Karlsruher Institut für Technologie

Betriebssysteme
(Operating Systems)

Prof. Dr. Frank Bellosa
Jonas Kiemel, M.Sc.
Mathias Gottschlag, M.Sc.

Musterlösung Nachklausur
26.09.2019

Alle Punkteangaben ohne Gewähr!

- Bitte tragen Sie zuerst auf dem Deckblatt Ihren Namen, Ihren Vornamen und Ihre Matrikelnummer ein. Tragen Sie dann auf den anderen Blättern (auch auf Konzeptblättern) Ihre Matrikelnummer ein.

  Please fill in your last name, your first name, and your matriculation number on this page and fill in your matriculation number on all other pages (including draft pages).

- Die Prüfung besteht aus 16 Blättern: Einem Deckblatt und 15 Aufgabenblättern mit insgesamt 5 Aufgaben.

  The examination consists of 16 pages: One cover sheet and 15 sheets containing 5 assignments.

- Es sind keinerlei Hilfsmittel erlaubt!

  No additional material is allowed!

- Die Prüfung gilt als nicht bestanden, wenn Sie versuchen, aktiv oder passiv zu betrügen.

  You fail the examination if you try to cheat actively or passively.

- Sie können auch die Rückseite der Aufgabenblätter für Ihre Antworten verwenden. Wenn Sie zusätzliches Konzeptpapier benötigen, verständigen Sie bitte die Klausuraufsicht.

  You can use the back side of the assignment sheets for your answers. If you need additional draft paper, please notify one of the supervisors.

- Bitte machen Sie eindeutig klar, was Ihre endgültige Lösung zu den jeweiligen Teilaufgaben ist. Teilaufgaben mit mehreren widersprüchlichen Lösungen werden mit 0 Punkten bewertet.

  Make sure to clearly mark your final solution to each question. Questions with multiple, contradicting answers are void (0 points).

Die folgende Tabelle wird von uns ausgefüllt!
The following table is completed by us!

Aufgabe	1	2	3	4	5	Total
Max. Punkte	12	12	12	12	12	60
Erreichte Punkte						
Note						

Aufgabe 1: Grundlagen

Assignment 1: Basics

a) Ein Prozess ruft den Kernel durch einen Systemaufruf auf. Nennen Sie zwei Möglichkeiten, die Parameter an den Kernel zu übergeben. **1 pt**

A process invokes the kernel by a system call. Name two ways of passing the parameters to the kernel.

Lösung:

- *Parameters can be passed in registers.*
- *Parameters can be pushed on the stack by a program and popped off by the kernel.*
- *When there are more parameters than registers, parameters can be stored in a block in memory and the address of the block can be passed as a parameter in a register.*

(0.5 P) *if one correct way is given,* **(1 P)** *if two correct ways are given*

Warum wird bei einem Systemaufruf nicht unbedingt ein Kontextwechsel benötigt? **1 pt**

Why does a system call not necessarily require a context switch?

Lösung:

Since the kernel is mapped into the address space of all processes **(1 P)***, system calls that do not block can be executed without changing address spaces.*

b) Geben Sie abseits möglicher Performancevorteile einen Grund an, warum mindestens ein Teil von Unterbrechungsroutinen typischerweise in Assemblersprache geschrieben ist. **1 pt**

Besides potential performance advantages, give a reason why at least a part of interrupt service routines is typically written in assembly language.

Lösung:

High-level languages usually do not allow the kind of low-level access **(0.5 P)** *to CPU hardware that is required in interrupt service routines (ISR). For instance, an ISR may be required to (de-)activate the interrupt servicing of a particular device or directly access certain registers* **(0.5 P)***.*

c) Nennen Sie jeweils eine durch den Prozessor ausgelöste Exception, bei welcher der Prozess nach Behandlung üblicherweise fortgesetzt beziehungsweise beendet wird. **1 pt**

Name one exception triggered by the processor for which the process is usually continued after exception handling and one for which the process is usually terminated.

