Aufgabe 1.1: Einfachauswahl-Fragen (18 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) Welche Aussage zum Thema Adressraumverwaltung ist richtig? 3 Punkte

- ❏ Da das Laufzeitsystem auf die Betriebssystemschnittstelle zur Speicherverwaltung zurückgreift, ist die Granularität der von `malloc()` zurückgegebenen Speicherblöcke vom Betriebssystem vorgegeben.
- ❏ Mit Hilfe des Systemaufrufes `malloc()` kann ein Programm zusätzliche Speicherblöcke von sehr feinkörniger Struktur vom Betriebssystem anfordern.
- ❏ Ein Speicherbereich, der mit Hilfe der Funktion `free()` freigegeben wurde, verbleibt im logischen Adressraum des zugehörigen Prozesses.
- ❏ Mit `malloc()` angeforderter Speicher, welcher vor Programmende nicht freigegeben wurde, kann vom Betriebssystem nicht mehr an andere Prozesse herausgegeben werden und ist damit bis zum Neustart des Systems verloren.

b) Man unterscheidet Traps und Interrupts. Welche Aussage ist richtig? 3 Punkte

- ❏ Normale Rechenoperationen können 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. Somit sind solche Unterbrechungen in die Kategorie Trap einzuordnen.
- ❏ Wenn ein Interrupt einen schwerwiegenden Fehler signalisiert, muss das unterbrochene Programm abgebrochen werden.

c) Ein *laufender* Prozess wird in den Zustand *blockiert* überführt. Welche Aussage passt zu diesem Vorgang? 2 Punkte

- ❏ Der Prozess terminiert.
- ❏ Der Prozess wartet auf Daten von der Festplatte.
- ❏ Es ist kein direkter Übergang von *laufend* nach *blockiert* möglich.
- ❏ Der Prozess liest von der Festplatte mit nichtblockierenden Eingabeoperationen.

d) Was passiert, wenn Sie in einem C-Programm über einen ungültigen Zeiger versuchen auf Speicher zuzugreifen? 2 Punkte

- ❏ Das Betriebssystem erkennt die ungültige Adresse bei der Weitergabe des Befehls an die CPU (partielle Interpretation) und leitet eine Ausnahmebehandlung ein.
- ❏ Die MMU erkennt die ungültige Adresse bei der Adressumsetzung und löst einen Trap aus.
- ❏ Beim Laden des Programms wird die ungültige Adresse erkannt und der Speicherzugriff durch einen Sprung auf eine Abbruchfunktion ersetzt. Diese Funktion beendet das Programm mit der Meldung "Segmentation fault".
- ❏ Der Compiler erkennt die problematische Code-Stelle und generiert Code, der zur Laufzeit bei dem Zugriff einen entsprechenden Fehler auslöst.

e) Welche Aussage über den Rückgabewert von `fork()` ist richtig? 1 Punkt

- ❏ Der Kind-Prozess bekommt die Prozess-ID des Vater-Prozesses.
- ❏ Im Fehlerfall wird im Kind-Prozess -1 zurückgeliefert.
- ❏ Der Rückgabewert ist in jedem Prozess (Kind und Vater) jeweils die eigene Prozess-ID.
- ❏ Dem Vater-Prozess wird die Prozess-ID des Kind-Prozesses zurückgeliefert.

f) Nehmen Sie an, der Ihnen bekannte Systemaufruf stat(2) wäre analog zu der Funktion readdir(3) mit folgender Schnittstelle implementiert: 3 Punkte
`struct stat *stat(const char *path);`
Welche Aussage ist richtig?

- ❏ Der Systemaufruf liefert einen Zeiger zurück, über den die aufrufende Funktion direkt auf eine Datenstruktur zugreifen kann, die die Dateiattribute enthält.
- ❏ Der Aufrufer muss sicherstellen, dass er den zurückgelieferten Speicher mit free(3) wieder freigibt, wenn er die Dateiattribute nicht mehr weiter benötigt.
- ❏ Ein Zugriff über den zurückgelieferten Zeiger liefert völlig zufällige Ergebnisse oder einen Segmentation fault.
- ❏ Durch den Zugriff über den zurückgegebenen Zeiger ist es möglich, die Inode-Informationen auf dem Datenträger direkt zu verändern.

g) Man unterscheidet die Begriffe Programm und Prozess. Welche der folgenden Aussagen zu diesem Themengebiet ist richtig? 2 Punkte

