Wintersemester 2018/2019

Name, Matrikelnummer

Prüfer:	Prof. Dr.-Ing. Rainer Keller	**Anzahl der Seiten:**	10
Studiengänge:	Softwaretechnik und Medieninformatik Technische Informatik Ingenieurpädagogik	**Semester:**	SWB2 TIB2 IEP2
Klausur:	**Betriebssysteme**	**Prüfungsnummern:**	IT 105 2004
Hilfsmittel:	**keine**, außer **1 DIN A4** Blatt, beidseitig **von Hand selbst** beschrieben	**Dauer der Klausur:**	90 Minuten

Bitte lesen Sie die Aufgaben sorgfältig durch. Jede Aufgabe besteht aus Unteraufgaben – für die es Teilpunkte gibt. Jeder Punkt entspricht ca. 1 Minute Arbeitszeit. Nutzen Sie also den zur Verfügung stehenden Raum und die Zeit aus, um möglichst sorgfältig und ausführlich zu antworten.

Allgemeines (5 Punkte)

a) Aus welchen Komponenten bestanden Rechner der 1. Generation (z.B. Zuse Z1)

b) Circa wie groß ist der Linux Kernel in Lines-of-Code und in welcher Programmiersprache ist der geschrieben?

c) Nennen Sie vier komplett unterschiedliche Betriebssysteme:

1. Betriebssystem: Windows	3. Betriebssystem: FreeRTOS
2. Betriebssystem: Linux, MacOS oder Android etc. (UNIX)	4. Betriebssystem: VxWorks

a) Lochkarten und Lochstreifen, Relais, Mechanische Addierer und Subtrahierer, Lochkartenleser und -stanzer, Speicher und Steuerungseinheit, Elektronen-/Vakuumröhren

b) 27 Millionen, in C geschrieben

Weitere Beispiele zu c:
• Heutige Real-time Operating Systems (RTOS):
QNX
Zephyr
ThreadX
TizenRT
StateOS FreeRTOS
pSOS
Xenomai
RIOT
RTEMS
RTX
RTAI
SCIOPTA
RTOS
Nucleus
Neutrino

Name, Matrikelnummer

Bash Shell (20 Punkte)

a) Benennen Sie für die jeweilige Shell-Funktion den passenden Shell-Befehl:

Systemcalls anzeigen:	strace
Netzwerk konfigurieren:	ifconfig oder ip
Link erstellen:	ln
Dateisystem erstellen:	mkfs
Zeile ausgeben:	echo
Archive ver-/auspacken:	tar
Prozess in den Hintergrund:	&
Signal an Prozess senden:	kill

b) Die Datei `gehaltsliste.txt` enthält die Namen, Stundenlohn und Arbeitsstunden durch Doppelpunkte getrennt (CSV) von Mitarbeitern einer großen Abteilung. Schreiben Sie ein Shell-Script, welches die Anzahl der Mitarbeiter mit 3-stelligem Lohn zählt, **sowie** den Mittelwert der Gehälter berechnet.

c) Was sind systemnahe Programme? Bitte geben Sie Beispiele an.

d) Was bedeuten die beiden Umgebungsvariablen PATH und LD_LIBRARY_PATH?

PATH	PATH enthält eine Liste an Verzeichnissen, in denen das Betriebssystem nach ausführbaren Dateien sucht. Wenn ein Befehl im Terminal eingegeben wird, wird dieser im PATH gespeicherten Verzeichnis ausgeführt.
LD_LIBRARY_PATH	LD_LIBARY_PATH gibt an, in welchen Verzeichnissen das Betriebssystem nach gemeinsam genutzten Bibliotheken, die Programme zur Laufzeit benötigen. Wenn ein Programm eine Bibliothek benötigt sucht es hier nach dieser.

e) Schreiben Sie die passenden Umgebungsvariablen hin:

Vorheriger Pfad:	$OLDPWD
Prompt String:	$PS1

Name, Matrikelnummer

Verwendete Shell	$SHELL
Pfad der Doku:	$MANPATH
Erster Parameter einer Funktion	
PID der Bash	

