Name, Matrikelnummer

| | | | |
|---|---|---|---|
| **Prüfer:** | Prof. Dr.-Ing. Rainer Keller | **Anzahl der Seiten:** | 9 |
| **Studiengänge:** | Softwaretechnik und Medieninformatik<br>Technische Informatik<br>Ingenieurpädagogik | **Semester:** | SWB2<br>TIB2<br>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 (9 Punkte)

a) Was war so exorbitant an der Entwicklung der Mikroprozessoren?

Der rasante Anstieg der Performance in sehr kurzen Zeitabständen.

3

b) Wie lautet die in der Vorlesung gegebenen Definition eines Betriebssystems?

„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.“

4

c) Circa wie groß ist ein L3 Cache eines modernen Intel Prozessors?

Circa 20MB

2

Name, Matrikelnummer

# Bash Shell (17 Punkte)

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

| | |
|---|---|
| Größe aller Dateien im Verzeichnis: | `du` |
| Hilfeseite von bash aufrufen: | `man` |
| Prozess 666 hart beenden: | `kill -9 666  (sigkill)` |
| Top Prozesse auflisten: | `top` |
| Datum ausgeben: | `date` |
| Archive ver-/auspacken: | `tar` |
| Prozess in den Hintergrund: | `bg` |
| Zeilen zählen: | `wc -l` |

4

b) Die Umgebungsvariable DATEI enthält den Namen einer Datei. Aus dieser Datei sollen Sie Vorname, Name, Matrikelnummer und Note von Studenten auslesen. Die Daten sind jeweils durch Doppelpunkte getrennt, die Note ist zweistellig, bspw.
`Max:Meier:12345:20`
`Moritz:Mueller:23456:1`
`Miriam:Musterfrau:34567:10`
Schreiben Sie ein Shell-Script, welches die Matrikelnummer des Studenten mit der besten Note ausgibt.

```
#!/bin/bash

datei="$DATEI"
beste_matrikelnummer=$(awk -F':' 'BEGIN
        {beste_note = 0} $4 > beste_note {beste_note = $4; matrikelnummer = $3} END {print matrikelnummer}' "$datei")
echo "Die Matrikelnummer des Studenten mit der besten Note ist: $beste_matrikelnummer"
```

7

Name, Matrikelnummer

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

Grundlegende Infrastruktur (direkter Zugriff auf Resourcen)

- dmesg
- strace
- lsdev
- lscpu
- gcc
- lnstat
- strings
- objdump
- modprobe
- insmod
- sysctl
- ...

3

d) Was machen die beiden folgenden Programme?

| | |
|---|---|
| mount | Bindet ein Dateisystem im Verzeichnisbaum ein, bspw.: mount -t ext2 /dev/sdb2 mein_dir/ |
| fsck | Prüft das auf dem Device befindliche Dateisystem, bspw. fsck /dev/sda1 |
| mknod | Auf Geräte wird mittels spezieller Dateien identifiziert (meistens in /dev); diese kann man selbst anlegen, bspw. mknod /dev/disk0 -b 14 0 |

3

Name, Matrikelnummer

# Scheduling und Systemcalls (22 Punkte)

a) Nennen Sie drei Optionen zum Setzen der Häufigkeit der Unterbrechung der CPU durch den Linux Scheduler – und was sind deren Vor- / Nachteile

| Wie häufig? | Vor- / Nachteil |
|---|---|
| 1. | |
| 2. | SIEHE NÄCHSTE SEITE |
| 3. | |

5

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?

1.) Verbrauchte CPU-Zeit und Art des Tasks

2.) Um Fairness/Gleichberechtigung zwischen den versch. Tasks herszustellen

⟶ I/O Tasks werden priorisiert (..., um das Nutzererlebnis angenehm zu gestalten)

3

c) Wie kann man Systemcalls einer Anwendung mit der PID 123 mitlesen?

```
strace -p 123
```

2

zu a)

## Allgemeine Scheduling Verfahren 2/6

HOCHSCHULE ESSLINGEN

- First-Come-First-Serve (FCFS), First-In-First-Out (FIFO):
  - » Prozesse werden in der Reihenfolge ihrer Ankunft **ohne Unterbrechung** bis zum Ende bzw. Blockierung bedient
  - \+ das einfachste Verfahren
  - \+ Implementierung über einfache Warteschlange
  - \+ im Allgemeinen gute Auslastung der Ressource (z.B. CPU)
  - – u. U. große Wartezeiten und damit schlechte Antwortzeiten
  - – kann zu langen Warteschlangen mit I/O-intensiven Prozessen führen, wenn ein rechenintensiver Prozess mit wenig I/O-Operationen immer wieder die CPU blockiert



