implementierung wird vorgeschlagen. Die globale Variable `L` ist dabei definiert mit `int L = 1`.

d) Beschreiben Sie eine Ausführung mit zwei Threads, bei der diese Implementierung den wechselseitigen Ausschluss nicht gewährleistet. **4 P**

```
int L = 1;
```

```
1 void lock() {
2     while (L == 0) {}
3     L = 0;
4 }
```

```
1 void unlock() {
2     L = 1;
3 }
```

Aufgabe 5 – Unix

13 Punkte

```
1  int main() {
2     int pid = fork();
3     printf("A");
4
5     if (pid < 0) {
6        printf("F");
7        exit(1);
8     } else if (pid == 0) {
9        printf("B");
10       wait();
11       exit(0);
12    } else {
13       printf("C");
14       wait();
15    }
16
17    printf("D");
18    exit(0);
19 }
```

a) Wählen Sie aus den vorgegebenen Ausgaben alle aus, die bei Ausführung des nebenstehenden Unix-Programms entstehen können und notieren Sie die zugeordneten Buchstaben. **5 P**

Hinweis:

- Die Funktion `wait` blockiert den aufrufenden Prozess so lange, bis sich einer seiner Kindprozesse beendet. Existiert kein solcher, so kehrt die Funktion sofort zurück.
- Die Auswahl falscher Ausgaben führt zu Punktabzug innerhalb dieser Teilaufgabe!

(a) ABACD	(e) ACABD	(i) ACADB
(b) AF	(f) AFD	(j) AACBD
(c) AACCDD	(g) ABBA	(k) ACDAB
(d) AABCD	(h) AFAF	(l) ABABDD

```
1  int main() {
2     int fd, pid;
3     pid = fork();
4
5     // Öffnen der existierenden
6     // und leeren Datei 'buffer'
7     fd = open("buffer", O_WRITE);
8
9     if (pid == 0) {
10       write(fd, "foo", 3);
11    } else {
12       wait();
13       write(fd, "bar", 3);
14    }
15
16    return 0;
17 }
```

b) Welchen Inhalt hat die Datei `buffer` nach Beendigung des nebenstehenden Unix-Programms? Wenn es mehrere Möglichkeiten gibt, nennen Sie diese. Wenn nicht, erklären Sie kurz warum. **3 P**

c) Nennen Sie drei Möglichkeiten, neben der Nutzung von Dateien, für den Austausch größerer Datenmengen zwischen Prozessen unter Unix? **3 P**

d) Erklären Sie den Prozesszustand *Zombie* und wofür er benötigt wird. **2 P**

Aufgabe 6 – Dateisysteme **14 Punkte**

a) In Unix-Dateisystemen gibt es „harte“ Verknüpfungen (hard links) und „symbolische“ Verknüpfungen (symbolic links). Erklären Sie kurz wie diese im Dateisystem repräsentiert werden. **2 P**

b) Nennen Sie zwei konkrete Aufgaben, die der Buffer Cache in einem Betriebssystem erfüllt. **2 P**

Ein Unix-Dateisystem organisiert alle Dateiinhalte und Metadaten in Blöcken auf dem Speichermedium. Die Größe eines Blocks beträgt 4 KiByte. Inodes sind jeweils 128 Byte groß. Jeder Verzeichniseintrag belegt 32 Byte.

c) Für einen bestimmten Einsatzzweck genügt es, wenn das Dateisystem höchstens 250 Dateien bzw. Verzeichnisse gleichzeitig verwalten kann. Wie viele Blöcke für Inodes sind in diesem Fall nötig? **1 P**

Gegeben ist der nachfolgende Ausschnitt aus dem Dateisystemzustand auf einem Speichermedium. Die dargestellten Blöcke enthalten entweder Inodes (I), Verzeichniseinträge (V) oder eine Allokations-Bitmap für Blöcke (BA) bzw. Inodes (IA). Weitere Blöcke (z. B. für Dateiinhalte) existieren, sind aber nicht dargestellt.

