Name: Matrikel-Nr.: Studiengang:

Klausur Betriebssysteme und Sicherheit, 16. 07. 2019

1	2	3	4	5	6	$\sum$
17	14	14	16	14	15	90

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

Aufgabe 1 – Betriebssystemgrundlagen **17 Punkte**

a) Nennen Sie zwei Möglichkeiten für die Kommunikation zwischen zwei Prozessen. **2 P**

b) Während Prozess P_1 läuft, wählt der Prozess-Scheduler Prozess P_2 zur Ausführung aus und möchte nun von P_1 zu P_2 umschalten. Wählen Sie die dazu nötigen Schritte aus und notieren Sie die zugeordneten Buchstaben in der korrekten Reihenfolge.

HINWEIS: Die Auswahl falscher Schritte führt zu Punktabzug innerhalb dieser Teilaufgabe! **6 P**

(a) Offene Dateien von P_1 schließen
(b) Prüfen, ob sich Kindprozesse von P_2 beendet haben
(c) Freien Rahmen finden/schaffen und als belegt markieren
(d) Hardwarezustand (Register, ...) herstellen
(e) Adressraum umschalten
(f) Scheduler-Informationen aktualisieren
(g) Dateisystem mit Festplatte synchronisieren (`sync`)
(h) Bankalgorithmus ausführen
(i) Hardwarezustand (Register, ...) sichern
(j) Prozessinformationen von P_2 finden

c) Nennen Sie eine Hardware-Funktionalität, die notwendig ist, um präemptives (aktiv unterbrochenes) Scheduling zu ermöglichen. Erläutern Sie kurz, warum diese Funktionalität erforderlich ist. **2 P**

d) Ein Programm greift schreibend auf eine virtuelle Adresse zu, für die die Memory Management Unit (MMU) keinen gültigen Seitentabelleneintrag findet („not present“). Wie reagiert die CPU auf diese Situation? **1 P**

e) Angenommen der Speicherzugriff aus Teilaufgabe d) war ein zulässiger Zugriff auf den Heap-Speicher der Anwendung. Welche Schritte muss das Betriebssystem durchführen, damit die Hardware den Zugriff erfolgreich ausführen kann? Wählen Sie die nötigen Schritte aus und notieren Sie die zugeordneten Buchstaben in der korrekten Reihenfolge.

HINWEIS: Die Auswahl falscher Schritte führt zu Punktabzug innerhalb dieser Teilaufgabe! **5 P**

(a) Bankalgorithmus ausführen	(e) Adressübersetzung durchführen
(b) Seitentabelle(n) aktualisieren	(f) Scheduler-Informationen aktualisieren
(c) Freien Rahmen finden/schaffen und als belegt markieren	(g) Programm fortsetzen, so dass der Zugriff erneut ausgeführt wird
(d) Limit-Register der CPU anpassen	(h) Fehleradresse aus Hardware auslesen

f) Wie würde ein Unix-System reagieren, wenn der Prozess einen Schreibzugriff auf ein nur-lesbares Segment des Speichers ausführte? **1 P**

Aufgabe 2 – Kryptographie und Sicherheit **14 Punkte**

Ein Chat-Dienst namens *myMessage* möchte seinen Nutzern sogenannte Ende-zu-Ende-Verschlüsselung bereitstellen. Das bedeutet, dass alle Nachrichten verschlüsselt und signiert über den Chat-Server des Dienstes versendet werden und selbst der Betreiber dieses Servers den Inhalt der Nachrichten nicht entschlüsseln kann.

Der Dienst verwaltet dafür auf einem Schlüssel-Server ein Verzeichnis von öffentlichen Schlüsseln (Pub_{Nutzer}) seiner Nutzer. Die entsprechenden privaten Schlüssel ($Priv_{Nutzer}$) werden vom jeweiligen Nutzer verwahrt und sind dem Betreiber nicht bekannt. Zusätzlich kennt jeder Nutzer den öffentlichen Schlüssel Pub_{Server} des Schlüssel-Servers.

a) Alice möchte Bob eine Nachricht über *myMessage* schicken. Die beiden sind bereits Nutzer des Dienstes, hatten aber bisher noch keinen Kontakt über *myMessage*. Sie kennen also keine Schlüssel vom jeweils anderen Teilnehmer.

