Aufgabe 1: (30 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) Beim Zugriff auf Datei-Inhalte wird typischerweise zwischen sequentiellem und wahlfreiem Zugriff unterschieden. Welche Aussage ist **richtig**? 2 Punkte

- ❏ bei sequentiellem Zugriff dürfen verschiedene Benutzer nur hintereinander auf die Datei zugreifen
- ❏ wahlfreier Zugriff erfolgt nach beliebigem Muster und ist ideal bei allen denkbaren Speichermedien geeignet
- ❏ bei wahlfreiem Zugriff ist ein wohlgeordnetes Zugriffsmuster nicht erkennbar und das Verfahren ist z. B. bei Festplatten geeignet
- ❏ bei wahlfrei organisiertem Massenspeicher ist sequentieller Zugriff nicht möglich

b) Welche Aussage zum Thema "Aktives Warten" ist **richtig**? 2 Punkte

- ❏ Aktives Warten vergeudet gegenüber passivem Warten immer CPU-Zeit
- ❏ Auf Mehrprozessorsystemen ist aktives Warten unproblematisch und deshalb dem passiven Warten immer vorzuziehen
- ❏ Aktives Warten darf bei nicht-verdrängenden Scheduling-Strategien auf einem Monoprozessorsystem nicht verwendet werden
- ❏ Bei verdrängenden Scheduling-Strategien verzögert aktives Warten nur den betroffenen Prozess, behindert aber nicht andere

c) Wie groß ist typischerweise eine Seite bei Seitenadressierung? 1 Punkt

- ❏ 8 bis 16 Bytes
- ❏ 512 bis 8.192 Bytes
- ❏ 32.768 bis 8.388.608 Bytes

d) In einem UNIX-UFS-Dateisystem gibt es symbolische Verweise (Symbolic Links) und feste Verweise (Hard Links) auf Dateien. Welche Aussage ist **falsch**? 2 Punkte

- ❏ Hard Links können nur vom Systemadministrator angelegt werden.
- ❏ Ein Hard Link kann nur auf Dateien, jedoch nicht auf Verzeichnisse verweisen.
- ❏ Ein Symbolic Link kann auf Verzeichnisse verweisen.
- ❏ Symbolic Links können existieren, obwohl die verwiesene Datei oder das verwiesene Verzeichnis bereits gelöscht wurde.

e) Welche Aussage über Prozesszustände ist in einem Monoprozessor-Betriebssystem mit blockierenden Ein-, Ausgabeoperationen **richtig**? 3 Punkte

- ❏ Wenn gerade keine Prozessumschaltung stattfindet und kein Prozess im Zustand *laufend* ist, so ist auch kein Prozess im Zustand *blockiert*.
- ❏ Es befinden sich bis zu zwei Prozesse im Zustand *laufend* und damit in Ausführung auf dem Prozessor (Vordergrund- und Hintergrundprozess).
- ❏ Wenn gerade keine Prozessumschaltung stattfindet und ein Prozess im Zustand laufend ist, so gibt es mindestens einen Prozess im Zustand *blockiert*.
- ❏ Ein Prozess im Zustand *laufend* wird in den Zustand *blockiert* überführt, wenn eine seiner Ein-, Ausgabeoperation nicht sofort abgeschlossen werden kann.

f) Sie kennen den Begriff Seitenflattern (Thrashing). Welche Aussage ist **richtig**? 3 Punkte

- ❏ Als Seitenflattern bezeichnet man das wiederholte löschen und neu laden des Translation-Look-Aside-Buffer (TLB), ausgelöst durch häufigen Prozesswechsel.
- ❏ Als Seitenflattern bezeichnet man das wiederholte Einlagern einer erst vor kurzem verdrängten Speicherseite. Die Prozesse verbringen als Folge die meiste Zeit mit dem Warten auf die Behebung von Seitenfehlern.
- ❏ Seitenflattern erkennt man an der starken Geräuschentwicklung der Festplatte, da auf Grund häufiger Seitenzugriffe der Lesekopf ständig neu positioniert wird. Bei Systemen ohne Festplatte (Thin-Clients) kann das Seitenflattern nicht auftreten.
- ❏ Durch die Verwendung von Semaphoren kann das unkontrollierte Flattern von Seiten synchronisiert werden.

