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) Welche Aussage bezüglich der Freispeicherverwaltung mittels einer nach Blockgröße sortierten verketteten Liste ist **falsch**? 2 Punkte

- ❏ Der Suchaufwand ist konstant.
- ❏ Eine Verschmelzung benachbarter Freispeicherbereiche ist aufgrund der Sortierung besonders einfach möglich.
- ❏ Bei einer Speicheranforderung muss die Liste u.U. vollständig durchlaufen werden.
- ❏ Die Datenstruktur eignet sich besonders zur Realisierung einer worst-fit Strategie.

b) Wie wird erkannt, dass eine Seite eines virtuellen Adressraums gerade ausgelagert ist? 2 Punkte

- ❏ Bei ausgelagerten Seiten ist im Seitendeskriptor das "present bit" nicht gesetzt. Das Betriebssystem erkennt dies und löst bei einer Adressauflösung für solch eine Seite einen Trap aus.
- ❏ Bei Programmen, die in virtuellen Adressräumen ausgeführt werden sollen, erzeugt der Compiler speziellen Code, der vor Betreten einer Seite die Anwesenheit überprüft und ggf. die Einlagerung veranlasst.
- ❏ Im Seitendeskriptor wird ein spezielles Bit geführt, das der MMU zeigt, ob eine Seite eingelagert ist oder nicht. Falls die Seite nicht eingelagert ist, löst die MMU einen Trap aus.
- ❏ Im Seitendeskriptor steht bei ausgelagerten Seiten eine Adresse des Hintergrundspeichers und der Speichercontroller leitet den Zugriff auf den Hintergrundspeicher um.

c) Mit logischen Adressräumen kann man mehrere Zwecke erreichen. Was gehört **nicht** dazu? 2 Punkte

- ❏ Sicherheit: man kann unbefugten Zugriff auf Daten verhindern.
- ❏ Schutzmechanismus: man kann die Auswirkungen von Berechnungsfehlern oder technischen Fehlern (wie z. B. Bitkipper) begrenzen.
- ❏ Virtualisierung: man kann in einem Programmlauf mehr Speicher nutzen, als physikalisch vorhanden ist.
- ❏ bessere Verwaltung: man kann Speicherbereiche mit unterschiedlicher Bedeutung voneinander abgrenzen.

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

- ❏ Einen speziellen Cache der MMU, der Informationen aus den zuletzt genutzten Seitendeskriptoren vorhält.
- ❏ Einen speziellen Cache der MMU, der den Inhalt der zuletzt angesprochenen Speicherzellen vorhält.
- ❏ Einen speziellen Cache der CPU, der die zuletzt ausgeführten Maschinenbefehle zwischenspeichert (beschleunigt vor allem den Ablauf von Schleifen).
- ❏ Der TLB ist eine schnelle Umsetzeinheit der MMU, die physikalische in logische Adressen umsetzt.

e) Was versteht man unter einem Interrupt? 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.
- ❏ Durch eine Signalleitung wird der Prozessor veranlasst, die gerade bearbeitete Maschineninstruktion abzubrechen.
- ❏ 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.

f) Welche Aussage über die Prozesszustände ist in einem Monoprozessor-Betriebssystem **richtig**? 3 Punkte

- ❏ Findet gerade keine Prozessumschaltung statt und ist kein Prozess im Zustand *laufend*, so ist auch kein Prozess im Zustand *bereit*.
- ❏ Ein Prozess im Zustand *blockiert* muss warten, bis der laufende Prozess den Prozessor abgibt und wird dann unmittelbar in den Zustand *laufend* überführt.
- ❏ Muss ein Prozess an einer Ein-, Ausgabeoperation warten, wird er vom Zustand *laufend* in den Zustand *bereit* überführt.
- ❏ Es befinden sich bis zu zwei Prozesse im Zustand *laufend* und damit in Ausführung auf dem Prozessor (Vordergrund- und Hintergrundprozess).

g) 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

h) Welche Aussage zum Thema Ablaufplanung ist **richtig**? 2 Punkte

- ❏ Bei der FCFS-Strategie kann es aufgrund des Konvoieffekts zu hohen Antwortzeiten kommen.
- ❏ Offline-Einplanungsverfahren eignen sich vor allem für den Einsatz auf mobilen Geräten, da diese auch ohne Internetverbindung arbeiten können.
- ❏ In Echtzeitsystemen kommt es auf maximalen Durchsatz an, weshalb hier ausschließlich nicht-unterbrechbare Schedulingverfahren verwendet werden.
- ❏ Asymmetrische Einplanungsverfahren zielen auf eine optimale Behandlung von Prozessmengen, die sich in E/A- und CPU-Intensität stark voneinander unterscheiden.

