Aufgabe 1: (6 Punkte)

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

Wollen Sie eine Multiple-Choice-Antwort korrigieren, kreisen Sie bitte die falsche Antwort ein und kreuzen die richtige an.

Lesen Sie die Frage genau, bevor Sie antworten.

a) Welche Aussage zum Thema Prozessmodelle ist **falsch**? 2 Punkte

- ❏ Beim schwergewichtigen Prozess bilden die Prozessinstanz und der Benutzeradressraum eine Einheit.
- ❏ Bei federgewichtigen Prozessen werden mehrere Kernfäden auf einen Benutzerfaden abgebildet.
- ❏ Die Umschaltung zweier federgewichtiger Prozesse ist ohne Wechsel des Adressraums möglich.
- ❏ Mehrere leichtgewichtige Prozesse können innerhalb eines Adressraums ablaufen.

b) Welche Aussage zum Thema Adressräume ist **richtig**? 2 Punkte

- ❏ In einem physikalischen Adressraum ist jede Adresse gültig.
- ❏ Die Abbildung von logischen Adressen auf physikalische Adressen erfolgt zur Laufzeit durch den Compiler.
- ❏ Segmentierung schränkt einen logischen Adressraum derart ein, dass nur auf gültige Speicheradressen erfolgreich zugegriffen werden kann.
- ❏ Virtuelle Adressräume sind Voraussetzung für die Realisierung logischer Adressräume.

c) Welche Aussage zum Thema Programmunterbrechungen ist **richtig**? 2 Punkte

- ❏ Der Zugriff auf eine logische Adresse kann zu einem Trap führen.
- ❏ Bei der mehrfachen Ausführung eines unveränderten Programms mit gleicher Eingabe treten Interrupts immer an den gleichen Stellen auf.
- ❏ Der Zeitgeber (Systemuhr) unterbricht die Programmbearbeitung in regelmäßigen Abständen. Die genaue Stelle der Unterbrechungen ist damit vorhersagbar.
- ❏ Wenn ein Interrupt einen schwerwiegenden Fehler signalisiert, muss das unterbrochene Programm abgebrochen werden.

Aufgabe 2: (15 Punkte)

Implementieren Sie eine Funktion **`sorted_exec`**, die ein Programm (Parameter `prg`) *n*-malig mit jeweils genau einem Argument ausführt. Der Funktion werden hierzu in einer `NULL`-terminierten, unsortierten Liste *n* Argumente übergeben (Parameter `args`).

Das Programm `prg` wird nun mit je einem Argument aus der alphabetisch sortierten Liste nacheinander in je einem eigenen Prozess ausgeführt. Die Funktion **`sorted_exec`** kehrt erst dann zurück, wenn alle gestarteten Prozesse beendet sind.

Tritt in der Funktion **`sorted_exec`** im Vaterprozess ein Fehler auf, so werden keine weiteren Prozesse mehr gestartet. Terminiert ein Kindprozess nicht mit Exit-Status 0, so werden die übrigen Prozesse aber noch ausgeführt.

Schnittstelle:

```
int sorted_exec(char *prg, char *args[]);
```

Rückgabewert:

- `-1`: bei der Bearbeitung trat ein Fehler auf oder mindestens einer der gestarteten Kindprozesse terminierte nicht mit dem Exit-Status 0
- `0`: die Ausführung der Funktion verlief frei von Fehlern, alle gestarteten Prozesse kehrten mit dem Exit-Status 0 zurück

Hinweise:

- Das Array **`args`** darf direkt sortiert werden.
- Die zum Sortieren notwendige Hilfs-Vergleichsfunktion muss ebenfalls von Ihnen implementiert werden.
- Das übergebene Programm soll im Systemsuchpfad für Programme gesucht werden, falls der Programmname keine Pfadangabe beinhaltet.
- Die Funktion soll keine Meldungen ausgeben, auch keine Fehlermeldungen.

```
#include <stdlib.h>
#include <unistd.h>
#include <string.h>

/* Hilfs-Vergleichsfunktion für qsort */
```

```
/* Funktion sorted_exec */
```


Aufgabe 3: (9 Punkte)

a) Beschreiben Sie anhand einer Skizze, wie hierarchische Namensräume in UNIX-Dateisystemen realisiert sind. Aus Ihrer Skizze soll der Zusammenhang zwischen Verzeichnissen (Katalogen), der Inodetabelle, den Inodes (Dateiköpfen) und Dateien hervorgehen. Die Skizze soll außerdem beinhalten, welche Art von Informationen in Verzeichniseinträgen und Inodes jeweils gespeichert werden.

b) Wie sind in diesem Zusammenhang *Hard Links* realisiert und welche Rolle spielen sie für die Navigation im Dateibaum? Warum wären beliebige Hard-Links auf Verzeichnisse problematisch und werden daher von gängigen Dateisystemen nicht erlaubt?