Aufgabe 1.1: Einfachauswahl-Fragen (3 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) Man unterscheidet Programmunterbrechungen in Traps und Interrupts. Welche Aussage ist richtig? 1,5 P.

- ❏ Bei der mehrfachen Ausführung eines unveränderten Programms mit gleichen Eingabedaten treten Interrupts immer an den gleichen Stellen auf.
- ❏ Normale Rechenoperationen können zu einem Trap führen.
- ❏ Ein gerade laufendes Maschinenprogramm kann bei Bedarf die Behandlung aller Programmunterbrechungen unterdrücken.
- ❏ Weil das Betriebssystem nicht vorhersagen kann, wann ein Prozess einen Systemaufruf tätigt, sind Systemaufrufe in die Kategorie Interrupt einzuordnen.

b) In einem UNIX-UFS-Dateisystem gibt es symbolische Namen/Verweise (Symbolic Links) und feste Links (Hard Links) auf Dateien. Welche Aussage ist richtig? 1,5 P.

- ❏ Auf jedes Verzeichnis verweist immer genau ein Hard-Link.
- ❏ Wird der letzte Symbolic Link auf eine Datei gelöscht, so wird auch die Datei selbst gelöscht.
- ❏ Für jede reguläre Datei existiert mindestens ein Hard-Link im selben Dateisystem.
- ❏ Ein Symbolic Link kann nicht auf Dateien anderer Dateisysteme verweisen.

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) Welche der folgenden Aussagen zum Thema Prozesse und Threads sind richtig? 3 Punkte

- ❍ Beim Blockieren eines schwergewichtigen Prozesses werden alle anderen schwergewichtigen Prozesse, die das selbe Progamm ausführen, ebenfalls blockiert.
- ❍ Leichtgewichtige Prozesse (Kernel-Level-Threads) können Multiprozessoren ausnutzen.
- ❍ Bei federgewichtigen Prozessen (User-Level-Threads) ist die Schedulingstrategie fest durch das Betriebssystem vorgegeben.
- ❍ Zu jedem leichtgewichtigen Prozess (Kernel-Level-Thread) gehört ein eigener, isolierter Adressraum.
- ❍ Federgewichtige Prozesse (User-Level-Threads) blockieren sich bei blockierenden Systemaufrufen gegenseitig.
- ❍ Die Umschaltung von leichtgewichtigen Prozesssen (Kernel-Level-Threads) kann nur im Systemkern erfolgen.

Aufgabe 2: (15 Punkte)

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

Schreiben Sie eine Funktion

```
int needsBackup(const char targetdir[]);
```

welche im aktuellen Arbeitsverzeichnis alle Dateien sucht, von denen ein Backup im Zielverzeichnis **`targetdir`** erstellt werden muss.

Hierzu wird das aktuelle Arbeitsverzeichnis (**`"."`**) nicht-rekursiv nach regulären Dateien durchsucht. Existiert im Zielverzeichnis keine Datei mit dem gleichen Namen oder ist das Änderungsdatum der Datei im aktuellen Arbeitsverzeichnis jünger als das der Version im Zielverzeichnis, so soll der Name der Datei auf der Standardausgabe ausgegeben werden.

Tritt bei der Bearbeitung des Verzeichnisses ein schwerwiegender Fehler auf, so gibt **`needsBackup`** den Wert -1 zurück, ansonsten den Wert 0.

Die Funktion soll außer den Dateinamen keine weiteren Meldungen ausgeben, auch keine Fehlermeldungen.

Anmerkungen:

- Pfad-Konkatenationen können entweder mit **`strcpy(3)`** und **`strcat(3)`** oder mit **`sprintf(3)`** durchgeführt werden.
- Zur Vereinfachung können Sie davon ausgehen, dass die Funktion **`lstat(2)`** nur dann fehlschlägt, wenn der übergebene Pfad nicht existiert.

```
#include <dirent.h>
#include <errno.h>
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>
#include <stdio.h>
```



```
// Funktion needsBackup()
```

Aufgabe 3: (9 Punkte)

a) Gegegeben sei das nachfolgende Programm.

Skizzieren Sie den Aufbau des logischen Adressraums eines Prozesses, der dieses Programm ausführt.

- Zeichnen Sie für jede Variable ein, wo diese ungefähr im logischen Adressraum zu finden sein wird und zu welchem Segment dieser Bereich gehört.
- Wohin zeigen die Zeiger-Variablen (ungefähre Position einzeichnen!)?

```
const char *str = "content";

int main() {
  static int var = 5;
  int *pt;

  pt = malloc(10*sizeof(int));
  pt[5] = 42;

  str[var] = 's';

  return 0;
}
```

b) Das obige Programmstück enthält Programmierfehler. Beschreiben Sie diese und deren (mögliche) Auswirkungen auf die Übersetzung oder Ausführung des Programmes.