Klausurvorbereitung

T210 - Betriebssysteme

1. **Single Choice**

   Welche der folgenden Aussagen ist wahr. In jeder Teilfrage ist nur genau eine Aussage wahr. Setzen Sie also genau ein Kreuz. Sollten Sie mehr als eine Aussage ankreuzen, erhalten Sie null Punkte für diese Teilfrage.

(1.1) (1 Punkt) Welche Operation ist **keine** übliche Dateioperation in herkömmlichen Betriebssystemen?

- ◯ `APPEND` - An eine vorhandene Datei anhängen
- ⊗ `ENCRYPT` - Datei verschlüsseln
- ◯ `OPEN` - Datei öffnen

(1.2) (1 Punkt) In einem I-Node-basierten Dateisystem...

- ◯ wird die max. Dateigröße durch die Anzahl der I-Nodes beschränkt.
- ◯ wird die max. Partitionsgröße durch die Anzahl von Links in der I-Node beschränkt.
- ⊗ wird die Dateianzahl durch die Anzahl der I-Nodes beschränkt.

(1.3) (1 Punkt) In einem FAT-Dateisystem...

- ⊗ wird die maximale Partitionsgröße u. a. durch Größe der FAT-Einträge beschränkt.
- ◯ wird die Dateianzahl durch die Blockgröße beschränkt.
- ◯ wird die maximale Dateigröße durch die Anzahl von FAT-Tabellen beschränkt.

(1.4) (1 Punkt) Der Referenzzähler einer Datei in einem Inode-basierten Dateisystem gibt an,

- ◯ wie oft die Inode der Datei verändert wurde.
- ⊗ wie oft die Inode in Verzeichnissen verlinkt ist.
- ◯ wie oft auf diese Datei lesend zugegriffen wurde.

(1.5) (1 Punkt) Prozesswechsel...

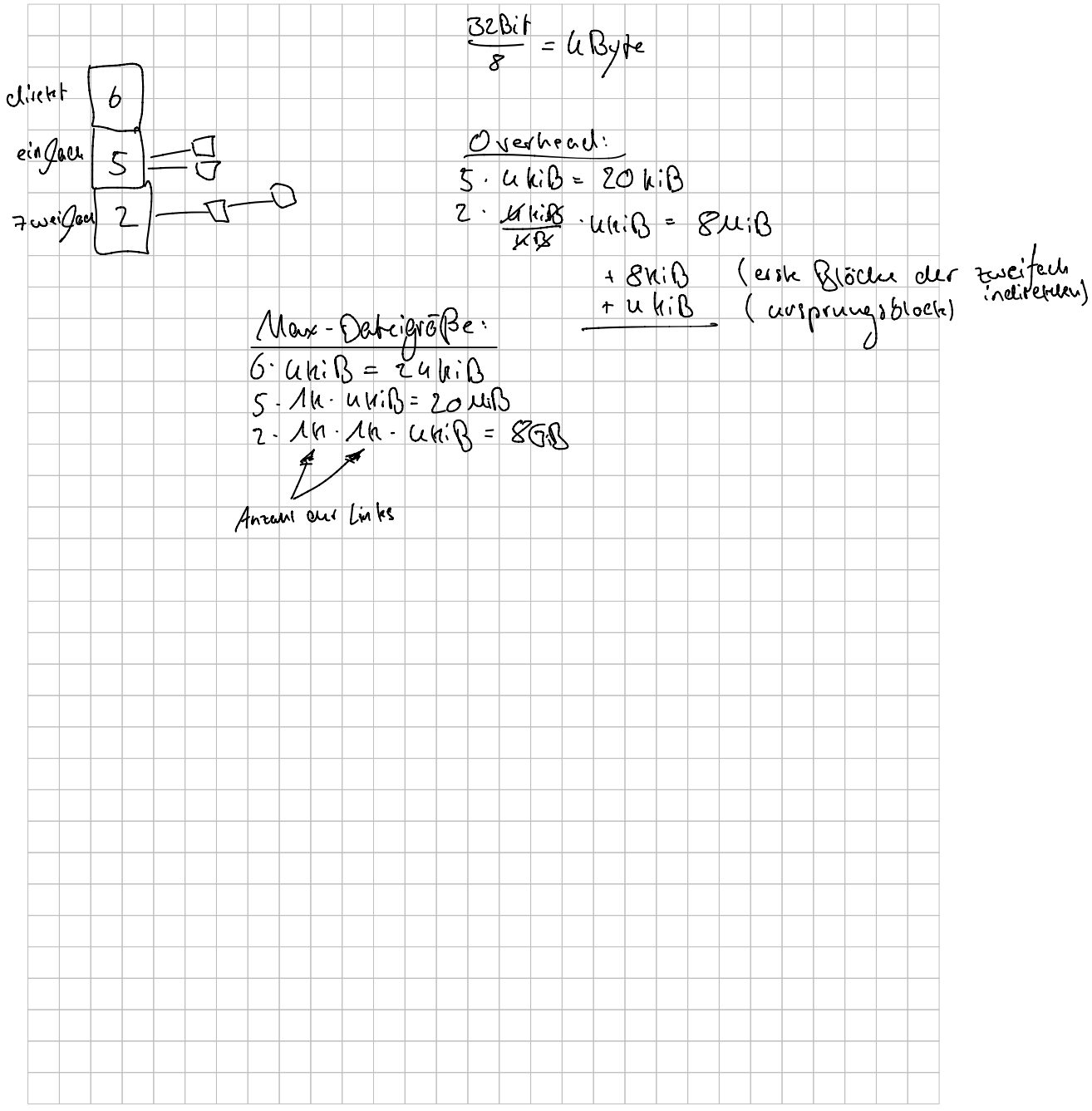
- ◯ haben die Aufgabe, Prozesse zwischen CPUs zu verschieben.
- ⊗ können bei jedem Interrupt durchgeführt werden.
- ◯ sichern nur Programmzähler und Statusregister des laufenden Prozesses.

