Musterlösung Nachklausur

13.09.2017

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 14 Blättern: Einem Deckblatt und 13 Aufgabenblättern mit insgesamt 5 Aufgaben.
  The examination consists of 14 pages: One cover sheet and 13 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.
- Wenn Sie zusätzliches Konzeptpapier benötigen, verständigen Sie bitte die Klausuraufsicht.
  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 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) Nennen Sie zwei allgemeine Aufgaben von Abstraktionen in einem Betriebssystem. Geben Sie je eine Beispielabstraktion sowie die passende Hardwareressource an. **2 pt**

Give two general purposes of abstractions in an operating system. For each one, provide an example abstraction and the corresponding hardware resource.

Lösung:

- *Simply interface to the hardware by hiding implementation details* **(0.5 P)**.
  Example: Files → Disk Storage **(0.5 P)**.
- *Multiplex hardware to multiple processes* **(0.5 P)**.
  Example: Processes/Threads → CPU **(0.5 P)**.

b) Diskutieren Sie Vor- und Nachteile von Interrupts gegenüber Polling. In welcher Situation ist Polling die bessere Wahl? Begründen Sie Ihre Antwort. **3 pt**

Discuss pros and cons of interrupts over polling. In which situation is polling the better choice? Justify your answer.

Lösung:

With polling the CPU constantly has to probe device registers for detecting events such as I/O completions **(0.5 P)**. *As long as no events occur, this method wastes CPU time* **(0.5 P)**. *While reducing the polling frequency (i.e., perform other work in between checks) preserves some CPU time, it also increases latency* **(0.5 P)**. *With interrupts, the CPU does not have to poll device registers, but instead it actively receives a signal as soon as its attention is required* **(0.5 P)**.

However, interrupts are computationally expensive as they imply a context switch **(0.5 P)**. *They are thus not particularly well suited for scenarios, where the CPU is predominantly processing events anyways and interrupts introduce unnecessary overhead* **(0.5 P)**.

c) Der Kernel verfügt in der Regel über einen eigenen vollständigen Adressraum. **0.5 pt**

The kernel usually possesses a full address space of its own.

Lösung:

☐ *Ja / Yes* [✓ ■ *Nein / No*]

d) Nennen Sie ein Anwendungsbeispiel für ein Betriebssystem *ohne* Trennung in Benutzer- und Kernelmodus. Begründen Sie Ihre Antwort. **1.5 pt**

Give a usage example for an operating system without *separation in user and kernel mode. Justify your answer.*

Lösung:

In small embedded systems **(0.5 P)** *a separation in user and kernel mode is often not necessary, because the system executes only a single application. This application will probably also perform frequent hardware accesses and one might not want to pay the overhead of privilege boundary crossings on a less potent hardware platform* **(1.0 P)**.

An alternative answer could putting forward operating systems for managed programming languages such as Singularity.

e) Skizzieren Sie die wichtigsten Schritte einer Anwendung und des Betriebssystems bei einem Systemaufruf. **3 pt**

Outline the most important steps by an application and the operating system for a system call.

Lösung:

1. *Place parameters (incl. system call number!) in registers and/or on stack* **(0.5 P)**.
2. *Execute `trap` instruction* **(0.5 P)**.
3. *Switch to kernel-mode stack and copy parameters from user stack to kernel stack* **(0.5 P)**.
4. *Look up and execute selected system service routine* **(0.5 P)**.
5. *Place result in register or on (user) stack and switch back to user stack* **(0.5 P)**.
6. *Return to user mode* **(0.5 P)**.

f) Erläutern Sie, warum der Stack den Heap nicht ersetzen kann. **2 pt**

Explain why the stack cannot replace the heap.

Lösung:

Although the stack can provide memory for variables and data structures just like the heap, the lifetime of data on the stack is inherently bound to the CPU's execution context **(1.0 P)**. *If the CPU leaves a function, which stored data on the stack, the data can no longer be safely accessed and should be considered dead. The heap, on the other side, decouples lifetime and execution context* **(1.0 P)**.

Total: 12.0pt

Aufgabe 2: Prozesse und Threads

Assignment 2: Processes and Threads

a) Betrachten Sie ein Betriebssystem, das jedem Prozess eine 16 Bit Prozess-ID (PID) zuweist. Das System bestimmt eine eindeutige PID, indem es beim Erstellen eines Prozesses einen globalen Zähler atomar inkrementiert und den neuen Zählerwert als PID verwendet.

