Aufgabe 1.1: Einfachauswahl-Fragen (22 Punkte)

Bei den Multiple-Choice-Fragen in dieser Aufgabe ist jeweils nur <u>eine</u> richtige Antwort eindeutig anzukreuzen. Auf die richtige Antwort gibt es die angegebene Punktzahl.

Wollen Sie eine Multiple-Choice-Antwort korrigieren, streichen Sie bitte die falsche Antwort mit drei waagrechten Strichen durch (**) und kreuzen die richtige an.

Lesen Sie die Frage genau, bevor Sie antworten!

a)	Wie	e 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.
		Jedes Programm bekommt zur Ladezeit mehrere Wertepaare aus Basis- und Längenregistern zugeordnet, die die Größe aller Segmente des darin laufenden Prozesses festlegen.
o)		s passiert, wenn Sie in einem C-Programm über einen ungültigen ger versuchen auf Speicher zuzugreifen?
		Das Betriebssystem erkennt die ungültige Adresse bei der Weitergabe des Befehls an die CPU (partielle Interpretation) und leitet eine Ausnahmebehandlung ein.
		Beim Laden des Programms wird die ungültige Adresse erkannt und der Speicherzugriff durch einen Sprung auf eine Abbruchfunktion ersetzt. Diese Funktion beendet das Programm mit der Meldung "Segmentation fault".
		Die MMU erkennt die ungültige Adresse bei der Adressumsetzung und löst einen Trap aus.
		Der Compiler erkennt die problematische Code-Stelle und generiert Code, der zur Laufzeit bei dem Zugriff einen entsprechenden Fehler auslöst.

c)		Prozess wird vom Zustand <i>blockiert</i> in den Zustand <i>bereit</i> übert. Welche Aussage passt zu diesem Vorgang?
		Der Prozess hat auf das Einlesen von Daten von der Festplatte gewartet, die nun verfügbar sind.
		Ein Prozess, der zu einem früheren Zeitpunkt aufgrund von Speichermangel auf den Hintergrundspeicher ausgelagert wurde, ist nun wieder eingelagert und kann weiterlaufen.
		Ein anderer Prozess wurde vom Betriebssystem verdrängt und der erstgenannte Prozess wird nun auf der CPU eingelastet.
		Es ist kein direkter Übergang von blockiert nach bereit möglich.
d)	Wo	durch kann es zu Seitenflattern kommen? 2 Punkte
		Wenn die Zahl der residenten Seiten die Größe des physikalischen Speichers überschreitet.
		Durch Programme, die eine Defragmentierung auf der Platte durchführen.
		Wenn ein Prozess zum Weiterarbeiten immer gerade die Seiten benötigt, die durch das Betriebssystem im Rahmen einer globalen Ersetzungsstrategie gerade erst ausgelagert wurden.
		Wenn zu viele Prozesse im Rahmen der mittelfristigen Einplanung auf den Hintergrundspeicher ausgelagert wurden (swap-out).
e)		n unterscheidet bei Programmunterbrechungen zwischen Traps und errupts. Welche Aussage dazu ist richtig?
		Die Behandlung eines Traps führt immer zur Beendigung des unterbrochenen Programms, da Traps nur durch schwerwiegende Fehler ausgelöst werden.
		Da das Betriebssystem nicht vorhersagen kann, wann ein Benutzerprogramm einen Systemaufruf absetzt, sind Systemaufrufe als Interrupts zu klassifizieren.
		Bei der mehrfachen Ausführung eines unveränderten Programms mit gleicher Eingabe treten Traps immer an den gleichen Stellen auf.
		Da Interrupts in keinem Zusammenhang mit dem unterbrochenen Programm stehen, muss der Prozessorstatus des unterbrochenen Programms während der Behandlung nicht speziell gesichert werden.

Klausur Systemprogrammierung	Februar 201
------------------------------	-------------