b)

```
#!/bin/bash

# Dateipfad zur Gehaltsliste
gehaltsliste="gehaltsliste.txt"

# Zähler für Mitarbeiter mit 3-stelligem Lohn
anzahl_3stellig=0

# Variable für die Summe der Gehälter
gesamtgehalt=0

# Variable für die Anzahl der Mitarbeiter
anzahl_mitarbeiter=0

# Schleife zum Lesen der Gehaltsliste
while IFS=":" read -r name stundenlohn arbeitsstunden
do
    # Prüfen, ob der Lohn 3-stellig ist
    if (( stundenlohn >= 100 && stundenlohn <= 999 ))
    then
        (( anzahl_3stellig++ ))
    fi

    # Berechnung des Gehalts
    gehalt=$(( stundenlohn * arbeitsstunden ))

    # Summierung des Gehalts
    (( gesamtgehalt += gehalt ))

    # Inkrementieren der Mitarbeiteranzahl
    (( anzahl_mitarbeiter++ ))
done < "$gehaltsliste"

# Berechnung des Mittelwerts
mittelwert=$(( gesamtgehalt / anzahl_mitarbeiter ))

# Ausgabe der Ergebnisse
echo "Anzahl der Mitarbeiter mit 3-stelligem Lohn: $anzahl_3stellig"
echo "Mittelwert der Gehälter: $mittelwert"
```

c)

Systemnahe Programme sind Softwareanwendungen, die eng mit dem Betriebssystem interagieren und auf einem niedrigeren Level arbeiten. Sie haben direkten Zugriff auf die Ressourcen des Computers und nutzen die Funktionen und Dienste des Betriebssystems, um bestimmte Aufgaben zu erfüllen.

z.B. Treiber, Bootloader und Kernel

[Diese Programme sind normalerweise in einer Sprache geschrieben, die eine direkte Systemsteuerung ermöglicht, wie z.B. C oder Assembler.]

Name, Matrikelnummer

Scheduling und Systemcalls (19 Punkte)

a) Wie lange dauert ein Systemaufruf circa und wieso war `getpid()` so schnell?

Wie lange?
Was war mit getpid()? getpid() fragt die aktuelle process id ab. Diese ist normalweise intern in der Kerneldatenstruktur mit dem laufenden Prozess gespeichert und kann deshalb sehr schnell abgefragt werden.

b) Welche beiden (temporären) Informationen verarbeitet ein CPU-Scheduler, um die Priorität eines Tasks neu zu berechnen? Wieso priorisiert er bestimmte Tasks höher und welche Tasks sind das?

c) Was passiert, wenn ein Task sein Zeitkontingent aufgebraucht hat und was passiert, wenn kein Task mehr Zeitkontingent übrig hat?

b) Execution Time and Priority
Echtzeittasks und interaktive Anwendungen werden beispielsweise höher priorisiert. Hier ist die Geschwindigkeit direkt für den Benutzer ersichtbar und längere Wartezeiten würden die Nutzbarkeit für den Menschen einschränken.

Hier scheint es noch ein gr0ßes Spektrum an Antwortmöglichkeiten zu geben

c) Wenn ein Task sein Zeitkontingent aufgebraucht hat wird er aus der CPU entfernt.
Wenn alle Task kein Zeitkontingent mehr haben, sucht der Scheduler nach neuen Tasks. Findet er keinen geht er in einen Ready-State um auf ein Ereignis für einen neuen Tasks zu warten.

Name, Matrikelnummer

d) Beurteilen Sie jede Aussage ob sie Wahr (W) oder Falsch (F) ist:

Aussage	**W/F?**
1:1 Modell heißt ein User-Thread ist ein Linux-Task	F
Das 1:1 Modell reduziert Komplexität ist aber langsam	F
Der Linux Kernel implementiert den Desolate Fair Scheduler (DFS)	F
Der Linux Scheduler läuft bei jedem Interrupt	F
Zur besseren Lastbalance werden Tasks auf freie Cores verschoben	W
Wenn ein Task blockiert, läuft der Scheduler	W
Das Zeitkontingent für alle Threads eines Tasks ist `vruntime`	F
Die `vruntime` ist in Millisekunden (ms) aufgeteilt	F
In `struct rq{}` stehen die lauffähigen und gestoppten Tasks	W
Die Migration eines Tasks von einem Core auf einen anderen ist teuer	W
Der Linux Scheduler läuft effizient in O(n)	W
Die CPU-Zeit wird an die `nr_running` Tasks verteilt	W

e) Welche Ressourcen teilen sich alle Threads eines Prozesses und welche sind pro Thread?

Geteilte Ressourcen? Speciherbereich, Dateidiskriptoren, Signale und Umgebungsvariablen
Per-Thread Ressourcen? Register, Stack und Ausführungszustand

Name, Matrikelnummer

Virtueller Speicher (17 Punkte)

a) Zeichnen Sie die Indexe und Pointer ein für die folgenden 64-Bit Virtuellen Adressen – bitte **beachten** Sie die binären Zahlenwerte (0...001 bedeutet eine 1 im niederwertigsten Bit, ansonsten Nullen). Geben Sie weiterhin die Anzahl Bits an den **unterstrichenen** Stellen ein.

