Aufgabe 1: (12 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) Wie funktioniert Adressraumschutz durch Eingrenzung? 2 Punkte

- ❏ Der Lader positioniert Programme immer so im Arbeistsspeicher, dass unerlaubte Adressen mit nicht-existierenden physikalischen Speicherbereichen zusammenfallen.
- ❏ Begrenzungsregister legen einen Adressbereich im logischen Adressraum fest, auf den alle Speicherzugriffe beschränkt werden.
- ❏ Begrenzungsregister legen einen Adressbereich im physikalischen Adressraum fest, auf den alle Speicherzugriffe beschränkt werden.
- ❏ Die MMU kennt die Länge eines Segments und verhindert Speicherzugriffe darüber hinaus.

b) Welche der folgenden Aussagen zu statischem bzw. dynamischem Binden ist **falsch**? 2 Punkte

- ❏ bei dynamischem Binden werden alle Adressen zum Bindezeitpunkt aufgelöst
- ❏ bei dynamischem Binden können auch zum Übersetzungszeitpunkt alle bekannten Adressbezüge bereits vollständig aufgelöst werden
- ❏ beim statischen Binden werden alle Adressen zum Bindezeitpunkt aufgelöst
- ❏ statisch gebundene Programme können zum Ladezeitpunkt an beliebige Speicheradressen platziert werden

c) Was gehört nicht zu den typischen Dateiattributen? 1 Punkte

- ❏ die Länge des Dateiinhalts
- ❏ der Zeitpunkt des letzten lesenden Zugriffs
- ❏ die Prozess-Identifikation des Prozesses, der die Datei gerade geöffnet hat
- ❏ die Benutzer-Identifikation des Eigentümers der Datei

d) Was versteht man unter virtuellem Speicher 2 Punkte

- ❏ Virtueller Speicher kann größer sein als der physikalisch vorhandene Arbeitsspeicher. Gerade nicht benötigte Speicherbereiche können auf Hintergrundspeicher ausgelagert werden.
- ❏ Virtueller Speicher kann größer sein als der physikalisch vorhandene Hintergrundspeicher zusammen mit dem Arbeitsspeicher. Über den tatsächlich existierenden Speicher hinausgehende Speicherbereiche sind dann nur scheinbar vorhanden.
- ❏ Virtueller Speicher sind die nicht vorhandenen Bereiche des physikalischen Adressraums.
- ❏ Virtueller Speicher kann dynamisch zur Laufzeit von einem Programm erzeugt werden (Funktion valloc(3)).

e) Was passiert, wenn Sie in einem Programm über einen ungültigen Zeiger versuchen auf Speicher zuzugreifen 2 Punkte

- ❏ Der Arbeitsspeicher erkennt, dass es sich um eine ungültige Adresse handelt und leitet eine Ausnahmebehandlung ein.
- ❏ Die MMU schickt an die CPU einen Interrupt. Hierdurch wird das Betriebssystem angesprungen, das den gerade laufenden Prozess mit einem "Segmentation fault"-Signal unterbricht.
- ❏ Die MMU erkennt die ungültige Adresse bei der Adressumsetzung und löst einen Trap aus.
- ❏ Das Betriebssystem erkennt die ungültige Adresse bei der Weitergabe des Befehls an die CPU (partielle Interpretation) und leitet eine Ausnahmebehandlung ein.

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.
- ❏ Solch eine Schnittstelle ist nicht schön, da dadurch die aufrufende Funktion auf internen Speicher des Betriebssystems zugreifen könnte.
- ❏ 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.

Aufgabe 2: (23 Punkte)

a) Programmieren Sie eine Funktion

`char **getdircontent(const char *dirname);`

die als Ergebnis ein Feld mit den in einem Directory verzeichneten Namen liefert.

Die Funktion erhält den Namen des auszulesenden Verzeichnisses übergeben und liefert ein Feld von Zeigern auf die Zeichenketten mit den darin verzeichneten Dateinamen.

Gehen Sie bei der Programmierung davon aus, dass ein Directory in der Regel nur selten mehr als 50 Einträge haben wird und organisieren Sie die dynamische Speicherverwaltung in Ihrer Funktion so, dass sie unter dieser Massgabe relativ effizient arbeitet.

Hinter dem letzten Zeiger auf einen Dateinamen soll ein NULL-Zeiger eingetragen werden, um das Ende des belegten Feldes zu markieren.

Im Fehlerfall soll die Funktion eine adäquate Fehlermeldung auf stderr ausgeben und einen NULL-Zeiger zurückliefern. Die in diesem Fall eigentlich notwendige Freigabe von bereits angefordertem Speicher dürfen Sie bei Ihrer Lösung hier ausnahmesweise vernachlässigen.

Die Funktion kann unter der Annahme programmert werden, dass sie Teil eines Moduls ist, der auch eine passende Funktion zur Freigabe des Speichers enthält. Die Freigabefunktion brauchen Sie aber hier nicht zu programmieren.


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


/* Makros, globale Variablen (falls notwendig) */

.....

.....

/* Funktion getdircontents */

..... a

.....
{
  /* lokale Variablen und was man sonst am
      Anfang so vorbereitet */

    .....

    .....

    .....

    .....

    .....

    .....

  /* Directory oeffnen */

    .....

    .....

    ..... d

    .....

    .....

    .....
```

```
/* Schleife */
    /* Allokation Zeigerfeld */
```
a

m

```
    /* Directory-Eintrag lesen und verarbeiten */
```
d

a

m

a

```
/* Ende Schleife */
```
d

a

Aufgabe 3: (10 Punkte)

Gegegeben sei das nachfolgende Programm.

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!)?

```
int ga;
static int gb;
char *m = "message";

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

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

}
```