i)	We	lche Aussage über Einplanungsverfahren ist richtig?	2 Punkte
		Bei kooperativen Verfahren können Prozesse die CPU nicht monopolisieren.	
		In einem asymmetrischen Multiprozessorsystem ist der Einsatz von as schen Verfahren zur Planung obligatorisch.	ymmetri-
		Im Round-Robin-Verfahren nutzen E/A-intensive Prozesse die ihnen z Zeitscheibe immer voll aus.	zugeteilte
		Probabilistische Verfahren eignen sich besonders für den Einsatz in Echtzeitsystemen, da sie die Einhaltung von Zeitgarantien sicherstelle	
j)		n unterscheidet die Begriffe Programm und Prozess. Welche der folden Aussagen zu diesem Themengebiet ist richtig?	2 Punkte
		Ein Programm kann immer nur von einem Prozess gleichzeitig ausgeführt werden.	
		Das Programm ist der statische Teil (Rechte, Speicher, etc.), der Proaktive Teil (Programmzähler, Register, Stack).	ozess dei
		Wenn ein Programm nur einen aktiven Ablauf enthält, nennt man die zess, enthält das Programm mehrere Abläufe, nennt man diese Thread	
		Ein Prozess ist ein Programm in Ausführung - ein Prozess kann aber seiner Lebenszeit auch mehrere verschiedene Programme ausführen.	während
k)	We	lche der folgenden Aussagen über UNIX-Dateisysteme ist richtig?	2 Punkte
		Auf ein Verzeichnis darf immer nur genau ein hard-link verweisen.	
		<i>Hard-links</i> auf Dateien können nur innerhalb des Dateisystems angelegt werden, in dem auch die Datei selbst liegt.	
		In einem Verzeichnis darf es mehrere Einträge mit dem selben Name falls diese Einträge auf unterschiedliche Dateiköpfe (<i>inodes</i>) verweise	-
		Wenn der letzte <i>symbolic link</i> , der auf eine Datei verweist, gelöscht wauch der zugehörige Dateikopf (<i>inode</i>) gelöscht.	vird, wird

Beim Binden mit einer statischen Bibliothek werden in einem Programm nur

Verweise auf verwendete Symbole der Bibliothek angelegt.

Klausur Systemprogrammierung

b)

Aufgabe 1.2: Mehrfachauswahl-Fragen (8 Punkte)

Bei den Multiple-Choice-Fragen in dieser Aufgabe sind jeweils m Aussagen angegeben, n ($0 \le n \le m$) Aussagen davon sind richtig. Kreuzen Sie <u>alle richtigen</u> Aussagen an. Jede korrekte Antwort in einer Teilaufgabe gibt einen halben Punkt, jede falsche Antwort einen halben Minuspunkt. Eine Teilaufgabe wird minimal mit 0 Punkten gewertet, d. h. falsche Antworten wirken sich nicht auf andere Teilaufgaben aus.

Wollen Sie eine falsch angekreuzte Antwort korrigieren, streichen Sie bitte das Kreuz mit drei waagrechten Strichen durch (***).

Le	esen	Sie die Frage genau, bevor Sie antworten!
a)		lche der folgenden Aussagen zum Thema Seiteneinlagerungs- und tenersetzungsstrategien ist richtig?
	О	Die Ersetzungsstrategie MIN ist in der Praxis nur schwer realisierbar, weil Wissen über das zukünftige Verhalten des Gesamtsystems notwendig ist.
	0	Bei der Verwendung von globalen Seitenersetzungsstrategien sind Seitenfehler vorhersagbar bzw. reproduzierbar.
	О	$\label{thm:presented} \mbox{Mit dem Systemaufruf free()} \ \ kann \ eine \ Speicherseite in \ den \ Freiseitenpufferen \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \ \$
	О	Die Ersetzungsstrategie LRU ersetzt die am längsten nicht mehr referenzierte Seite.
	0	Bei der Verwendung von lokalen Seitenersetzungsstrategien sind Seitenfehler vorhersagbar bzw. reproduzierbar.
	О	Die Ersetzungsstrategie LRU benötigt im Vergleich zu FIFO immer weniger Versuche, bis eine zu ersetzende Seite gefunden werden kann.
	О	Lokale Seitenersetzungsstrategien wählen die zu ersetzende Seite immer aus der Menge aller im System verfügbaren Seitenrahmen aus.
	О	Bei der Ersetzungsstrategie LFU wird die am seltensten referenzierte Seite aus dem Speicher verdrängt.

	Iche der folgenden Aussagen zum Thema Synchronisation sind 4 Punkte ntig?
О	Ein Mutex kann ausschließlich für einseitige Synchronisation verwendet werden.
О	Der Einsatz von nicht-blockierenden Synchronisationsmechanismen kann zu Verklemmungen $(dead\text{-}locks)$ führen.
О	Die V-Operation kann auf einem Semaphor auch von einem Faden aufgerufen werden, der zuvor keine P-Operation auf dem selben Semaphor ausgeführt hat.
О	Ein Anwendungsprozess muss bei der Verwendung von Semaphoren Interrupts sperren, um Probleme durch Nebenläufigkeit zu verhindern.
О	Für nichtblockierende Synchronisation werden spezielle Befehle der Hardware genutzt, die wechselseitigen Ausschluss garantieren.
О	Semaphore können sowohl für einseitige als auch für mehrseitige Synchronisation verwendet werden.
О	Zur Synchronisation eines kritischen Abschnitts ist passives Warten immer besser geeignet als aktives Warten.
O	Gibt ein Faden einen Mutex frei, den er selbst zuvor nicht angefordert hatte, stellt dies einen Programmierfehler dar; der fehlerhafte Prozess sollte dann abgebrochen werden.