## Allgemeine Scheduling Verfahren 3/6

HOCHSCHULE ESSLINGEN

- First-In-First-Out (FIFO) mit Prioritätenwarteschlange:
  - » Prozesse werden in der Reihenfolge ihrer Ankunft **mit Unterbrechung** bis zum Ende bzw. Blockierung bedient, aber rechenintensive Prozesse erhalten eine geringe Priorität und I/O-intensive eine hohe Priorität. Prozesse derselben Prioritätsklasse werden in der Reihenfolge ihrer Ankunft ohne Unterbrechung bis zu ihrem Ende bzw. ihrer Blockierung bedient. Prozesse können durch Prozesse höherer Priorität unterbrochen werden. Der Scheduler kann die Prioritäten der Prozesse ändern.
  - \+ das einfachste Verfahren
  - \+ Implementierung über einfache Warteschlange**n**
  - \+ im Allgemeinen gute Auslastung der CPU
  - \+ Je nach Priorisierung faire Verteilung und
  - \+ Gutes Antwortverhalten (für I/O-intensive Prozesse)

- Shortest Job First:
  - » Der Prozess mit der kürzesten Rechenzeit wird als nächster **ohne Unterbrechung** bedient; Kenntnis der Rechenzeit ist erforderlich; Prozesse mit gleicher Rechenzeit werden nach FCFS bedient
  - \+ Verfahren ist optimal bezogen auf die mittlere Wartezeit für eine vorgegebene Menge von Prozessen
  - – Voraussichtliche Rechenzeit muss bekannt sein
  - » nicht geeignet für kurzfristiges *Scheduling* innerhalb des OS, da die Rechenzeiten i.A. nicht bekannt sind,
  - » Wird eingesetzt für langfristiges *Scheduling* im Batchbetrieb (Das Zeitlimit wird vom Benutzer angegeben / geschätzt...).





- Round-Robin Scheduling:
  - » Der Prozess darf den Prozessor für eine **Time-Slice** benutzen; falls er in dieser Zeit nicht fertig wird, wird er **unterbrochen** und am Ende der Warteschlange eingereiht; es handelt sich hier um ein zyklisches Verfahren ohne Prioritäten.
  - \+ gleichmäßige Aufteilung der CPU-Zeit auf die Prozesse
  - – Größe der Zeitscheibe ist stark situationsabhängig:
    - ➢ zu kleine Zeitscheibe
      ⇒ hoher Verwaltungsaufwand durch Prozesswechsel
      ⇒ Effizienz niedrig, da Cache immer „cold“
    - ➢ zu große Zeitscheibe
      ⇒ nähert sich dem FCFS-Verfahren, da mit zunehmender Zeitscheibe blockierende Systemaufrufe wahrscheinlicher werden
      ⇒ mittlere Wartezeiten und Antwortzeiten werden größer

- Fair-Share Scheduling:
  - » Bindung eines Anteils einer CPU an den Process-Owner bzw. User, anstelle des Prozesses selbst. Einer User, der mehrere Prozesse laufen lässt, erhält für die Summe aller seiner Prozesse den gleichen Anteil an CPU Ressourcen wie ein anderer User mit nur einem Prozess.
  - \+ Benutzer die das System vollmachen werden bestraft.
  - – Verwaltung der Information ist ziemlich aufwändig.

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 | F |
| Früher hatte Linux ein 1:n Task:Thread Modell | F |
| Zur besseren Lastbalance werden Tasks auf freie Cores verschoben | W |
| Die `vruntime` ist in Millisekunden (ms) aufgeteilt | F |
| In `struct task{}` stehen die lauffähigen und gestoppten Tasks | F |
| Ein Task läuft nachdem er auf einem core gestartet wurde nur auf diesem | F |
| Ein Prozess besteht aus einem Thread und ist ein Task | F |
| Die CPU-Zeit wird an die `nr_running` Tasks verteilt | F |

5

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

Geteilte Ressourcen?

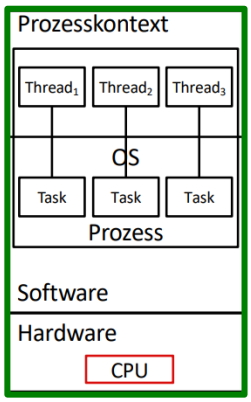
- Deskriptoren
- Signal-Handler
- Virtueller Adressraum
- Umgebungsvariablen

Per-Thread Ressourcen?

