Name: Matrikel-Nr.: Studiengang:

Klausur Betriebssysteme und Sicherheit, 02. 03. 2022

— Bearbeitungszeit: 90 Minuten — Prüfer: Prof. Dr. Schirmeier, Dr. Köpsell —

1	2	3	4	5	6	$\sum$
17	13	15	15	15	15	90

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

Aufgabe 1 – Sicherheit

17 Punkte

a) Nennen Sie drei wesentliche Schutzziele mit jeweils einer kurzen Beschreibung in IT-Systemen. **3 P**

b) Beschreiben Sie den allgemeinen Ablauf der Authentikation mit einem *symmetrischen* Authentikationssystem: Was berechnet der Sender? Was wird gesendet? Wie prüft der Empfänger? Kann der Empfänger nach erfolgreicher Prüfung eine dritte Partei davon überzeugen, dass er die Nachricht vom Sender erhalten hat? Begründen Sie Ihre Antwort. **4 P**

c) Was besagt das Prinzip von Kerckhoffs? Geben Sie zwei Argumente an, die begründen, warum dieses Prinzip sinnvoll ist. **2 P**

d) Beschreiben Sie die Überlegungen hinter dem Sphärenmodell bezüglich Datenschutz. Welches Datenschutzprinzip lässt sich diesem Modell zuordnen? Erläutern Sie dieses Prinzip kurz. **2 P**

e) Nennen und erläutern Sie kurz mögliche Faktoren, die verwendet werden können, um Mensch durch Rechner zu identifizieren. Nennen Sie für jeden Faktor jeweils ein Beispiel. **3 P**

Gegeben sei folgendes Coverbild (Grauwertbild, 8 Bit/Pixel) in das die Nachricht *emb*=`001101` sequentiell (wie unten im Bild gezeigt) eingebettet werden soll.

255	254	253
254	253	252
251	254	255

f) Geben Sie das resultierende Stegobild an, unter Verwendung von *LSB-Ersetzung* für die Einbettung. **3 P**

Aufgabe 2 – Virtueller Speicher **13 Punkte**

Ein System verwendet 64 Bit Adressen und 3-stufige Seitentabellen mit 16 Bit Index je Seitentabelle und 16 Bit Offset. Wie üblich werden einzelne Byte adressiert. Die Einträge in den Tabellen sind jeweils 64 Bit groß.

a) Wie groß ist eine Seite? **1 P**

b) Wie viele Einträge gibt es in einer Seitentabelle? **1 P**

c) Wie viele Rahmen braucht das System zum Speichern einer Seitentabelle? **1 P**

d) Wie viele Tabellen sind mindestens nötig, um einen Adressraum mit 128 Seiten zu verwalten? **1 P**

Ein Programm wird mit einer leeren Seitentabelle gestartet und greift während seiner Ausführung erfolgreich auf folgende Adressen zu:

- `0x0815 0815 0815 0815`,
- `0xC01D C0FF EE00 0000` und
- `0x09F9 1102 9D74 E35B`.

Danach wurde ein Seitenfehler beim Zugriff auf `0xD841 56C5 6356 88C0` erfolgreich aufgelöst. Danach verändern sich die Seitentabellen nicht.

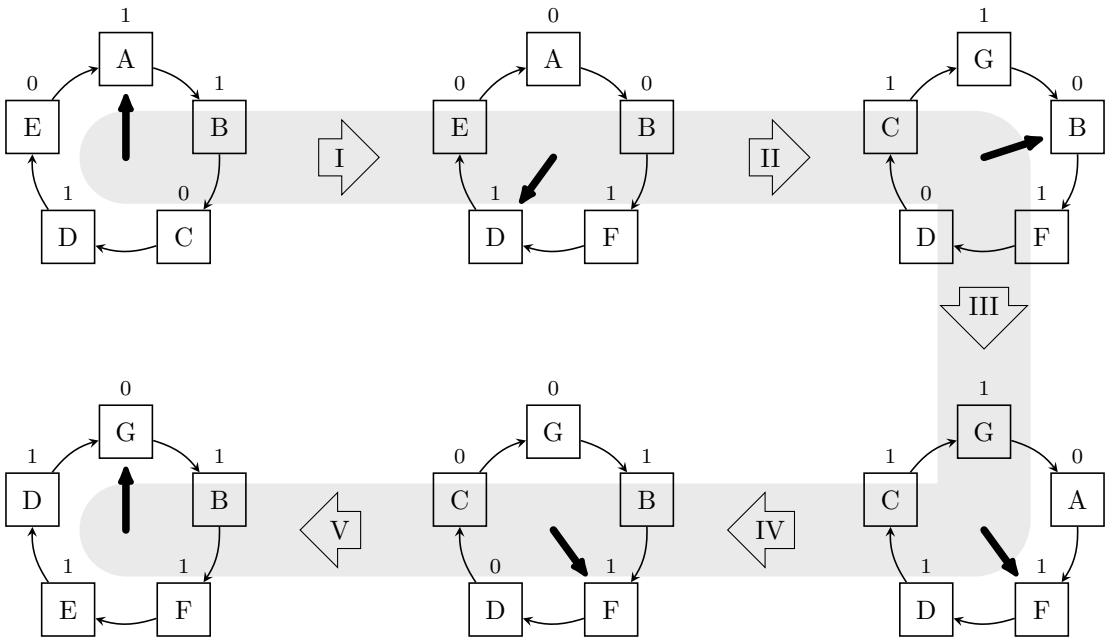
e) Welche der folgenden lesenden Zugriffe führen zu einem Seitenfehler?
HINWEIS: Falsche Antworten führen zu Punktabzug innerhalb der Teilaufgabe. **3 P**