Block	1	2	3	4	5
Typ	BA	IA	I	I	V
Inhalt	$0-5,$ $8, 9,$ $23-25$	$0,$ $1,$ 2	$I_0 \rightarrow V \mid 64 \mid 5$ $I_1 \rightarrow D \mid 7.042 \mid 8, 9$ $I_2 \rightarrow D \mid 9.042 \mid 23, 24, 25$		`"a.jpg"` $\rightarrow I_1$ `"b.tex"` $\rightarrow I_2$

Blöcke 0 bis 5, usw. belegt

Inodes 0, 1, 2 belegt

Inode → Typ | Größe (in Byte) | Blockzeiger

Dateiname → Inode

Die Datei `c.tmp` wird im selben Verzeichnis wie `a.jpg` und `b.tex` neu angelegt und bekommt dabei das Inode I_{42} in Block 4 zugewiesen. Dann werden 15.000 Byte Daten in die neue Datei geschrieben.

d) Verändern Sie die Metadaten im oben stehenden Schema durch Ergänzung / Korrektur / Streichung so, dass der endgültige Zustand nach dem Schreiben des Inhalts von `c.tmp` ensteht. **5 P**

Das Speichermedium besteht aus neuartigen *Blitz*-Speicherzellen und ist so schnell, dass das Konsistenzverfahren *Synchrones Schreiben* mit guter Schreibleistung eingesetzt werden kann.

e) Betrachtet wird das Anlegen von `c.tmp` und das anschließende Schreiben der Daten in diese Datei (siehe Teilaufgabe d)). Geben Sie, getrennt für beide Teiloperationen, die Reihenfolge gemäß *Synchronem Schreiben* an, in der alle neuen oder geänderten Blöcke geschrieben werden müssen. Vermerken Sie jeweils, wann eine *Write Barrier* (WB) erforderlich ist. **4 P**

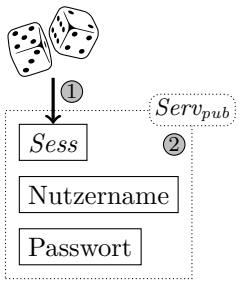
- Anlegen der leeren Datei:

- Schreiben der Daten:

Aufgabe 7 – Kryptographie und Sicherheit **14 Punkte**

Fletnix, ein Anbieter für den Verleih von Videos über das Internet, betreibt einen Server mit den Kunden- und Videodaten. Zur Nutzung des Dienstes stellt er seinen Kunden das ausführbare Programm *DRaMa* bereit. Verbindungen zum Server sollen stets sicher verschlüsselt erfolgen.

Aus Effizienzgründen kommt ein hybrides Verschlüsselungssystem zum Einsatz: Der öffentliche RSA-Schlüssel des Servers ($Serv_{pub}$) ist in *DRaMa* fest integriert. Beim Verbindungsaufbau erzeugt das Programm zunächst einen neuen, zufälligen Sitzungsschlüssel *Sess* ①. Anschließend werden die Anmeldeinformationen (Nutzername und Passwort) sowie *Sess* unter Verwendung von $Serv_{pub}$ verschlüsselt ② und an den Server gesandt. Der Server entschlüsselt die Nachricht und prüft die Anmeldeinformationen. Für alle nachfolgende Kommunikation wird *Sess* verwendet.



a) Nennen Sie einen geeigneten kryptografischen Algorithmus, der für die Verschlüsselung der Kommunikation mittels *Sess* eingesetzt werden kann. **1 P**

b) Welches Problem träte auf, wenn $Serv_{pub}$ nicht in *DRaMa* integriert wäre, sondern vor jeder Anmeldung vom Server geladen würde? **1 P**

c) Eva findet das beschriebene Verfahren viel zu aufwendig. Sie schlägt vor, stattdessen einmalig einen Schlüssel (*Serv*) für ein symmetrisches Verschlüsselungsverfahren zu generieren und diesen in *DRaMa* zu hinterlegen. *Serv* soll dann direkt für die Absicherung aller Kommunikation mit dem Server genutzt werden. Wie beurteilen Sie diesen Vorschlag? **2 P**

Wir betrachten nun eine konkrete Anmeldung (mit stark vereinfachten Werten). Der öffentliche RSA-Schlüssel $Serv_{pub}$ des Servers setze sich zusammen aus dem Modul $n = 91 = 7 \cdot 13$ und dem Exponenten $c = 59$. Einem Angreifer ist es gelungen den geheimen Exponenten des Servers ($d = 11$) zu bestimmen und die Anmeldenachricht $x' = 42$ abzufangen.

d) Weisen Sie nach, dass es sich bei d tatsächlich um den geheimen Exponenten des Servers handelt. **3 P**

e) Entschlüsseln Sie die abgefangene, verschlüsselte Nachricht $x' = 42$. **3 P**

Passwörter werden auf dem *Fletnix*-Server unter Verwendung einer kryptografischen Hash-Funktion gespeichert.

f) Was ist eine Hash-Funktion? Welche (zusätzliche) Eigenschaft muss eine *kryptografische* Hash-Funktion besitzen? Nennen Sie einen geeigneten Algorithmus. **3 P**

g) Warum werden kryptografische Hash-Funktionen bei der Speicherung von Passwörtern verwendet? **1 P**