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

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

j) Welches der folgenden Verfahren ist zur Synchronisation des Zugriffs auf gemeinsame Daten in einem Multiprozessorsystem **nicht geeignet**? 2 Punkte

- ❏ Aktives Warten bis die Sperre aufgehoben wird.
- ❏ Binäre Semaphore
- ❏ Spezialbefehle wie `cas`
- ❏ Sperre der Interrupts

k) Was versteht man unter einem unsicheren Zustand im Rahmen der Verklemmungsvermeidung? 2 Punkte

- ❏ Bei einem unsicheren Zustand ist unklar, ob eine P-Operation blockieren wird oder nicht.
- ❏ Wenn sich ein System in einem unsicheren Zustand befindet, wird es früher oder später mit Sicherheit zu einer Verklemmung kommen.
- ❏ Wenn sich ein System in einem unsicheren Zustand befindet, ist es verklemmt.
- ❏ Wenn ein System sich in einem unsicheren Zustand befindet, dürfen keine weiteren Ressourcen mehr angefordert werden.

l) 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

- ❏ 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.
- ❏ Über Zeiger kann man alle Daten des Aktivierungsblocks der aufrufenden Funktion verändern.
- ❏ 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.

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

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

n) Welche Aussage über das aktuelle Arbeitsverzeichnis (Current Working Directory) **trifft zu**? 2 Punkte

- ❏ Besitzt ein UNIX-Prozess kein Current Working Directory, so beendet sich der Prozess mit einem Segmentation Fault.
- ❏ Mit dem Systemaufruf `chdir()` kann das aktuelle Arbeitsverzeichnis durch den Vaterprozess verändert werden.
- ❏ Das aktuelle Arbeitsverzeichnis erbt der Prozess vom aktuellen Benutzer
- ❏ Pfadnamen, die nicht mit dem Zeichen ’/’ beginnen, werden relativ zu dem aktuellen Arbeitsverzeichnis interpretiert

Aufgabe 2: (60 Punkte)

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

a) Schreiben Sie ein Programm `websrv` (Webserver), welches Dateien aus einem Basisverzeichnis über ein vereinfachtes HTTP-Protokoll ausliefert. Für jede aufgebaute Verbindung wird ein Kindprozess ereugt, der genau eine Datei an die Gegenstelle ausliefert und die Verbindung dann schließt. Das Programm erhält den Pfad des Basisverzeichnisses, eine Grenze für die maximale Zahl an laufenden Kindprozessen sowie optional die Portnummer, auf welcher der Server Anfragen entgegennimmt. Wird der Port nicht übergegeben, wird der Standardport 80 verwendet.
Eine Anfrage an den Server besteht aus exakt einer Zeile in folgendem Format:

```
GET /lame/node6.html
```

Der Pfad ist relativ zum Basisverzeichnis zu interpretieren, der führende ’/’ muss hierbei ignoriert werden. Sie können davon ausgehen, dass der so erhaltene Pfadname nicht länger als 4096 Zeichen ist.