1. `0x0816 0816 0815 0816`
2. `0xC01D DEAD BEEF 0000`
3. `0xD841 56C5 6356 ABCD`

Zur Verwaltung des limitierten physischen Speichers wird der Clock-Algorithmus verwendet.

f) Mit welchem Ziel wird ein Seitenverdrängungsalgorithmus, wie beispielsweise Clock, verwendet? **1 P**

In einem System mit fünf physischen Rahmen werden die folgenden Zustände beobachtet:



g) Geben Sie für jeden der Übergänge eine mögliche Zugriffsfolge an. **5 P**

Aufgabe 3 – Synchronisation **15 Punkte**

a) Nennen Sie neben Semaphoren drei weitere Techniken für wechselseitigen Ausschluss. **3 P**

b) Nennen Sie einen Vorteil von Semaphoren gegenüber Spinlocks. **1 P**

c) Übernehmen Sie das unten gegebene Code-Beispiel und ergänzen Sie die nötigen Funktionen eines Semaphors um ordnungsgemäß wechselseitigen Ausschluss zu gewährleisten.

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. **2 P**

```
sem_t s(    );

void thread_fn() {
  
  /* Kritischer Abschnitt */
  
}
```

Für das Verarbeiten einer Datei soll das Erzeuger-Verbraucher-Modell mit einem Erzeuger und einem Verbraucher genutzt werden. Busy Waiting soll, sofern möglich, vermieden werden. Für den Austausch der Daten steht ein 10-elementiger FIFO-Ringpuffer zur Verfügung.

d) Wie viele Semaphore werden benötigt? Wie sind diese Semaphore jeweils initialisiert? **3 P**

e) Übernehmen Sie den unten stehenden Code und vervollständigen Sie ihn. Für die schnellere Verarbeitung wurde außerdem entschieden 2 Verbraucher zu verwenden. Sollten weitere Semaphore oder Variablen benötigt werden, so führen Sie diese ein. **6 P**

Globale Variablen

```
ele_t b[10];
int front = 0;
int back = 0;
```

Erzeuger

```
while (1) {
  
  ele_t e = erzeugen();
  
  buffer[back] = e;
  back = (back + 1)%10;
  
}
```

Verbraucher

```
while (1) {
  
  e = buffer[front];
  front = (front + 1)%10;
  
  verbrauchen(e);
  
}
```

Aufgabe 4 – Dateisysteme **15 Punkte**

In einem Dateisystem, das das Unix-Rechtesystem umsetzt, gibt es folgende Dateien und Verzeichnisse mit folgenden Rechten:

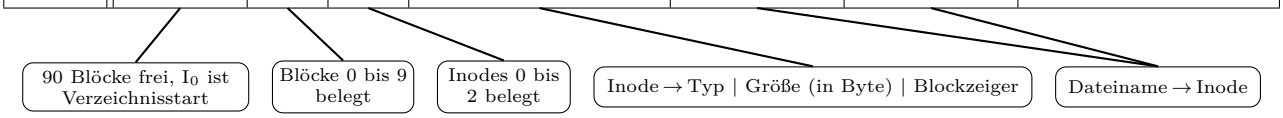
```
/          root  root  rwx rwx r-x
/data      root  g1    rwx rwx r-x
/data/foo  u1    g1    rwx r-- r-x
/data/bar  u1    g2    rw- rw- rwx
```

a) Beantworten Sie für die folgenden Zugriffe, ob sie erlaubt wären oder nicht. Begründen Sie ihre Antwort. **3 P**

1. Nutzer `u1`, in Gruppe `g1` möchte die Datei `/data/foo` zum Lesen öffnen.
2. Nutzer `u3`, in Gruppe `g2` möchte die Datei `/data/foo` ausführen.
3. Nutzer `u1`, in Gruppe `g1` möchte die Datei `/data/bar` ausführen.