Lösung:

The process is usually resumed after a page fault. **(0.5 P)**

The process is usually terminated after a division by zero or an invalid opcode. **(0.5 P)**

Further information on exception/signal handling can be found at `https://www.oreilly.com/library/view/understanding-the-linux/0596005652/ch04s02.html` *and by calling man 7 signal.*

d) Nennen Sie fünf im Benutzeradressraum üblicherweise vorhandene Segmente. **2.5 pt**

Name five segments typically present in the user address space.

Lösung:

(0.5 P) *per correct segment: .data, .rodata, .bss, .text, stack, heap*

Note that the OS is not in the user *address space.*

Nennen Sie eines dieser Segmente, dessen Speicher üblicherweise zwischen mehreren Prozessen geteilt wird. Weshalb wird es geteilt? **1 pt**

Name one of these segments whose memory is usually shared between multiple processes. Why is it shared?

Lösung:

(0.5 P) *for any correct answer: .text, .rodata*

Sharing saves memory **(0.5 P)** *(and load time).*

Begründen Sie für eines der Segmente, weshalb es *nicht* geteilt werden kann. **1.5 pt**

Give a reason for one of the segments why it cannot *be shared.*

Lösung:

For example, the heap **(0.5 P)** *cannot be shared because then allocations and modifications from one process would be visible in other processes, breaking the isolation* **(1.0 P)**.

e) Nennen Sie vier Operationen, die das Betriebssystem zur Verwendung von Dateien bereitstellt. **2 pt**

Name four operations the operating system offers for working with files.

Lösung:

- `create()`
- `write()`
- `read()`
- `reposition()` or `seek()`
- `delete()`
- `truncate()`
- `open()`
- `close()`

f) Über welche zwei Wege kann der Kernel auf Geräteregister zugreifen? **1 pt**

Which two ways can the kernel use to access device registers?

Lösung:

- *port-mapped I/O*
- *memory-mapped I/O*

Total: 12.0pt

Aufgabe 2: Prozesse und Threads

Assignment 2: Processes and Threads

a) Bei welchem Threadmodell kann ein Prozess die Leistung von Multikernsystemen nicht vollständig ausnutzen? **0.5 pt**

Which thread model does not allow processes to utilize the full performance of multi-core systems?

Lösung:

The Many-to-One Model (User Level Threads) **(0.5 P)**

b) Geben Sie zwei Gründe an, warum das Betriebssystem die Adressräume verschiedener Prozesse voneinander isolieren sollte. **1 pt**

Give two reasons why the operating system should isolate the address spaces of different processes.

Lösung:

(0.5 P) *per reason,* **(1.0 P)** *maximum:*

- *Buggy program could trash other programs.*
- *Malicious user could steal other users sensitive data like passwords.*
- *Selfish user could use up all (virtual) memory for himself.*

c) Wie viele Kindprozesse werden im unten gezeigten Code bei der Ausführung von `main()` erzeugt. Begründen Sie Ihre Antwort. **1.5 pt**

How many child processes are created in the code shown below when executing `main()`? Justify your answer.

```
1 void main() {
2     fork();
3     fork();
4 }
```

Lösung:

Three child processes are created. **(0.5 P)** *After the initial process forks in line 2, there are two processes running, a parent and a child.* **(0.5 P)** *Each of them then forks in line 3, creating two additional processes.* **(0.5 P)** *Then all the processes exit.*

Wie entstehen Zombie-Prozesse? Welchen Zweck erfüllen Sie? **2 pt**

How do zombie processes arise? Which purpose do they serve?

Lösung:

A zombie process is a process that has completed execution (via the exit system call) but still has an entry in the process table. **(1 P)** *The entry in the process table is still needed to allow the parent process to read its child's exit status via wait().* **(1 P)**

d) Das Shortest-Job-First-Scheduling-Verfahren wählt zur Ausführung den Prozess mit der kürzesten Gesamtausführungszeit. Erklären Sie, warum sich das Verfahren nicht für beliebige Prozesse umsetzen lässt. **0.5 pt**

The shortest job first scheduling policy selects the process for execution which has the smallest total execution time. Explain why the policy cannot be implemented for arbitrary processes.

