Klausur Betriebssysteme und Sicherheit, 04.03.2026

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

1	**2**	**3**	**4**	**5**	**6**	$\sum$
15	15	15	15	15	15	90

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

Aufgabe 1 – Kryptographie und Authentifikation **15 Punkte**

Die affine Chiffre ist ein Verschlüsselungsverfahren, bei dem jedem Buchstaben des Klartextes ein Zahlenwert $x \in \{0,\ldots,25\}$ zugeordnet wird. Die Verschlüsselung erfolgt durch

$$y = (a \cdot x + b) \mod 26,$$

wobei y den Zahlenwert des entsprechenden Geheimtextbuchstabens bezeichnet. Dabei bilden a und b ein Schlüsselpaar mit

- $a \in \{1, 3, 5, 7, 9, 11, 15, 17, 19, 21, 23, 25\}$
- $b \in [0, 25]$

Jeder Buchstabe wird entsprechend seinem Zahlenwert einzeln verschlüsselt. Zur Entschlüsselung wird die inverse Abbildung $x = a^{-1}(y - b) \mod 26$ verwendet.

Gehen Sie im Folgenden vom lateinischen Alphabet mit fortlaufend durchnummerierten 26 Großbuchstaben gemäß Tabelle 1 aus.

Tabelle 1: Alphabet

Buchstabe	A	B	C	D	E	F	G	H	I	J	K	L	M
Zahlenwert	0	1	2	3	4	5	6	7	8	9	10	11	12
Buchstabe	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
Zahlenwert	13	14	15	16	17	18	19	20	21	22	23	24	25

a) Gegeben sei das Schlüsselpaar $a = 5, b = 3$. Verschlüsseln Sie den Klartext „`EVE`“ mit der affinen Chiffre. **2 P**

b) Erklären Sie, warum jedes Schlüsselpaar mit $a = 1$ einer Caesar-Verschlüsselung entspricht. **2 P**

c) Sie haben eine verschlüsselte Nachricht abgefangen. Wie viele eindeutige Schlüsselpaare kommen für die Verschlüsselung infrage? Es genügt, wenn Sie eine Formel zur Berechnung angeben. **2 P**

d) Nennen Sie einen Grund und erläutern Sie kurz, warum dieses Verschlüsselungsverfahren nicht sicher ist. **2 P**

Ein Unternehmen verwaltet die Passwörter seiner Benutzer in einer Datenbank. Dabei speichert es für jeden Benutzer den Benutzernamen, einen Salt sowie den Hashwert des Passworts.

e) Authentifikation kann auf drei Faktoren basieren: Wissen, Besitz und Biometrie. Nennen Sie jeweils ein Beispiel für jede dieser Authentifikationsarten. **3 P**

f) Erklären Sie, was Preimage Resistance (Präbildresistenz) bedeutet und warum sie für die Sicherheit von Passwörtern relevant ist. **2 P**

g) Erklären Sie den Unterschied zwischen einem Salt und einem Pepper im Kontext von Passwort-Hashfunktionen. Gehen Sie darauf ein, welche unterschiedlichen Angriffe durch die beiden Mechanismen erschwert werden. **2 P**

Aufgabe 2 – Sicherheit **15 Punkte**

Gegeben sei der C-Code in Abb. 1.

Die Funktion `strcpy (str1, str2)` schreibt alle Zeichen von `str2` in `str1` bis ein Null-Terminator gelesen wird, welches das Ende von `str2` signalisiert.

Nehmen Sie an, dass keine zusätzlichen Schutzmechanismen, insbesondere keine vom Compiler bereitgestellten Sicherheitsfunktionen, aktiv sind. Gehen Sie außerdem von einer x86-Architektur mit 32 Bit sowie von den C-Aufrufkonventionen `cdecl` aus.

```
1  #include <stdio.h>
2  #include <string.h>
3
4  void copy(char* input) {
5    char reserved[128];
6    strcpy(reserved, input);
7    printf("Copy complete\n");
8  }
9
10 int main() {
11   int length = 256;
12   char input[length];
13   printf("Waiting for input:");
14   fgets(input, length);
15   copy(input);
16   return 0;
17 }
```

Abbildung 1: C-Code.

