
 Name, Matrikelnummer

Prüfer:	Prof. Dr.-Ing. Rainer Keller	Anzahl der Seiten:	14
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 selbst geschrieben (keine Kopie). Das Hilfsmittel ist abzugeben. Sollte kein Hilfsmittel verwendet werden ist dies in der Klausur anzugeben.		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. Achten Sie auf Schlüsselwörter wie „in Stichworten“, „kurz“, „ausführlich“ „nennen“, „erklären“ oder „im Detail“. Der Umfang Ihrer Antwort soll sich danach richten.

1. Allgemeines (8 Punkte)

- a) Nennen Sie die allgemeine Definition eines Betriebssystems der Vorlesung?

Ein Betriebssystem ist eine Software, die auf effiziente Weise die Komplexität eines Computers vor dem Benutzer und dem Programmierer versteckt und einer Gruppe von Benutzern und Programmen gemeinsamen, sicheren Zugriff auf Rechen-, Speicher-, Kommunikationsmittel zur Verfügung stellt (01, S.16)

2

- b) Nennen Sie 3 unterschiedliche (Klassen von) Systemen auf denen Linux läuft

1.	Mobilgeräte (zB Smartphones)	
2.	Server	
3.	Personal Computer	

2

4.Embedded Systems (zB Rasperri (01, S.19/26)

Name

c) Nennen Sie 2 unterschiedliche (Klassen von) Systemen auf denen Windows läuft?

1.	Personal Computer
2.	Server (01, S.19)

1

d) Warum ist der Linux Kernel (hauptsächlich) in der Programmiersprache C programmiert?

Programmiersprache C wird hauptsächlich verwendet weil, Sie eine effiziente hardwarenahe Sprache ist und effizienten Zugriff auf die niedrige Systemhardware bietet und direkte Manipulation dieser ermöglicht.

Dabei ist sie auch flexibel und portabel zwischen verschiedenen Hardwareplattformen was die Entwicklung eines Betriebssystems erlaubt, das auf verschiedenster Hardware läuft.

Außerdem vereint C eine gute Balance zwischen Maschinennähe und Lesbarkeit. (GPT4)

2

e) Sie können die Datei check als User me nicht editieren:

d-wx----- 3 me you 42 Feb 29 11:55 check

Warum nicht - und wie können Sie den Fehler beheben?

Das liegt daran, dass der User (me) nur die Rechte schreiben (w bzw. 2) und ausführen (x bzw. 1) hat, also in Oktal ausgedrückt 3. Um die Datei zu editieren, braucht man auch noch Leserechte, sodass auch der Inhalt gelesen werden kann. Also mit chmod 700 Check oder chmod u+r check zusätzlich Leserechte erteilen. Nun habe ich nicht nur Schreib- und Ausführungsrechte, sondern auch Leserechte, sodass man als User auch editieren kann

1

2. Bash

(16 Punkte)

a) Welche Vorteile haben Script-Sprachen wie Bash, welche Nachteile haben sie?

Vorteile?

- schnelle, einfache Automatisierung von Aufgaben
- benutzerfreundlich, flexibel, ermöglichen Kombination mehrerer Befehle in einem Skript, Zugriff auf viele Linux-Werkzeuge

2

Nachteile?

- im Vergleich zu vollwertigen Programmiersprachen weniger Leistungsfähig, im Bezug auf Geschwindigkeit, Effizienz begrenzt
- anfälliger für Fehler, Sicherheitsrisiken, vor allem bei Handhabung Benutzereingaben