Erklären Sie, warum dieses System in der Praxis Probleme verursachen würde. **1 pt**

Consider an operating system, which assigns a 16 bit process ID (PID) to each process. On process creation, the system determines a unique PID by atomically incrementing a global counter and using the new counter value as PID.

Explain why this system would cause problems in practice.

Lösung:

With a PID of 16 bits, a maximum of 65536 processes can be created before the counter overflows. **(0.5 P)** *Once this happens, the system will start to reuse previously used PIDs, possibly even those of existing processes.* **(0.5 P)**

Wie könnte die PID-Vergabe angepasst werden? **1 pt**

How could the PID allocation be adjusted?

Lösung:

The system must find the next unused *PID instead of simply incrementing the counter,* **(1.0 P)** *for example by incrementing the counter repeatedly until reaching a PID that is not currently assigned to a process.*

b) Welches der in der Vorlesung vorgestellten Threadmodelle würden Sie für eine I/O-intensive Anwendung verwenden? Begründen Sie Ihre Antwort. **2 pt**

Which one of the thread models presented in the lecture would you choose for an I/O-intensive application? Justify your answer.

Lösung:

The one-to-one model **(1.0 P)***, because I/O-intensive applications can be expected to block frequently. Since all threads are known to the kernel in the one-to-one model, it is possible to block only those threads actually performing I/O operations.* **(1.0 P)**

In the many-to-one model, all user-level threads would be blocked as soon as one of them performs I/O, because the I/O operation would block the underlying kernel-level thread.

The many-to-many model suffers from the same problem, albeit to a lesser extent: Each user-level thread performing I/O would block one kernel-level worker thread, which is then unavailable to the user-level scheduler until the I/O operation completes. If a sufficient number of user-level threads perform I/O simultaneously, it is possible to block all kernel-level worker threads, leaving the other user-level threads unable to execute even though they are not waiting for I/O.

c) Erklären Sie den Begriff *Long-Term Scheduling* im Kontext der Vorlesung. **1 pt**

Explain the term long-term scheduling *in the context of the lecture.*

Lösung:

Long-term scheduling is used to decide at what time processes are started in a system. In other words, the long-term scheduler decides which (and how many) processes are visible to the short-term scheduler. **(1.0 P)**

d) Erklären Sie den Unterschied zwischen einem Scheduler und einem Dispatcher. **2 pt**

Explain the difference between a scheduler and a dispatcher.

Lösung:

A dispatcher implements the mechanism *to switch threads. It is given a thread to switch to, and then performs the tasks necessary to make that thread run on the CPU, such as saving the old thread's context to memory and restoring the new thread's context.* **(1.0 P)**

In contrast, a scheduler implements the scheduling policy*. It is thus responsible for selecting the next thread to run after the previous thread has exhausted its time slice or blocked for I/O. After making a selection, the scheduler typically calls into the dispatcher, passing an identifier for the selected thread.* **(1.0 P)**

e) Diskutieren Sie Vor- und Nachteile von kurzen gegenüber langen Zeitscheiben bei präemptivem Round-Robin Scheduling (nicht Virtual Round-Robin). Nutzen Sie dabei die in der Vorlesung vorgestellten Bewertungskriterien für Schedulingverfahren. **3 pt**

Discuss advantages and disadvantages of short over long time slices in preemptive round-robin scheduling (not virtual round-robin). Use the criteria presented in the lecture for assessing scheduling policies.

Lösung:

- *Pro: improved interactivity* **(0.5 P)**
- *Pro: shorter response time* **(0.5 P)**
- *Pro: Improved fairness for I/O-bound jobs* **(0.5 P)**
- *Contra: increased turnaround time* **(0.5 P)**
- *Contra: increased scheduling and dispatching overhead* **(0.5 P)**
- *Contra: decreased throughput* **(0.5 P)**

f) Erläutern Sie das Konzept der *Prioritätsvererbung (Priority Inheritance).* **2 pt**

Explain the concept of priority inheritance.

Lösung:

Priority inheritance (or priority donation) can be useful if a high-priority task A depends on the completion of a low-priority task B **(1.0 P)***, in which case A donates its priority to B so B completes faster* **(1.0 P)***. Otherwise, A effectively gets the lower priority of B, because A cannot run, until B has finished (i.e., the scheduler selected the low priority task).*

Total: 12.0pt

Aufgabe 3: Koordination und Kommunikation von Prozessen

Assignment 3: Process Coordination and Communication

a) Welche der Bedingungen für einen Deadlock wird nicht erfüllt, wenn sämtliche Ressourcen sortiert und ausschließlich in dieser Reihenfolge alloziert werden? Begründen Sie Ihre Antwort. **1 pt**

Which of the requirements for a deadlock is negated by ordering all resources and always acquiring them in this order? Justify your answer.