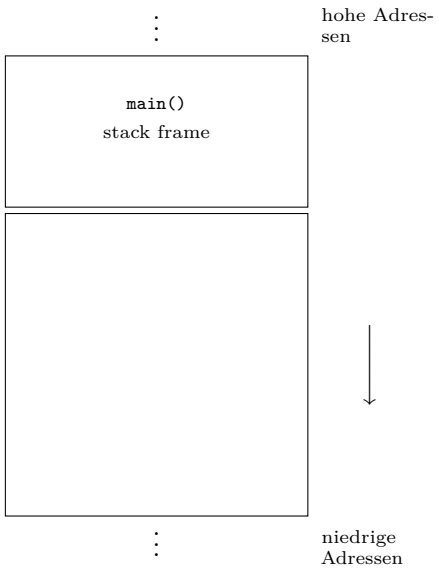


Abbildung 2: Stack-Frame.

a) Vervollständigen Sie Abbildung 2, indem Sie die typischerweise gesicherten Register, die Funktionsargumente sowie die lokalen Variablen der Funktion `copy` so einzeichnen, wie sie sich vor dem Funktionsaufruf `strcpy(reserved, input)` (Zeile 6) auf dem Stack befinden.

Geben Sie hierbei die Größen der jeweiligen Speicherbereiche an; ein maßstabsgetreues Größenverhältnis in der Zeichnung ist nicht erforderlich. **3 P**

b) Was ist ein Buffer-Overflow? Kann es in Abb. 1 zu einem Buffer-Overflow kommen? Falls ja, beschreiben Sie, warum. Falls nein, begründen Sie Ihre Antwort. **2 P**

c) Nehmen Sie an, dass der Stack in der umgekehrten Richtung, d. h. von niedrigen zu hohen Adressen, wächst. Das würde bedeuten, dass die Rücksprungaddresse eine niedrigere Adresse als die lokalen Variablen hat. Sind stackbasierte Buffer-Overflow-Angriffe immer noch möglich? Begründen Sie Ihre Antwort. **2 P**

d) Als wirksame Gegenmaßnahme gegen Buffer-Overflows wurde „Address Space Layout Randomization (ASLR)“ eingeführt. Beschreiben Sie kurz die Funktionsweise von ASLR und warum ASLR gegen Buffer-Overflow-Angriffe schützt. **2 P**

Eine Angreiferin betreibt eine Webseite und platziert dort das folgende HTML-Element, um einen CSRF-Angriff (Cross-Site Request Forgery) durchzuführen:

```
<img src="https://bank.example.com/transfer?to=eve&amount=1000">
```

Analysieren Sie den Angriff und beantworten Sie die folgenden Fragen.

e) Wer ist das Opfer? **1 P**

f) Was ist das Ziel des Angriffs? **1 P**

g) Welche Schwachstelle der Bankanwendung wird bei dem Angriff ausgenutzt? **1 P**

h) Warum ist das präparierte HTML-Element besonders problematisch? **1 P**

i) Welche Voraussetzungen müssen beim Opfer für einen erfolgreichen Angriff erfüllt sein? **1 P**

j) Welche Schutzmaßnahme würden den Angriff erschweren oder verhindern? **1 P**

Aufgabe 3 – Dateisysteme **15 Punkte**

a) Nennen und erläutern Sie *jeweils einen* Nachteil für eine unterbrechungsgetriebene E/A sowie aktives Warten. **2 P**