Lösung:

The total execution time of a process must be known before execution. **(0.5 P)**

Erklären Sie, wie die Shortest-Job-First-Strategie in der Praxis approximiert werden kann. **1.5 pt**

Explain how the shortest job first strategy can be approximated in practice.

Lösung:

The shortest job first strategy can be approximated by selecting the next process based on the length of its next CPU burst rather than its total execution time. **(1 P)** *The length of the next CPU burst can be predicted as an exponential average of previous CPU bursts* **(0.5 P)**.

e) Erklären Sie, wie ein egoistischer Prozess seine CPU-Zeit maximieren kann, wenn Multi-Level Feedback Queue Scheduling verwendet wird. **2 pt**

Explain how a selfish process can maximize its amount of CPU time when multi-level feedback queue scheduling is used.

Lösung:

A selfish process can maximize its CPU time by relinquishing the CPU right before its time quantum is fully used up **(1 P)** *, thus keeping or increasing its priority.* **(1 P)** ***Note:*** *Trying to be demoted to the lowest queue by repeatedly using up the full time quantum does not maximize the total CPU time since all processes in queues with higher priorities are scheduled first.*

f) Ein Scheduling-Verfahren wählt zur Ausführung aus allen lauffähigen Prozessen den Prozess mit der höchsten Priorität aus. Nehmen Sie an, dass es drei Prozesse L, M und H mit den Prioritäten p(L) < p(M) < p(H) gibt. Erklären Sie, wie es zu Prioritätsinversion kommen kann. **2 pt**

A scheduling policy selects among all runnable processes the one with the highest priority. Assume, there are three processes L, M, and H with priorities p(L) < p(M) < p(H). Explain how priority inversion can occur.

Lösung:

If process H requires a resource, which is currently held by process L **(0.5 P)**, *process H has to wait for process L to release the resource* **(0.5 P)**. *Process L cannot release the resource as long as process M is runnable, since process M is scheduled prior to process L* **(0.5 P)**. *Indirectly, process M affects how long process H has to wait for the resource, although process M has a lower priority than process H* **(0.5 P)**. *This effect is called priority inversion.*

Erklären Sie, wie Prioritätsinversion verhindert werden kann. **1 pt**

Explain how priority inversion can be prevented.

Lösung:

Priority donation **(0.5 P)** *(also known as priority inheritance) prevents the problem of priority inversion. A process that is blocking a resource needed by a higher-priority process inherits the higher priority until it releases the resource.* **(0.5 P)** **Total: 12.0pt**

Aufgabe 3: Koordination und Kommunikation von Prozessen

Assignment 3: Process Coordination and Communication

a) Erklären Sie den Unterschied zwischen synchroner und asynchroner IPC. **1 pt**

Explain the difference between synchronous and asynchronous IPC.

Lösung:

Whereas asynchronous IPC system calls return immediately even if the message has not been sent/received **(0.5 P)**, *synchronous IPC system calls block until a message is sent/received* **(0.5 P)**.

b) Gegeben sei eine Integervariable `x`. Nehmen Sie an, dass `x++` von zwei Threads parallel ausgeführt wird. Erklären Sie, weshalb die Variable manchmal insgesamt nur um eins erhöht wird. **1.5 pt**

Given an integer variable `x`, *assume that* `x++` *is executed by two threads in parallel. Explain why overall the variable is sometimes only incremented by one.*

Lösung:

The `++` *operator first loads the old value, then stores the incremented value in the memory location of* `x`, *thus executing two distinct memory accesses* **(0.5 P)**. *If the second thread loads the value before the first has stored the incremented value, the original value is incremented in both cases, so one update is lost* **(1.0 P)**.