   Sollten Sie Offset-Längen korrigieren wollen, nutzen Sie bitte das Feld unten.

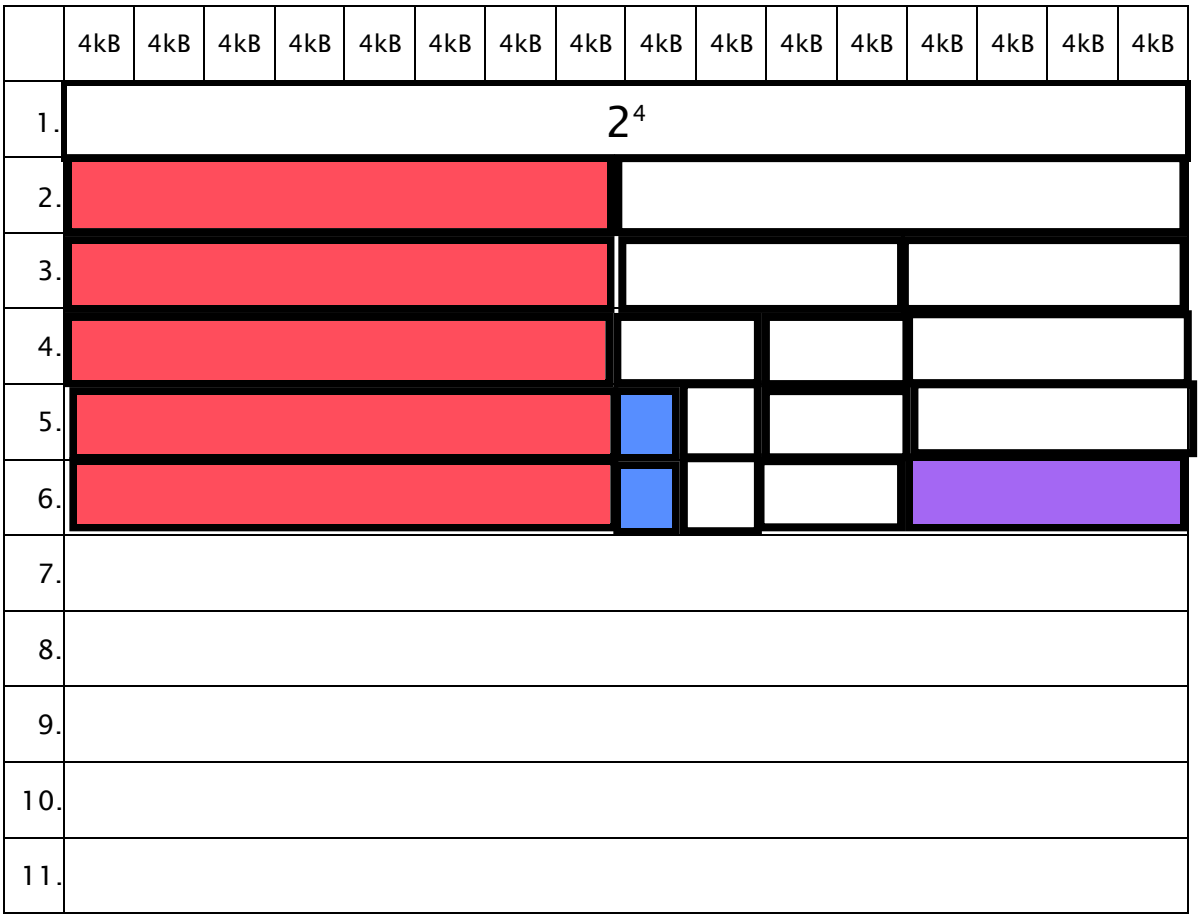
   Keinen Plan was er will

Name, Matrikelnummer

b) Der Buddy-Allokator erlaubt, sehr schnell freie Speicherbereiche zu identifizieren. Die untenstehende Ansicht entspricht der Darstellung von Wikipedia. Zuerst ist der Speicher komplett frei. Zeichnen Sie die folgenden Allokationen (mit Unterschritten) ein:

1. Programm A alloziiert 17 kB Speicher
2. Programm B alloziiert 3 kB Speicher
3. Programm A alloziiert 13 kB Speicher

Hoffe mal das stimmt so



c) Wie viele Bits bietet der Intel Prozessor für Schutzebenen, wie viele Ebenen erlaubt dies und wie viele nutzt Linux?