Das Dateisystem einer SSD nutzt eine Blockgröße von 4 KiB. Die Dateien liegen auf der SSD wie folgt im unten skizzierten Dateisystem. Die dargestellten Blöcke enthalten entweder Inodes (IB), den Superblock (SB), Verzeichnisblöcke (VB) oder eine Allokations-Bitmap für Blöcke (ABM) bzw. Inodes (IBM). Weitere Blöcke (z. B. für Dateiinhalte) existieren, sind aber nicht dargestellt. Einträge in den Inodeblöcken haben dabei die Form:

`<Inode-Nummer>` → `<Liste der Datenblöcke> / <Dateigröße in Bytes>`

Block	0	1	2	3	4	5	6
Typ	SB	ABM	IBM	IB	IB	VB	VB
Inhalt	free: 40 root: I_0	0 – 10	0 – 3	$I_0 \rightarrow 5\,/\,32$ $I_1 \rightarrow 7, 8\,/\,7000$	$I_2 \rightarrow 6\,/\,64$ $I_3 \rightarrow 9, 10\,/\,8000$	`etc` → I_2	`passwd` → I_1 `data` → I_3

b) Welche Blöcke des Dateisystems müssen in welcher Reihenfolge gelesen werden, wenn aus der Datei `/etc/passwd` 1 KiB ab einem Offset von 5 KiB eingelesen werden soll? Nehmen Sie dabei an, dass die Inhalte eines Blockes immer vollständig gelesen werden (also kein Block ein zweites Mal gelesen werden muss). **6 P**

c) Die Datei `/etc/data` wird gelöscht. Geben Sie den aktualisierten Zustand des Dateisystems in der in Aufgabe a) angegebenen Form an: **4 P**

Block	0	1	2	3	4	5	6
Typ	SB	ABM	IBM	IB	IB	VB	VB
Inhalt							

d) In einem Verzeichnis befindet sich ausschließlich eine leere Datei `out.txt` mit folgender Rechtezuweisung:

Dateiname	Besitzer	Gruppe	Rechte
`out.txt`	`alex`	`users`	`--- rw- rw-`

Der Benutzer `alex` gehört zur Gruppe `users` und führt folgenden Befehl im Verzeichnis von `out.txt` aus:

```
ls > out.txt
```

Zerlegen Sie die Befehlskette in ihre Bestandteile und beschreiben Sie jeweils kurz deren Funktion. Welchen Inhalt hat die Datei `out.txt` nach der Ausführung? **3 P**

Aufgabe 4 – Speicher

15 Punkte

Gegeben sei ein fiktives 32-Bit-System. Das System verwaltet seinen virtuellen Speicher mit zweistufigen Seitentabellen. Die Indizes der beiden Stufen sind jeweils 10 Bit groß. Jede Seite ist 4 KiB groß.

Hinweis: Bei den folgenden Aufgaben genügt die Angabe von Zweierpotenzen.

a) Beantworten Sie folgende Fragen:

Hinweis:

Bei den folgenden Aufgaben genügt die Angabe von Zweierpotenzen.

Es ist immer auch eine geeignete Einheit anzugeben.

- Wie viele Einträge umfasst eine Seitentabelle einer Stufe?
- Wie groß ist der Offset in diesem System?
- Wie groß ist eine Kachel?

3 P

b) Das System verwaltet folgende Status- und Rechte-Bits in seinen Seitentabellen:

p	w	u
Present Bit	Write Bit	User Bit

Gegeben sind die folgenden Seitentabellen. Die Seitentabelle der ersten Stufe liegt bei `0x1000`.

