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) Wie funktioniert Adressraumschutz durch Eingrenzung? 2 Punkte

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

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

- ❏ buddy-Verfahren sind nur bei sehr leistungsfähigen Rechnern und großem Speicher einsetzbar, weil sie aufwändig in der Berechnung sind und viel Verschnitt verursachen.
- ❏ best-fit ist in jedem Fall das beste Verfahren
- ❏ die worst-fit-Strategie kann einem mit der kürzesten Antwortzeit einen ausreichend großen Speicherbereich liefern
- ❏ die worst-fit-Strategie ist lediglich theoretisch interessant, da es in der Praxis nie sinnvoll ist, den am schlechtesten passenden Speicherplatz zuzuweisen.

d) Wie gross ist die Seitenkacheltabelle zur Adressabbildung von logischen auf physikalische Adressen in einem System mit Seitenadressierung wenn ein Eintrag in der Seitenkacheltabelle 32 Bit gross ist, der virtuelle Adressraum 4 GigaByte umfasst und eine Speicherseite 4096 Byte gross ist. 2 Punkte

- ❏ 32 Byte
- ❏ 512 bis 8.192 Byte
- ❏ 1.048.576 Byte
- ❏ 4.194.304 Byte

e) Sie kennen den Begriff Demand-Paging. Welche Aussage dazu ist **richtig**? 3 Punkte

- ❏ Demand-Paging setzt eine segmentierte Speicherverwaltung voraus.
- ❏ Demand-Paging benötigt keinerlei Hardware-Unterstützung, da sich alle benötigten Mechanismen auch ohne MMU realisieren lassen.
- ❏ Demand-Paging erlaubt es größere logische Adressräume anzulegen, als Hauptspeicher vorhanden ist. Allerdings muss vorausgesetzt werden, dass ein Prozess nicht alle Seiten des logischen Adressraums tatsächlich anspricht.
- ❏ Demand-Paging lädt eine Seite erst dann in den Hauptspeicher, wenn sie tatsächlich angesprochen wird. Nicht benutzte Seiten werden unter Umständen aus dem Hauptspeicher ausgelagert.

f) Was versteht man unter Virtuellem Speicher? 2 Punkte

- ❏ Adressierbarer Speicher in dem sich keine Daten speichern lassen, weil er physikalisch nicht vorhanden ist.
- ❏ Einen logischen Adressraum.
- ❏ Virtueller Speicher kann größer sein als der physikalisch vorhandene Arbeitsspeicher. Gerade nicht benötigte Speicherbereiche können auf Hintergrundspeicher ausgelagert werden.
- ❏ Virtueller Speicher wird vom Compiler beim Binden angelegt.

g) 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.
- ❏ Durch die Verwendung von Semaphoren kann das unkontrollierte Flattern von Seiten synchronisiert werden.
- ❏ Seitenflattern tritt auf, wenn Seiten zur Defragmentierung im Speicher verschoben werden.

h) Ein Betriebssystem setzt logische Adressräume auf der Basis von Segmentierung ein. Welche Aussage ist **falsch**? 3 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.

i) In welcher der folgenden Situationen wird **kein** Trap ausgelöst? 2 Punkte

- ❏ Bei einem Systemaufruf.
- ❏ Wenn in einer Interrupt-Bearbeitung auf eine ungültige Adresse zugegriffen wird.
- ❏ Bei Aufruf eines privilegierten Maschinenbefehls in einem Programm auf Benutzerebene
- ❏ Wenn man auf der Tastatur <CTRL>-T eingibt.

j) Bei der Behandlung von Ausnahmen (Traps oder Interrupts) unterscheidet man zwei Bearbeitungsmodelle. Welche Aussage hierzu ist **richtig**? 3 Punkte