(0.5 P) *if only "race condition" is mentioned without an explanation.*

c) Erklären Sie, wie Virtualisierung von Ressourcen Deadlocks verhindern kann. Welche der Bedingungen für das Auftreten von Deadlocks wird negiert? **1.5 pt**

Explain how virtualization of resources can prevent deadlocks. Which of the requirements for the occurrance of deadlocks is negated?

Lösung:

Virtualization makes a larger (potentially infinite) number of virtual resource instances available to programs **(1.0 P)**, *through some form of multiplexing. Therefore, programs can share resources, so the mutual exclusion requirement does not hold anymore* **(0.5 P)**.

Alternative: Virtualization can prevent deadlocks if it implements multiplexing based on preemption. In that case, the no-preemption requirement does not hold anymore.

In welche der drei Kategorien *Deadlock Prevention*, *Deadlock Avoidance* und *Deadlock Detection* fällt dieser Ansatz? **0.5 pt**

Into which of the three categories deadlock prevention, deadlock avoidance, *and* deadlock detection *does this approach fall?*

Lösung:

Deadlock prevention **(0.5 P)**, *as virtualization of resources is a mechanism applied to prevent deadlocks already during the development of the software.*

d) Nennen Sie die drei gewünschten Eigenschaften einer Lösung für das Kritischer-Abschnitt-Problem (ohne Erklärung). **1.5 pt**

Name the three desired properties for a solution of the critical section problem (without explanation).

Lösung:

Mutual exclusion **(0.5 P)**, *progress* **(0.5 P)**, *bounded waiting* **(0.5 P)**.

Das folgende Codebeispiel verwendet ein schwaches Semaphor (d. h. ein `signal()`-Aufruf weckt einen beliebigen wartenden Thread auf), sorgt jedoch mit einer Zählervariable dafür, dass die Threads in der Reihenfolge, in der sie Zeile 2 ausführen, den kritischen Abschnitt betreten. Der Code wird von mehreren Threads eines Prozesses ausgeführt. Nehmen Sie an, dass die Zählervariablen nicht überlaufen und dass Zeile 2 atomar ausgeführt wird.

Nennen Sie für die drei oben genannten Anforderungen, ob diese Lösung sie erfüllt. Begründen Sie jeweils, weshalb die Anforderung erfüllt oder nicht erfüllt wird. **6 pt**

The following code section uses a weak semaphore (i.e., a call to `signal()` wakes up an arbitrary waiting thread), but uses a counter variable to ensure that the threads enter the critical section in the order in which they execute line 2. The code is executed by multiple threads of one process. Assume that the counter variables do not overflow and that line 2 is executed atomically.

For the three requirements named above, state whether the solution fulfills them. Each time, justify why or why not the requirement is fulfilled.

Globale Variablen / *Global variables:*

```
Semaphore s = 1; /* initially, one thread can enter */
int next_ticket = 0;
int allowed_to_enter = 0;
```

Von allen Threads ausgeführter Code / *Code executed by each thread:*

```
1  /* the ticket determines the order in the critical section */
2  int my_ticket = next_ticket++; /* atomic! */
3  while (true) {
4          wait(s);
5          if (my_ticket == allowed_to_enter) {
6                  break;
7          } else {
8                  /* wake up a different thread */
9                  signal(s);
10         }
11 }
12
13 /* ... critical section ... */
14
15 allowed_to_enter++;
16 signal(s);
```

Lösung:

- *Mutual exclusion is fulfilled* **(0.5 P)**, *because each thread has its individual ticket and only one ticket can pass the condition in line 5. The next thread can only enter once the previous thread has left the critical section and has executed line 15* **(1.5 P)**.
  Alternative: The semaphore ensures mutual exclusion because a thread in the critical section has executed `wait()` *more often than* `signal()`.
- *Progress is not fulfilled* **(0.5 P)**, *because if three threads wait for the critical section, the semaphore can randomly wake up only the two of the three threads whose ticket does not match, in which case no thread will ever enter the critical section* **(1.5 P)**.
- *Bounded waiting is fulfilled* **(0.5 P)**, *because the order of the threads executing line 1 determines the order in the critical section. Therefore, once a thread has executed line 1, there is a bound on the number of other threads entering the critical section before the thread enters it, which fulfills the definition given in the lecture* **(1.5 P)**.