Addr.: `0x1000`

0x0	0xD	pwu
0x1	0x2	pwu
0x2	0x1	wu
...		
0xBA	0x9	p u
...		
0x3FF	0xD	pw

Addr.: `0x2000`

0x0	0x2F8	pwu
...		
0x51	0x007	pw
...		
0x1AD	0x0D9	p u
...		
0x3FF	0xDEA	wu

Addr.: `0x9000`

0x0	0xABC	pw
0x1	0x05E	p
0x2	0xC00	pw
...		
0x12	0x1A3	pwu
...		
0x3FF	0x03C	p u

Addr.: `0xD000`

0x0	0x01F	p u
...		
0x200	0xAF4	p u
0x201	0x897	pw
...		
0x3FF	0x0B2	wu

Ein Nutzerprogramm führt nun die nachfolgend gegebenen Zugriffe aus. Bestimmen Sie mithilfe der Seitentabellen für jeden erfolgreichen Zugriff die physische Adresse. Geben Sie andernfalls an, warum und an welcher Stelle ein Seitenfehler auftritt. **4 P**

- Lesender Zugriff auf `0x2001A7`
- Lesender Zugriff auf `0x2E80203F`
- Schreibender Zugriff auf `0x3FFB2C`
- Schreibender Zugriff auf `0x8D9`

c) Für die Ermittlung möglicher Ersetzungskandidaten bei Speicherplatzmangel nutzt das System das Verfahren **OPT**. Der aus drei Rahmen bestehende Hauptspeicher sei zu Beginn leer. Betrachten Sie nun folgende Seitenreferenzfolge:

5 – 1 – 2 – 3 – 2 – 5 – 4 – 1 – 3 – 4 – 2

Ergänzen Sie in der gegebenen Tabelle die Zuordnung von Seiten zu Rahmen gemäß der verwendeten Seitenersetzungsstrategie. Notieren Sie außerdem das Auftreten von Seitenfehlern.

HINWEIS: Die Kontrollzustände dienen ausschließlich als Gedankenstütze und sind **nicht** Gegenstand der Bewertung. **6 P**

		5	1	2	3	2	5	4	1	3	4	2
Hauptspeicher	Rahmen 1											
	2											
	3											
	Seitenfehler											
Kontrollzustände	Rahmen 1											
	2											
	3											

d) Begründen Sie kurz, weshalb das Verfahren OPT im Allgemeinen für die Seitenersetzung in der Praxis nicht umsetzbar ist. Nennen Sie zudem ein alternatives Verfahren, welches in der Realität Verwendung findet.

2 P

Ersatztabelle für c) (Das Verwenden dieser Tabelle macht die andere Lösung automatisch ungültig!):

		5	1	2	3	2	5	4	1	3	4	2
Hauptspeicher	Rahmen 1											
	2											
	3											
	Seitenfehler											
Kontrollzustände	Rahmen 1											
	2											
	3											

Aufgabe 5 – Scheduling

15 Punkte

Zwei Threads, T_D und T_W, arbeiten auf einem Einprozessorsystem zusammen. T_D empfängt Videoframes von einem Server, dekodiert sie und legt sie in einer gemeinsamen Datenstruktur ab, T_W entnimmt die Frames und schreibt sie in eine Videodatei. Beide Threads wiederholen sich unendlich lange und sind zum Zeitpunkt t=0 rechenbereit.

HINWEIS: Es kann davon ausgegangen werden, dass jederzeit zu verarbeitende Daten vorliegen, sowohl in der gemeinsamen Datenstruktur als auch auf dem Server.

HINWEIS: Nehmen Sie an, dass in diesem System Threads genauso wie Prozesse eingeplant werden.

Funktion	Beschreibung	Dauer
`receive_frame`	Empfängt einen Frame von einem Server (blockierend)	20 ms
`decode`	Dekodiert einen Frame	40 ms
`insert`	Schreibt einen Frame in die gemeinsame Datenstruktur	10 ms
`remove`	Liest und entfernt einen Frame aus der gemeinsamen Datenstruktur	30 ms
`store_frame`	Speichert einen Frame in die Video-Datei (blockierend)	40 ms

