Aufgabe 1: (20 Punkte)

Bei den Multiple-Choice-Fragen ist jeweils nur **eine** richtige Antwort anzukreuzen. Falsche Beantwortung führt bei der einzelnen Frage zu Null Punkten. Lesen Sie die Frage genau, bevor Sie antworten.

Wollen Sie eine Multiple-Choice-Antwort korrigieren, kreisen sie bitte die falsche Antwort ein und kreuzen die richtige an. Eine Frage, bei der nicht eindeutig die eine richtige Lösung angekreuzt ist, kann nicht gewertet werden.

a) Welche Antwort stellt kein Attribut einer Datei eines UNIX-UFS-Dateisystems dar? (1 Punkt)

- ❏ Dateilänge
- ❏ Anzahl der symbolischen Links
- ❏ Zugriffsrechte
- ❏ Eigentümer

b) Welche Angaben enthält die Implementierung eines Verzeichniseintrags in einem Standard-UNIX-Dateisystem (z.B. UFS, EXT2)? (2 Punkte)

- ❏ Blocknummer des Inode-Plattenblocks und Dateiname
- ❏ Inode-Nummer und Dateiname
- ❏ Dateiname, Blocknummer des ersten Datenblocks, Dateigröße, Eigentümer und Zugriffsrechte
- ❏ nur Inode-Nummer

c) Welche der Aussagen bzgl. eines logischen Adressraums, der auf dem Prinzip der Segmentierung aufgebaut wurde, ist falsch? (3 Punkte)

- ❏ Segmentierung unterstützt die Ein- und Auslagerung von Segmenten auf eine Festplatte, da ein Segment nicht an den selben Ort eingelagert werden muss, an dem es vor einer Auslagerung platziert war.
- ❏ Über gemeinsame Segmente können zwei logische Adressräume auf die selben Speicherzellen zugreifen.
- ❏ Die Segmentierung erlaubt bei der Abbildung eines logischen Adressraums keinen Zugriff auf Speicherzellen, die auch Bestandteil von anderen logischen Adressräumen sind (Zugriffschutz).
- ❏ Segmente können verschiedene Länge haben. Eine Längenbegrenzung wird üblicherweise bei der Speicherabbildung geprüft.

d) Welche Aussage ist in einem Monoprozessor-Betriebssystem falsch? (3 Punkte)

- ❏ Ein Prozess im Zustand *blockiert* muss warten, bis der laufende Prozess den Prozessor abgibt und kann dann in den Zustand *laufend* überführt werden.
- ❏ Es befindet sich maximal ein Prozess im Zustand *laufend* und damit in Ausführung auf dem Prozessor.
- ❏ Ist zu einem Zeitpunkt kein Prozess im Zustand *laufend*, so ist auch kein Prozess im Zustand *bereit*.
- ❏ In den Zustand *blockiert* gelangen Prozesse in der Regel nur, wenn sie vorher den Zustand *laufend* inne hatten.

e) Sie müssen einen kritischen Abschnitt implementieren. Welches der aufgeführten Koordinierungsmittel ist dazu am Besten geeignet? (1 Punkt)

- ❏ binärer Semaphor
- ❏ Algorithmus von Peterson
- ❏ Up-Down-Systeme
- ❏ PV-Chunk-Semaphor

f) Welche Aussagen treffen für Dateisystemimplementierungen zu? Finden Sie die richtige Aussagenkombination. (3 Punkte)

A) Journalled-Dateisysteme benötigen keine zusätzliche Operation beim Hochfahren des Systems, da ihre Daten immer konsistent sind.

B) Das Prüfen der Konsistenz eines Dateisystems beim Hochfahren kann unterbleiben, wenn das System vor dem Herunterfahren die Konsistenz erkannt und im Dateisystem markiert hat. Vor der Veränderung eines so markierten Dateisystems muss die Marke entfernt werden.

C) Die Prüfoperation beim Hochfahren eines UNIX-Systems kann auf einem typischen UNIX-Dateisystem (z.B. UFS) inkonsistene Zustände erkennen und beseitigen. Dabei gehen unter Umständen Daten verloren.

- ❏ Die Aussagen B und C sind richtig.
- ❏ Alle Aussagen sind richtig.
- ❏ Die Aussagen A und C sind richtig.
- ❏ Nur die Aussage C ist richtig.

g) Bei einer prioritätengesteuerten Prozess-Auswahlstrategie (Scheduling-Strategie) kann es zu Problemen kommen. Welches der folgenden Probleme kann auftreten? (2 Punkte)

- ❏ Die Anzahl der Prioritäten reicht nicht aus, wenn nur wenige Prozesse vorhanden sind.
- ❏ Prozesse können ausgehungert werden.
- ❏ Das Phänomen der Prioritätsumkehr hungert niedrig-priore Prozesse aus.
- ❏ Die Auswahlstrategie arbeitet ineffizient, wenn viele Prozesse im Zustand bereit sind.