Name _____

b) Was machen die folgenden Bash-Befehle. Bitte stichpunktartig erklären:

du -h	- "disk usage", "human readable" Zeigt Größe von Dateien, Verzeichnissen, Großmengen menschenlesbar format (z.B. MB/Byte)
fsck	- "Filesystem Check" überprüft Integrität von Dateisystemen, Fehler auf Speichermedien korrigieren
dmesg	- "display Message" zeigt Kernel benötigte Logs an, nützlich Debuggen Hardware/Firmwareproblemen zeigt Kernel Logs aus
objdump	- Disassembly von Objekt Dateien
ps	- "Process status", Anzeige der Prozesse die momentan laufen
echo	- angabe String auf Terminal
kill	sendet Prozess, in den kill Signal an diesen gesendet wird
less	"less is more than more", Anzeige Datei auf Terminal

3

c) Geben Sie die passenden Shell Befehle an:

Tastendruck um Programm abzubrechen:	Ctrl + C
Den Prozess 666 freundlicherweise auf Prio 10 setzen:	nice dd -p 666
Freien Plattenplatz menschenlesbar anzeigen lassen:	df -h
Datei f in Verzeichnis d verschieben:	mv f d/
Datei b anfassen (bspw. damit es ins Backup kommt!):	touch b
Filesystem der 1. Partition auf der 2. Platte prüfen:	fsck /dev/sdb1
Dateien in allen Verzeichnissen finden mit Endung .h:	find / -name "*.h"
Symbolischer Link: b (Neu) zeigt auf a (Original):	ln -s a b

4

Name _____

d) Schreiben Sie das Bash Programm count: Es basiert auf dem Output von last:

```
rakeller pts/6 134.108.34.30      Wed Feb 29 11:55 still logged in  
rakeller pts/7 134.108.34.67      Wed Feb 30 12:01 - 12:03 (00:03)
```

Schreiben Sie ein vollständiges Skript, welches die IP-Adresse als erstes Parameter nimmt und die Ausgabe von last nach dieser IP durchsucht und am Ende die IP und die Anzahl der Logins ausgibt. Eine Fehlerprüfung auf Anzahl Parameter ist Teil der Aufgabe – nicht aber Prüfung auf korrekte IPv4/IPv6 Adresse (dies ergibt aber Zusatzpunkte)!)

Die Ausgabe im obigen Fall wäre also:

```
$ ./count 134.108.43.30
```

```
134.108.43.30 1
```

```
#!/bin/bash ← wichtig immer am Anfang von Bash-Skript
```

```
# überprüfen ob genau ein Parameter übergeben wurde
```

```
if ["$#" -ne 1]; then  
    echo "Fehler: genau 1 IP als Parameter erforderlich!"  
    exit 1
```

```
fi
```

```
# IP-Adresse als Parameter
```

```
IP = $1
```

```
# Zähler der Logins, die dieser IP entsprechen
```

```
LOGIN-COUNT = $(last | grep $IP | wc -l) wc -l für word count, -l nur Anzahl der Zeilen  
↑ ↑ ↑ also wc-l gibt Anzahl der Zeilen an, wo IP vorkommt  
Ausgabe von last wird durch grep nach dem übergebenen Parameter $IP  
durchsucht
```

```
# Ausgabe der IP und Anzahl der Logins
```

```
echo "$IP $LOGIN-COUNT"
```

