

Aufgabe 1: (40 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 groß ist typischerweise eine Seite bei Seitenadressierung?

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

1 Punkt

b) Welche Antwort stellt **kein** Attribut einer Datei eines UNIX-UFS-Dateisystems dar?

- Zeitpunkt der letzten Änderung
- Gruppenzugehörigkeit
- Zugriffsrechte
- Dateiendung (z.B. .html oder .jpeg)

1 Punkt

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

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

2 Punkte

d) Welche Aussage bezüglich des Systemaufrufs `accept()` ist richtig?

- Der Systemaufruf nimmt eine Verbindung aus einer Warteschlange von neuen Verbindungen und bekommt für diese neue Verbindung eine neue Portnummer.
- Ein `accept()` Systemaufruf kann nicht blockieren, da durch den Aufruf von `listen()` sichergestellt wurde, dass immer eine Verbindungsanfrage vorliegt.
- Der `accept()` Systemaufruf nimmt eine Verbindung aus einer mit `listen()` eingerichteten Warteschlange und erzeugt für diese Verbindung einen neuen Socket.
- Der Systemaufruf `accept()` teilt dem Betriebssystem mit, dass der Prozess bereit ist Verbindungen entgegen zu nehmen.

2 Punkte

e) Welche Antwort trifft für die Eigenschaften eines UNIX/Linux-Filedeskriptors zu?

- 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.

2 Punkte

f) Ein *laufender* Prozess wird in den Zustand *blockiert* überführt. Welche Aussage passt **nicht** zu diesem Vorgang?

- Der Prozess wartet auf Daten von der Festplatte.
- Der Prozess wartet mit dem Systemaufruf `wait()` auf die Terminierung eines Sohn-Prozesses.
- Der Prozess hat einen Seitenfehler für eine Seite, die bereits in die Freiliste der Seiten eingetragen aber noch im Hauptspeicher vorhanden ist.
- Der Prozess wartet auf eine Tastatureingabe.

2 Punkte

- g) 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_0 -Strategie an und wird daher in fast allen realen Systemen benutzt.
 - Die Second-Chance-Strategie benötigt ein Referenzbit in jedem Eintrag der Seitenkachelntabelle.
- h) Für welchen Zweck wird der Systemaufruf `shmat()` benutzt? 2 Punkte
- Mit dem Systemaufruf `shmat()` kann ein Prozess einen Teil seines Speichers exportieren, sodass andere Prozesse darauf zugreifen können.
 - Der Systemaufruf `shmat()` blendet ein bestehendes Shared-Memory-Segment in den logischen Adressraum des aufrufenden Prozesses ein.
 - Der Aufruf von `shmat()` erzeugt ein neues Speichersegment, auf das verschiedene Prozesse gleichberechtigt zugreifen können.
 - Damit zwei Prozesse auf einen gemeinsam genutzten Speicherbereich zugreifen können müssen diese zuerst vom Betriebssysteme jeweils einen Semaphor anfordern. Mit der Hilfe des Semaphors und dem Systemaufruf `shmat()` können die Prozesse anschliessend auf den gemeinsamen Speicher zugreifen.
- i) Sie kennen den Translation-Look-Aside-Buffer (TLB). Welche Aussage ist richtig? 2 Punkte
- Der TLB verkürzt die Zugriffszeit auf den physikalischen Speicher, da ein Teil des möglichen Speichers in einem sehr schnellen Pufferspeicher vorgehalten wird.
 - Der TLB puffert Daten bei der Ein-, Ausgabebehandlung und beschleunigt diese damit.
 - Verändert sich die Speicherabbildung von logische auf physikalische Adressen aufgrund einer Adressraumumschaltung, so werden auch die Daten im TLB ungültig.
 - Wird eine Speicherabbildung im TLB nicht gefunden, wird der auf den Speicher zugreifende Prozess mit einer Schutzraumverletzung (Segmentation Fault) abgebrochen.

- j) Welche Aussage über UNIX-Semaphoren ist richtig? 3 Punkte
- UNIX-Semaphoren können unmittelbar das Verhalten von PV-Chunk- und UP-DOWN-Systemen nachbilden.
 - Die Operation `semop()` kann sich auf maximal zwei UNIX-Semaphoren gleichzeitig beziehen.
 - UNIX-Semaphoren können das Verhalten von PV-Multiple-Semaphoren nachbilden.
 - UNIX-Semaphoren sind nur für die Koordinierung von Speicherzugriffen geeignet, nicht jedoch für die Koordinierung von Aktivitätsträgern in mehreren Prozessen.
- k) 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

- l) Ein System setzt Segmentierung und Seitenadressierung ein. Zwei Prozesse sollen ein Segment gemeinsam benutzen. Wie geht das Betriebssystem vor, um das gemeinsame Segment einzurichten? 3 Punkte
- Das Betriebssystem legt eine Seitenkacheltable für das gemeinsame Segment an und trägt diese bei beiden Prozessen in die jeweiligen Segmenttabellen ein.
 - Das Betriebssystem trägt jeweils eine Seitenkacheltable in die prozesseigene Segmenttable ein und sorgt dafür, dass die Einträge jeweils auf die gleichen Kacheln im Hauptspeicher verweisen.
 - Gemeinsame Segmente können nur durch den Prozess jedoch nicht vom Betriebssystem eingerichtet werden.
 - Die beiden Prozesse bekommen die gleiche Segmenttable zugewiesen.
- m) 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.
- n) Welche Aussage über den Rückgabewert von `fork()` ist richtig? 1 Punkt
- Der Sohn-Prozess bekommt die Prozess-ID des Vater-Prozesses.
 - Im Fehlerfall wird im Sohn-Prozess -1 zurückgeliefert.
 - Dem Vater-Prozess wird die Prozess-ID des Sohn-Prozesses zurückgeliefert.
 - Bei erfolgreicher Ausführung kehrt `fork()` nicht zum Vater-Prozess zurück.

- o) Welche Aussage bzgl. eines asymmetrischen Verschlüsselungsverfahrens ist richtig? 3 Punkte
- Beim Erstellen einer digitalen Signatur wird mit Hilfe des öffentlichen Schlüssels ein Hash-Wert einer Nachricht verschlüsselt.
 - Bei asymmetrischen Verfahren und vielen Kommunikationsteilnehmern wird lediglich pro Teilnehmer ein Schlüsselpaar mit öffentlichem und privatem Schlüssel benötigt.
 - Das RSA-Verfahren nach Rivest, Shamir und Adleman basiert auf großen Primzahlen und ist daher besonders effizient implementierbar. So lassen sich große Nachrichten leicht direkt verschlüsseln.
 - Aus dem privaten Schlüssel muss sich leicht ein öffentlicher Schlüssel ableiten lassen.
- p) Welcher UNIX-Systemaufruf wird bei der Verwendung von Sockets auf keinen Fall gebraucht? 1 Punkt
- `open()`
 - `read()`
 - `write()`
 - `close()`
- q) Bei RAID 1 wird durch das Verteilen der Daten auf mehreren Platten ein sogenanntes gestreiftes Plattensystem erzeugt. Welche Aussage dazu ist richtig? 2 Punkt
- Es dürfen nicht mehr als 5 Festplatten beteiligt sein, da sonst die Paritätsinformation nicht mehr gebildet werden kann.
 - Der Lesezugriff auf ein gestreiftes Plattensystem ist schneller, da mehrere Platten gleichzeitig beauftragt werden können.
 - Die Paritätsinformation wird gleichmäßig über alle beteiligten Platten verteilt.
 - Die Daten auf einem RAID 1 Festplattensystem sind besonders sicher, da beim Ausfall von nur einer Platte die verlorengegangenen Daten wieder hergestellt werden können.

- r) 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.
- s) User-Level- und Kernel-Level-Threads unterscheiden sich in verschiedenen Eigenschaften. Welche Kombination ist richtig? 3 Punkte
- Bei User-Level-Threads können anwendungsabhängig Schedulingstrategien eingesetzt werden; blockierende Systemaufrufe von Kernel-Level-Threads blockieren keine anderen Threads.
 - Kernel-Level-Threads werden sehr effizient umgeschaltet; User-Level-Threads blockieren sich bei blockierenden Systemaufrufen gegenseitig.
 - Bei Kernel-Level-Threads ist die Schedulingstrategie meist vorgegeben; User-Level-Threads können Multiprozessoren ausnutzen.
 - User-Level-Threads werden nicht effizient umgeschaltet; blockierende Systemaufrufe von Kernel-Level-Threads blockieren keine anderen Threads.
- t) 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 `r1sd`, das als Server-Prozess Anfragen nach Directory-Inhalten über TCP/IP beantwortet.

Das `r1sd`-Programm erhält als Argument die Nummer des Ports auf dem Verbindungen angenommen werden sollen.

Das Programm soll folgendermaßen arbeiten:

- Es nimmt Verbindungen über beliebige IP-Adressen auf dem angegebenen Port an.
- Es wird jeweils eine Verbindung angenommen und in einem Sohnprozess bearbeitet während der Vaterprozess für die nächste Verbindung wieder bereitsteht.
- Der Client sendet über die Verbindung den Namen des aufzulistenden Directories.
- Der Server gibt die Namen aller Dateien, die in dem angeforderten Directory angelegt sind aus (unsortiert, jeweils ein Dateiname in einer Zeile) — dieser Teil der Aufgabe wird in einer Funktion mit Namen "ls" bearbeitet.
- Der bearbeitende Sohnprozess terminiert anschließend, der Vaterprozess braucht sich in der Lösung nicht um terminierende Sohnprozesse kümmern.

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

```

```

/* Funktionsdeklarationen, globale Variablen */

```

```

/* Funktion main */

```

```

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

```

 A:

```

/* Socket oeffnen */

```

```

/* Socket an angegebenen Port
und beliebige IP-Adressen binden */

```

```

/* Warteschlange ankommender Verbindungen
auf 5 einstellen */

```

 S:

/ Server-Schleife */*

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

/ Verbindung annehmen */*

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

/ Prozess zur Kommandoausführung erzeugen */*

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

/ Fehler bei Prozesserzeugung */*

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

S:
P:

/ Sohnprozess */*

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

3P

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

2P

/ Vaterprozess */*

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

1S

} / Ende Funktion main */*

P: S:

```
/* ls-Funktion */
```

d

o

r

a

L:

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

Ein Aufruf von

```
make rlsd
```

soll das Programm erzeugen, ein Aufruf von

```
make clean
```

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

Aufgabe 3: (40 Punkte)

Festplatten werden in vielen Betriebssystemen zum Speichern von Dateien verwendet.

- Beschreiben Sie kurz den Aufbau einer Festplatte und wie einzelne Datenblöcke auf einer Festplatte organisiert werden. (6 Punkte)
- Dateien benötigen oft mehr als einen Datenblock. Beschreiben sie kurz die nachfolgenden Speicherungsarten und deren Vor- und Nachteile sowie ein Dateisystem als Beispiel. (17 Punkte)
 - Kontinuierliche Speicherung
 - Verkettete Speicherung
 - Indizierte Speicherung
 - Baumsequentielle Speicherung
- Was ist ein Journalled-File-Systems und welche Vor- und Nachteile bringt der Einsatz eines solchen Dateisystems? (5 Punkte)
- Da Festplattenzugriffe sehr langsam sind, entstehen für einen Prozess lange Wartezeiten. Beschreiben Sie die Zustände, die ein Prozess in diesem Falle und allgemein im Laufe seiner Ausführung einnehmen kann und erläutern Sie, warum/wann ein Übergang in einen anderen Zustand stattfindet (evtl. mittels einer Skizze). (12 Punkte)