Ergänzen Sie die Lücken im folgenden Austausch, um den Versand der ersten Nachricht durchzuführen. Kreuzen Sie Felder durch, die im jeweiligen Schritt nicht benutzt werden. **10 P**

Sender	verschlüsselt mit	signiert mit	Empfänger	entschlüsselt mit	prüft mit
Schritt 1: Alice fordert Bobs Schlüssel vom Schlüssel-Server an					
Alice	Pub_{Server}	$Priv_{Alice}$	Schlüssel-Server	$Priv_{Server}$	Pub_{Alice}
Schritt 2: Server sendet Bobs Schlüssel an Alice					
Schlüssel-Server			Alice		
Schritt 3: Alice sendet Nachricht an Bob					
Alice			Bob		
Schritt 4: Bob möchte die Integrität der Nachricht prüfen. Dazu					
Bob					
Schritt 5:					

b) Nennen Sie einen Algorithmus, der für die kryptografischen Operationen in *myMessage* verwendet werden kann. Handelt es sich um symmetrische oder asymmetrische Kryptografie? **2 P**

c) Skizzieren Sie kurz einen möglichen Angriff, den der Betreiber von *myMessage* trotz der beworbenen Ende-zu-Ende-Verschlüsselung durchführen kann. Wir gehen dabei davon aus, dass die Sicherheit der kryptografischen Algorithmen selbst nicht gebrochen wird. **2 P**

Aufgabe 3 – Dateisysteme **14 Punkte**

Ein Unix-Dateisystem organisiert alle Dateiinhalte und Metadaten in Blöcken auf dem Speichermedium. Die Größe eines Blocks beträgt 4 KiByte.

a) Was ist die maximale Größe des Dateisystems, wenn die Blockzeiger 31 Bit lang sind? **1 P**

b) Nennen Sie zwei Arten von Metadaten, die bei UNIX-Dateisystemen im Inode gespeichert werden. Begründen Sie, warum diese Arten von Metadaten nicht in einer anderen Dateisystemstruktur (z.B. Superblock) abgelegt werden können. **3 P**

Gegeben ist der nachfolgende Ausschnitt aus dem Dateisystemzustand auf einem USB-Stick. Die dargestellten Blöcke enthalten entweder den Superblock (SB), 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	0	1	2	3	4	...
Typ	SB	BA	IA	I	V	...
Inhalt	/ → 0	0 – 9	0,1,2	$I_0 \rightarrow$ Verz. | 32 | 4 $I_1 \rightarrow$ Datei | 7.042 | 5, 6 $I_2 \rightarrow$ Datei | 10.133 | 7, 8, 9 $I_3 \rightarrow$ Datei | 420 | 10	`"scan.jpg"` → I_1 `"secret.key"` → I_3	...

Inode-Verweis für Wurzelverzeichnis

Blöcke 0 bis 9 belegt

Inodes 0 bis 2 belegt

Inode → Typ | Größe (in Byte) | Blockzeiger

Dateiname → Inode

c) Die im Schema dargestellten Metadatenstrukturen sind inkonsistent, da der Nutzer den USB-Stick während eines Schreibvorgangs abgezogen hat. Worin genau bestehen die vorhandenen Inkonsistenzen? Erläutern Sie, welche Probleme jeweils auftreten können, wenn das Betriebssystem diese Datenstrukturen ohne vorherige Reparatur für zukünftige Lese- und Schreiboperationen konsultiert und/oder verändert. **6 P**

d) Auf Anraten einer befreundeten Informatikerin stellt der Nutzer die Schreibstrategie des Betriebssystems für externe Speichergeräte auf *Synchrones Schreiben*. Welche der in Teilaufgabe c) gezeigten Inkonsistenzen könnten auch zukünftig wieder auftreten, wenn der USB-Stick zu früh abgezogen wird? Bei welchen ist das ausgeschlossen? Begründen Sie Ihre Aussagen mit Ihrem Wissen über die Funktionsweise des Konsistenzverfahrens *Synchrones Schreiben*. **3 P**

e) *Synchrones Schreiben* und auch andere Verfahren, wie *Journaling*, benötigen sogenannte *Write Barriers*, um korrekt zu funktionieren. Wozu dient eine *Write Barrier*? **1 P**