- ❏ Das Wiederaufnahmemodell dient zur Behandlung von Interrupts (Fortführung des Programms nach einer zufällig eingetretenen Unterbrechung). Da Traps deterministisch auftreten, ist bei ihnen das Modell nicht sinnvoll anwendbar.
- ❏ Nach dem Beendigungmodell werden Interrupts bearbeitet. Gibt man z. B. CTRL-C unter UNIX über die Tastatur ein, wird ein Interrupt-Signal an den gerade laufenden Prozess gesendet und dieser dadurch beendet.
- ❏ Interrupts dürfen auf keinen Fall nach dem Beendigungsmodell behandelt werden, weil überhaupt kein Zusammenhang zwischen dem unterbrochenen Prozess und dem Grund des Interrupts besteht.
- ❏ Das Betriebssystem kann Interrupts, die in ursächlichem Zusammenhang mit dem gerade laufenden Prozess stehen, nach dem Beendigungsmodell behandeln, wenn eine sinnvolle Fortführung des Prozesses nicht mehr möglich ist.

k) Wozu dient der Maschinenbefehl cas (compare-and-swap)? 2 Punkte

- ❏ Um bei Monoprozessorsystemen Interrupts zu sperren.
- ❏ Um auf einem Multiprozessorsystem einfache Modifikationen an Variablen ohne Sperren implementieren zu können.
- ❏ Um bei der Implementierung von Schlossvariablen (Locks) aktives Warten zu vermeiden.
- ❏ Zur Realisierung einer effizienten Verdrängungssteuerung bei einseitiger Symchronisation.

l) Welche Aussage zum Thema Prozesseinplanung ist **richtig**? 2 Punkte

- ❏ Preemptive Schedulingstrategien bilden die Grundlage zur Implementierung von CPU-Schutz.
- ❏ Kooperative Schedulingstrategien sind vor allem für Mehrbenutzersysteme geeignet.
- ❏ Probabilistische Strategien erlauben eine exakte Vorhersage der CPU-Auslastung
- ❏ Für deterministisches Scheduling ist kein Wissen über das Laufzeitverhalten der beteiligten Prozesse erforderlich.

m) Welche Aussage zur Verklemmungsvermeidung (deadlock avoidance) ist **richtig**? 2 Punkte

- ❏ Bei Verklemmungsvermeidung 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 Bankiers-Algorithmus wird beim Belegen eines Betriebsmittels geprüft, ob mit erfolgreicher Belegung ein unsicherer Zustand eintreten würde.
- ❏ Im Falle einer Verklemmung wird einer der beteiligten Prozesse beendet.

Aufgabe 2: (60 Punkte)

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

a) Schreiben Sie ein Programm `prspool` (Print Spooler), welches in dem Verzeichnis */var/spool* nach Druckaufträgen (Dateien mit der Dateiendung *.ps*, Dateiname max. `PATH_MAX` Zeichen lang) sucht und diese an eine Reihe von Netzwerkdruckern verteilt.

`prspool` erhält die Namen der Drucker, welche den Hostnamen der Drucker entsprechen, auf der Kommandozeile. Es muss mindestens ein Druckername übergeben werden, ansonsten soll das Programm mit einer Fehlermeldung terminieren.

Zur Bestimmung freier Drucker verwaltet `prspool` ein Feld mit einem Eintrag pro Drucker. Der Eintrag enthält entweder die Prozess-ID des gerade mit dem Drucker kommunizierenden Sohnprozesses, falls der Drucker belegt ist, oder den Wert 0, falls der Drucker frei ist. Ein belegter Drucker wird beim Terminieren des entsprechenden Kindprozesses in der Signalbehandlungsroutine wieder freigegeben. Steht für einen Druckauftrag kein freier Drucker zur Verfügung, soll das Hauptprogramm mittels `sigsuspend(2)` blockieren und auf das Freiwerden eines Druckers warten. **Achtung**: An dieser Stelle ist Synchronisation erforderlich: Während oder nach der Suche nach freien Druckern könnte ein Drucker frei werden und das Programm blockiert trotzdem weil es dies nicht mehr mitbekommt. Außerdem könnte ein Kindprozess terminieren, bevor der entsprechende Drucker überhaupt als belegt markiert werden konnte.