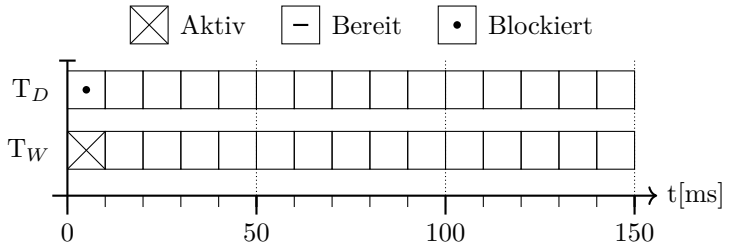
```
// Frame Decoder (T_D)
encoded_frame = receive_frame();
frame = decode(encoded_frame);
insert(frame);
```

```
// Frame Writer (T_W)
frame = remove();
store_frame(frame);
```

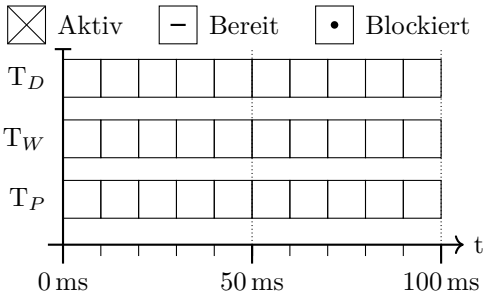
a) Geben Sie für beide Threads die Länge des CPU- und des E/A-Stoßes an. **2 P**

b) Zeichnen Sie in das folgende Gantt-Diagramm ein, wie die beiden Threads T_D und T_W bearbeitet werden, wenn das Scheduling nach der „Virtual Round Robin“-Strategie mit einer Zeitscheibe von 20 ms vorgenommen wird.

HINWEIS: Die ersten 10 ms sind bereits vorgegeben. **3 P**



c) Angenommen, im System läuft zusätzlich ein weiterer Thread T_P. Dieser rechnet 20 ms lang und gibt anschließend etwas auf der Konsole aus. Die Ausgabe dauert 10 ms. Zeichnen Sie in das nebenstehende Gantt-Diagramm ein, wie die drei Threads T_D, T_W und T_P bearbeitet werden, wenn das Scheduling nach der „Shortest Remaining Time First“-Strategie vorgenommen wird. Gehen Sie davon aus, dass dem Scheduler die entsprechenden Laufzeiten bekannt sind und er auf Basis dieses Wissens entscheiden kann. **3 P**

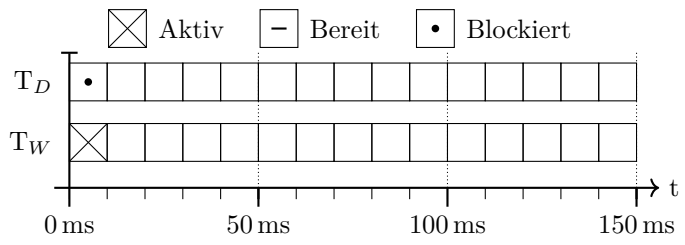


d) Benennen und erläutern Sie das Problem, das beim Einsatz der Scheduling-Strategie „Shortest Remaining Time First“ auftreten kann. Gehen Sie dabei auf die Ursache dieses Problems ein und erklären Sie, inwiefern die Strategie „Highest Response Ratio Next“ das Problem adressiert. **3 P**

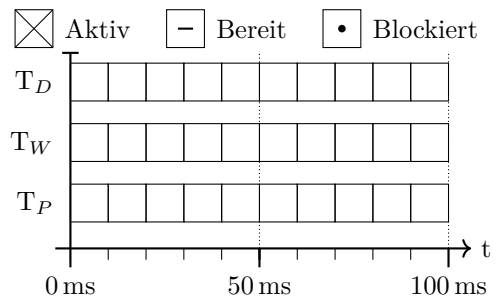
e) Beide Threads nutzen eine gemeinsame Datenstruktur. Welches Problem kann beim präemptiven Scheduling entstehen? Warum ist es beim kooperativen Scheduling besser vorhersehbar? Wie kann das Problem allgemein gelöst werden? **3 P**

f) Warum ist es sinnvoll, die Aufgaben in dieser Form zu trennen, anstatt einen einzigen Thread zu verwenden, der sowohl das Empfangen und Dekodieren als auch das Schreiben der Frames übernimmt? **1 P**