(1.5 P) *for identifying the livelock which prevents progress, even if it is (incorrectly!) given as an argument against bounded waiting.* **(0.5 P)** *for correct definitions of the requirements.*

Total: 12.0pt

Aufgabe 4: Speicher

Assignment 4: Memory

a) Nennen Sie zwei Vorteile sowie zwei Nachteile von seitenbasiertem virtuellen Speicher gegenüber direkter physischer Adressierung. **2 pt**

Give two advantages and two disadvantages of page-based virtual memory compared to direct physical addressing.

Lösung:

Advantages, **(0.5 P)** *each:*

- *Isolation (Process ↔ Process, Process ↔ Kernel)*
- *Sharing (e.g., libraries)*
- *Memory overcommitment can be implemented*
- *Easier allocation of physical memory (i.e., fragmentation allowed)*

Disadvantages, **(0.5 P)** *each:*

- *More complex hardware necessary (MMU + TLB)*
- *Higher access latency (TLB lookup, page table traverse on TLB miss)*
- *Context switching costs increase due to necessary TLB flushes (strictly speaking, this is only a disadvantage of operating systems with* multiple *virtual address spaces)*
- *Kernel entries become necessary as only the OS is allowed to manipulate the address space*
- *More complex to implement*
- *Memory overhead for page tables*

b) Welche generellen Aufgaben muss ein Betriebssystem erfüllen, um seitenbasierte virtuelle Adressräume zu implementieren? Gehen Sie von einem System mit hardwaregesteuertem TLB aus. **2 pt**

What general tasks does an operating system have to perform in order to implement page-based virtual address spaces? Assume a system with a hardware-managed TLB.

Lösung:

The operating system has to perform the following tasks, **(0.5 P)** *each:*

- *Allocation / management of page tables*
- *Implement a page fault handler*
- *Implement page allocation, loading, and replacement*
- *Implement address space switching*

c) Mit welcher Information lässt sich auf modernen Systemen das Working Set eines Prozesses einfach bestimmen? **0.5 pt**

What information can be used on modern systems to easily determine the working set of a process?

Lösung:

The working set is the set of pages accessed in the last delta time frame. If the hardware supplies a reference bit **(0.5 P)**, *the OS can periodically read and reset the bit to determine which pages have been accessed in the given scan interval.*

d) Beim ersten Zugriff auf eine Seite in einem System mit Demand Paging kommt es zu einem Seitenfehler. Warum? **0.5 pt**

The first time a page is accessed in a system with demand paging, a page fault occurs. Why?

Lösung:

In a system with demand paging, pages are allocated and filled only on the first access **(0.5 P)**. *To detect the first access, the page is marked invalid (no access) in the page table.*

Bei der anschließenden Behandlung des Seitenfehlers steht kein freier physischer Speicher zur Verfügung. Geben Sie stichwortartig die nötigen Schritte zur Auflösung des Seitenfehlers an. Es sollen dabei keine Prozesse beendet werden. Geben Sie notwendige Interaktionen mit der Hardware explizit an. **3.5 pt**

During the subsequent handling of the page fault, no free physical memory is available. Give the necessary steps to resolve the page fault without terminating processes. Explicitly state any necessary interactions with the hardware.

Lösung:

Necessary steps to do page replacement are, **(0.5 P)** *each:*

- *Find victim page*
- *Invalidate user mode references to the page frame (in all page tables)*
- *Write back contents, if necessary*
- *Flush/invalidate entries in TLB*
- *Load new contents into page frame*
- *Update mapping in current page table*
- *Restart instruction*

e) Erläutern Sie das grundlegende Prinzip eines Slab-Speicherallokators. **1 pt**

Explain the basic principle of a slab memory allocator.