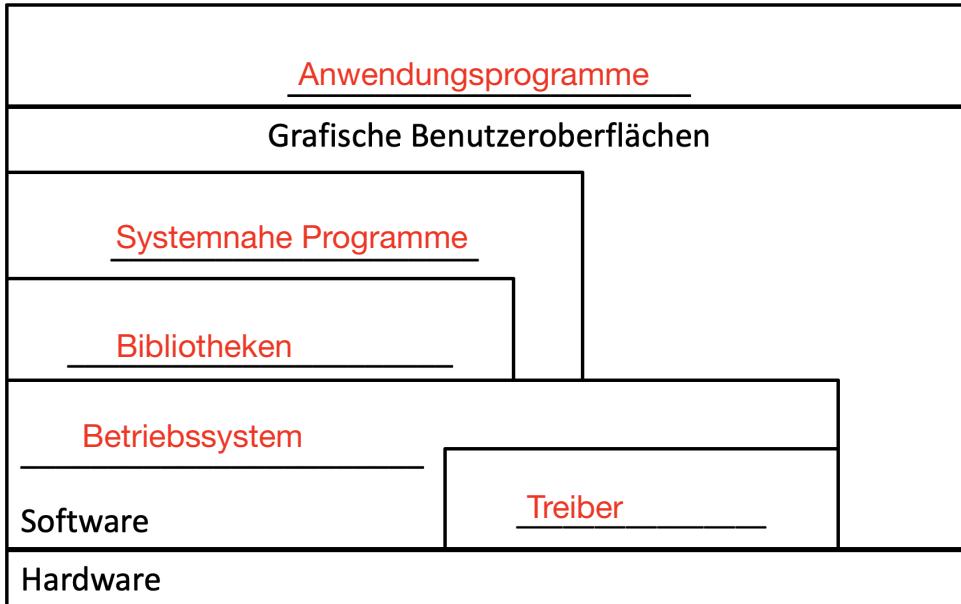
7

Name _____

3. System Calls

(7 Punkte)

- a) Beschriften Sie die (SW-)Hierarchien auf modernen UNIX/Windows-Systemen:



2

- b) Auf welche Arten kann der User-Kontext auf Intel x86 System-Calls aufrufen?

Mithilfe von Software Interrupts wie int \$0x80 oder spezialisierten schnellen Systemaufrufmechanismen SYSENTER / SYSEXIT oder syscall initiieren

2

- c) Ein HW-Interrupt tritt auf. Was passiert? Bitte den Ablauf kurz beschreiben!

Ablauf:

1. Ein Gerät meldet Interrupt über Interrupt Leitung
2. Der Mikroprozessor wird unterbrochen
3. Die ausgeführte Anwendung bemerkt davon nichts (außer dass sie evtl. verzögert reagiert)
4. Eine Interrupt-Behandlungsroutine (Interrupt Service Routine (ISR)) liest Daten (in BS-internen Puffer)
5. BS springt in die unterbrochene Anwendung zurück
6. Periodisch muss der BS-interne Puffer kopiert werden
7. Wartet die Anwendung auf die Daten (bsp. Daten von der Festplatte) können diese gelesen werden (04, S. 20)

3

Name _____

4. Linux Kernel & Scheduling (20 Punkte)

a) Was muss die HW mind. bieten, damit Scheduling implementiert werden kann?

- | | |
|----|--|
| 1. | Timer-Interrupt: Ermöglicht Betriebssystem in regelmäßigen Abständen Kontrolle zu übernehmen, um den Scheduler auszuführen |
| 2. | Prioritätsregister: Ermöglicht Priorisierung von Prozessen basierend auf ihrer Wichtigkeit oder ihrem Ressourcenbedarf |

2

b) In Linux Kernel arbeitet der Completely Fair Scheduler (CFS). Nach welchem Prinzip und mit welcher Datenstruktur arbeitet dieser?

Der CFS arbeitet nach dem Prinzip der fairen Verteilung von CPU-Zeit unter den Prozessen. Er verwendet eine Rot-Schwarz-Baum-Datenstruktur, um Prozesse zu verwalten und sicherzustellen, dass jeder Prozess proportional zu seiner Priorität CPU-Zeit erhält

2

c) Ein Prozess ruft einen blockierenden Systemcall (wie z.B. read()) zum Lesen von Daten von der Festplatte). Was passiert? Erklären Sie die Schritte.