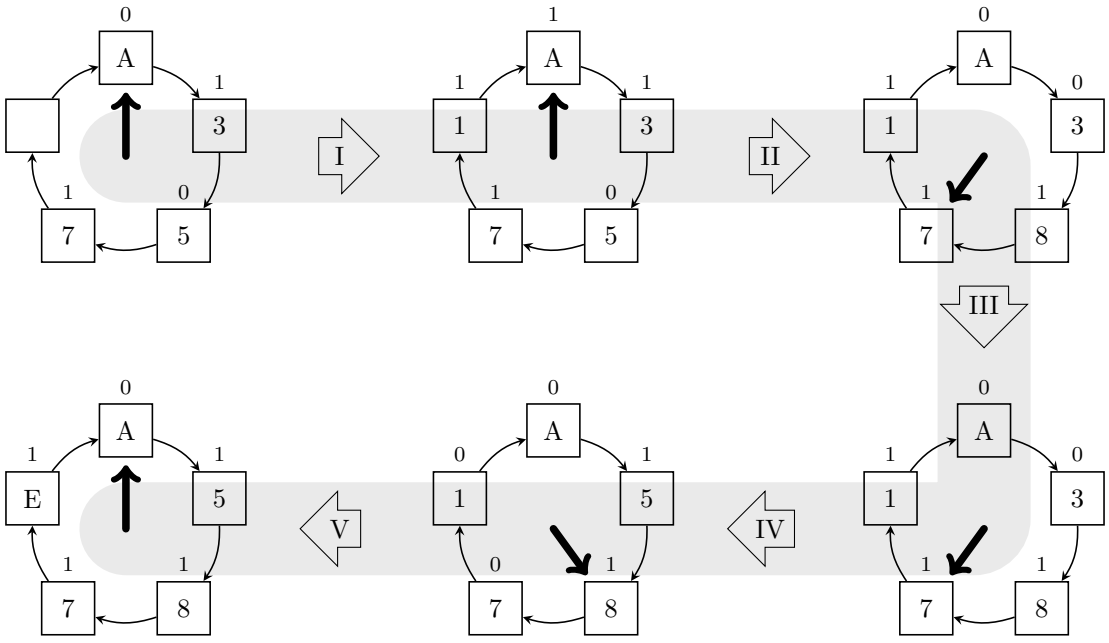
Aufgabe 4 – Speicherverwaltung **16 Punkte**

Ein Unix-System verfügt über 5 Rahmen physischen Speichers und setzt zur Seitenverdrängung den *Clock*-Algorithmus ein.

a) Warum ist ein Seitenverdrängungsalgorithmus, wie bspw. *Clock*, notwendig? **1 P**

b) Diskutieren Sie jeweils einen Vor- und Nachteil von *Clock* gegenüber *FIFO* als Seitenverdrängungsstrategie. **2 P**

Während der Ausführung des beschriebenen Systems wurden die folgenden Zustände im *Clock*-Algorithmus beobachtet.



c) Geben Sie für jeden Übergang eine Seitenreferenzfolge an, aus der sich die gezeigten Zustandsänderungen ergeben. **8 P**

I.

II.

III.

IV.

V.

d) Ist Ihre Lösung für Schritt III aus Teilaufgabe c) eindeutig oder sind noch andere Referenzfolgen für diesen Zustandsübergang möglich? Begründen Sie ihre Antwort kurz. **1 P**

e) Der *Clock*-Algorithmus gilt als gute Näherung des Verfahrens *LRU*. Beschreiben Sie kurz nach welchem Prinzip *LRU* arbeitet und erläutern Sie, weshalb *Clock* lediglich eine Näherung darstellt? **2 P**

f) Welche Schwierigkeit ergibt sich beim Einsatz von *Clock*, wenn mehrere Prozesse gemeinsamen Speicher verwenden? **2 P**