Das Programm soll folgendermaßen ablaufen:

- `prspool` durchsucht in einer Endlosschleife das Verzeichnis */var/spool* nach Druckaufträgen (Dateien mit der Endung *.ps*). Um eine erneute Bearbeitung in weiteren Suchläufen zu vermeiden, wird die Endung der Auftragsdatei zunächst in *.px* geändert. Der Druckauftrag wird dann in der Funktion `process` weiterbearbeitet. Der Funktion wird das Feld mit den Druckern und der Name der Auftragsdatei übergeben.
- Funktion `void process(char **printers, char *jobname)`: Die Funktion ermittelt zunächst einen freien Drucker (wartet ggf. wie oben beschrieben). Dann wird ein Kindprozess erzeugt, der in der Funktion `printjob()` den Auftrag an den Drucker übermittelt. Der Drucker wird als belegt markiert.
- Funktion `void printjob(char *jobfile, char *printer)`: In dieser Funktion wird zunächst der Name des Druckers zu einer IP-Adresse aufgelöst und eine TCP/IP-Verbindung zum Drucker auf dem Port 9100 aufgebaut. Über diese Verbindung wird der Inhalt der Auftragsdatei an den Drucker übertragen und die Auftragsdatei danach gelöscht.

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 <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <stdlib.h>
#include <stdio.h>
#include <signal.h>
#include <netinet/in.h>
#include <netdb.h>
#include <errno.h>
#include <dirent.h>
#include <string.h>
#include <sys/stat.h>
#include <fcntl.h>

#define SPOOLDIR "/var/spool"
#define PRPORT 9100

/* Funktionsdeklarationen, globale Variablen */
```

A:


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

```
   /* Argumente pruefen */
```

```
   /* Feld zur Verwaltung der Druckerbelegung allokieren */
```

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

```
   if(chdir(SPOOLDIR) == -1) {
      perror(SPOOLDIR); exit(EXIT_FAILURE);
   }
```

A:
S:

```
   /* Hauptschleife */
```

```
   /* Spool Verzeichnis lesen und Auftraege bearbeiten */
```

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

D:


```
/* Auftragsbearbeitung: Funktion process */
static void process(char **printers, char *jobname)
{
    /* Variablendefinitionen, Vorbereitungen */
```

```
    /* Freien Drucker bestimmen oder warten */
```

—>

```
    /* Kindprozess zur Bearbeitung erzeugen */
```

```
} /* Ende Funktion process */

/* Signalbehandlung für SIGCHLD */
static void chldhandler(int s)
{
```

```
} /* Ende Funktion chldhandler */
```

P:


```
/* Datei-Uebertragung: Funktion printjob */
static void printjob(const char *jobname, const char *printername)
{
    /* Variablendefinitionen, Vorbereitungen */
   struct sockaddr_in sin;
   struct hostent *paddr;
```

```
   /* Hostnamen zu Adresse aufloesen */
   paddr = gethostbyname(printername);
   if(paddr == NULL) {
      herror(printername); exit(EXIT_FAILURE);
   }
   sin.sin_family = AF_INET;
   sin.sin_port=htons(PRPORT);
   memcpy(&sin.sin_addr.s_addr, paddr->h_addr, paddr->h_length);

   /* Socket erzeugen und mit Drucker verbinden  */
```

—>

```
    /* Auftragsdatei oeffnen und uebertragen  */
```

```
} /* Ende Funktion printjob */
```

J:

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

Ein Aufruf von

```
make prspool
```

soll das Programm erzeugen, ein Aufruf von

```
make clean
```

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

M:

Aufgabe 3: (23 Punkte)

a) Erläutern Sie die Begriffe Programm und Prozess.

b) Beschreiben Sie die unterschiedlichen Arten ("Gewichtsklassen") von Prozessen/ Threads.

c) Bei der Prozesseinplanung (Scheduling) unterscheidet man kurz-, mittel- und langfristige Einplanung.
Skizzieren Sie die dabei auftretenden Prozesszustände und die möglichen Übergänge. Schreiben Sie an die Übergänge jeweils ein kurzes Stichwort, das die Ursache für so einen Übergang beschreibt.

Aufgabe 4: (7 Punkte)

Erklären Sie die Funktionsweise von Bedingungsvariablen und zu welchem Zweck sie eingesetzt werden.