Aufgabe 1: (20 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 Antwort beschreibt ein Attribut einer Datei eines UNIX-UFS-Dateisystems? 1 Punkt

- ❏ Anzahl der symbolischen Links
- ❏ Dateiname
- ❏ Position des Schreib-Lese-Zeigers
- ❏ Anzahl der Hard links

b) Welche Antwort trifft für die Eigenschaften eines UNIX/Linux Filedeskriptors zu? 2 Punkte

- ❏ Ein Filedeskriptor ist eine prozesslokale Integerzahl, die der Prozess zum Zugriff auf eine Datei, ein Gerät, einen Socket oder eine Pipe benutzen kann.
- ❏ Filedeskriptoren sind Zeiger auf Betriebssystemstrukturen, die von den Systemaufrufen ausgewertet werden, um auf Dateien zuzugreifen.
- ❏ Ein Filedeskriptor ist eine Integerzahl, die über gemeinsamen Speicher an einen anderen Prozess übergeben werden kann, und von letzterem zum Zugriff auf eine geöffnete Datei verwendet werden kann.
- ❏ Beim Öffnen ein und derselben Datei erhält ein Prozess jeweils die gleiche Integerzahl als Filedeskriptor zum Zugriff zurück.

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

- ❏ Der Prozess wartet auf Daten von der Festplatte.
- ❏ Der Prozess hat einen Seitenfehler für eine Seite, die bereits in die Freiliste der Seiten eingetragen aber noch im Hauptspeicher vorhanden ist.
- ❏ Der Prozess liest Daten von der Festplatte mit nichtblockierenden Eingabeoperationen.
- ❏ Der Prozess terminiert.

d) Was versteht man unter einer Unterbrechung bei der Ausführung von Instruktionen durch einen Prozessor? 2 Punkte

- ❏ Eine Signalleitung teilt dem Prozessor mit, dass er den aktuellen Prozess anhalten und auf das Ende der Unterbrechung warten soll.
- ❏ Der Prozessor wird veranlasst eine Unterbrechungsbehandlung durchzuführen. Der gerade laufende Prozess kann die Unterbrechungsbehandlung ignorieren.
- ❏ Mit einer Signalleitung wird dem Prozessor eine Unterbrechung angezeigt. Der Prozessor sichert den aktuellen Zustand bestimmter Register insbesondere des Programmzählers und springt eine vordefinierte Behandlungsfunktion an.
- ❏ Durch eine Signalleitung wird der Prozessor veranlasst, die gerade bearbeitete Maschineninstruktion zu unterbrechen und in den Benutzermodus umzuschalten.

e) Bei einer prioritätengesteuerten Prozess-Auswahlstrategie (Scheduling-Strategie) kann es zu Problemen kommen. Welches der folgenden Probleme kann auftreten? 2 Punkte

- ❏ Eine prioritätenbasierte Auswahlstrategie arbeitet sehr ineffizient, wenn viele Prozesse im Zustand bereit sind.
- ❏ Prioritätenbasierte Auswahlstrategien führen zwangsläufig zur Aushungerung von Prozessen, wenn mindestens zwei verschiedene Prioritäten vergeben werden.
- ❏ Ein hochpriorer Prozesse muss evtentuell auf ein Betriebsmittel warten, das von einem niedrigpriorien Prozess exklusiv benutzt wird. Der niedrigpriore Prozess kann das Betriebsmittel jedoch wegen eines mittelhochprioren Prozesses nicht freigeben (Prioritätenumkehr).
- ❏ Das Phänomen der Prioritätsumkehr hungert niedrig-priore Prozesse aus.

f) Welche Aussage ist bezüglich der Seitenersetzungsstrategie Second-Chance richtig? 2 Punkte

- ❏ Die Second-Chance Strategie nähert sich der LRU-Strategie an, wenn alle Seiten sehr häufig benutzt werden.
- ❏ Die Second-Chance Strategie zeigt keine FIFO-Anomalie.
- ❏ Die Second-Chance Strategie nähert sich nahezu der optimalen B_O-Strategie an und wird daher in fast allen realen Systemen benutzt.
- ❏ Die Second-Chance Strategie benötigt ein Referenzbit in jedem Eintrag der Seitenkacheltabelle.

g) Für welchen Zweck wird der Systemaufruf `listen()` benutzt? 2 Punkte

- ❏ Mit `listen()` wird ein Socket für die Verbindungsannnahme vorbereitet. Ein Parameter gibt an, wieviele Verbindungsanfragen vor deren Annahme gepuffert werden können.
- ❏ Damit das Betriebssysteme für einen Socket überhaupt Systemaufrufe annimmt, muss es erst mit `listen()` in einen Modus des „Zuhörens“ gebracht werden.
- ❏ Der Aufruf von `listen()` wartet solange an einem Socket, bis eine einkommende Verbindungsanfrage vorliegt.
- ❏ Mit `listen()` wird ein Socket für die Verbindungsannahme vorbereitet. Ein Parameter gibt an, wieviele laufende Verbindungen maximal möglich sind.