h) Was muss ein(e) Software-Entwickler(in) unbedingt beachten, damit seine/ihre Programme nicht Opfer eines Hacker-Angriffs werden? (2 Punkte)

- ❏ Der Quellcode darf nicht herausgegeben werden, damit Schwachstellen nicht entdeckt werden können.
- ❏ Bei Verwendung der Programmiersprache C muss darauf geachtet werden, dass die Stringoperationen strcpy() und strcat() nur eingesetzt werden, wenn die Länge des zu übertragenden Strings noch in das Ziel-Array passt.
- ❏ Es dürfen keine vorgefertigten Bibliotheksfunktionen verwendet werden, weil deren Implementierung als nicht vertrauenswürdig eingestuft werden muss.
- ❏ Der Einsatz der Bibliotheksfunktion system() muss verboten werden, da diese Funktion nicht sicher ausgeführt werden kann.

i) Welcher UNIX-Systemaufruf wird bei der Verwendung von Sockets auf keinen Fall gebraucht? (1 Punkt)

- ❏ close()
- ❏ open()
- ❏ accept()
- ❏ listen()

j) Was ist eine Capability? (2 Punkte)

- ❏ Eine bestimmte Fähigkeit des Anwendungsprogramms
- ❏ Ein Zugriffsrecht, das bei dem zugegriffenen Objekt gespeichert wird.
- ❏ Eine Referenz auf ein Objekt. Dabei trägt die Referenz die Zugriffsrechte auf das Objekt mit sich, die der Besitzer der Referenz ausüben darf.
- ❏ Ein Leistungsparameter der Sicherheitskomponente des Betriebssystems.

Aufgabe 2: (40 Punkte)

Schreiben Sie ein Server-Programm **fileserver**, welches über eine TCP-Netzwerkverbindung Informationen über Dateien bereitstellt. Der Client schickt eine durch Newline abgeschlossenen Zeile Text, welche einen Dateinamen enthält. Der Server schickt daraufhin den Dateinamen und die Dateigröße zurück und wartet auf die nächste Zeile vom Client. Die Verbindung zum Client soll beendet werden, wenn der Client eine Zeile bestehend aus dem Wort "exit" schickt. Jede Client-Verbindung soll vom Server in einem eigenen Prozeß bearbeitet werden.

Der Server soll folgendermaßen gestartet werden: **fileserver *Portnummer***
und arbeitet dann wie folgt beschrieben:

- Aufrufparameter auswerten, bei falschem Aufruf Fehlermeldung und terminierung
- Installation eines SIGCHLD Signalhandlers
- Erzeugen und Initialisieren eines Sockets und Warten auf Verbindungen
- Erzeugen eines neuen Prozesses zur Bearbeitung der Socket-Verbindung
- Eintragen dieses Prozesses in eine Jobliste
- Bearbeiten der Verbindung in der Funktion **`verbindung_bearbeiten:`**
  - Eine durch Newline abgeschlossene Zeile wird vom Socket gelesen.
  - Falls diese Zeile aus dem Wort "exit" besteht, soll die Funktion beendet werden.
  - Anderenfalls wird die Zeile als Dateiname interpretiert und die Größe der entsprechenden Datei ermittelt.
  - Eine Zeile Text bestehend aus Dateiname und Dateigröße wird an den Client geschickt.
- Wenn die Funktion **`verbindung_bearbeiten`** beendet ist, soll der Kindprozess terminieren.

Die vom Server gestarteten Kindprozesse melden ihr Terminieren durch das Signal SIGCHLD. Der Signalhandler für SIGCHLD soll die terminierten Kindprozesse aus der Jobliste austragen. Wenn während der Verbindungsbearbeitung Fehler auftreten, soll dies durch den Exit-Status des Kindprozesses an den fileserver gemeldet werden, welcher dann eine geeignete Meldung (im Signal-Handler) ausgeben soll.

Auf den folgenden Seiten finden Sie ein Gerüst für das beschriebene Programm. In den Kommentaren sind die Aufgaben der einzelnen, zu ergänzenden Programmteile beschrieben.