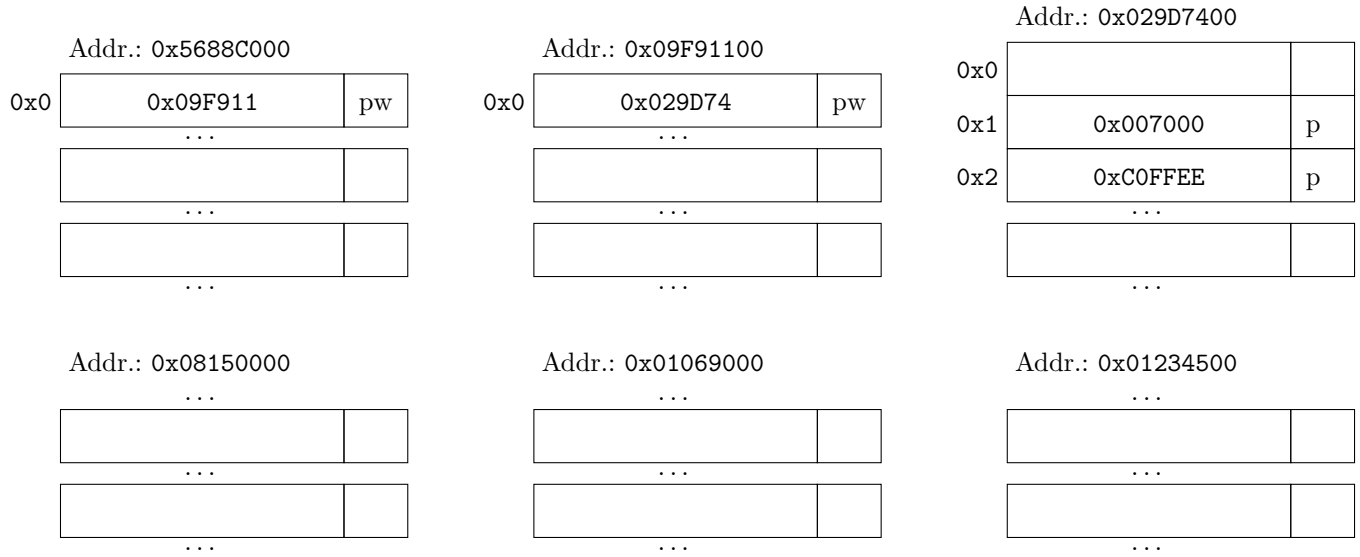
Aufgabe 5 – Virtueller Speicher **14 Punkte**

Ein hypothetisches System verwende virtuellen Speicher mit 32-Bit-Adressen und dreistufige Tabellen zur Adressübersetzung. Die Indizes aller Stufen seien jeweils 8 Bit groß und der Speicher sei Byte-adressierbar. In den Tabellen werden die Zugriffsrechte eines Eintrags durch die Bits „present“ (`p`) und „writable“ (`w`) bezeichnet. Bei einem Zugriff werden diese Rechte jeweils auf allen Stufen überprüft.

a) Wie groß (in Byte) ist eine Seite? **1 P**

b) Wie viele Seitentabellen (über alle Stufen) werden maximal benötigt, um einen Adressraum zu verwalten, in den 100 Seiten eingeblendet sind? **1 P**

Einem Prozess stehen folgende Seitentabellen zur Verfügung. Alle nicht aufgeführten oder frei gelassenen Einträge werden dabei als ungültig („not present“) angenommen. Der Einsprungpunkt für die Adressübersetzung, also die Tabelle der ersten Stufe, liegt an Adresse `0x5688C000`.



Weiterhin habe der Prozess die nebenstehenden Regionen:

`0x00000100` – `0x000002FF`	Code (lesbar)
`0x00000300` – `0x000004FF`	Heap (lesbar, schreibbar)
`0x7FFFFF00` – `0x7FFFFFFF`	Stack (lesbar, schreibbar)

c) Prüfen Sie für jeden der folgenden Zugriffe, ob dieser durch die MMU ausgeführt werden kann. Geben Sie in diesem Fall die zugehörige physische Adresse an. Für nicht unmittelbar auflösbare Adressen, prüfen Sie zunächst, ob das Betriebssystem Änderungen an den Seitentabellen vornehmen würde und tragen Sie diese ggf. in obiges Schema ein. Nennen Sie dann ebenfalls die sich ergebende physische Adresse des Zugriffs.