- ❏ Das Programm ist der statische Teil (Rechte, Speicher, etc.), der Prozess der aktive Teil (Programmzähler, Register, Stack).
- ❏ Wenn ein Programm nur einen aktiven Ablauf enthält, nennt man diesen Prozess, enthält das Programm mehrere Abläufe, nennt man diese Threads.
- ❏ Ein Prozess ist ein Programm in Ausführung - ein Prozess kann aber auch mehrere verschiedene Programme ausführen
- ❏ Ein Programm kann immer nur von einem Prozess ausgeführt werden

h) Welche der folgenden Aussagen zu statischem bzw. dynamischem Binden ist richtig? 2 Punkte

- ❏ bei dynamischem Binden müssen zum Übersetzungszeitpunkt alle Adressbezüge vollständig aufgelöst werden
- ❏ beim statischen Binden werden alle Adressen zum Ladezeitpunkt aufgelöst
- ❏ dynamisch gebundene Programme können auch noch zur Laufzeit durch das Nachladen neuer Programmodule (plug-ins) ergänzt werden
- ❏ bei statischem Binden werden durch den Compiler alle Adressbezüge vollständig aufgelöst

Aufgabe 1.2: Mehrfachauswahl-Fragen (4 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 Teilaufgabse 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: 4 Punkte

```
static int a = 2012;
int f1 (const int *y) {
    static int b;
    int c;
    char *d = malloc(2407);
    int (*e)(const int *) = f1;
    y++;
    ...
}
```

Welche der folgenden Aussagen zum obigen Programmfragment sind richtig?

- ❍ `a` liegt im Datensegment.
- ❍ `b` liegt im Stacksegment.
- ❍ `c` ist mit dem Wert 0 initialisiert.
- ❍ `d` ist ein Zeiger, der in den Heap zeigt.
- ❍ Die Speicherstelle, auf die `d` zeigt, verliert beim Rücksprung aus der Funktion `f1()` ihre Gültigkeit.
- ❍ `e` liegt im Stacksegment und zeigt in das Textsegment.
- ❍ Die Anweisung `y++` führt zu einem Laufzeitfehler, da `y` konstant ist.
- ❍ `y` liegt im Stacksegment.

Aufgabe 2: pdu (45 Punkte)

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

Schreiben Sie ein Programm **`pdu`** (parallel disk usage), das die Größe aller übergebenen Verzeichnisbäume parallel ermittelt und anschließend sortiert ausgibt. Der Grad der Parallelität kann durch den Aufrufer per Befehlszeile festgelegt werden.

Das Programm bekommt auf der Befehlszeile vom Benutzer als ersten Parameter die maximale Anzahl an gleichzeitig laufenden Threads. Für diesen Parameter sind nur Werte zwischen 1 und 399 zulässig. Die folgenden Parameter sind die Namen der zu durchsuchenden Verzeichnisbäume.

Das Programm soll folgendermaßen arbeiten:

- Der Haupt-Thread startet für jeden übergebenen Verzeichnisbaum die Funktion `tstart` in einem eigenen Thread. Ist die maximale Anzahl an gleichzeitig laufenden Threads erreicht wartet der Haupt-Thread passiv, bis wieder ein neuer Thread gestart werden kann.
  Nachdem für jeden Verzeichnisbaum die Größe parallel ermittelt wurde, gibt der Hauptthread eine Liste der Verzeichnisnamen zusammen mit der Größe aus.
  Trat bei der Bearbeitung eines Vereichnisbaums ein Fehler auf, so wird als Größe -1 angezeigt.
- Funktion **`void *tstart(struct threadArg *arg)`**: Berechnet für das per Struktur übergebene Verzeichnis die Größe mit Hilfe der rekusiven Funktion `getDirSize` und legt das Ergebnis in der Struktur ab.
- Funktion **`int getDirSize(char *path)`**: Durchsucht das als Parameter übergebene Verzeichnis und summiert die Größe aller Einträge rekursiv auf. Dabei sollen nur Verzeichnise und reguläre Dateien berücksichtigt werden. Tritt bei der Bearbeitung ein Fehler auf, wird die Rekursion abgebrochen und -1 zurückgegeben.

Ihnen steht das aus der Übung bekannte Semaphoren-Modul zur Verfügung. Die Schnittstelle finden Sie im folgenen Programmgerüst nach den #include-Anweisungen.

Auf den folgenden Seiten finden Sie ein Gerüst für das beschriebene Programm. In den Kommentaren sind nur die wesentlichen Aufgaben der einzelnen zu ergänzenden Programmteile beschrieben, um Ihnen eine gewisse Leitlinie zu geben. Es ist überall sehr großzügig Platz gelassen, damit Sie auch weitere notwendige Programmanweisungen entsprechend Ihrer Programmierung einfügen können.