Alle Programmteile, an denen Ergänzungen vorgenommen werden müssen, sind wie dieser Absatz am Seitenrand markiert. Ergänzungen sind eventuell auch innerhalb vorgegebener Zeilen nötig.

```
/* Diese Funktionen bieten eine Joblisten-Verwaltung.
 * Die Funktionen entsprechen den Funktionen der
 * Übungsaufgabe 5 (jsh) und sind als gegeben vorauszusetzen.
 */


/* Erzeugt eine neue Jobliste.
 * Bei Erfolg wird ein Zeiger auf die Jobliste zurückgegeben,
 * im Fehlerfall NULL.
 */
jl_t jl_new(void);


/* Löscht eine Jobliste. Ein Zeiger auf die Jobliste wird als
 * Parameter jobs übergeben.
 * Im Erfolgsfall wird 0 zurückgegeben, sonst -1.
 */
int jl_delete(jl_t jobs);


/* Setzt die Suchposition auf den Anfang der Joblist.
 * Im Erfolgsfall wird 0 zurückgegeben, sonst -1.
 */
int jl_rewind(jl_t jobs);


/* Schreibt den pid-Wert der aktuellen Suchposition in die
 * Speicherstelle, auf die der Zeiger pid zeigt und setzt
 * die Suchposition auf das nächste Element der Liste.
 * Im Erfolgsfall wird 0 zurückgegeben, sonst -1.
 */
int jl_next(jl_t jobs, int *pid);


/* Fügt ein Element pid in die Jobliste jobs ein.
 * Im Erfolgsfall wird 0 zurückgegeben, sonst -1.
 */
int jl_insert(jl_t jobs, int pid);


/* Entfernt das aktuelle Element aus der Jobliste.
 * Das aktuelle Element ist das Element, für welches das
 * zuletzt aufgerufene jl_next Informationen geliefert hat.
 * Im Erfolgsfall wird 0 zurückgegeben, sonst -1.
 */
int jl_remove(jl_t jobs);
```



```
/*
 * Funktion zum Entfernen von Carriage-Return und/oder Newline
 * am Ende des eines Strings. Diese Funktion wird ebenfall
 * gegeben und braucht nicht programmiert werden.
 */
void trim(char *line);

/*
 * Deklaration von Funktionen, die bei der Programmierung von
 * main benötigt werden und die im Rahmen dieser
 * Aufgabe (zusätzlich zu main) zu programmieren sind
 */

/* Funktion, welche die Anforderung eines Clients bearbeitet.
 * Der Filedeskriptor fd bezeichnet die Verbindung zum Client.
 * Im Erfolgsfall wird 0 zurückgegeben, sonst -1.
 */
int verbindung_bearbeiten(int client_sock_fd);

/* Signalhandler */
void sigchld_handler();

/* Funktion, die den Signalhandler installiert */
void install_signalhandler();

/* Blockieren und Deblockieren des Signals SIGCHLD */
void sigblock();
void sigunblock();

/*
 * Globale Variablen
 */
jl_t jobs;

/* Includes */
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <sys/wait.h>
#include <errno.h>
#include <unistd.h>
#include <string.h>
#include <sys/stat.h>
#include <fctnl.h>
#include <signal.h>
#include <netinet/in.h>
```

```
/* Funktion main
 */
int main(int argc, char *argv[]) {
      /* Variablendefinitionen */
      int port;

      struct sockaddr_in sin;

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


      /* Auswerten der Aufrufparameter */
      if (argc != ..... ) {
          fprintf(stderr, "Falsche Anzahl an Parametern.\n");
          exit(EXIT_FAILURE);
      }
      port = atoi(argv[.....]); /* wandle String in int um */

      jobs = jl_new();

      /* Signalhandler für SIGCHLD installieren */
      install_signalhandler();

      /*
       * Socket erzeugen und initialisieren
       */

      ..........................................................................
      ..........................................................................
      ..........................................................................
      ..........................................................................
      ..........................................................................
      ..........................................................................
      sin.sin_family = ..........................................................
      sin.sin_addr.s_addr = .....................................................
      sin.sin_port = ............................................................
      ..........................................................................
      ..........................................................................
      ..........................................................................
      ..........................................................................
      ..........................................................................
```



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

      while (1) {
        /*
         * Neue Verbindung akzeptieren.
         */

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

```
        /* Neuen Prozeß erzeugen und in Jobliste eintragen.
         * Verbindung in diesem Prozeß bearbeiten (durch
         * Aufruf von verbindung_bearbeiten() )
         * Fehler beim Bearbeiten der Verbindung soll der
         * Kindprozeß durch seinen Exit-Status anzeigen.
         */

        } /* gehört zu while(1) */
} /* gehört zu main() */
```



```
/* Funktion, welche die Anforderung eines Clients bearbeitet.
 * Der Filedeskriptor fd bezeichnet die Verbindung zum Client.
 * Im Erfolgsfall wird 0 zurückgegeben, sonst -1.
 */
int verbindung_bearbeiten(int fd) {

}
```

```
/*
 * Signalhandler für SIGCHLD installieren.
 */
void install_signalhandler() {

}
/*
 * Signalhandler für SIGCHLD.
 */
void sigchld_handler(int sig) {

}
```



```
/*
 * Signal SIGCHLD blockieren.
 */
void sigblock() {

}

/*
 * Signal SIGCHLD deblockieren.
 */
void sigunblock() {

}
```

Schreiben Sie ein Makefile zum Erzeugen des fileserver-Programms.

Ein Aufruf von `make fileserver` soll das Programm erzeugen, ein Aufruf von `make clean` soll das fileserver-Programm und evtl. erzeugte .o-Dateien entfernen.