- arbeitet, da er auf die Festplatte warten muss
- | |
|---|
| 1. Der Prozess wird in den Wartewall verschoben, nicht read() sondern Prozess der read |
| 2. Das BS weckt Prozess, plant anderen Prozess zur Ausführung ein |
| 3. Sobald der andere bereit ist, wird ursprünglicher Prozess wieder in Bereit-Warte Schlange aufgenommen (also Prozess der read() arbeitet) |
| 4. Wenn Prozess wieder an der Reihe ist wird read() Arbeit fortgesetzt
Daten werden in den Speicher des Prozesses geladen |

unterdrückt laufenden Prozess der read() arbeitet, weil dieser Prozess nicht weiterarbeiten kann
2 bis Pausen verfügbar sind

d) Welche (temporären) Informationen sollte ein Scheduler erfassen, um die Priorität eines Tasks neu zu berechnen?

- | |
|---|
| 1. Cpu Nutzung: Wie viel Prozessorarbeit Task bereits verwendet hat |
| 2. Wartezeit: Wie lange Task bereits auf Ressourcen gewartet hat |
| 3. Priorität: aktuelle Priorität des Tasks, die sich aus Wartezeit und anderen Faktoren, wie Art des Prozesses ergibt |

2

Name

e) Beurteilen Sie jede Aussagen ob diese Wahr (W) oder Falsch (F) ist:

Aussage	W/F?
Das 1:1 Modell reduziert Komplexität, ist damit einfacher <i>weinen Komplexität sein, da es mehr Arbeit für den Kernel haben für Verwaltung der Threads</i>	F/W
Der Kernel weiß beim 1:1 Modell, daß ein Thread auf einen Lock wartet <i>Bei 1:1 Kernel über Status jeder Threads informiert, ob Thread auf Lock wartet</i>	✓
1:1 Modell heißt ein User-Thread entspricht einem Task im OS <i>Im 1:1 Thread-Modell entspricht jeder User Thread direkt mit einem Kernel Thread</i>	✓
In struct task{} stehen die lauffähigen und gestoppten Tasks <i>Enthalten Informationen über alle Tasks</i>	✓
Der Scheduler wählt den Task mit der längstem vruntime <i>Der CFS wählt Task mit höherer vruntime an um Fairness zu gewährleisten</i>	F
Die vruntime ist in Millisekunden (ms) aufgeteilt <i>in nanosekunden</i>	F
Ein Task läuft nachdem er auf einem core gestartet wurde nur auf diesem <i>Blieb von einem Core zu anderem verhängen</i>	F
Als Nutzer kann ich nicht die Priorität meiner Prozesse setzen <i>auch als Nutzer möglich z.B. mit dem Befehl nice -n -p nach</i>	F
Die CPU-Zeit wird an die nr_running Tasks verteilt <i>CPU Zeit wird unter den laufenden Tasks verteilt</i>	✓
Java Anwendungen mit mehreren Threads nutzen auf Linux nur 1 Core <i>wollen auf Multicore anwenden lassen, wenn sie mehrere Threads haben</i>	F

f) Geben Sie mind. vier Kommandos an, um den Linux Kernel zu konfigurieren, zu kompilieren und zu installieren.

1.	0. <u>make oldconfig</u> : alte config übernehmen
	1. <u>make menuconfig</u> : textbasierter Konfigurationsverzweiger durch um Funktion des Kernels auszuwählen
2.	2. <u>make</u> : kompiliert Kernel
3.	3. <u>make modules</u> : kompiliert alle Module, die während make menuconfig ausgewählt wurden
	4. <u>make modules_install</u> : installiert kompilierte Module in entsprechender Verzeichnis
4.	5. <u>make install</u> : installiert Kernel selbst

4

2

Name _____

- g) Programmieren Sie ein minimales Kernel-Module, welches beim Laden prüft, ob der ladende Task die FPU nutzt und dies auf der Debug-Konsole ausdrückt. 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
```