Ein Unix-artiges Dateisystem organisiert alle Dateiinhalte und Metadaten in Blöcken auf dem Speichermedium. Die Größe eines Blocks beträgt 4 KiByte. Gegeben ist der nachfolgende Ausschnitt aus dem Dateisystemzustand auf der Festplatte. Die dargestellten Blöcke enthalten entweder Inodes (IB), Superblöcke (SB), Verzeichnisblöcke (VB) 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	5	6
Typ	SB	BA	IA	IB	VB	VB	IB
Inhalt	`free`: 90 `root`: I_0	0 – 9	0,1,2	$I_0 \to$ V | 32 | 4 $I_1 \to$ V | 64 | 5 $I_2 \to$ D | 5.000 | 7,8	`"data"` $\to I_1$	`"bar"` $\to I_2$ `"foo"` $\to I_3$	$I_3 \to$ D | 3.000 | 9



b) Welche Blöcke müssen vom Betriebssystem gelesen werden, um die Datei `/data/bar` zu öffnen und ihren vollständigen Inhalt zu lesen? **6 P**

c) Das verwendete Dateisystem nutzt Journaling (nur Metadaten) um Inkonsistenzen bei einem Absturz zu vermeiden. Geben Sie die Reihenfolge der Blöcke (mit Write-Barriers) an, die geschrieben werden müssen, wenn an die Datei `/data/foo` 16 KiByte angefügt werden. **6 P**

Hinweis:

- Für das Journal stehen die Blöcke 100-120 zur Verfügung.
- Das Entfernen der Journaleinträge muss nicht angegeben werden.
- Nutzen Sie für normale Blöcke folgende Notation:
  - *Blocknummer* (*Typ*)
  - **Beispiel:** Schreiben von Datenblock an Blockadresse 30 ⇒ 30 (D)
- Nutzen Sie für die Journalblöcke folgende Notation:
  - *Blocknummer* (J, *Typ*, *Zielblock*) für normale Journaleinträge oder
  - *Blocknummer* (J, TX) für den Transaktionsblock (ohne Prüfsumme) im Journal.
  - **Beispiel:** Schreiben von Inode an Blockadresse 28 in Journalblock 100 ⇒ 100 (J, IB, 28)
- Alle nicht genannten Blöcke im obigen Schema sind verfügbar und können verwendet werden.

Aufgabe 5 – Unix

15 Punkte

Ein Unix-Prozess führt den unten stehenden Code aus. Dabei hat die Datei `/home/sweet/home` den Inhalt: `LS`

```
1  int fd1, fd2;
2  int bytes = -1;
3
4  int main() {
5    char buffer[1];
6    int counter = 0;
7
8    fd1 = open("/home/sweet/home", O_RDONLY);
9    unlink("/home/sweet/home");
10
11   fd2 = creat("/sweet/alabama", O_WRONLY);
12
13   bytes = read(fd1, buffer, 1);
14
15   while (bytes > 0) {
16     int status;
17     int pid = fork();
18
19     if (pid == 0) {
20       write(fd2, buffer, 1);
21
22       counter = counter + 1;
23       printf("Zähler: %d\n", counter);
24
25       exit(0);
26     }
27
28     wait(&status);
29     bytes = read(fd1, buffer, 1);
30   }
31
32   close(fd1);
33   close(fd2);
34   printf("Zähler: %d\n", counter);
35 }
```

a) In welchen Segmenten des Unix-Adressraumes liegen die Symbole `main`, `fd1`, `bytes` und `counter`? **4 P**

b) Welchen Inhalt hat `/sweet/alabama`, nachdem das Programm sich erfolgreich beendet hat? Gibt es mehrere Möglichkeiten? Begründen Sie kurz warum mehrere Möglichkeiten existieren oder nicht existieren. **3 P**

c) Was wird auf der Konsole ausgegeben? Erläutern Sie kurz, wie es zu dieser Ausgabe kommt. **3 P**

d) Was passiert beim Aufruf von `unlink` in Zeile 9? Wann wird der Inhalt der zugehörigen Datei gelöscht? **2 P**

e) Was unterscheidet einen System Call von einem normalen Funktionsaufruf innerhalb eines Programms? **1 P**

f) Nennen Sie zwei mögliche Ursachen dafür, dass die Funktion `exec()` fehlschlägt. **2 P**

Hinweis:

- `open(pfad, modus)` öffnet die Datei `pfad` im Modus `modus`. `O_RDONLY` öffnet zum Lesen, `O_WRONLY` öffnet zum Schreiben.
- `creat(pfad, modus)` erzeugt die Datei `pfad` und öffnet sie anschließend in dem Modus `modus`.
- `read(fd, puffer, n)` liest bis zu `n` Bytes aus dem File Descriptor `fd` in den Puffer `puffer`. Die Funktion gibt die Menge der gelesenen Bytes zurück oder `-1`, wenn sie am Ende der Datei angelangt ist.
- `write(fd, puffer, n)` schreibt `n` Bytes aus dem Puffer `puffer` in den File Descriptor `fd`.
- `printf(format, zahl)` erzeugt eine Ausgabe wie in `format` angegeben. Dabei wird `%d` durch den Wert der Variable `zahl` ersetzt.
- Alle Funktionsaufrufe sind *stets erfolgreich*.