2. I-Nodes

Dateisysteme dienen der Verwaltung von Daten in Dateien. I-Nodes sind Datenstrukturen, die Dateien repräsentieren. In einem I-Node-basierten Dateisystem gelten die folgenden Bedingungen:

- Die Blockgröße in dem Dateisystem beträgt 4 kiB.
- Jede I-Node speichere 6 direkte Links auf die ersten 6 Blöcke der Datei, 5 einfach-indirekte Links und 2 zweifach indirekte Links auf weitere Dateiblöcke.
- Blöcke, die für die indirekte Verlinkung weiterer Blöcke verwendet werden, speichern ausschließlich Links auf Dateiblöcke und keine weiteren Informationen.
- Jeder Link sei 32 Bit groß.

(2.1) (7 Punkte) Wieviel Speicherplatz benötigen die Blöcke zum Ablegen der Dateiorganisationsinformationen einer Datei maximal, d. h. bei der größtmöglichen Datei? Gehen Sie davon aus, dass die I-Node selbst einen Block benötigt.

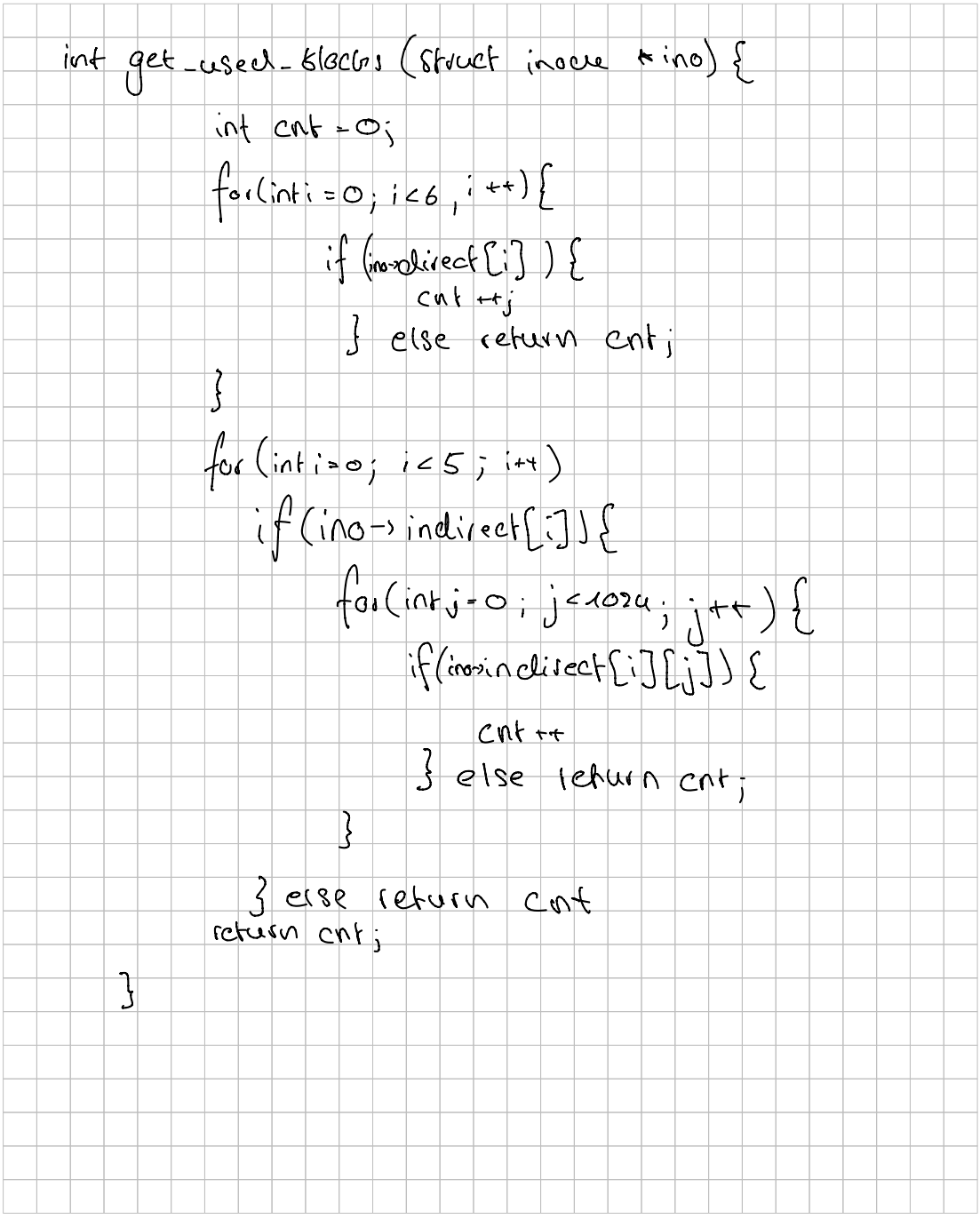


(2.2) (15 Punkte) Die Datenstruktur einer I-Node, die nur direkte und einfach indirekte Links habe, sieht in C wie folgt aus:

```
struct inode {
        . . .
        _p32    direct[6];
