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) Was versteht man unter einem Translation Lookaside Buffer (TLB)? 2 Punkte

- ❏ Einen speziellen Cache der MMU, der den Inhalt der zuletzt angesprochenen Speicherzellen vorhält.
- ❏ Einen speziellen Cache der MMU, der Informationen aus den zuletzt genutzten Seitendeskriptoren vorhält.
- ❏ Einen Pufferspeicher des Compilers um den Übersetzungsvorgang zu beschleunigen (es werden die Codes der zuletzt übersetzten Statements vorgehalten).
- ❏ Einen speziellen Cache der CPU, der die zuletzt ausgeführten Maschinenbefehle zwischenspeichert (beschleunigt vor allem den Ablauf von Schleifen).

b) Welche Aussage zu Speicherzuteilungsverfahren ist **richtig**? 2 Punkte

- ❏ die worst-fit-Strategie ist lediglich theoretisch interessant, da es in der Praxis nie sinnvoll ist, den am schlechtesten passenden Speicherplatz zuzuweisen.
- ❏ best-fit ist in jedem Fall das beste Verfahren
- ❏ bei buddy-Verfahren gibt es keinen externen Verschnitt
- ❏ buddy-Verfahren sind nur bei sehr leistungsfähigen Rechnern und großem Speicher einsetzbar, weil sie aufwändig in der Berechnung sind und viel Verschnitt verursachen.

c) Es gibt verschiedene Ursachen, wie Nebenläufigkeit in einem System entstehen kann (gewollt oder auch ungewollt). Was gehört **nicht** dazu? 2 Punkte

- ❏ durch Interrupts
- ❏ durch die MMU
- ❏ durch preemptives Scheduling
- ❏ durch Threads auf einem Monoprozessorsystem

d) Im regulären Ablauf eines Anwendungsprogramms und einer Signalbehandlungsfunktion (bzw. im Ablauf eines Betriebssystems und einer Unterbrechungsbehandlungsfunktion) wird auf gemeinsame Daten modifizierend zugegriffen. Welche Aussage ist **falsch**? 3 Punkte

- ❏ Falscher Einsatz von Schlossvariablen (lock/unlock-Operationen) kann zu Verklemmungen führen
- ❏ Der Aufruf von P-Operationen im Anwendungsprogramm und dazu korrespondierenden V-Operationen in der Signalbehandlung führt nicht zu Verklemmungen
- ❏ Alle Verfahren der mehrseitigen Synchronisation können eingesetzt werden
- ❏ Alle Verfahren nicht-blockierender Synchronisation sind einsetzbar

e) Gegeben sei folgendes Szenario: zwei Threads werden auf einem Monoprozessorsystem mit der Strategie "First Come First Served" verwaltet. In jedem Faden wird die Anweisung "i++;" auf die gemeinsame, globale Variable i ausgeführt. 2 Punkte
Welche der folgenden Aussagen ist **richtig**:

- ❏ Während der Inkrementoperation müssen Interrupts vorübergehend unterbunden werden.
- ❏ Die Inkrementoperation muss mit einer CAS-Anweisung nicht-blockierend synchronisiert werden.
- ❏ Die Operation i++ ist auf einem Monoprozessorsystem immer atomar.
- ❏ In einem Monoprozessorsystem ohne Verdrängung ist keinerlei Synchronisation erforderlich.

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

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

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

- ❏ Der Compiler erkennt die problematische Code-Stelle und generiert Code, der zur Laufzeit bei dem Zugriff einen entsprechenden Fehler auslöst.
- ❏ Der Speicher schickt an die CPU einen Interrupt. Hierdurch wird das Betriebssystem angesprungen, das den gerade laufenden Prozess mit einem "Segmentation fault"-Signal unterbricht.
- ❏ Beim Zugriff über den Zeiger muss die MMU die erforderliche Adressumsetzung vornehmen, erkennt die ungültige Adresse 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.

h) Man unterscheidet zwischen privilegierten und nicht-privilegierten Maschinenbefehlen. Welche Aussage ist **richtig**? — 2 Punkte

- ❏ Privilegierte Maschinenbefehle dürfen in Anwendungsprogrammen grundsätzlich nicht verwendet werden.
- ❏ Die Benutzung eines privilegierten Maschinenbefehls in einem Anwendungsprogramm führt zu einer asynchronen Programmunterbrechung.
- ❏ Privilegierte Maschinenbefehle können durch Betriebssystemprogramme (partielle Interpretation) implementiert werden.
- ❏ Privilegierte Befehle sind die einzige Möglichkeit, auf Geräteregister zuzugreifen.