Aufgabe 2: timebox (60 Punkte)

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

Schreiben Sie ein Programm timebox, das auf dem TCP/IPv6-Port 2016 (LISTEN_PORT) einen Dienst anbietet, um die Modifikationszeit von auf dem Server abgelegten Dateien abzufragen. Die Dateien liegen im Verzeichnis /proj/timebox (BASEDIR). Die Abarbeitung parallel eintreffender Anfragen soll von einem Arbeiter-Thread-Pool übernommen werden; die Threads werden über einen entsprechend synchronisierten Ringpuffer mit Verbindungen versorgt.

Ein Client sendet pro Verbindung eine Befehlszeile mit einer Maximallänge von 1024 Zeichen (MAX_LINE_LEN) an den Server. Der angebotene Dienst unterstützt genau einen Befehl:

 TIME relPath liefert das Änderungsdatum (modification date) der Datei am übergebenen Pfad (relativ zum Verzeichnis BASEDIR) an den Client zurück.

Der Server soll eine dynamische Aktualisierung der Anzahl aktiver Arbeiter-Threads erlauben. Dazu wird eine Konfigurationsdatei (CONFIG) eingelesen, die als einzigen Inhalt die Anzahl der Threads enthält. Nach einer Änderung signalisiert der Server-Administrator dem Serverprozess durch ein SIGUSR1-Signal, dass die Konfigurationsdatei einen neuen Wert enthält. Spätestens nach der Annahme der nächsten Verbindung soll die Datei neu eingelesen und die Anzahl der laufenden Threads entsprechend angepasst werden.

Alle Threads des Servers sollen außerdem das Signal SIGPIPE ignorieren, damit eine fehlerhafte Verbindung zu einem Client nicht zum Beenden des Servers führt.

Das Programm soll folgendermaßen strukturiert sein:

- Das Hauptprogramm initialisiert zunächst alle benötigten Datenstrukturen und nimmt auf einem Socket Verbindungen an. Wird festgestellt, dass in der Zwischenzeit ein SIGUSR1-Signal verarbeitet wurde, soll die Konfigurationsdatei neu eingelesen werden. Eine erfolgreich angenommene Verbindung soll zur weiteren Verarbeitung in den Ringpuffer eingefügt werden.
- Funktion void parse_config(void): Liest die Zahl der Threads aus der Konfigurationsdatei ein. Anschließend soll die Anzahl der laufenden Threads entsprechend der Differenz zum aktuellen Wert angepasst werden. Zum Beenden von Threads sollen dazu spezielle Werte (DEAD_PILL) in den Ringpuffer gelegt werden. Die Funktion soll auch zum initialen Starten der Arbeiter-Threads verwendet werden. Tritt beim Einlesen ein Fehler auf oder ist der gelesene Wert kleiner oder gleich 0, soll stattdessen der Standardwert (DEFAULT_THREADS) verwendet werden.
- Funktion void* thread_handler(void *): Hauptfunktion der Arbeiter-Threads. Entnimmt dem Ringpuffer eine Verbindung. Falls der Wert DEAD_PILL gelesen wird, soll sich der aktuelle Thread beenden. Ansonsten wird zur weiteren Verarbeitung die Funktion handleCommand aufgerufen.
- Funktion void handleCommand(FILE *rx, FILE *tx): Liest die Anfragezeile vom Client aus und überprüft, ob sie dem unterstützten Kommando entspricht. Ist dies der Fall, wird der Zeitstempel der Datei ausgelesen und an den Client gesendet (Formatstring: %1d). Bei auftretenden Fehlern wird eine kurze Fehlermeldung an den Client gesendet (Beispiel: "Error: unknown command\n").

Zusätzlich sollen Sie die Funktionen bbPut (int value) und bbGet () implementieren. Der Ringpuffer soll dabei statisch als Feld der Länge 16 (BUFFERSIZE) angelegt werden. Zur Koordination stehen Ihnen Semaphore mit den Funktionen semCreate, P und V zur Verfügung. Diese Funktionen müssen Sie nicht selbst implementieren. Die Schnittstellen entsprechen dabei dem Modul, das sie aus den Übungen kennen, und sind auf der folgenden Seite am Anfang des Programms deklariert.