```
#include <linux/module.h> ← gmaß legt in jedem Modul drin  
#include <linux/sched.h> ← für Task struct, PF-* kann man in Aufgabe nutzen  
#include <linux/init.h>  
#ifndef module_init  
MODULE_LICENSE("GPL");  
MODULE_AUTHOR("kevin");  
MODULE_DESCRIPTION("Prüft FPU Nutzung ladender Task");  
#endif  
# module lädt diese Funktion beim Start  
static int __init fpu_check_init(void){ ← Funktion zum laden des Moduls  
    struct task_struct *task; ← Zugriff auf den aktuellen Task  
  
    if (task->flags && PF_USED_MATH){ ← hier wird geprüft ob FPU verwendet  
        printk(KERN_INFO "ladende Task hat die FPU verwendet.\n");  
    } else {  
        printk(KERN_INFO "ladende Task FPU nicht verwendet.");  
    }  
    return 0;  
}  
  
static void __exit fpu_check_exit(void){ ← entlädt das Modul, aber reguläre Funktion  
    printk(KERN_INFO "Modul entwölft");  
}  
  
module_init(fpu_check_init); ← zum laden des Moduls  
module_exit(fpu_check_exit); ← zum entladen des Moduls
```

6

Name _____

5. Virtueller Speicher (18 Punkte)

- a) Wofür stehen und was bedeuten DPL, W, A und P im Deskriptor? (Zusatz: B)

	31	23	20	15	11	7	0
Base 31:24	G	B	0	A V L	Limit 19:16	P P L	D 1 Type 0 E W A
Base Address 15:00				Segment Limit 15:00			
31 16 15 0							
DPL	-Sind Bits 13,14 -geben Privileglevel des Segments an						
W	Bit 9, gibt an ob Segment beschreibbar ist						
A	Bit 8, gibt an ob Durch zugegriffen werden						
P	Bit 7 gibt an ob Segment im Speicher vorhanden ist						
B	2 Zusatzpunkte Bei Stack Segmenten gibt Bit 22 die Größe des Stacks, bestimmt Standardgröße der Operenden						

- b) Was macht der TLB, wann wird er ausgelesen, wer darf auf ihn schreibend zugreifen?

Was macht der TLB?	Ist eine Cache-Speicherart, die die virtuellen Speicheradressen, die von Programmern verwendet werden, in physische Adressen im Speicher des Computers übersetzt
Wann gelesen?	Wird jedes mal gelesen, wenn eine virtuelle Adresse abgesetzt werden muss, also bei jedem Speicherzugriff eines Programms
Wer schreiben?	Schreiben in TLB wird vom OS bzw. Kernel durchgeführt

- c) Was passiert bei einem TLB-Miss?

Übersetzung konnte nicht im TLB gefunden werden. BS muss Page-Table-Strukturen durchsuchen, sobald gefunden in Pages, wird Übersetzung in TLB geladen. Wenn Seite nicht in RAM, wird Page-fault ausgelöst, BS durchsucht Festplatte nach Page um sie in RAM zu laden.
--

- TLB-Miss lief Exception aus und damit Page-Table Walk des OS