        _p32    *indirect[5];
};
```

Gehen Sie davon aus, dass eine Blockadresse den Datentyp `_p32` hat, d. h. `_p32` ist ein Zeiger auf einen Block.

Implementieren Sie nun eine Funktion `get_used_blocks(struct inode *ino)` in C, die die Anzahl der benutzten Blöcke für eine Datei ermittelt und zurückgibt, die durch den Zeiger *ino auf eine I-Node charakterisiert wird. Unbenutzte Einträge in den Zeigerstrukturen sollen durch den Eintrag `NULL` markiert sein.



3. **Freispeicherverwaltung**

Eine bitmap-basierte Freispeicherverwaltung in einem Dateisystem speichere markiere belegte Blöcke in der Bitmap mit einer 1 und freie Blöcke mit einer 0. Dabei werden die Belegtbits von 16 aufeinanderfolgenden Blöcke in einer 16-Bit-Zahl abgelegt, wobei das niedrigstwertige Bit immer dem Block mit der kleinsten Adresse entspricht.

Das zu Block 0 gehörende Belegtbit ist also im niedrigstwertigen Bit der ersten 16-Bit-Zahl abgelegt.

Gegeben sei eine Partition der Größe 10 GiB und einer Blockgröße von 2 kiB.

(3.1) (3 Punkte) Ermitteln Sie die Größe der Freispeicher-Bitmap.