1.	Wie viele Bits?	4
2.	Wie viele Ebenen?	16
3.	Linux nutzt?	2

Name, Matrikelnummer

Linux Kernel Module (10 Punkte)

a) Programmieren Sie ein minimales Kernel-Modul, welches beim Laden prüft ob der ladende Task die FPU nutzt und dies ausdruckt. Die relevante Datenstruktur `task_struct` sowie die `PF_*` Flags (beide aus `include/linux/sched.h`) sind:

```
struct task_struct { ...
    volatile long state; // -1 unrunnable, 0 runnable, >0 stopped
    unsigned int flags; // Defined by PF_* below
… }
#define PF_IDLE      0x00000002
#define PF_USED_MATH 0x00002000 // If set the fpu was used
```

b) Ihr Modul soll den Parameter „alloc_size“ als Integer beim Laden übernehmen können. Schreiben Sie hierzu den C Code:

c) Bitte geben Sie den Befehl an, um ihr Modul `modfpu.ko` zu laden:

```
MODULE_LICENSE("GPL");
MODULE_DESCRIPTION("Kernel-Modul zur Überprüfung der FPU-Nutzung");

#include <linux/module.h>
#include <linux/init.h>
#include <linux/moduleparam.h>
#include <linux/sched.h>

static int alloc_size = 0;

module_param(alloc_size, int, 0);
MODULE_PARM_DESC(alloc_size, "Size of allocation");

static int __init check_fpu_init(void)
{
   struct task_struct *current_task = current;

   if (current_task->flags & PF_USED_MATH) {
      printk(KERN_INFO "Der ladende Task nutzt die FPU.\n");
   } else {
      printk(KERN_INFO "Der ladende Task nutzt nicht die FPU.\n");
   }

   printk(KERN_INFO "alloc_size: %d\n", alloc_size);

   return 0;
}

static void __exit check_fpu_exit(void)
{
   printk(KERN_INFO "Kernel-Modul wird entladen.\n");
}

module_init(check_fpu_init);
module_exit(check_fpu_exit);
```

```
sudo insmod modfpu.ko alloc_size=1234
```

Name, Matrikelnummer

Interprozesskommunikation (8 Punkte)

a) Bitte geben Sie min. einen Unix-Funktionsaufruf je Kommunikationsmodell an:

Uni-direktionaler Daten-transfer via Kernel	write()
Gemeinsamer Speicher	mmap()
Asynchrone Benach-richtigung eines Events	eventfd()
Direkt in den Speicher eines Prozesses schreiben	ptrace()

b) Nennen Sie Vor- und Nachteile Daten zwischen Prozessen mittels Dateien auszutauschen:

Vorteile: -Kann einfach implementiert werden -Daten bleiben gespeichert und können nach einem Neustart weiter verwendet werden -Verschiedene Prozesse und Systeme sind dadurch kompatibel
Nachteile: -Es wird zusätzlicher Overhead erzeugt, der besonders bei gr0ßen Datenmengen zu geringer Effizienz führen kann, da auf die Festplatte oder einen anderen dauerhaften Datenträger zugegriffen werden muss -Zeitverzögerung durch viele Schreib- und Leseoperationen -Synchronisationsprobleme, wenn mehrere Prozesse auf den gleiche Speicher zugreifen -Sicherheitsrisiken beim Austausch von Daten (unautorisierte Zugriffe und Manipulation der Daten)

Name, Matrikelnummer

Virtualisierung, Echtzeit & andere OS (11 Punkte)

a) Nennen Sie die 4 Klassifikationen für Antwortverhalten:

1.	Real-Time
2.	Interactive
3.	Batch
4.	Time-Sharing

b) Warum ist der Linux Kernel in der Standardkonfiguration nicht echtzeitfähig?

c) Worauf basiert das Betriebssystem MacOS und wie heißt der Kernel?

Basis	XNU
Kernel:	Mach

d) Wie unterscheidet sich Microsoft Windows Server von einem Standard Windows 10?

b) Der Linux-Kernel in Standardkonfiguration ist auf gute Balance zwischen Leistung, Funktionalität und Flexibilität ausgelegt:
-Der Linux-Scheduler zielt darauf ab Ressourcen gerecht zwischen Prozessen zu verteilen
-Interrupts werden nicht deterministisch behandelt und können so zu Unterbrechungen im Ausführungsfluss führen
-Durch das Ranking nach Priorität im Scheduler ist nicht gegeben, dass ein TAsk in einer bestimmten Zeit ausgeführt ist
-Kernel hat durch hohe Komplexität hohe Latenzzeiten