Lösung:

The allocator forms a so called slab cache from one or multiple slabs, themselves being pages of contiguous physical memory **(0.5 P)**. *The cache is split into chunks of (fixed) equal size. Allocations can only be made with this chunk size* **(0.5 P)**.

Für welches Szenario eignet sich der Slab-Allokator besonders? Begründen Sie Ihre Antwort. **1 pt**

For which scenario is the slab allocator particularly suited? Justify your answer.

Lösung:

Slab allocators are commonly used for allocating (kernel) objects of the same size **(0.5 P)**. *This is because, (1) the allocator effectively minimizes fragmentation, which is usually a problem with many but small allocations. And (2) by reusing freed objects some initialization overhead may be saved* **(0.5 P)**.

f) Heutzutage werden Datenstrukturen häufig auf eine Länge von 64 Byte optimiert. Welche Bedeutung hat diese Zahl? **0.5 pt**

Nowadays, data structures are often optimized to a length of 64 bytes. What is the meaning of this number?

Lösung:

On many processors, 64 bytes is the cache line size (or a multiple thereof).

g) Welche Art von Cache Miss kann in einem vollassoziativen Cache nicht auftreten? Begründen Sie Ihre Antwort. **1 pt**

What type of cache miss cannot occur in a fully associative cache? Justify your answer.

Lösung:

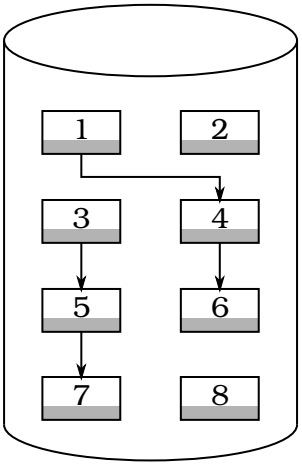
Fully associative caches are not susceptible to conflict misses **(0.5 P)**, *because in a conflict miss the requested entry has previously been removed from the selected set due to capacity reasons although there was room left in other sets. As fully associative caches posses a single set only, conflict misses cannot occur* **(0.5 P)**. **Total: 12.0pt**

Aufgabe 5: I/O, Hintergrundspeicher und Dateisysteme

Assignment 5: I/O, Secondary Storage, and File Systems

a) Die unten stehende Abbildung zeigt einen Sekundärspeicher mit einem Dateisystem, in dem zwei Dateien gespeichert sind. Welche Dateiallokationsstrategie kommt in dem abgebildeten Dateisystem zum Einsatz? **1 pt**

The figure below shows a secondary storage system with a file system that stores two files. Which file allocation strategy does the depicted file system use?



File Table

Name	Start	Length
FileA	1	3
FileB	3	3

Lösung:

Chained Allocation

Nennen Sie ein Zugriffsmuster, mit dem auf Dateien in dem Dateisystem effizient zugegriffen werden kann (1) und eines, mit dem der Zugriff langsam ist (2). **1 pt**

Name an access pattern to files that the file system can handle efficiently (1), and another with which access is slow (2).

Lösung:

efficient (1): sequential access **(0.5 P)**

slow (2): random access **(0.5 P)**

Nennen Sie eine Allokationsstrategie, mit der unabhängig vom Zugriffsmuster ein effizienter Zugriff möglich ist. Erklären Sie, wie der Dateizugriff funktioniert. **1.5 pt**

Name an allocation strategy that allows efficient access independent from the access pattern. Explain how file access works.