Ist ein Zugriff unzulässig, so genügt es dies anzugeben. Diese Zugriffe werden nicht weiter betrachtet und führen auch nicht zum Abbruch des Programms.

Hinweis: Freie Rahmen stehen an den Adressen `0x74E35B00`, `0xD8415600` und `0xC5635600` zur Verfügung. **12 P**

- Lesend auf `0x00000262`:
- Lesend auf `0x000102CD`:
- Schreibend auf `0x00000123`:
- Schreibend auf `0x7FFFFF0A`:
- Schreibend auf `0x00000500`:

Aufgabe 6 – Synchronisation

15 Punkte

a) Nennen Sie jeweils ein Beispiel für aktiv wartende (busy waiting) und blockierende Synchronisationsmechanismen. **2 P**

b) Diskutieren Sie Vor- und Nachteile aktiv wartender gegenüber blockierender Synchronisationsmechanismen. **2 P**

Gegeben sei nachfolgendes, aus vier Threads bestehendes, Programm. Die Threads greifen auf die gemeinsamen Variablen `A`, `B` und `C` zu, welche durch die Locks `l_A`, `l_B` und `l_C` geschützt werden.

Thread 1

```
1 lock(l_A);
2 lock(l_B);
3 lock(l_C);
4 f(A,B,C);
5 unlock(l_C);
6 unlock(l_B);
7 unlock(l_A);
```

Thread 2

```
1 lock(l_C);
2 lock(l_B);
3 lock(l_A);
4 f(C,B,A);
5 unlock(l_C);
6 unlock(l_B);
7 unlock(l_A);
```

Thread 3

```
1 lock(l_B);
2 lock(l_A);
3 lock(l_C);
4 f(B,A,C);
5 unlock(l_B);
6 unlock(l_A);
7 unlock(l_B);
```

Thread 4

```
1 lock(l_A);
2 lock(l_C);
3 lock(l_A);
4 f(A,C,B);
5 unlock(l_A);
6 unlock(l_C);
7 unlock(l_A);
```

c) Das Analyseprogramm *LockMaster 3000* hat diesen Programmcode als potenziell fehlerbehaftet markiert. Welche (potentiellen) Probleme sehen Sie in dem Programm? Geben Sie das korrigierte Programm in unten stehendem Schema an. **7 P**

Thread 1

```
lock(l_   );
lock(l_   );
lock(l_   );
f(A,B,C);
unlock(l_   );
unlock(l_   );
unlock(l_   );
```

Thread 2

```
lock(l_   );
lock(l_   );
lock(l_   );
f(C,B,A);
unlock(l_   );
unlock(l_   );
unlock(l_   );
```

Thread 3

```
lock(l_   );
lock(l_   );
lock(l_   );
f(B,A,C);
unlock(l_   );
unlock(l_   );
unlock(l_   );
```

Thread 4

```
lock(l_   );
lock(l_   );
lock(l_   );
f(A,C,B);
unlock(l_   );
unlock(l_   );
unlock(l_   );
```

Auf einer Hardware-Architektur, die keine atomaren Instruktionen unterstützt, soll ein Spinlock implementiert werden, das von beliebig vielen Prozessen benutzbar ist. Ein Entwickler hat für Ein-Prozessor-Systeme die folgenden Implementierung vorgeschlagen.

```
void init(int *lck) {
   *lck = 0;
}
```

```
void lock(int *lck) {
   cli();
   while (*lck == 1) {}
   *lck = 1;
   sti();
}
```

```
void unlock(int *lck) {
   *lck = 0;
}
```

Hinweis: `cli` blockiert die Zustellung von Interrupts auf dem Prozessor, wohingegen `sti` die Zustellung wieder erlaubt.

d) Welches Problem hat diese Implementierung? Geben Sie eine korrekte Implementierung an. Auch diese soll durch beliebig viele Prozesse nutzbar sein und ohne atomare Instruktionen auskommen. **4 P**