i) In einem Seitendeskriptor werden verschiedene Informationen über eine Seite eines virtuellen Adressraums gehalten. Was gehört sicher **nicht** dazu? — 2 Punkte

- ❏ Die Adresse der Seite im physikalischen Hauptspeicher
- ❏ Die Position der Seite im logischen Adressraum
- ❏ Zugriffsrechte (z. B. lesen, schreiben, ausführen)
- ❏ Ein Zähler, der ein Maß für das Alter der Seite enthält

j) Beim Blockieren in einem Monitor muss der Monitor freigegeben werden. Warum? — 2 Punkte

- ❏ weil sonst die Monitordaten inkonsistent sind.
- ❏ weil ein anderer Thread die Blockierungsbedingung nur aufheben kann, wenn er den Monitor betreten darf.
- ❏ weil kritische Abschnitte immer nur kurz belegt sein dürfen.
- ❏ weil der Thread sonst aktiv warten würde.

k) Ein Betriebssystem setzt logische Adressräume auf der Basis von Segmentierung ein. Welche Aussage ist **falsch**? — 2 Punkte

- ❏ Die Segmentierung schränkt den logischen Adressraum derart ein, dass nur auf gültige Speicheradressen erfolgreich zugegriffen werden kann.
- ❏ Über gemeinsame Segmente kann innerhalb von zwei verschiedenen logischen Adressräumen auf die selben Speicherzellen zugegriffen werden.
- ❏ Mit der Segmentierung kann einem Prozess mehr Speicher zugeordnet werden als physikalisch vorhanden ist, da ein Segment teilweise ausgelagert werden kann.
- ❏ Segmente können verschiedene Länge haben. Eine Längenbegrenzung wird üblicherweise bei der Speicherabbildung geprüft.

l) Welches der folgenden Verfahren ist **nicht** zur Verklemmungsvorbeugung in dem am Beispiel der "speisenden Philosophen" beschriebenen Verklemmungsszenario geeignet? — 2 Punkte

- ❏ Einer der Philosophen nimmt immer nur beide Stäbchen (oder Gabeln) gleichzeitig.
- ❏ Jeder Philosoph nimmt grundsätzlich immer zuerst das rechte und dann das linke Stäbchen. Dadurch wird eine Ordnung im System hergestellt, durch die keine Verklemmungen auftreten können.
- ❏ Alle Philosophen nehmen immer nur beide Stäbchen gleichzeitig.
- ❏ Ein Philosoph kann von seinem Nachbarn die Rückgabe eines Stäbchens fordern.

m) 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

n) Für lokale Variablen, Aufrufparameter, etc. einer Funktion wird bei vielen Prozessoren ein sog. Aktivierungsblock (activation record oder stack frame) auf dem Stack angelegt. Welche Aussage ist **richtig**? 3 Punkte

- ❏ Über Zeiger kann man alle Daten des Aktivierungsblocks der aufrufenden Funktion verändern.
- ❏ Nach dem Rücksprung aus einer Funktion sind Zeiger auf die Speicherzellen ihres Aktivierungsblocks nicht mehr gültig. Ein Zugriff über solch einen Zeiger führt dann zu einem Segmentation fault.
- ❏ Bei rekursiven Funktionsaufrufen kann der Aktivierungsblock in jedem Fall wiederverwendet werden, weil die gleiche Funktion aufgerufen wird.
- ❏ Der Compiler legt zur Übersetzungszeit fest, an welcher Position im Aktivierungsblock der main-Funktion die globalen Variablen angelegt werden.

Aufgabe 2: (60 Punkte)

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

a) Schreiben Sie ein Programm `mailq` (Mail-Queue-Bearbeiter), das Mail aus einem Warteschlangendirectory an entfernte Rechner zustellt.

Wird `mailq` ohne Parameter aufgerufen, werden maximal 5 Aufträge gleichzeitig bearbeitet. Bei Aufruf `mailq` *`n`* werden bis zu n Aufträge durch Threads gleichzeitig bearbeitet.

Das Programm soll folgendermaßen arbeiten:

- `mailq` sucht in dem Directory `/var/spool/mailq` nach Dateien, deren Name mit dem Zeichen `m` beginnt und bearbeitet den Auftrag durch Aufruf der Funktion `void pjob(char *mfile)`. Andere Dateien werden ignoriert.
- Funktion `pjob`: Falls die maximale Zahl gleichzeitig zu bearbeitender Aufträge bereits erreicht ist, wird gewartet, bis ein laufender Auftrag fertig wird. Sonst wird für den Auftrag ein Thread (*Arbeits-Thread*) erzeugt. Die Thread-Startfunktion `tstart` führt zur eigentlichen Auftragsbearbeitung die Funktion `void send_mail(char *mfile)` aus.
- Zur Koordinierung der maximalen Threadzahl müssen Sie die Anzahl der erzeugten Threads in einer normalen Variablen mitzählen. Das Inkrementieren erfolgt bei der Threaderzeugung, das Dekrementieren führt der Thread vor dem Terminieren aus. Beachten Sie die Nebenläufigkeit dieser Operationen! Wenn die maximale Threadzahl erreicht ist, wartet der Haupt-Thread mit `pthread_cond_wait` (und nicht mit `pthread_join`!) auf das Terminieren eines Arbeits-Threads.
- Funktion `send_mail`: In einer `mfile`-Datei steht in der ersten Zeile der Empfänger in der Form *`user@hostname`* und ab der zweiten Zeile bis zum Dateiende der Inhalt der Mail. Es wird eine Verbindung zu *`hostname`*, Port 25, aufgebaut und Folgendes übertragen:

  ```
  RCPT TO: user@hostname
  DATA
  Mailinhalt
  .
  ```

  Abschliessend wird die `mfile`-Datei gelöscht.
- Sie können davon ausgehen, dass alle benutzten Dateinamen sowie die Zeilen im Mailinhalt jeweils nie länger als 1024 Zeichen sein werden und dass der Inhalt der `mfile`-Datei in jedem Fall korrekt aufgebaut ist.

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 - Sie können aber nicht davon ausgehen, dass Sie ausschließlich nur diese Funktionen benötigen.

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


/* Funktionsdeklarationen, globale Variablen */

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

/* Funktion main */

.....
.....
{
  /* Argumente prüfen */

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

A:


```
    /* Initialisierungen */
    {

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

    /* Auftraege suchen und bearbeiten */
    {

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

} /* Ende Funktion main */
```

T:
D:

```
/* Auftragsbearbeitung: Funktion pjob */
void pjob(char *mfile) {
    /* Variablendefinitionen, Vorbereitungen */
```

```
/* Regelung der Threadzahl. Koordinierung mit Threads */
```

T:
K:


```
    /* Thread erzeugen  */
```

```
} /* Ende Funktion pjob */


/* Thread-Startfunktion: Funktion tstart */

void *tstart(void *mfile) {
    /*  Mail zustellen */
```

```
    /*  Ressourcen freigeben, Thread beenden */
```

```
} /* Ende Funktion tstart */
```

T:
K:

```
/* Mailuebertragung: Funktion send_mail */

void send_mail(char *mfile) {

  /* Variablendefinitionen */
  char user[1024], hostname[1024];
  /* m-Datei oeffnen */
  /* Empfaenger ermitteln */
   fscanf(mf, "%s@%s\n", user, hostname);
   /* Adress-Struktur fuer hostname aufbauen */
   {
      struct sockaddr_in addr;
      struct hostent *h;
      if ((h=gethostbyname(hostname)) == NULL) {
          perror(hostname);
          return;
      }
      memcpy(&addr.sin_addr, h->h_addr_list[0], sizeof(addr.sin_addr));
      addr.sin_port = htons(25);
      addr.sin_family = AF_INET;
```



```
      /* Verbindung aufbauen */
  }
  /* Mailinhalt uebertragen */
  /* Auftrag abschliessen */
} /* Ende Funktion send_mail */
```

<-

U:

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

Ein Aufruf von

```
make mailq
```

soll das Programm erzeugen, ein Aufruf von

```
make clean
```

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

M:

Aufgabe 3: (20 Punkte)

Man kann Schedulingverfahren nach verschiedenen Kriterien in Kategorien einteilen. Beschreiben Sie für die folgenden Kategorisierungen jeweils die Eigenschaften entsprechender Schedulingstrategien, nennen Sie konkrete Beispiele für solche Strategien und geben Sie an, in welchen Anwendungssituationen entsprechende Strategien sinnvoll eingesetzt werden.

a) Kooperatives bzw. präemptives Scheduling

b) Deterministisches bzw. probabilistisches Scheduling

c) Statisches bzw. dynamisches Scheduling

d) Asymmetrisches bzw. symmetrisches Scheduling

Aufgabe 4: (10 Punkte)

Bei virtuellen Adressräumen können Teile des Speichers auf Hintergrundspeicher ausgelagert sein.

a) Wie wird im System bei einem Speicherzugriff erkannt, dass der entsprechende Speicher ausgelagert ist?

b) Was läuft im System nach dieser Erkennung ab? Beschreiben Sie die einzelnen Schritte, die im Betriebssystem abgewickelt werden, um den Speicher verfügbar zu machen und geben Sie an, welche Prozesszustände der auslösende Prozess dabei jeweils einnimmt.