Ersatzdiagramm für b) (Das Verwenden dieses Diagramms macht die andere Lösung automatisch ungültig!):



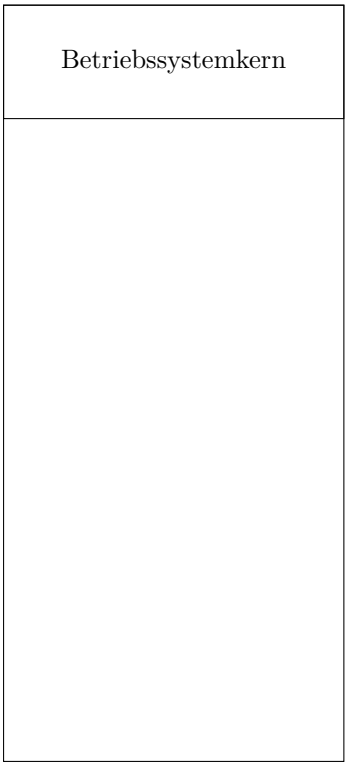
Ersatzdiagramm für c) (Das Verwenden dieses Diagramms macht die andere Lösung automatisch ungültig!):



Aufgabe 6 – UNIX und IPC

15 Punkte

a) Ergänzen Sie in der nebenstehenden Zeichnung die fünf Segmente eines UNIX-Adressraums, die aus der Vorlesung bekannt sind. Ordnen Sie die Segmente entsprechend der üblichen UNIX-Konvention an. Beschreiben Sie zudem kurz, welche Art von Daten in dem jeweiligen Segment gespeichert werden. **5 P**



b) Nennen Sie den Mechanismus der Interprozesskommunikation (IPC), der bei der Ausführung dieser Befehlskette genutzt wird: `ls | wc` **1 P**

Nebenstehender C-ähnlicher Pseudocode soll dafür sorgen, dass Eltern- und Kindprozess über den in Teilaufgabe b) abgefragten IPC-Mechanismus miteinander kommunizieren können. Für die Implementierung sind die Funktionen folgender Systemaufrufe gegeben:

- `ipc_channel() -> (fd_in, fd_out)`: erstellt einen Kommunikationskanal für IPC und gibt ein Tupel mit File-Deskriptoren auf die Eingabe in den Kanal (`fd_in`) und die Ausgabe aus dem Kanal (`fd_out`) zurück.
- `dup2(fd_0, fd_1)`: manipuliert den File-Deskriptor `fd_1`, sodass dieser auf die gleiche offene Datei zeigt wie der File-Deskriptor `fd_0`.
- `close(fd_0)`: schließt den File-Deskriptor `fd_0`.

```
int STDIN = 0, STDOUT = 1, STDERR = 2;
int fd_in, fd_out;

[fd_in, fd_out] = ipc_channel();
int res = fork();
if (res > 0) {
    
    execlp("ls", "ls", NULL); }
else if (res == 0) {
    
    execlp("wc", "wc", NULL); }
```

c) Erklären Sie, warum der Systemaufruf `ipc_channel()` vor dem Aufruf von `fork()` stattfinden muss, damit die Kommunikation zwischen Eltern- und Kindprozess funktioniert. Gehen Sie insbesondere auf die File-Deskriptoren ein. **3 P**

d) Füllen Sie die grau hinterlegten Bereiche mit passendem Pseudocode aus. Achten Sie darauf, alle nicht länger benötigten File-Deskriptoren zu schließen. Die in Teilaufgabe b) vorgegebene Kommunikationsrichtung soll beibehalten werden. Bei Bedarf können Sie die vordefinierten File-Deskriptoren für die Standardeingabe (`STDIN`), die Standardausgabe (`STDOUT`) und die Standardfehlerausgabe (`STDERR`) verwenden. **6 P**