Aufgabe 1: Ankreuzfragen (7 Punkte)

1) Einfachauswahlfragen (4 Punkte)

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

Wollen Sie eine 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) Welche der folgenden Aussagen zum Thema Threads ist richtig? 2 Punkte

- ☐ Die Umschaltung von User-Threads ist eine privilegierte Operation und muss deshalb im Systemkern erfolgen.
- ☐ Bei User-Threads ist die Scheduling-Strategie nicht durch das Betriebssystem vorgegeben.
- ☐ Zu jedem Kernel-Thread gehört ein eigener, geschützter Adressraum.
- ☐ Kernel-Threads können Multiprozessoren nicht ausnutzen.

b) Welche der folgenden Aussagen zum Thema Dateisysteme sind richtig? 2 Punkte

- ☐ Symbolische Links können nur auf Dateien innerhalb des selben Dateisystems verweisen.
- ☐ Auf das Wurzelverzeichnis (root directory, „/“) darf immer nur genau ein hard-link verweisen.
- ☐ Zum Anlegen oder Löschen von Dateien sind die Schreibzugriffsrechte auf das übergeordnete Verzeichnis irrelevant.
- ☐ In einem hierarchisch organisierten Namensraum dürfen gleiche Namen in unterschiedlichen Kontexten enthalten sein.

2) Mehrfachauswahlfragen (3 Punkte)

Bei den Mehrfachauswahlfragen in dieser Aufgabe sind jeweils m Aussagen angegeben, davon sind n ($0 \leq n \leq m$) Aussagen richtig. Kreuzen Sie alle richtigen Aussagen an.
Jede korrekte Antwort in einer Teilaufgabe gibt einen Punkt, jede falsche Antwort einen 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 bzgl. dieses Programms sind korrekt?

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

Aufgabe 2: spquota (15 Punkte)

Sie dürfen diese Seite zur besseren Übersicht bei der Programmierung heraustrennen!

Schreiben Sie ein Program `spquota`, das den im aktuellen Verzeichnis beanspruchten Speicherplatz erhöht und Dateiinhalte zerstört. Dies soll geschehen, indem `spquota` den momentanen Speicherverbrauch aller **regulären** Dateien im Verzeichnis ermittelt und der Inhalt einer beliebigen Datei überschrieben wird. Sollte das Verzeichnis leer sein, soll eine neue Datei „spquota“ angelegt werden. Der Inhalt der ausgewählten Datei soll mit dem Byte-Wert `0x42`, in ASCII `'B'`, überschrieben werden. Die Anzahl der zu schreibenden Bytes ergibt sich aus der Summe der Größe aller regulären Dateien plus eins. Nutzen Sie `fopen(3)` mit `"w"` zum Schreiben der Datei.

Nur Dateien im aktuellen Verzeichnis sollen berücksichtigt werden. Es ist **kein** rekursiver Abstieg zu implementieren. Achten Sie auf korrekte Ressourcenverwaltung.

Beispielhafter Aufruf von `spquota` und dessen Verhalten in verschiedenen Verzeichnissen:

```
sp@~: stat -c '%s B: %n' ./*
106764 B: ./2019s-SP-klausur.pdf
112044 B: ./2019s-SP-loesung.pdf
sp@~: spquota
sp@~: stat -c '%s B: %n' ./*
218809 B: ./2019s-SP-klausur.pdf
112044 B: ./2019s-SP-loesung.pdf
```

```
sp@~: ls
sp@~: spquota
sp@~: ls
spquota
sp@~: stat -c '%s B: %n' ./*
1 B: ./spquota
```

Hinweise:

- Zur Vereinfachung dürfen Sie die Ausführung im Fehlerfall mittels `die()` abbrechen.
- Die Konstante `NAME_MAX` beschreibt die maximale Länge eines Dateinamens in Bytes.

```c
#include <dirent.h>
#include <errno.h>
#include <limits.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

// Hilfsfunktionen
static void die(const char *msg) {
  perror(msg);
  exit(EXIT_FAILURE);
}
```



```c
int main(int argc, char **argv) {
  // lokale Variablen

  // Auslesen des aktuellen Verzeichnisses
```

```
  // Schreiben der Datei
  return EXIT_SUCCESS;
}
```

S:

Aufgabe 3: Traps/Interrupts (8 Punkte)

1) In der Systemprogrammierung werden zwei Kategorien von Programmunterbrechungen (Ausnahmen) unterschieden. Welche sind dies? Grenzen Sie diese kurz voneinander ab. (2 Punkte)

2) Wenn das folgende Programmstück in einem UNIX-System abläuft, wird ein Fehler auftreten: (3 Punkte)

```
int *p = NULL;
*p = -1;
```

a) Warum tritt bei der Ausführung des Programmstückes ein Fehler auf? (1 Punkt)

b) Welche Hardwarekomponente entdeckt dieses Fehler zuerst und wie teilt sie diesen Fehler dem Betriebssystem mit? (2 Punkte)

3) Zur Behandlung von Unterbrechungen werden Mantelprozeduren (*wrapper*) angesprungen. Welche Arbeitsschritte sind in diesen Prozeduren notwendig, um die Unterbrechung korrekt zu behandeln? (3 Punkte)