Lösung:

Contiguous Allocation **(0.5 P)***: The data is stored in consecutive blocks, so the operating system can calculate the needed block from the address of the first block and the access offset.* **(1 P)**

or

Indexed Allocation **(0.5 P)***: The file block addresses are stored in inodes. When accessing a file, the operating system looks up the inode and reads the addresses from there.* **(1 P)**

or

Linked List Allocation **(0.5 P)***: When accessing a file block, the operating system walks the File Allocation Table (which is a linked list of file blocks) in RAM.* **(1 P)** *Thus, the penalty for random accesses is not as large as with Chained Allocation.*

Im abgebildeten Dateisystem soll die effiziente Unterstützung von Hardlinks ergänzt werden. Beschreiben Sie, wie die *File Table* dafür verändert bzw. durch weitere Tabellen ergänzt werden muss. **2 pt**

The depicted file system should be extended with efficient support for hard links. Describe how the File Table *needs to be adapted and/or what tables need to be added.*

Lösung:

- *New column in File Table for the hard link count* **(0.5 P)**
- *Remove Name column from File Table* **(0.5 P)**
- *New "Hard Link Table" maps from link name to index in the File Table* **(1 P)**

b) Bei dem Adressierungsmodus *Cylinder-Head-Sector* (CHS) hat das Betriebssystem direkte Kontrolle über den physischen Speicherort auf Festplatten. Nennen und erklären Sie zwei Gründe, warum eine solche Adressierung bei SSDs nicht verwendet wird. **2 pt**

With the addressing mode Cylinder-Head-Sector (CHS), the operating system has direct control over the physical location of data on hard disks. Name and explain two reasons why SSDs do not offer such an addressing mode.

Lösung:

SSDs need to erase blocks before writing to them. They cannot erase individual blocks, but only pages composed of multiple blocks. With logical addressing, the SSD can redirect writes to previously-erased blocks.

SSDs have limited write durability. They employ wear leveling, distributing writes across physical blocks with the Flash Translation Layer. With physical addressing, the SSD could not implement wear leveling, which would greatly reduce its lifetime.

HDDs have potentially long seek times when move the read/write head to another track. CHS *allows the operating system to optimize HDD access by locating data on the disk in a way that minimizes seek times. SSDs do not have a similar penalty for random access, so there is no benefit from a* CHS*-like addressing mode.*

Welcher Adressierungsmodus kommt bei modernen Festplatten und SSDs stattdessen zum Einsatz? **0.5 pt**

Which addressing mode is used on modern hard disks and SSDs instead?

Lösung:

Logical Block Addressing (LBA)

c) Sowohl Journaling-Dateisysteme als auch Log-Structured-Dateisysteme dokumentieren Änderungen in einer Log-Datenstruktur. Wie unterscheidet sich die Nutzung des Logs bei diesen Dateisystemarten? **1 pt**

Both journaling file systems and log-structured file systems store updates in a log data structure. How does the usage of the log differ between these two types of file systems?

Lösung:

In a journaling file system, the log only stores data temporarily to protect file system integrity in case of crashes. In a log-structured file system, the log serves as permanent storage location for all data in the file system.

***Note:** In both file system types, the log ensures that the file system keeps a consistent state in case of crashes.*

d) Erklären Sie den Unterschied zwischen Mandatory- und Advisory-Locking. **1 pt**

Explain the difference between mandatory locking and advisory locking.

Lösung:

With mandatory locking, the operating system denies all access to locked files for applications that do not hold the lock. With advisory locking, the application is responsible to check for any locks with dedicated system calls.

e) Beschreiben Sie einen Vor- und einen Nachteil der *Virtual File System* (VFS)-Ebene im Betriebssystem. **2 pt**

Describe an advantage and a disadvantage of the Virtual File System (VFS) layer in the operating system.

Lösung:

Advantages:

- *The VFS layer offers a uniform interface to all file system types.*
- *The VFS layer allows combining multiple file systems in a single namespace (mounting).*

Disadvantage: By abstracting over different file system types, the VFS layer might not allow access to special optimized operations of some file systems. (Example: copy-on-write links in log-structured file systems)

***Note:** The VFS layer is not a hardware abstraction. Operating systems usually implement a block device layer for this purpose that the file systems can use.* **Total: 12.0pt**