- für jeden TLB Miss muss das OS die Pages eines Prozesses durchsuchen: OS nennt sich Page-Table-Walk

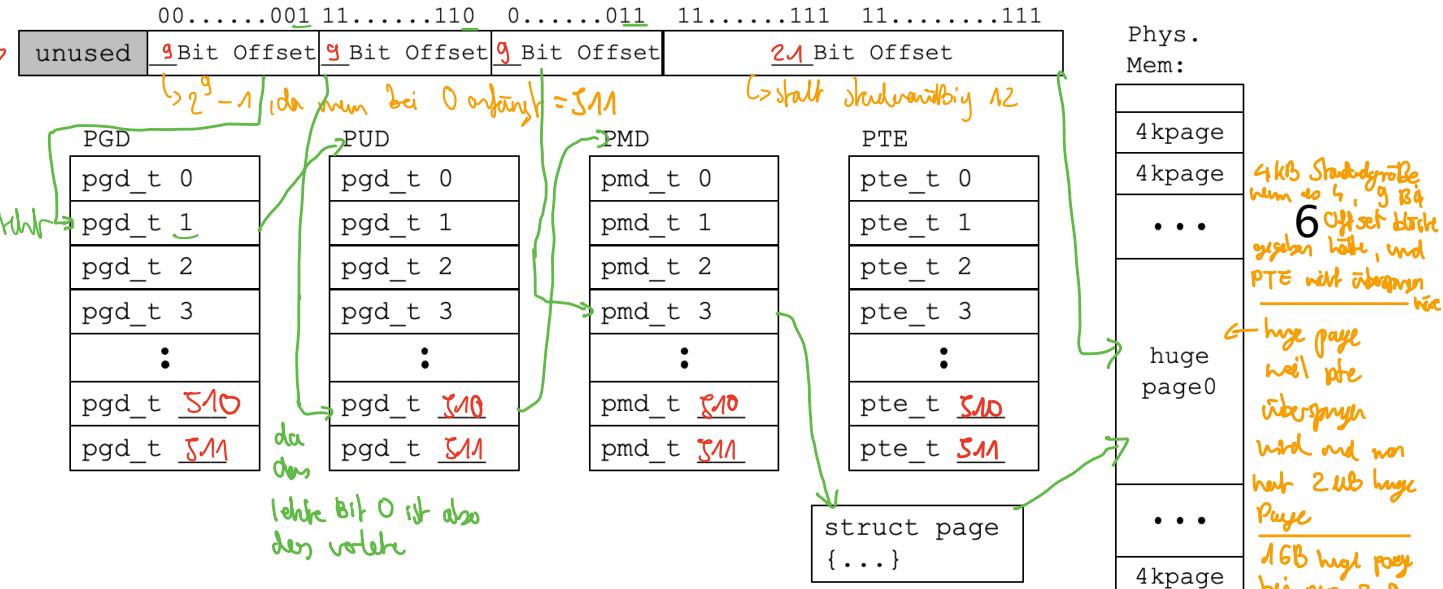
- ist eine Page im RAM aber nicht im L1B : Minor Fault
- ist eine Page nicht im RAM (auf Reopferte) : Major Fault

Sommersemester 2023

Name

- d) Zeichnen Sie alle Pointer für die folgende 64-Bit Virtuelle Adresse ein – bitte beachten Sie die binären Zahlenwerte (0...001 bedeutet eine 1 im niedrigerwertigsten Bit, ansonsten Nullen). Geben Sie weiterhin die Anzahl an den unterstrichenen Stellen ein.

Sollten Sie etwas korrigieren wollen, nutzen Sie bitte das Feld unten.



Platz für Kommentare und Verbesserungen

- e) Für besonders große Server und Speicheranforderungen, wie wird diese Speicherhierarchie erweitert werden?

Entweder durch Erweiterung der Seitentabellen wie fünfstufige Seitentabellen oder Verwendung von Large/Huge Pages: statt Standard 4-KB Seiten können größere Seitengrößen wie 2 MB oder 1 GB verwendet werden.

Sommersemester 2023

Name _____

- f) Der Buddy Allokator erlaubt, sehr effizient freie Speicherbereiche zu identifizieren. Die untenstehende Ansicht entspricht der Darstellung von Wikipedia. Im initialen Zustand 1 ist der gesamte Speicher frei. Zeichnen Sie die folgenden Allokationen ein:

1. Programm A allokiert 13 kB Speicher
2. Programm B allokiert 5 kB Speicher
3. Programm A gibt den Speicher wieder frei
4. Programm C allokiert 15 kB Speicher

	4kB	4kB	4kB	4kB	4kB	4kB	4kB	4kB	4kB	4kB	4kB	4kB	4kB	4kB	4kB	4kB
1.	2^4															
2.																
3.																
4.																
5.																
6.																
7.																
8.																
9.																
10.																
11.																

4

Name _____

6. IPC & Synchronisation

(9 Punkte)

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

Asynchrone Benachrichtigung eines Events:	<u>Signal()</u> , <u>sigaction()</u> : Systemaufrufe verwendet um Signalbehandler einzurichten oder zu verändern
Gemeinsamer Speicher	<u>shmget()</u> <u>shmat()</u> , <u>munmap()</u>
Uni-direktonaler Datentransfer via Kernel	<u>pipe()</u> um unidirektionale Pipe zu erstellen über die Daten von einem Prozess zu einem anderen fließen können
Direkt in den Speicher eines Prozesses schreiben	<u>writen()</u> , <u>ptrace()</u> kann verwendet werden um in Speicher eines anderen Prozesses zu schreiben <u>write()</u> kann dann auf den Speicher angewendet werden

2

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

