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

- ❏ Ein Prozess kann durch mehrere Programme ausgeführt werden.
- ❏ Mit Hilfe des Systemaufrufs *exec* wird das bestehende Programm im aktuell laufenden Prozess ersetzt.
- ❏ Der Binder erzeugt aus einer oder mehreren Objekt-Dateien einen Prozess.
- ❏ Der Prozess ist der statische Teil (Rechte, Speicher, etc.), das Programm der aktive Teil (Programmzähler, Register, Stack).

b) Man unterscheidet Traps und Interrupts. Welche Aussage ist richtig? 2 Punkte

- ❏ Das Betriebssystem kann Interrupts, die in ursächlichem Zusammenhang mit dem gerade laufenden Prozess stehen, nach dem Beendigungsmodell behandeln, wenn eine sinnvolle Fortführung des Prozesses nicht mehr möglich ist
- ❏ Der Zugriff auf eine physikalische Adresse kann zu einem Trap führen.
- ❏ Traps dürfen nicht nach dem Wiederaufnahmemodell behandelt werden, da ein Trap immer einen schwerwiegenden Fehler signalisiert.
- ❏ Ganzzahl-Rechenoperationen können nicht zu einem Trap führen.

c) Welche Aussage zum Thema Threads ist richtig? 2 Punkte

- ❏ Zu jedem Kernel-Thread gehört ein eigener isolierter Adressraum.
- ❏ Kernel-Threads können Multiprozessoren nicht ausnutzen.
- ❏ Userlevel-Threads blockieren sich bei blockierenden Systemaufrufen gegenseitig.
- ❏ Zur Umschaltung von Userlevel-Threads ist ein Adressraumwechsel erforderlich.

d) Welche Aussage zum Thema Adressraumschutz ist richtig? 2 Punkte

- ❏ Beim Adressraumschutz durch Abteilung wird der logische Adressraum in mehrere Segmente mit unterschiedlicher Semantik unterteilt.
- ❏ Adressraumschutz durch Eingrenzung benötigt keinen verschiebenden Lader, um Programmadressen an Arbeitsspeicheradressen zu binden
- ❏ Beim Einsatz von Segmentierung ist es möglich, dass dieselbe logische Adresse in unterschiedlichen logischen Adressräumen auf unterschiedliche physikalische Adressen verweist.
- ❏ In einem segmentierten Adressraum kann zur Laufzeit kein weiterer Speicher mehr dynamisch nachgefordert werden.

e) Welche Aussage zum Thema Betriebsarten ist richtig? 2 Punkte

- ❏ Mehrzugangsbetrieb ist nur in Verbindung mit CPU- und Speicherschutz sinnvoll realisierbar.
- ❏ Beim Stapelbetrieb können keine globalen Variablen existieren, weil alle Daten im Stapel-Segment (Stack) abgelegt sind.
- ❏ Mehrprogrammbetrieb ermöglicht die simultane Ausführung mehrerer Programme innerhalb desselben Prozesses.
- ❏ Echtzeitsysteme findet man hauptsächlich auf großen Serversystemen, die eine enorme Menge an Anfragen zu bearbeiten haben.

f) Welche Aussage zum Thema Speicherverwaltung ist richtig? 2 Punkte

- ❏ Sollte bei einer Speicheranforderung mittels `malloc` nicht genügend Speicher vorhanden sein, wird der Prozess vom Betriebssystem beendet.
- ❏ Ohne spezielle Unterstützung durch das Betriebssystem kann das Laufzeitsystem nur eine vorab statisch festgelegte Menge an Speicher feingranular verwalten.
- ❏ Speicherbereiche, die vor Beendigung des Prozesses nicht mit `free` freigegeben wurden, sind bis zum Neustart des Systems unwiederbringlich verloren.
- ❏ Speicherbereiche, die im logischen Adressraum zusammenhängend sind, müssen auch im physikalischen Hauptspeicher zusammenhängend sein.

g) 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.
- ❏ Ein Kindprozess des Prozesses terminiert.

h) Was versteht man unter Virtuellem Speicher? 2 Punkte

- ❏ Speicher, der nur im Betriebssystem sichtbar ist, jedoch nicht für einen Anwendungsprozess.
- ❏ Speicher, der einem Prozess durch Ein- und Auslagern von Speicherbereichen durch das Betriebssystem und die Hardware vorgespiegelt wird.
- ❏ Virtueller Speicher kann dynamisch zur Laufzeit von einem Programm mit der Funktion `malloc` erzeugt werden.
- ❏ Virtueller Speicher ist in unbegrenzter Menge vorhanden.

i) Welche Aussage zu Zeigern ist richtig? 2 Punkte

- ❏ Die Übergabesemantik für Zeiger als Funktionsparameter ist *call-by-reference*.
- ❏ Ein Zeiger kann zur Manipulation von schreibgeschützten Datenbereichen verwendet werden.
- ❏ Zeiger vom Typ `void*` existieren in C nicht, da solche "Zeiger auf Nichts" keinen sinnvollen Einsatzzweck hätten.
- ❏ Zeiger können verwendet werden, um in C eine call-by-reference Übergabesemantik nachzubilden.