g) Welche der Aussagen bzgl. eines logischen Adressraums, der auf dem Prinzip der Segmentierung aufgebaut wurde, ist **richtig**? 3 Punkte

- ❏ Segmentierung unterstützt die Ein- und Auslagerung von Segmenten auf eine Festplatte besonderst gut, da die Größe von Segmenten den Größen von Blöcken auf der Festplatte entsprechen.
- ❏ Über gemeinsame Segmente können zwei logische Adressräume auf die selben Speicherzellen zugreifen.
- ❏ Die Segmentierung erlaubt bei der Abbildung eines logischen Adressraums keinen Zugriff auf Speicherzellen, die auch Bestandteil von anderen logischen Adressräumen sind (Zugriffschutz).
- ❏ Segmente können verschiedene Länge haben, jedoch nie mehr als 4 KiloByte. Die Längenbegrenzung wird üblicherweise bei der Speicherabbildung geprüft.

h) Welche Aussage zur kontinuierlicher Speicherung der Daten einer Datei auf Festplatte ist **falsch**? 2 Punkte

- ❏ Eine kontinuierliche Speicherung der Daten in aufeinanderfolgenden Blöcken erhöht die Leseleistung gegenüber verstreuter Speicherung.
- ❏ Eine kontinuierliche Speicherung der Daten in aufeinanderfolgenden Blöcken erhöht die Schreibleistung gegenüber verstreuter Speicherung.
- ❏ Bei kontinuierlicher Speicherung kann die Ortsinformation lediglich aus der Nummer des ersten Plattenblocks und der Dateilänge bestehen.
- ❏ Kontinuierliche Speicherung ist gänzlich unmöglich, falls Dateien dynamisch erweiterbar sein sollen.

i) Welche Aussage ist bezüglich der Festplattentreiber eines UNIX-Betriebssystems **richtig**? 2 Punkte

- ❏ Der Festplattentreiber übernimmt Betriebssystemaufgaben wie z.B. die Implementierung eines Dateisystems.
- ❏ Ein Festplattentreiber in UNIX wird in einem eigenen Prozess realisiert.
- ❏ Festplattentreiber werden innerhalb des UNIX-Betriebssystems über eine einheitliche Schnittstelle angesprochen.
- ❏ Jede einzelne Festplatte benötigt ihren eigenen Treiber.

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

- ❏ Damit das Betriebssystem 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 Verbindungsannnahme vorbereitet. Ein Parameter gibt an, wieviele Verbindungsanfragen vor deren Annahme gepuffert werden können.
- ❏ Mit `listen()` wird ein Socket für die Verbindungsannahme vorbereitet. Ein Parameter gibt an, wieviele laufende Verbindungen maximal möglich sind.

k) Was versteht man unter Verklemmungsvermeidung? 2 Punkte

- ❏ Das System überprüft vor dem Belegen von Betriebsmitteln, ob ein unsicherer Zustand eintreten würde.
- ❏ Bei einem Zyklus im Betriebsmittelgraph wird einer der Prozesse aus dem Zyklus terminiert.
- ❏ In einem System wird dafür gesorgt, dass eine der vier notwendigen Bedingungen für Verklemmungen nicht eintreten kann.
- ❏ In einem System wird dafür gesorgt, dass eine der vier hinreichenden Bedingungen für Verklemmungen nicht eintreten kann.

l) Ein Prozess wird in den Zustand bereit überführt. Welche Aussage passt **nicht** zu diesem Vorgang? 2 Punkte

- ❏ Der Prozess wurde von einem Prozess mit einer höherer Priorität verdrängt.
- ❏ Der Prozess hat auf Daten von der Festplatte gewartet und die Daten stehen nun zur Weiterbearbeitung bereit.
- ❏ Der Prozess hat einen Seitenfehler für eine Seite, die noch nicht im Hauptspeicher vorhanden ist.
- ❏ Der Prozess wartet auf eine Tastatureingabe.