$\frac{10\,GiB}{2\,kiB} = 5\,Mi$ (Bits)

Größe in B: $\frac{5\,Mi}{8} \approx 640\,kiB$

(3.2) (8 Punkte) Schreiben Sie in C eine Funktion, die ermittelt, ob es einen freien Block in der Bitmap des o. a. Dateisystems gibt.

```
int is_block_available(_u16 *bitmap)
```

```
int is_block_available(_u16 *bitmap){
    int size = 655360 /2;
    for(int i=0; i<size; i++){
        if(bitmap[i] != 0xFFFF){
            return 1;
        }
    }
    return 0;
}
```

4. **Links**

Viele Dateisysteme implementieren sowohl symbolische Links als auch Hardlinks.

(4.1) (2 Punkte) Erklären Sie kurz, was Links sind und wofür diese sinnvoll eingesetzt werden können.

um auf Datei von unterschiedlichen Orten zuzugreifen ohne den Inhalt jedes mal zu kopieren

(4.2) (3 Punkte) Erklären Sie den Unterschied zwischen Hardlinks und symbolischen Links. Warum kann ein Hardlink nicht auf eine Datei eines anderen Dateisystems zeigen?

1) Hardlink: 2 Dateien die auf den gleichen Speicherbereich zugreifen
SoftLink: Eine neue Datei vom Typ LINK, die die Adresse der Datei beinhaltet

2) Weil ein anderes Dateisystem anderen Speicherbereiche/ Adressen hat

Symbolische Links: ln -s
hard links: ln

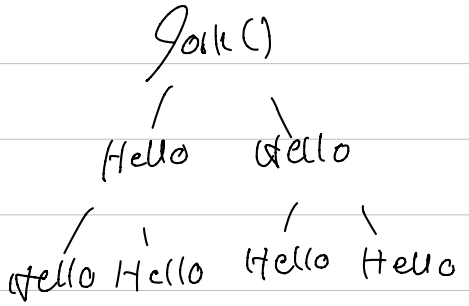
5. Prozesserzeugung

Prozesse unter UNIX-Systemen werden mittels des `fork()`-Systemaufrufs erzeugt.

(5.1) (4 Punkte) Gegeben sei das folgende C-Programm. Wie lautet die Ausgabe des Programmes? Begründen Sie kurz!

```
void main(){
    int x;
    for (x = 0; x < 2; x++){
        fork();
        printf("Hello!\n");
    }
}
```

6



(5.2) (3 Punkte) Wie findet ein Prozess nach einem `fork()`-Systemaufruf heraus, ob er der Eltern- der Kindprozess ist? Welche Rückgabewerte können auftreten?

return = 0 : Kindprozess

return = P-ID Kindprozess: Elternprozess

```
if (fork()) {
        // Elternprozess
} else {
        // Kindprozess
}
```

6. Scheduling

(6.1) (4 Punkte) Zielstellungen des Prozess-Schedulings in Betriebssystemen sind in der Regel vom Einsatzzweck des Systems abhängig.

Nennen Sie jeweils ein Schedulingverfahren, das vor allem bei interaktiven Systemen und eines, das vor allem bei Stapelverarbeitungssystemen eingesetzt wird. Begründen Sie kurz Ihre Entscheidung!

First-Come-First-Out (alle Prozesse wie beim Stapel von oben nach unten)

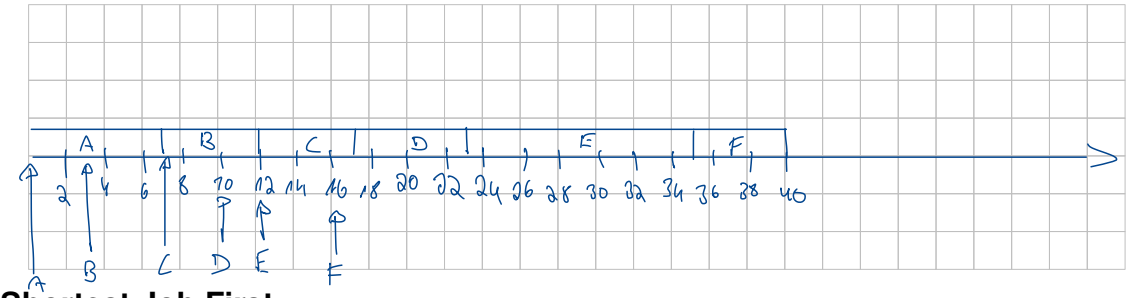
Round-Robin (interaktiv, da aktiv eingegriffen wird)

(6.2) (4 Punkte) In einem System gibt es 6 Prozesse mit in der folgenden Tabelle angegebenen Ankunft- und Laufzeiten:

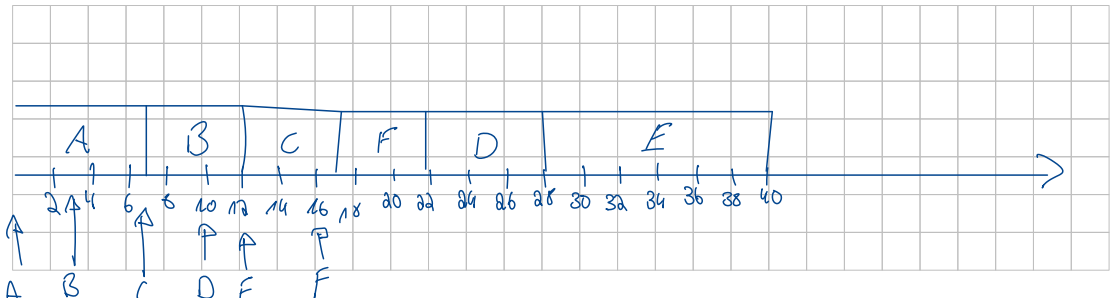
Prozess	CPU-Laufzeit [ms]	Ankunftszeit [ms]
A	7	0
B	5	3
C	5	7
D	6	10
E	12	12
F	5	16

Auf einem Einprozessorrechner sollen die Prozesse mit unterschiedlichen Ankunftszeiten verarbeitet werden. Skizzieren Sie die Ausführungsreihenfolge der Prozesse und die Zeitpunkte von deren Beginn und Ende auf einer Zeitleiste für

First Come First Serve



Shortest Job First



(6.3) (4 Punkte) Berechnen Sie die gesamten Verweilzeiten der Prozesse im System von Prozessankunft bis Prozessende. Tragen Sie dazu die Verweilzeiten aller Prozesse von der Ankunftzeit im System bis zur Beendigung für das jeweilige Verfahren

in die folgende Tabelle ein und berechnen Sie im Anschluss die durchschnittliche Verweilzeit pro Verfahren.

Verfahren	A	B	C	D	E	F	Durch-schnitt
First Come First Serve	7	9	10	13	23	24	$\frac{86}{6} = 14{,}33$
Shortest Job First	7	9	10	18	28	6	$\frac{78}{6} = 13$

7. Deadlockvermeidung

Ein Deadlock-Zustand kann auftreten, wenn in einer Menge von Prozessen jeder Prozess aus der Menge auf ein Ereignis wartet, das nur ein anderer Prozess aus der Menge auslösen kann.

(7.1) (4 Punkte) Was sind die vier Voraussetzungen, die erfüllt sein müssen, damit ein Deadlock entstehen kann?

- Wechselseitiger Ausschluss
- Hold-and-Wait-Bedingung
- Ununterbrechbarkeit
- Zyklische Wartebedingung

(7.2) (4 Punkte) Der Bankier-Algorithmus von Dijkstra ist eine Möglichkeit, Deadlocks zu verhindern. Berechnen Sie mit Hilfe des Bankier-Algorithmus', ob die im Folgenden dargestellte Zuteilung *sicher* ist:

Gegeben sei ein System mit 3 Prozessen A, B, C und einer Ressource, die **15** Mal im System verfügbar ist. Die folgende Tabelle gibt an, wie oft die Ressource von jedem Prozess verwendet wird und wie oft der Prozess sie maximal verwendet.

	nutzt	**max**
A	1	4
B	4	11
C	5	7

Ermitteln Sie, ob die gegebene Zuteilung sicher ist. Stellen Sie dafür die einzelnen Schritte des Bankiers-Algorithmus dar.

	nutzt	max
A	1	4
B	4	11
C	5	7

frei: 5

	nutzt	max
A	1	4
B	4	11
C	7	7

frei: 3

	nutzt	max
A	1	4
B	11	11
C	0	–

frei: 3

	nutzt	max
A	4	4
B	0	–
C	0	–

frei: 11

→ Zuteilung ist sicher