Aufgabe 1.1: Einfachauswahl-Fragen (4 Punkte)

Bei den Multiple-Choice-Fragen in dieser Aufgabe ist jeweils nur **eine** richtige Antwort eindeutig anzukreuzen. Auf die richtige Antwort gibt es die angegebene Punktzahl.

Wollen Sie eine Multiple-Choice-Antwort korrigieren, streichen Sie bitte die falsche Antwort mit drei waagrechten Strichen durch (☒)und kreuzen die richtige an.

Lesen Sie die Frage genau, bevor Sie antworten.

a) Wie funktioniert Adressraumschutz durch Eingrenzung? 2 Punkte

- ❏ Der Lader positioniert Programme immer so im Arbeitsspeicher, dass unerlaubte Adressen mit nicht-existierenden physikalischen Speicherbereichen zusammenfallen.
- ❏ Begrenzungsregister legen einen Adressbereich im logischen Adressraum fest, auf den alle Speicherzugriffe beschränkt werden.
- ❏ Begrenzungsregister legen einen Adressbereich im physikalischen Adressraum fest, auf den alle Speicherzugriffe beschränkt werden.
- ❏ Die MMU kennt die Länge eines Segments und verhindert Speicherzugriffe darüber hinaus.

b) Namensräume dienen u. a. der Organisation von Dateisystemen. Welche Aussage ist richtig? 2 Punkte

- ❏ Flache Namensräume sind besonders einfach implementierbar und damit vor allem für Mehrbenutzersysteme gut geeignet
- ❏ Der Nachteil von hierarchischen Namensräumen besteht darin, dass das Dateisystem spezielle Funktionen zum Auflösen von Namenskonflikten implementieren muss
- ❏ Hierarchische Namensräume werden erzeugt, indem man in einem Kontext symbolische Verweise auf Dateien einträgt
- ❏ In einem hierarchisch organisierten Namensraum dürfen gleiche Namen in unterschiedlichen Kontexten enthalten sein

Aufgabe 1.2: Mehrfachauswahl-Fragen (3 Punkte)

Bei den Multiple-Choice-Fragen in dieser Aufgabe sind jeweils *m* Aussagen angegeben, *n* ($0 \leq n \leq m$) Aussagen davon sind richtig. Kreuzen Sie **alle richtigen** Aussagen an. Jede korrekte Antwort in einer Teilaufgabe gibt einen halben Punkt, jede falsche Antwort einen halben Minuspunkt. Eine Teilaufgabe wird minimal mit 0 Punkten gewertet, d. h. falsche Antworten wirken sich nicht auf andere Teilaufgaben aus.

Wollen Sie eine falsch angekreuzte Antwort korrigieren, streichen Sie bitte das Kreuz mit drei waagrechten Strichen durch (☒).

Lesen Sie die Frage genau, bevor Sie antworten.

a) Gegeben sei folgendes Programmfragment: 3 Punkte

```
static int a;
int f1 (int x, const int* y) {
    static int b;
    int c;
    y++;
    ...
}
```

Welche der folgenden Aussagen zu den Variablen im Programm sind richtig?

- ❍ `b` ist uninitialisiert und enthält einen zufälligen Wert.
- ❍ Auf `a` kann von anderen Modulen aus zugegriffen werden.
- ❍ Wenn `f1` den Wert von `x` ändert, so beeinflusst dies den Aufrufer.
- ❍ `c` verliert beim Rücksprung aus `f1` seine Gültigkeit.
- ❍ `y` kann sowohl auf einen einzelnen `int` als auch auf ein Array von `int`s verweisen.
- ❍ Die Anweisung `y++` führt zu einem Compiler-Fehler, weil `y` als `const` deklariert ist.

Aufgabe 2: (15 Punkte)

Sie dürfen diese Seite und die Manual-Seite am Ende der Klausur zur besseren Übersicht bei der Programmierung heraustrennen!

Implementieren Sie eine Funktion

```
struct jobDesc* multithread(char* filename, int numLines,
                            void (*threadFunc)(struct jobDesc*));
```

die für jede Zeile einer Datei die als Parameter übergebene Funktion `threadFunc` in einem eigenen Thread ausführt. `multithread` liefert einen Zeiger auf ein Feld von Strukturen zurück, die die wesentlichen Informationen zu jedem einzelnen Thread enthalten. Für das Warten auf die Beendigung der gestarteten Threads ist der Aufrufer zuständig - es ist nicht Bestandteil dieser Aufgabe.

Die Funktion `multithread` liest die als Parameter übergebene Datei `filename` zeilenweise ein und startet für jede Zeile einen neuen Thread, der die Funktion `threadFunc` ausführt. (Sie können davon ausgehen, dass die Datei immer genau `numLines` Zeilen enthält und keine Zeile länger als 100 Zeichen ist)

Die Funktion `threadFunc` führt eine beliebige, nicht näher beschriebene Operation auf den Eingabedaten aus, die als Ergebnis eine Zahl (`int`) liefert.

Die Übergabe der eingelesenen Zeile an den neu erzeugten Thread und die Rückgabe des Ergebnisses erfolgen mit Hilfe der Struktur `jobDesc`. Das Feld `line` ist für die eingelesene Zeile vorgesehen. Die Funktion `threadFunc` speichert das Ergebnis ihrer Berechnung im Feld `output`. Das Feld `tid` benötigt der Aufrufer, um auf die Beendigung des Threads zu warten.

Tritt beim Einlesen einer Zeile oder der Erzeugung eines Threads ein Fehler auf, so wird der Wert der `errno`-Variable im Feld `error` der dazugehörigen Struktur gespeichert und die Ausführung der Funktion `multithread` normal fortgesetzt. Tritt ein andererer Fehler bei der Ausführung von `multithread` auf, so liefert die Funktion `NULL` zurück. Sonst wird ein Zeiger auf ein Feld aus `jobDesc`-Strukturen zurückgeliefert.

```
#include <unistd.h>
#include <stdlib.h>
#include <string.h>
#include <pthread.h>
#include <stdio.h>
#include <errno.h>

struct jobDesc {
    pthread_t tid;
    char input[101];
    int output;
    int error;
};
```



```
/* Funktion multithread */
```

Aufgabe 3: (8 Punkte)

In welchen Zuständen kann sich ein Prozess befinden? Beschreiben Sie die Zustände kurz in Stichworten. Durch welche Ereignisse werden die Zustandsübergänge ausgelöst? Illustrieren Sie Ihre Erläuterungen ggf. durch eine kleine Skizze.