Das Programm soll folgendermaßen ablaufen:

- Nach der Initialisierung nimmt das Hauptprogramm Verbindungen entgegen. Hierbei sollen max. 13 Verbindungsanfragen vom Betriebssystem gepuffert werden. Jeweils vor der Annahme einer Verbindung prüft das Programm, ob die maximale Zahl von Kindprozessen bereits erreicht ist und wartet in diesem Fall passiv auf das Ende eines Kindprozesses.
- Nach Annahme der Verbindung wird ein Kindprozess erzeugt. Dieser ruft zur Anfragebearbeitung die Funktion **`serve`** auf. Der Vaterprozess nimmt nach Erzeugung des Kindes weitere Anfragen entgegen.
- Funktion **`void serve(int filedesc)`**: Die Funktion liest zunächst die Anfragezeile vom übergebenen Filedeskriptor und prüft dann, ob es sich beim angeforderten Pfad um ein Verzeichnis handelt. Ist dies der Fall, so wird eine Datei **`index.html`** in diesem Verzeichnis ausgeliefert, ansonsten die angeforderte Datei selbst. Hierzu wird die Funktion **`sendFile`** mit dem entsprechenden Pfadnamen sowie einem **`FILE*`** der Socketverbindung aufgerufen.
- Funktion **`void sendFile(char *path, FILE *csp)`**: Die Funktion öffnet die übergebene Datei und überträgt deren Inhalt auf die Socketverbindung.
- Im Fehlerfall soll das Programm eine Fehlermeldung auf den Standardfehlerkanal ausgeben und den Prozess, in welchem der Fehler aufgetreten ist, beenden. Steht der Fehler im Zusammenhang mit der Bearbeitung einer Anfrage, soll ausserdem an den Client eine Fehlermeldung übertragen werden.
- Zombieprozesse sollen unmittelbar nach deren Entstehen beseitigt werden. Richten Sie hierzu eine entsprechende Signalbehandlung ein.

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 <stdio.h>
#include <sys/stat.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <signal.h>
#include <netinet/in.h>
#include <netinet/ip.h>
#include <errno.h>
#include <sys/wait.h>

/* Funktionsdeklarationen, globale Variablen */

/* Funktion main */
```

A:


```
/* Argumente pruefen und auswerten*/

/* Signalbehandlung einrichten */
```

A:
S:

```
/* Verbindungsannahme vorbereiten */
```

T:


```
/* Hauptschleife */

    /* Verbindung annehmen */

    /* Sohnprozess zur Bearbeitung der Anfrage erzeugen */

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

T:
S:

```
/* Signalbehandlung für SIGCHLD */
```

```
/* Ende Signalbehandlung für SIGCHLD */

/* Anfragebearbeitung: Funktion serve */
void serve(int cs)
{
    /* Variablendefinitionen, Vorbereitungen */
```

```
    /* URL lesen */
```

S:
V:


```
    /* Dateityp ermitteln und Uebertragungsfunktion aufrufen */
```

```
} /* Ende Funktion serve */
```

V:

```
/* Dateiuebermittlung: Funktion sendFile */
void sendFile(const char *path, FILE *csp)
{
   /* Datei oeffnen */
```

```
/   /* Daten senden */
```

```
} /* Ende Funktion sendFile */
```

F:

Aufgabe 3: (9 Punkte)

Man unterscheidet verschiedene Arten von Prozessen bzw. Threads im Hinblick auf ihre "Gewichtsklasse". Erläutern Sie die verschiedenen Kategorien und ihre Eigenschaften. Beschreiben Sie vor allem auch, wodurch das unterschiedliche "Gewicht" bedingt ist.

Aufgabe 4: (21 Punkte)

a) Skizzieren Sie in einer programmiersprachen-ähnlichen Form die Programmierung der bekannten Semaphore-Operationen **`P()`** und **`V()`** mit Hilfe der z. B. von der pthreads-Koordinierung bekannten mutex-Operationen `lock()` und `unlock()` sowie der Condition-Variablen-Funktionen `cond_wait()` und `cond_broadcast()`.

Der Semaphore-Wert selbst sei eine globale Variable `int s`.

Bei dieser Aufgabe kommt es nicht auf die genaue Parametrierung der lock- und cond-Funktionen an, sondern nur auf ihren prinzipiellen Einsatz an den richtigen Stellen.

b) Was versteht man unter einer Verklemmung und was sind die Voraussetzungen, damit eine Verklemmung auftreten kann?

c) Eine Maßnahme gegen Verklemmungen ist Verklemmungsvorbeugung (*deadlock prevention*). Beschreiben Sie die zwei Methoden, wie man hierbei vorgehen kann und geben Sie ein konkretes Beispiel für eine Vorbeugungsmaßnahme an.