m) Sie kennen den Begriff Demand-Paging. Welche Aussage dazu ist **<u>richtig</u>**? 2 Punkte

- ❏ Demand-Paging benötigt keinerlei Hardware-Unterstützung, da sich alle benötigten Mechanismen auch ohne MMU realisieren lassen.
- ❏ Demand-Paging lädt eine Seite erst dann in den Hauptspeicher, wenn die Festplatte bereits angesprochen wird. Nicht benutzte Daten werden dabei auf die Festplatte geschrieben.
- ❏ Demand-Paging setzt eine segmentierte Speicherverwaltung voraus.
- ❏ Demand-Paging erlaubt es größere logische Adressräume anzulegen, als Hauptspeicher vorhanden ist. Die Seiten werden erst dann in den Hauptspeicher geladen, wenn sie tatsächlich angesprochen werden. Nicht benutzte Seiten werden unter Umständen aus dem Hauptspeicher ausgelagert.

n) Welche Aussage über UNIX-Semaphore ist **<u>falsch</u>**? 2 Punkte

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

Aufgabe 2: (60 Punkte)

a) Schreiben Sie ein Programm `msgd` (Message-Daemon), das einen Mitteilungstext bei Anfragen über eine Netzverbindung ausgibt. Der Text wird dem Programm über eine Konfigurationsdatei zur Verfügung gestellt.

Dem Programm `msgd` kann als Parameter der Name der Konfigurationsdatei übergeben werden. Bei Aufruf ohne Parameter wird die Datei `msgd.conf` verwendet.

Das Programm soll folgendermaßen arbeiten:

- `msgd` startet zuerst einen Sohnprozess, der die eigentliche Arbeit im Hintergrund übernimmt (deshalb auch Daemon-Prozess genannt), gibt dessen Prozess-Id aus ("`msgd startet, PID=...`") und terminiert dann sofort wieder. Die eigentliche Arbeit wird in der Funktion `void msgserver()` erledigt.
- Funktion `msgserver`: der Text aus der Konfigurationsdatei wird unter Verwendung der Funktion `char *getconf(int *size)` eingelesen. Anschliessend werden TCP-Verbindungen von beliebigen IP-Adressen auf Port 9999 erwartet.
- Es wird jeweils eine Verbindung angenommen, der konfigurierte Text auf die Verbindung ausgegeben und anschliessend die Verbindung beendet. Es sollen bis zu 20 Verbindungsanforderungen warten dürfen.
- Funktion `getconf`: Die Funktion ermittelt die aktuelle Größe der Konfigurationsdatei, allokiert den dafür benötigten Speicher, liest die komplette Konfigurationsdatei in diesen Speicherbereich ein und gibt einen Zeiger auf den Speicherbereich als Ergebnis zurück. Im Fehlerfall wird NULL geliefert. Im Parameter `size` wird die Länge des Textes zurückgegeben.
- Änderungen der Konfigurationsdatei können dem Daemon-Prozess durch ein SIGHUP-Signal mitgeteilt werden. Der Empfang des Signals soll bewirken, dass ab der nächsten TCP-Verbindungsanfrage der neue Text ausgegeben wird. Die Textausgabe auf einer gerade bestehenden Verbindung darf durch die Neukonfiguration nicht zerstört werden. Es muss entweder noch der alte oder schon der neue Text ausgegeben werden.
  Bedenken Sie, dass ein neuer Text eine andere Länge haben kann. Eine Neukonfiguration des Textes soll während der Laufzeit des Daemon-Prozesses beliebig oft möglich sein.
- Sie können davon ausgehen, dass der Name der Konfigurationsdatei nicht länger als 255 Zeichen ist. Die Länge der Datei selbst kann beliebig sein.