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) Welche der folgenden Aussagen zu UNIX-Dateisystemen sind richtig? 4 Punkte

- ❍ In einem Verzeichnis darf es mehrere Einträge mit identischem Namen geben, sofern sie auf unterschiedliche *Inodes* verweisen.
- ❍ Für das Löschen einer Datei sind die Rechte-Informationen im *Inode* des Verzeichnisses, das die Datei enthält, irrelevant.
- ❍ Nach dem Löschen eines *Inodes* und der dazugehörigen Datenblöcke ist garatiert, dass kein *hard link* mehr darauf verweist.
- ❍ Ein *Inode* kann im Dateisystem nicht über mehrere Namen referenziert werden.
- ❍ Beim lesenden Zugriff auf eine Datei über einen *symbolic link* kann ein Prozess den Fehler *Permission denied* erhalten, obwohl er das Leserecht auf dem *symbolic link* besitzt.
- ❍ Der *Inode* einer Datei wird getrennt von ihrem Inhalt auf der Platte gespeichert.
- ❍ Ein *Inode* enthält u.a. die Anzahl der *symbolic links*, die auf ihn verweisen.
- ❍ Die Anzahl der *hard links*, die auf ein Verzeichnis verweisen, hängt von der Anzahl seiner Unterverzeichnisse ab.

Aufgabe 2: syslogd (45 Punkte)

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

Schreiben Sie ein Programm **`syslogd`** (syslog daemon), das Protokollnachrichten von mehreren UNIX-Pipes einliest, puffert und gebündelt auf die Standardausgabe ausgibt.

Mehrere Lese-Threads lesen die Protokollnachrichten parallel ein und ein Thread (Haupt-Thread) gibt diese aus. Zum Austausch der Logmeldungen zwischen den Lesern und dem Schreiber wird eine verkettete Liste mit FIFO-Eigenschaft (Listenelemente: `struct logLine`) verwaltet. Der Zugriff auf die verkettete Liste ist geeignet zu synchronisieren.

Das Programm soll folgendermaßen arbeiten:

- Der Haupt-Thread sucht im aktuellen Arbeitsverzeichnis nach UNIX-Pipes (`S_ISFIFO`) und startet für jede Pipe einen neuen Lese-Thread, der die Funktion `readThread` ausführt. Danach wartet der Haupt-Thread passiv, bis er von einem Lese-Thread benachrichtigt wird, dass die verkettete Liste genügend Elemente enthält.
  Nun sichert der Haupt-Thread die verkettete Liste und erzeugt eine neue, leere Liste. (Tipp: Es ist nicht notwendig die einzelnen Listenelemente zu kopieren. Es reicht aus den Kopfzeiger geeignet zu verändern).
  Danach gibt er alle in der Liste gepufferten Protokollnachrichten aus (Format: "*NameDerPipe*: *Protokollnachricht*") und wartet wieder passiv auf weitere Protokollnachrichten.
- Funktion **`void *readThread(const char *filename)`**: Öffnet die als Parameter übergebene Datei zum Lesen und liest zeilenweise (Sie können davon ausgehen, dass Protokollnachrichten maximal 4095 Zeichen lang sind). Jede eingelesene Zeile wird zusammen mit dem Dateinamen der UNIX-Pipe in die verkettete Liste eingefügt. Sobald die Liste 20 Elemente umfasst, wird der Haupt-Thread benachrichtigt.

Ihnen steht das aus der Übung bekannte Semaphor-Modul zur Verfügung. Die Schnittstelle finden Sie im folgenden 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.


```
#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>

typedef struct SEM SEM;
SEM *semCreate(int initVal);
int semDestroy(SEM *sem);
void P(SEM *sem);
void V(SEM *sem);

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

typedef struct logLine {
   char source[256]; // Name der Pipe, von der die Zeile stammt
   char line[4097];  // Inhalt der Zeile
   struct logLine* next;
} logLine;

// Makros, Funktionsdeklarationen, globale Variablen
```

```
// Funktion main()
```

```
   // Verzeichnis öffnen und durchsuchen
```


```
   // Threads starten
```

```
   // Logmeldungen entnehmen und ausgeben
```

```
// Ende Funktion main
```

A:

```
// Funktion readThread()
```

```
  // Logmeldungen zeilenweise einlesen
```


```
    // gelesene Meldung in Liste einfuegen
```

```
// Ende Funktion readThread
```

R:

Aufgabe 3: (23 Punkte)

a) Was versteht man unter einem logischen Adressraum? Beschreiben Sie eine Möglichkeit, wie zur Ladezeit oder zur Laufzeit eine Abbildung auf den realen Adressraum vorgenommen werden kann. (4 Punkte)

b) Logische Adressräume sorgen nicht nur für die Abbildung auf den realen Hauptspeicher, sie sind auch eine Schutzdomäne. Welche Arten von Schutz gewährleisten sie? (3 Punkte)

c) Gegegeben sei das unten stehende Programm. (11 Punkte)

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

- Wozu werden welche Segmente angelegt?
- Zeichnen Sie für jede Variable ein, wo diese ungefähr im logischen Adressraum zu finden sein wird.
- Wohin zeigen die Zeiger m, p und q (ungefähre Position einzeichnen!)?

```
char *m = "Weltmeister";
int wm = 20;
static int de = 14;

int main() {
  int la;
  static int lb;
  int *p;
  static double *q;

  p = malloc(100 * sizeof(int));
  q = malloc(100 * sizeof(double);

}
```

d) Wenn das folgende Programmstück in einem UNIX-System abläuft, wird ein Fehler auftreten. (5 Punkte)

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

Bitte möglichst knappe, aber präzise Antworten:

d1) Welcher Fehler tritt auf?

d2) Warum tritt der Fehler auf?

d3) Der Fehler wird von einer Hardwarekomponente zuerst entdeckt - welche Komponente ist das?

d4) Mit welchem Mechanismus wird der Fehler dem Betriebssystem mitgeteilt?

d5) Was macht das Betriebssystem mit dem Prozess, der gerade das Programmstück ausführt?