h) Welche Aussage zur Verklemmungsverhinderung ist richtig? 2 Punkte

- ❏ Bei Verklemmungsverhinderung wird dafür gesorgt, dass eine der vier notwendigen Bedingungen für Verklemmungen nicht auftreten kann.
- ❏ Das System überprüft vor dem Freigeben von Betriebsmitteln, ob ein unsicherer Zustand eintreten würde.
- ❏ Mit Hilfe des Bankers-Algorithmus wird beim Belegen eines Betriebsmittels geprüft, ob mit erfolgreicher Belegung ein unsicherer Zustand eintreten würde.
- ❏ Bei einem Zyklus im erweiterten Betriebsmittelgraph liegt ein unsicherer Zustand vor.

i) Welche Aussage über UNIX-Semaphoren ist richtig? 3 Punkte

- ❏ UNIX-Semaphoren können unmittelbar das Verhalten von PV-Chunk- und PV-Multiple-Semaphoren nachbilden.
- ❏ UNIX-Semaphoren können das Verhalten von UP-DOWN-Systemen nachbilden.
- ❏ Eine Operation `semop()` auf einem UNIX-Semaphor kann sich auf mehrere Einzelsemaphoren maximal zweier UNIX-Semaphoren gleichzeitig beziehen.
- ❏ UNIX-Semaphoren sind nur für die Koordinierung von Aktivitätsträgern innerhalb eines Prozesses geeignet, nicht jedoch für die Koordinierung mehrerer Prozesse.

j) Alle aufgeführten Koordinierungsmittel erlauben die Implementierung eines kritischen Abschnitts. Welches der aufgeführten Verfahren ist jedoch ungeeignet, weil die Prozesse aktiv warten müssen? 1 Punkt

- ❏ binärer Semaphor
- ❏ Algorithmus von Peterson
- ❏ Up-Down-Systeme
- ❏ Sperren von Unterbrechungen

k) Wie wird in einem UNIX-Dateisystem (z.B. EXT2 oder Sun UFS) das Attribut des Eigentümers einer Datei realisiert? 1 Punkt

- ❏ Die User-ID (UID) des Eigentümers steht im Verzeichniseintrag der Datei.
- ❏ Eine Integerzahl repräsentiert die User-ID (UID) des Eigentümers im Inode der Datei.
- ❏ Der Benutzername des Eigentümers wird im Inode der Datei abgespeichert.
- ❏ Die GID des Eigentümers wird als Integerzahl im Inode der Datei gespeichert.

Aufgabe 2: (40 Punkte)

a) Schreiben Sie ein Programm **`time`**, das die Rechenzeit eines anderen Programms ermittelt.

Das **`time`**-Programm erhält als Argumente den Namen eines anderen Programms sowie ggf. dessen Argumente (z. B. `time ls -al /tmp`), führt dieses Programm aus, gibt anschließend auf dem Fehlerausgabekanal die Rechenzeit im Benutzer-Modus aus und terminiert dann wieder.

Die Ausgabe der Rechenzeit kann in der Einheit clock-ticks, in der sie vom System geliefert wird, erfolgen.

Wird während der Ausführung von **`time`** ein Interrupt-Signal (CTRL-C) ausgelöst, soll folgendes passieren:

- Trifft das Signal ein, bevor der Prozess für das zu messende Programm gestartet wurde, soll das Signal einfach ignoriert werden.
- Wurde der Prozess für das zu messende Programm bereits gestartet, gibt **`time`** die Meldung "`Programm xxx läuft noch`" aus
  (`xxx` steht hierbei für den Namen des gerade zu messenden Programms).
- Das zu messende Programm selbst soll das Signal ignorieren (Sie können hierbei davon ausgehen, dass das zu messende Programm die Einstellungen für das Signal SIGINT von sich aus nicht verändert).

Die Ermittlung und Ausgabe der Rechenzeit am Ende darf durch zwischenzeitlich eingetroffene Signale nicht verhindert werden.

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.


```
/* includes */
#include <stdio.h>
#include <errno.h>
#include <signal.h>
#include <unistd.h>
#include <sys/times.h>
#include <sys/wait.h>
#include <sys/types.h>

/* Funktionsdeklarationen, globale Variablen */
```

```
/* Funktion main */
```

```
{
  /* lokale Variablen und was man sonst am Anfang so braucht */
```

```
/* Signalbehandlung einrichten */
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

/* Prozess zur Kommandoausführung erzeugen */
.....
.....
.....
.....

/* Fehler bei Prozesserzeugung */
.....
.....
.....
.....
.....

 /* Sohnprozess */
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
```

S:
P:


```
/* Vaterprozess */
.....
.....
.....
.....
.....

    /* auf Ende des zu messenden Progr. warten */
.....
.....
.....
.....
.....
.....
.....

    /* Laufzeit feststellen und ausgeben*/
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

} /* Ende Funktion main */
```

S:
P:
T:

```
/* Handler-Funktion für SIGINT */
```

b) Schreiben Sie ein Makefile zum Erzeugen des time-Programms.

Ein Aufruf von

```
make time
```

soll das Programm erzeugen, ein Aufruf von

```
make clean
```

soll das **`time`**-Programm und evtl. erzeugte **`.o`**-Dateien entfernen.

S:
M: