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) Sie kennen den Translation-Look-Aside-Buffer (TLB). Welche Aussage ist richtig? 2 Punkte

- ❏ Der TLB verkürzt die Speicherzugriffszeit, da ein Teil der möglichen Speicherabbildungen in einem sehr schnellen Pufferspeicher vorgehalten wird.
- ❏ Der TLB puffert Daten bei der Ein-/Ausgabebehandlung und beschleunigt diese damit.
- ❏ Der TLB ist eine schnelle Umsetzeinheit der MMU, die physikalische in logische Adressen umsetzt.
- ❏ Wird eine Speicherabbildung im TLB nicht gefunden, wird der auf den Speicher zugreifende Prozess mit einer Schutzraumverletzung (Segmentation Fault) abgebrochen.

c) Welche Aussage ist bezüglich der Seitenersetzungsstrategie Least Recently Used (LRU) richtig? 2 Punkte

- ❏ Die LRU-Strategie benötigt ein Referenzbit in jedem Eintrag der Seitentabelle.
- ❏ Als Auswahlkriterium für die Ersetzung einer Seite wird die Zeit seit dem letzten Zugriff auf die Seite verwendet.
- ❏ Zur Implementierung von LRU benötigt man eine sehr genaue Systemuhr.
- ❏ Die LRU-Strategie gewährleistet, dass immer die Seiten eingelagert sind, auf die in der Zukunft zugegriffen wird.

d) Wie groß ist die Seitentabelle zur Adressabbildung von logischen auf physikalische Adressen in einem System mit Seitenadressierung, wenn ein Eintrag in der Seitentabelle 32 Bit groß ist, der virtuelle Adressraum 4 GigaByte umfasst und eine Speicherseite 1024 Byte groß ist. 2 Punkte

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

e) Welche Antwort trifft für die Eigenschaften eines UNIX/Linux-Filedeskriptors zu? 2 Punkte

- ❏ 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.

f) Betrachten Sie folgende Aussagen zur Speicherung der Daten einer Datei auf Festplatte. Welche Aussage ist **falsch**? 2 Punkte

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

g) Namensräume dienen u. a. der Organisation von Dateisystemen. Welche Aussage ist **richtig**? 2 Punkte

- ❏ Flache Namensräume sind besonders einfach implementierbar und damit vor allem für Mehrbenutzersysteme gut geeignet.
- ❏ Flache Namensräume erlauben pro Benutzer nur einen Kontext.
- ❏ Der Nachteil von hierarchischen Namensräumen besteht darin, dass das Dateisystem spezielle Funktionen zum Auflösen von Namenskonflikten implementieren muss.
- ❏ In einem hierarchisch organisierten Namensraum dürfen gleiche Namen in unterschiedlichen Kontexten enthalten sein.

h) Beim Einsatz von RAID-Systemen wird durch zusätzliche Festplatten ein fehlertolerierendes Verhalten erzielt. Welche Aussage dazu ist richtig? 2 Punkte

- ❏ Bei RAID-4-Systemen wird die Paritätsinformation gleichmäßig über alle beteiligten Platten verteilt.
- ❏ Der Lesezugriff auf ein gestreiftes Plattensystem – insbesondere auch auf ein RAID-5-System – ist schneller, da mehrere Platten gleichzeitig beauftragt werden können.
- ❏ Bei RAID 4 und 5 darf eine bestimmte Menge von Festplatten nicht überschritten werden, da es sonst nicht mehr möglich ist, die Paritätsinformation zu bilden.
- ❏ Bei RAID-5-Systemen sind mindestens 5 Festplatten nötig.

i) Was gehört nicht zu den typischen Dateiattributen? 2 Punkte

- ❏ die Länge des Dateiinhalts
- ❏ der Zeitpunkt des letzten lesenden Zugriffs
- ❏ die Prozess-Identifikation des Prozesses, der die Datei gerade geöffnet hat
- ❏ die Benutzer-Identifikation des Eigentümers der Datei

j) In einem UNIX-UFS-Dateisystem gibt es symbolische Namen/Verweise (Symbolic Links) und feste Links (Hard Links) auf Dateien. Welche Aussage ist richtig. 2 Punkt

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

k) 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.

l) User- und Kernel-Threads unterscheiden sich in verschiedenen Eigenschaften. Welche Kombination ist richtig? 3 Punkte

- ❏ Bei User-Threads können anwendungsabhängig Schedulingstrategien eingesetzt werden; Kernel-Threads können Multiprozessoren nicht ausnutzen.
- ❏ Kernel-Threads werden äußerst effizient umgeschaltet; User-Threads blockieren sich bei blockierenden Systemaufrufen gegenseitig.
- ❏ Bei Kernel-Threads ist die Schedulingstrategie meist vorgegeben; User-Threads können Multiprozessoren ausnutzen.
- ❏ User-Threads werden effizient umgeschaltet; blockierende Systemaufrufe von Kernel-Threads blockieren keine anderen Threads.

m) Sie kennen die vier notwendigen/hinreichenden Bedingungen für Verklemmungen. Welche Aussage ist **falsch**? 2 Punkte

- ❏ Verklemmungsvermeidung (Deadlock Avoidance) sorgt zur Laufzeit dafür, dass die 4. Bedingung (zirkuläres Warten) nicht eintreten kann.
- ❏ Bei Verklemmungsvorbeugung (Deadlock Prevention) muss dafür gesorgt werden, dass keine der notwendigen Bedingungen erfüllt ist.
- ❏ Indem man ein Auftreten der 4. Bedingung (zirkuläres Warten) durch Anwendung von Regeln ausschließt, wird Verklemmungsvorbeugung (Deadlock Prevention) erreicht.
- ❏ Bei Verklemmungserkennung wird keine Überprüfung der drei notwendigen Bedingungen durchgeführt.

n) In einem Segmentdeskriptor werden verschiedene Informationen über ein Segment eines logischen Adressraums gehalten. Was gehört sicher **nicht** dazu? 2 Punkte

- ❏ die Benutzer-Ids, für die diese Zugriffsrechte gelten
- ❏ die physikalische Adresse des Segmentanfangs im Hauptspeicher
- ❏ Zugriffsrechte (z. B. lesen, schreiben, ausführen)
- ❏ die Länge des Segments

o) 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.

Aufgabe 2: (60 Punkte)

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

a) Schreiben Sie ein Programm `ftpd` (File-Transfer-Protocol-Daemon), das als Server die Übertragung von Dateien und das Anzeigen von Directory-Inhalten über TCP/IP-Verbindungen unterstützt.

`ftpd` erzeugt beim Start eine feste Zahl von "Arbeiter-Threads", welche die eigentliche Bearbeitung der FTP-Verbindungen übernehmen. Die Anzahl der Threads muss dem Programm als Kommandozeilenparameter übergeben werden.

Das Programm soll folgendermaßen arbeiten:

- `ftpd` nimmt auf Port 21 Verbindungen von beliebigen Adressen entgegen. Es sollen maximal 8 Verbindungen anstehen können.
- Es wird jeweils eine Verbindung angenommen und ein File-Handle (`FILE *`) auf die Socketverbindung über einen Ringpuffer an einen Arbeiter-Thread übergeben. Die maximale Zahl bereits angenommener aber noch auf einen Arbeiter-Thread wartender Verbindungen sei auf 12 beschränkt.
- Ist der Ringpuffer voll, gibt `ftpd` nur die Meldung "`Server ueberlastet`" auf die Socket-Verbindung aus und bricht diese sofort wieder ab.
- Die Arbeiter-Threads entnehmen in ihrer Startfunktion `tstart` ein File-Handle aus dem Ringpuffer und rufen die Funktion `ftpConn` zur weiteren Bearbeitung der Verbindung auf. Bei einem leeren Ringpuffer wartet der Thread passiv auf eine neue eingehende Verbindung. Nach Behandlung einer Verbindung beginnt der Thread von vorn mit der Bearbeitung der nächsten Verbindung.
- Die Funktion `ftpConn` liest von der Socketverbindung eine Folge von Kommandos (maximale Länge 1023 Zeichen) und delegiert die Bearbeitung jeweils an andere Funktionen. Die Ausgabe der Kommandos (und Fehlermeldungen) soll jeweils (soweit sinnvoll möglich) auf den Socket erfolgen.
- Implementieren Sie folgende Kommandos:
  `QUIT` beendet die Verbindung
  `GET` *`datei`* liefert den Inhalt der Datei
  `LIST` *`directory`* liefert eine Liste der Dateien in dem angegebenen Directory (Datei- und Directorynamen enthalten keine Leerzeichen).
  Die Bearbeitung erfolgt in den Funktionen

```
void getFile(FILE *sockp, const char *filename);
```

und

```
void listDir(FILE *sockp, const char *dirname);
```

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 <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <errno.h>
#include <pthread.h>
#include <sys/socket.h>
#include <netinet/in.h>
#include <dirent.h>