- Registerinhalte
- Stack-Speicher
- Thread-Variablen

3

f) Malen Sie die Zuordnung von Linux Threads eines Prozesses auf Tasks, sodaß die Zugehörigkeit zu den Ebenen klar wird:



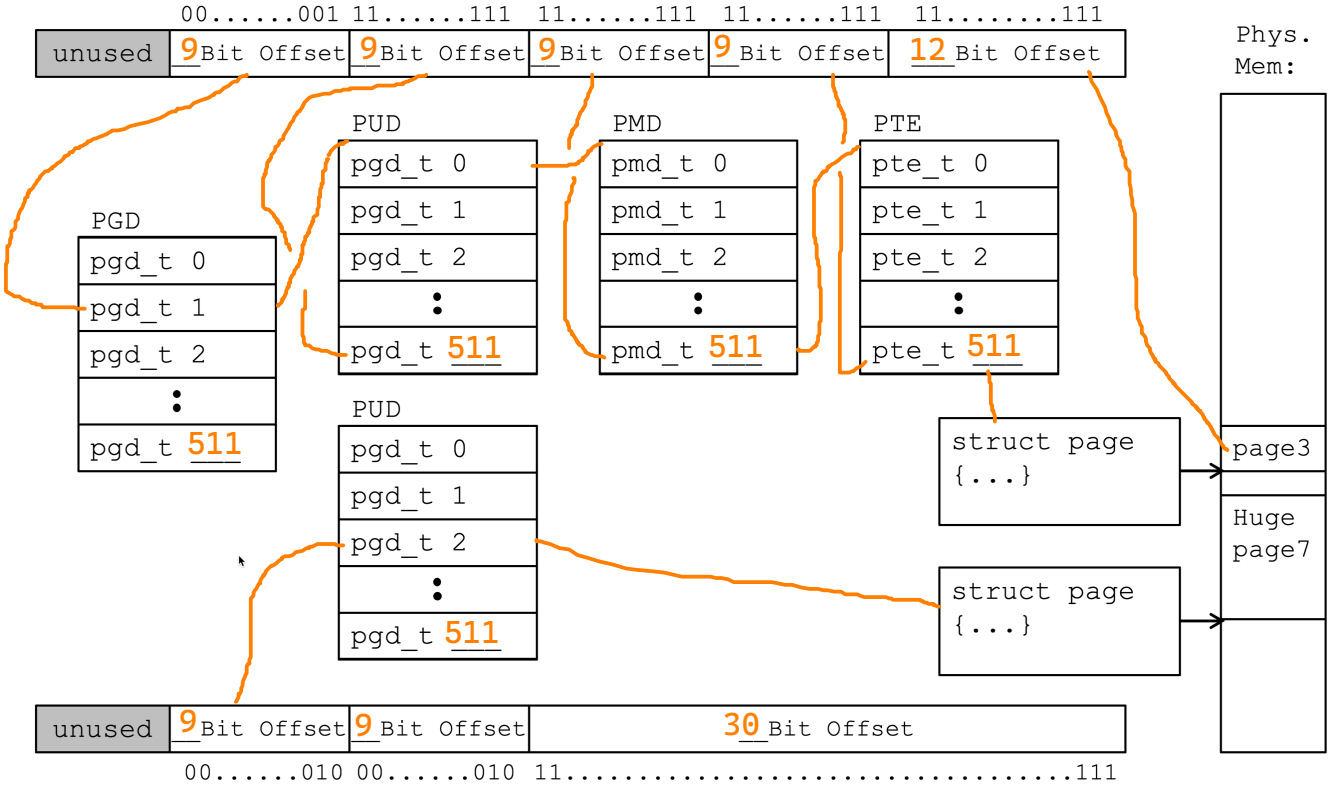

4

Name, Matrikelnummer

# Virtueller Speicher (19 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. 8

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




Korrektur:

Würde mich freuen, wenn's mir nochmal jemand erklärt.
Ich denke das stimmt absolut nicht ...

Name, Matrikelnummer

b) Der Data Segment Descriptor bietet 3 Bits für Speicherschutz: Writable, Accessed und Present. Welche Konzepte lassen sich mit diesen Bits umsetzen?