Vorteile:	<u>Einfachheit</u> : Austauschen von Daten über Dateien ist einfache IPC-Methode, leicht zu verstehen, implementiert
	<u>Dauerhaftigkeit</u> : Daten, die in Dateien gespeichert sind, bleiben nach Ende Prozesse erhalten. Nötig wenn Daten über lange Zeiträume persistent werden sollen
	<u>Unabhängigkeit von der Prozesslebenszeit</u> : Daten in Dateien sind unabhängig von Lebensdauer der Prozesse die sie erzeugen oder konsumieren
	<u>Kein Bedarf an komplexer Synchronisation</u> : Dateioperationen sind atomic, für nahezu keine zusätzliche Synchronisation erforderlich
	<u>Zugriffskontrolle</u> : Dateibasierte IPC profitiert von vorhandenen Dateisystemberechtigungen die Zugriff regeln
Nachteile:	<u>Effizienz</u> : Lesen/Schreiben kann langsamer sein als andere IPC Methoden
	<u>Synchronisation</u> : gleichzeitiger Zugriff mehrere Prozesse auf dieselbe Datei kann Synchronisationsmechanismen erfordern
	<u>Ressourcenverteilung</u> : kann zu ineffizienter Ressourcenverteilung führen
	<u>Komplexität bei gleichzeitigem Zugriff</u> : mehrere Prozesse gleichzeitig auf Datei zugreifen, kann Handhabung komplex werden
	<u>Dateisystemabhängigkeit</u> : IPC ist abhängig vom Zustand des Dateisystems

5

- c) Nennen Sie vier Synchronisationsmechanismen:

1.	<u>Semaphore</u> :
2.	<u>Mutex</u> :
3.	<u>Bedingungsvariable</u> :
4.	<u>Read-Write Locks</u> : ermöglicht mehreren Threads, gleichzeitig lesenden Zugriff auf Ressource zu haben

2

Name _____

7. Virtualisierung & Echtzeit-OS (12 Punkte)

- a) Warum ist die x86-Architektur so besonders schwierig, effizient zu virtualisieren? (erinnern Sie sich an das Paper von VMware und die Virtualisierung mittels trap-and-emulate und eine der Instruktionen).

Gast Betriebssystem kann sehen, dann es nutzt die

3

Trap-and-Emulate: Wenn VM eine sensible Operation ausführt

Popf:

- b) Nennen Sie drei Vorteile von Virtuellen Maschinen – und drei Nachteile:

Vorteile:	
1.	
2.	
3.	
Nachteile:	
1.	
2.	
3.	

3

- c) Nennen Sie drei Vorteile von Container-Technologien (z.B. Docker):

1.	
2.	
3.	
4.	

2

Name

- d) Bei einem System gehen m periodische Ereignisse i mit der jeweiligen Periode P_i ein. Das Ereignis i verbraucht C_i CPU-Zeit. Wann ist dieses System noch stabil und kann diese Ereignisse zeitlich verarbeiten?

Geben Sie die Formel an.

2

- e) Was heißt ein Standard Linux-Kernel (wie in der VL demonstriert) ist nicht echtzeitfähig. Was zeigte der demonstrierte Kernel?

Was heißt nicht echtzeitfähig:

Was zeigte dieser Kernel:

2