/* Funktionsdeklarationen, Makros, globale Variablen */

static void *tstart(void *arg);
static void ftpConn(FILE *sockp);
static void getFile(FILE *sockp, const char *filename);
static void listDir(FILE *sockp, const char *dirname);

/* Funktion main */

{
   /* Variablendefinitionen */
```

A:


```
   /* Argumente prüfen und auswerten */

   /* Synchronisation vorbereiten */

   /* Thread-Pool erzeugen */
```

A:
S:

```
/* Server-Socket oeffnen
   und Verbindungsannahme vorbereiten */
```

O:


```
/* Server-Schleife:
   Verbindungen annehmen und bearbeiten */
```

O:

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

```
/* Hauptschleife der Arbeiter-Threads: Funktion tstart */
static void *tstart(void *arg)
{
```




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

T:


```
/* Auftragsentgegennahme: Funktion ftpConn */
static void ftpConn(FILE *sockp)
{
```


```
} /* Ende Funktion ftpConn */
```

C:

```
/* Kommandobearbeitung: Funktion getFile */
static void getFile(FILE *sockp, const char *filename)
{
```

```
} /* Ende Funktion getFile */
```

G:


```
/* Kommandobearbeitung: Funktion listDir */
static void listDir(FILE *sockp, const char *dirname)
{
```

```
} /* Ende Funktion listDir */
```

D:

Aufgabe 3: (20 Punkte)

Zur Verwaltung von freiem Speicher (z. B. in Funktionen wie malloc und free) gibt es verschiedene Strategien bei der Speicherzuteilung.

a) Nehmen Sie einen Speicher von 1024 Byte an, initial ist ein Datenblock der Größe 130 Byte an Adresse 0 vergeben. Ein Programm führt die im folgenden Bild angegeben malloc-Aufrufe aus.
Geben Sie für das das *worst-fit*- und für das *Buddy*-Verfahren an, welches Ergebnis diese Aufrufe jeweils zurückliefern, und skizzieren Sie in der Grafik, wie der Speicher danach aussieht (kennzeichnen Sie die vergebenen Speicherbereiche jeweils mit der Nummer der malloc-Aufrufe). Auch der initial vergebene Block wird mit dem jeweilgen Verfahren verwaltet und ist in der Skizze zu berücksichtigen.

worst-fit **buddy**

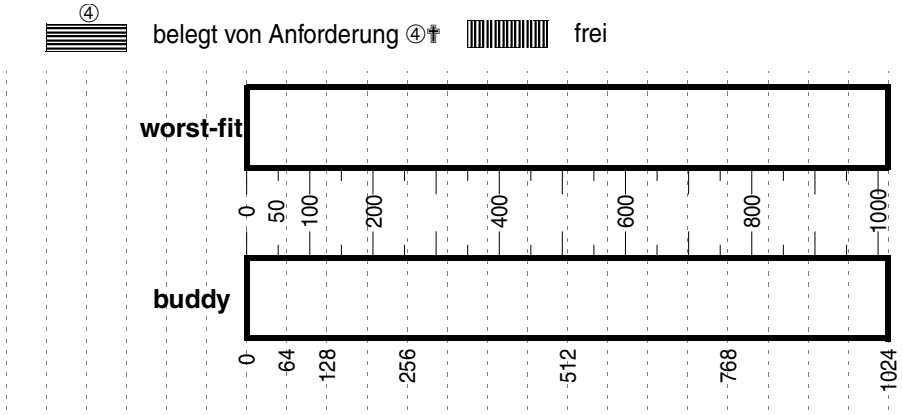
① `malloc(70);`

② `malloc(200);`

③ `malloc(50);`

④ `malloc(300);`

⑤ `malloc(50);`



b) nun werden der Reihe nach in den Schritten A, B, C und D die Bereiche
A: ①, B: ②, C: ③, D: ⑤
freigegeben.
Geben Sie den initialen Aufbau der Freispeicherstrukturen (nach malloc ⑤) und jeweils nach jedem Schritt an.

worst-fit

init.

A

B

C

D

Buddy

init.	A	B	C	D

Aufgabe 4: (10 Punkte)

Skizzieren Sie (graphisch) die Abbildung von einer logischen Adresse in eine physikalische Adresse in einem seitennummerierten Adressraum mit einer Seitengröße von **1024 Byte**.

Tragen Sie konkret die Abbildung für die logische Adresse 0x1A67 ein, wobei die Basisadresse des Seitenrahmens von Seite Nr. 1 0x01502800 und von Seite Nr. 6 0x01504800 sein soll.