Einige wichtige Manual-Seiten liegen bei - es kann aber durchaus sein, dass Sie bei Ihrer Lösung nicht alle diese Funktionen oder gegebenenfalls auch weitere Funktionen benötigen.


```
/* includes */
#include <dirent.h>
#include <errno.h>
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>

#include "sem.h"

SEM *sem_init(int initVal);
int sem_del(SEM *sem);
void P(SEM *sem);
void V(SEM *sem);

static void die(const char message[]) {
        perror(message);
        exit(EXIT_FAILURE);
}

// Makros, Funktionsdeklarationen, Strukturen, globale Variablen
```

```
// Funktion main
```

```
    // Argumente auswerten und weitere Initialisierungen
```

A:


```
//  Verzeichnisse parallel bearbeiten
```

```
// Auf Terminierung der Threads warten, Ergebnis ausgeben
```

```
// Ende Funktion main
```

T:

```
// Thread-Funktion
```

```
// Funktion getDirSize
```

```
   // Directory-Eintraege lesen
```



```
   // Groesse der relevanten Eintraege ermitteln und summieren
```

D:

Aufgabe 3: (11 Punkte)

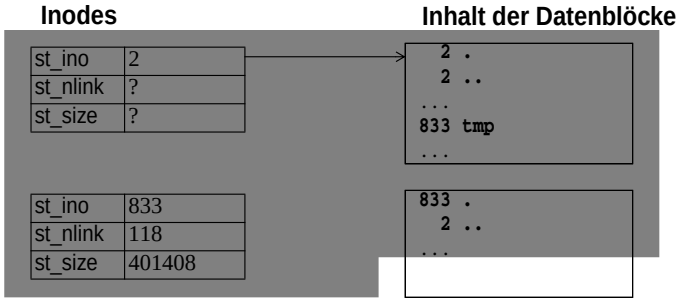
Gegeben ist die folgende Ausgabe des Kommandos **`ls -alRi /tmp/sp`** (rekursiv absteigende Ausgabe aller Dateien und Verzeichnisse unter `/tmp/sp` mit Angabe der Inode-Nummer) auf einem Linux-System. (11 Punkte)

```
fauixx>  ls -alRi /tmp/sp
/tmp/sp:
total 408
 91 drwxrwxr-x   4 jklein i4staff   4096 Jul 19 15:17 .
833 drwxrwxrwt 118 root   root    401408 Jul 19 15:30 ..
 96 drwxrwxr-x   2 jklein i4staff   4096 Jul 19 15:21 dir1
 98 drwxrwxr-x   2 jklein i4staff   4096 Jul 19 15:22 dir2

/tmp/sp/dir1:
total 68
 96 drwxrwxr-x 2 jklein i4staff  4096 Jul 19 15:21 .
 91 drwxrwxr-x 4 jklein i4staff  4096 Jul 19 15:17 ..
536 -rwxr--r-- 2 jklein i4staff 52224 Jul 19 15:20 datei1
258 -rw-rw-r-- 1 jklein i4staff    30 Jul 19 15:21 datei2

/tmp/sp/dir2:
total 68
 98 drwxrwxr-x 2 jklein i4staff  4096 Jul 19 15:22 .
 91 drwxrwxr-x 4 jklein i4staff  4096 Jul 19 15:17 ..
896 lrwxrwxrwx 1 jklein i4staff    14 Jul 19 15:22 dat1 -> ../dir1/datei2
536 -rwxr--r-- 2 jklein i4staff 52224 Jul 19 15:20 xxx
900 -rw-rw-r-- 1 jens   i4staff    30 Jul 19 15:22 yyy
```

Ergänzen Sie im weißen Bereich die auf der folgenden Seite im grauen Bereich bereits angefangene Skizze der Inodes und Datenblöcke des Linux-Dateisystems um alle entsprechenden Informationen, die Sie aus obiger Ausgabe entnehmen können.



st_ino	
st_nlink	
st_size	

st_ino	
st_nlink	
st_size	

st_ino	
st_nlink	
st_size	

st_ino	
st_nlink	
st_size	

st_ino	
st_nlink	
st_size	

st_ino	
st_nlink	
st_size	

st_ino	
st_nlink	
st_size	

st_ino	
st_nlink	
st_size	

Aufgabe 4: (12 Punkte)

Beschreiben Sie in Stichworten die unterschiedlichen Arten ("Gewichtsklassen") von Prozessen/Threads (Eigenschaften, Vorteile, Nachteile).