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 ggf. auch weitere Funktionen benötigen.

```
/* includes */
#include <sys/types.h>
#include <stdlib.h>
#include <unistd.h>
#include <errno.h>
#include <string.h>
#include <stdio.h>
#include <signal.h>
#include <netdb.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <sys/stat.h>


/* Funktionsdeklarationen, globale Variablen */
```



```
/* Funktion main */

{
  /* Argumente prüfen */

  /* Serverprozess starten */
  {

  }
} /* Ende Funktion main */
```

A:
P:

```
/* msgserver-Funktion */
void msgserver()
{
    /* Variablendefinitionen */

    char *message;       /* Text der auszugebenden Meldung */
    int msg_size;        /* Laenge der Meldung */
    int sock_accept;     /* Socketdeskriptor der angenommenen Verb. */

    .....
    .....
    .....
    .....

    /* Socket oeffnen, binden, usw. */

    .....
    .....
    .....
    .....
    .....
    .....
    .....

    .....
    .....
    .....
    .....
    .....
    .....
    .....
    .....
    .....
    .....
    .....
    .....
```

S:


```
    /* Signal-Handler für SIGHUP einrichten */
    {

    .....
    .....
    .....
    .....
    .....
    .....
    .....
    .....
    .....
    }

    .....
    .....
    .....
    .....
    .....
    .....
    .....
    .....
    .....
    .....

    /* Server-Schleife */

    .....
    .....
    .....
    .....
    .....
    .....
    .....
    .....
    .....
    .....
    .....
```

g

I:
G:

```
        /* Verbindung annehmen */
        {
```

s

```
        }

        /* Text ausgeben */
        {

          FILE *conn;
          int i;

          if ((conn = fdopen(sock_accept, "w")) == NULL ) {
            perror("fdopen socket");
            exit(EXIT_FAILURE);
          }
          for(i=0; i<msg_size; i++) putc(message[i], conn);
```

s

```
        }
    } /* Ende Server-Schleife */
} /* Ende msgserver-Funktion */
```

S:


```
/* Konfigurationsdatei einlesen: Funktion getconf */
char *getconf(int *s)
{
  /* Variablendefinitionen */
```

```
  /* Groesse der Datei ermitteln */
```

```
  /* Speicher allokieren */
```

```
  /* Datei einlesen */
```

```
  /* Ergebnis liefern */
```

```
} /* Ende Funktion getconf */
```

G:

```
/* signal-handler für SIGHUP */
```

i

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

Ein Aufruf von

```
make msgd
```

soll - falls erforderlich - die aktuelle Version der Quelldatei aus der RCS-Datei "auschecken" und das Programm erzeugen, ein Aufruf von

```
make clean
```

soll das `msgd`-Programm und evtl. bei vorherigen make-Läufen erzeugte `.o`-Dateien entfernen.

I:
M:

Aufgabe 3: (18 Punkte)

a) Skizzieren Sie (graphisch) die Abbildung von einer logischen Adresse in eine physikalische Adresse in einem System mit Segmentierung.

b) Was muss das Betriebssystem tun, wenn aufgrund von Hauptspeichermangel ein Segment ausgelagert werden soll? Was passiert, wenn der Prozess nach dem Auslagern auf die Daten des Segments zugreift?

c) Ein Segment soll von zwei Prozessen gemeinsam genutzt werden. Was muss das Betriebssystem hierfür tun?

d) Nennen Sie 3 wesentliche Unterschiede zwischen Segmentierung und Seitenadressierung.

e) Was ist ein TLB?

Aufgabe 4: (12 Punkte)

Skizzieren Sie in einer programmiersprachen-ähnlichen Form die Programmierung der zwei Funktionen *put* und *get* (entspricht *store* und *fetch*), die in der üblichen Art und Weise als Erzeuger und Verbraucher auf einem Puffer fester Größe (Bounded Buffer) operieren.

Koordinieren Sie die Funktionen mit Hilfe von Semaphoren.

Beschreiben Sie kurz die Bedeutung der von Ihnen eingesetzten Semaphore und welche Werte sie initial haben müssen.

Gehen Sie davon aus, dass auch jeweils mehrere Erzeuger und Verbraucher gleichzeitig einen Zugriff versuchen könnten.