- W: Writeable bit = Speichersegment ist beschreibbar
- A: Access Bit = Speichersegment wurde angefasst
- P: Present Bit = Segment ist im Hauptspeicher

4

c) Der Intel Prozessor unterstützt 4 Ebenen für den Priviledge Level (CPL). Welche Softwareschicht auf einem Betriebssystem beispielhaft zugeordnet sein?

| | |
|---|---|
| 0. Ebene: | Kernel |
| 1. Ebene: | Treiber |
| 2. Ebene: | Software |
| 3. Ebene: | Nutzer |

4

d) Der Buddy Allocator für Speicherverwaltung hat ein Problem mit Fragmentierung. Wieso ist das für Verwaltung von VM-Pages kein Problem?

- Aufteilung in feste Einheiten (pages)
- Fragmentierung des Speichers wird 'maskiert'
  ⟶ durch Zuordnung der virtuellen Seiten auf physischen Speicher
- Dynamische Speicherverwaltung
-

3

Name, Matrikelnummer

# Interprozesskommunikation (14 Punkte)

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

| | |
|---|---|
| Direkt in Speicher eines Prozesses schreiben: | write(), memcpy() |
| Einem Prozess eine Benachrichtigung schicken: | pipe(), kill() |
| Datei beschreiben: | write(), mmap() |
| Uni-direktionale Verbindung: | send(), close() |

2

b) Warum könnte eine Uni-direktionale Verbindung langsamer sein – aber wieso ist das Konzept dennoch so gut?

| |
|---|
| Gründe für langsame Performance:<br>- Datenrate wird nicht ausgenutzt (nur einseitig)<br>- ...? |
| Gründe für gutes Konzept:<br>- Minimiertes Sicherheitsrisiko<br>- Skalierbarkeit<br>- Zuverlässigkeit |

6

c) Nennen Sie den vermutlich schnellsten IPC-Kommunikation bezüglich Bandbreite und eine Begründung warum:

| Schnellster Call: | Begründung: |
|---|---|
| shared memory | - kein Overhead<br>- Direkter Zugriff über Arbeitsspeicher |

6

Name, Matrikelnummer

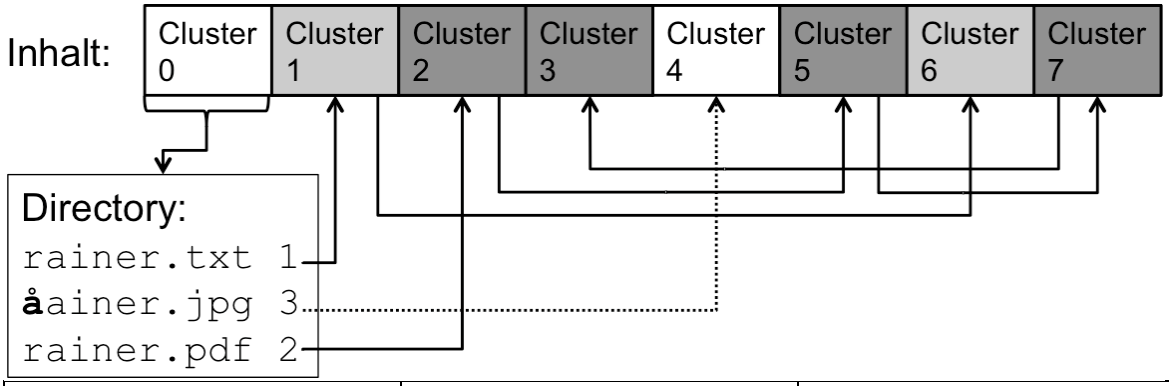
# Dateisysteme (9 Punkte)

a) Zur korrekten Unterscheidung von Datenträgergrößen wurde eine IEC-Norm eingefügt. Nennen Sie mindestens 3 Größen und geben grob an was das an Unterschied ausmacht.

| Bezeichnung | Prozentualer Unterschied |
|---|---|
| KibiByte | 2-3% |
| MebiByte | circa 5% |
| GibiByte | 7-8% |

3

b) Defragmentieren Sie das unten stehende FAT-Verzeichnis. Geben Sie Schritte mit an, sowie die Folge der Cluster je Datei.




| | | |
|---|---|---|
| 1. | 2. | 3. |
| 4. | 5. | 6. |
| Resultat für Folge rainer.txt | | |
| Resultat für Folge rainer.pdf: | | |

Das hab' ich noch nicht verstanden!
Schau' ich mir nochmal an.

6