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 Anweisungen 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 gegebenenfalls auch weitere Funktionen benötigen.

```
#include <arpa/inet.h>
#include <errno.h>
#include <pthread.h>
#include <signal.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/stat.h>
#include <svs/tvpes.h>
#include <unistd.h>
SEM *semCreate(int initVal);
void P(SEM *sem);
void V(SEM *sem);
#define BUFFERSIZE 16
#define MAX LINE LEN 1024
static const char BASEDIR[] = "/proj/timebox";
static const char CONFIG[] = "/etc/timebox/config";
static const unsigned int DEFAULT_THREADS = 2;
static const int LISTEN PORT = 2016;
static const int DEAD PILL = 0xdeaddead:
static void die(const char msg[]){
    fprintf(stderr, "Error: %s\n", msg); exit(EXIT_FAILURE);
// Funktionsdeklarationen, globale Variablen usw.
```

Klausur Systemprogrammierung

/ Funktion main()	_
// Socket erstellen und auf Verbindungsannahme vorbereiten	
/ Socket erstellen und auf Verbindungsannahme vorbereiten	

// Verbindungen annehmen und bearbeiten	
// Ende Funktion main	
// Signalbehandlung für SIGUSR1	
// Ende Signalbehandlung	M:

// Funktion parse_config()		// Funktion thread_handler()	
// Ende Funktion parse_config()	P:	// Ende Funktion thread_handler()	Н:
			-

Klausur Systemprogrammierung

 h handleCommand()	_
 	L
 	L
 	L
 	_

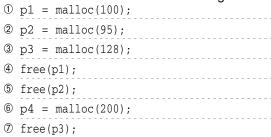
// Funktion bbPut()	
// Ende Funktion bbPut()	
// Funktion bbGet()	
// Ende Funktion bbGet()	

Aufgabe 3: (18 Punkte)

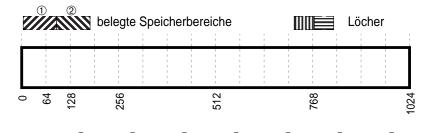
Ein in der Praxis häufig eingesetztes Verfahren zur Verwaltung von freiem Speicher ist das *Buddy*-Verfahren.

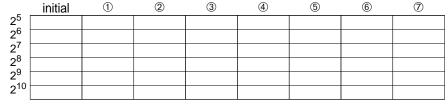
Nehmen Sie einen Speicher von 1024 Byte an und gehen Sie davon aus, dass die Freispeicher-Verwaltungsstrukturen separat liegen. Initial ist bereits ein Datenblock der Größe 50 Byte vergeben worden. Ein Programm führt nacheinander die im folgenden Bild angegebenen Anweisungen aus.

Ergebnis



a) Tragen Sie hinter den obigen Anweisungen jeweils ein, welches Ergebnis die malloc()-Aufrufe zurückliefern. Skizzieren Sie in der folgenden Grafik, wie der Speicher nach Schritt ® (dem vorletzten Schritt!) aussieht, und tragen Sie in der Tabelle den aktuellen Zustand der Lochliste nach jedem Schritt ein. Für Löcher gleicher Größe schreiben Sie die Adressen einfach nebeneinander in die Tabellenzeile (es ist nicht notwendig, verkettete Buddys wie in der Vorlesung beschrieben einzutragen). (13 Punkte)





In einem 8. Schritt erfolgt schließlich noch die Speicheranforderung ® p5 = malloc(300);
Welche Stellen in den Freispeicher-Verwaltungsstrukturen werden im Rahmen die-
ser Anforderung überprüft, welche Stellen werden verändert und was ist das Ergeb-
nis des Aufrufs? (3 Punkte)
Erläutern Sie den Unterschied zwischen externer und interner Fragmentierung.
Erläutern Sie den Unterschied zwischen externer und interner Fragmentierung. (2 Punkte)
(2 Punkte)
(2 Punkte)

Aufgabe 4: (12 Punkte)

Bei der Einplanung von Prozessen werden Gütemerkmale bzw. Kriterien zur Aufstellung der konkreten Einlastungsreihenfolge von Prozessen unterschieden, die von den verschiedenen Einplanungsverfahren optimiert werden.

Februar 2016

a)	Nennen Sie die zwei übergreifenden Kategorien, in die diese Kriterien generell ein geteilt werden, und beschreiben Sie kurz deren jeweiligen Fokus. (2 Punkte).
))	Nennen Sie nun mindestens je 2 Vertreter beider Kategorien und beschreiben Sie welches Verhalten durch diese Kriterien jeweils erreicht werden soll. (8 Punkte)

ge	'n	? '	W	el	cl	ne	e]	K	or	ıfl	li	kt	e	k	Ö	n	ne	er	1 (da	ab	e	i	n	ii	t a	ar	ıd	le	re	en	ŀ	ζ1	it	e	ri	eı	1 8	aı	ıf	tr	et	te	n	?	(2	2	P	uı	nk	cto	e)
		-										-					-			-			-			-		-			-		-		-			-		-	-		-					-			-	-
		-										-					-			-			-			-		-			-				-			-		-	-		-					-			-	-
		-										-					-			-			-			-					-							-		-	-		-					-			-	-
		-										_					_			-			-			_					_							_		-	_		-					-			_	-
		_																		-			-			_					_							_			_		_			_		_			_	-
																				_			_								_							_			_		_			_		_				
		-			-							-					-			-			-			-		-			-		-		-			-		-	-		-					-			-	-