Aufgabe 6 – Scheduling **15 Punkte**

a) In der kurzfristigen Einplanung werden die Abfertigungszustände bereit (ready), aktiv (running) und blockiert (blocked) verwendet. Zeichnen Sie das Zustandsdiagramm mit den Zuständen und allen gültigen Übergängen zwischen diesen. Beschreiben Sie für jeden Zustandsübergang *kurz*, ein Ereignis, das ihn auslöst. **5 P**

b) In einem Multiprozessor-Betriebssystem können laufbereite Prozesse in einer einzigen, *globalen* Ready-Liste verwaltet werden, oder aber in je einer *lokalen* Ready-Liste pro CPU. Erklären Sie kurz *einen* Vorteil und *einen* Nachteil der Verwaltung in *einer globalen* Listen. **2 P**

Ein Betriebssystem verwaltet drei Prozesse P_1, P_2 und P_3. Die Prozesse treffen in dieser Reihenfolge im System ein und sind alle zum Zeitpunkt $t = 0$ rechenbereit. Jeder Prozess hat drei CPU-Stöße, deren Dauer (in ms) in nebenstehender Tabelle angegeben sind.

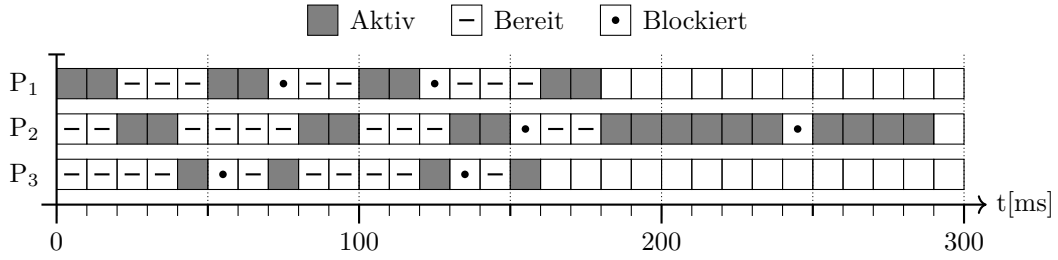
P_1	40	20	20
P_2	60	60	40
P_3	10	20	10

Zusätzlich hat jeder Prozess zwei E/A-Stöße von je 10 ms Dauer, die jeweils zwischen den drei CPU-Stößen stattfinden. Die Prozessumschaltzeit kann vernachlässigt werden.

Als Strategien stehen folgende aus der Vorlesung bekannte Möglichkeiten zur Verfügung:

- First-Come First-Served (FCFS)
- Least Recently Used (LRU)
- Round Robin (RR) mit Zeitscheibenlänge 20 ms
- Shortest Process Next (SPN)
- Shortest Seek Time First (SSTF)
- Virtual Round Robin (VRR) mit Zeitscheibenlänge 20 ms

c) Das folgende Gantt-Diagramm zeigt eine mögliche Abarbeitungsreihenfolge der Prozesse auf einem Einprozessorsystem. Welche der möglichen Strategien führt zu diesem Verhalten? Erklären Sie *kurz*, woran Sie das erkannt haben. **3 P**



In einem Einprozessor-Echtzeitsystem sind drei periodische Tasks T_1, T_2, T_3 einzuplanen mit folgenden Werten (p_i Periodenlänge, t_i Bearbeitungszeit, Periodenende = Zeitschranke):

$$p_1 = 15,\ t_1 = 4 \qquad p_2 = 10,\ t_2 = 2 \qquad p_3 = 4,\ t_3 = 1$$

Zwischen den Tasks bestehen keine Abhängigkeiten und sie sind an beliebiger Stelle unterbrechbar.

d) Ist eine Einplanung mittels *statischer* Prioritäten möglich? Begründen Sie Ihre Antwort und ordnen Sie, wenn möglich, den Tasks Prioritäten zu. **3 P**

e) Die Taskmenge aus Teilaufgabe (d) soll nun mit *dynamischen* Prioritäten eingeplant werden. Außerdem sollen, wenn möglich, noch zusätzliche Tasks hinzugefügt werden. Diese zusätzlichen Tasks T_n haben alle die gleichen Eigenschaften ($p_n = 20$, $t_n = 1$). Um wie viele solcher Tasks kann die Taskmenge maximal erweitert werden, ohne die Einplanbarkeit zu gefährden? **2 P**