Lösung:

This breaks circular wait **(0.5 P)**, *because a cycle in the resource acquisition graph would require at least one process to acquire a resource with a lower order than the resource it already holds* **(0.5 P)**.

b) Warum lässt sich in der Praxis Deadlock Avoidance nur selten umsetzen? **1 pt**

Why is it in practice usually impossible to implement deadlock avoidance?

Lösung:

Deadlock avoidance is often impossible to implement, because it requires information about future resource requests in order to restrict resource allocation **(1.0 P)**.

c) Nennen und beschreiben Sie jeweils kurz die beiden in der Vorlesung vorgestellten Möglichkeiten, ein System nach einem aufgetretenen Deadlock zu reparieren (*Deadlock Recovery*). **2 pt**

*List and briefly describe the two techniques presented in the lecture to repair a system after a deadlock has occurred (*deadlock recovery*).*

Lösung:

- *Process Termination* **(0.5 P)**: *Abort processes until the deadlock cycle is eliminated* **(0.5 P)**.
- *Resource Preemption* **(0.5 P)**: *Roll back processes to a safe state if they hold resources involved in the deadlock* **(0.5 P)**.

d) Worin unterscheiden sich direktes und indirektes Message Passing? **1 pt**

What is the difference between direct and indirect message passing?

Lösung:

- *Direct message passing: Communication operations are always targeted at a specific process directly* **(0.5 P)**.
- *Indirect message passing: Communication operations write to and read from a mailbox* **(0.5 P)**, *which increases flexibility and allows the communication partners to change.*

Sind POSIX Message Queues in diesem Sinne direkt oder indirekt? **0.5 pt**

In this sense, are POSIX message queues direct or indirect?

Lösung:

Indirect **(0.5 P)***: The message queue itself is the mailbox.*

e) Warum werden bei Message Passing keine Puffer benötigt, wenn blockierende `send`-Operationen verwendet werden? **1 pt**

Why does message passing not require any buffers if blocking `send` operations are used?

Lösung:

Because messages are then only sent while both sender and receiver are executing the corresponding send/receive operations **(0.5 P)**, *so the data can be copied directly* **(0.5 P)**.

f) Erklären Sie den Unterschied zwischen schwachen und starken Semaphoren. Welche der Anforderungen an Synchronisierungsprimitive wird durch starke Semaphoren erfüllt, durch schwache aber nicht? **2 pt**

Explain the difference between weak and strong semaphores. Which of the requirements for synchronization primitives is met by strong semaphores, but not by weak ones?

Lösung:

Strong semaphores wake waiting threads up in the order in which the thread started waiting **(0.5 P)**, *whereas weak semaphores wake up waiting threads in random order* **(0.5 P)**. *Therefore, strong semaphores meet the requirement of bounded waiting, whereas weak semaphores do not* **(1.0 P)**.

g) Die folgenden Codeabschnitte werden parallel auf zwei verschiedenen Kernen eines Systems ausgeführt. `shared` zeigt in beiden Fällen auf den selben Speicher (Shared Memory). Dieser Speicher sei zu Beginn der Ausführung mit Nullen initialisiert.

Warum kann es passieren, dass auf aktuellen Prozessoren `do_work()` mit `0`, anstelle dem von `generate_input()` generierten Wert, aufgerufen wird? Erklären Sie schrittweise, wie es zu der fehlerhaften Ausführung kommt. **2 pt**

The following code sections execute on two different cores of a system in parallel. `shared` points to the same memory in both cases (shared memory). At the beginning of the execution, this memory is initialized with zeros.

Why is it possible that on modern processors `do_work()` is called with `0`, instead of the value generated by `generate_input()`? Explain step by step what leads to the erroneous execution.

Prozess 1:

```
1 int *shared;
2 // [...]
3 shared[0] = generate_input();
4
5 // signal other process
6 // that data is available
7 shared[100] = 1;
```

Prozess 2:

```
1 int *shared;
2 // [...]
3 // wait for available data
4 while(shared[100] == 0)
5     ;
6 // fetch and process data
7 do_work(shared[0]);
```

Lösung:

On some processor architectures, the processor is allowed to reorder memory operations **(1.0 P)***:*

In process 1, the first memory access can take effect after the second **(0.5 P)**, *so process 2 can execute line 7 and read invalid data even before line 3 of the first process takes effect* **(0.5 P)**.

Mit welcher Maßnahme lässt sich das Problem beheben, ohne dass Synchronisierungsprimitive wie Spinlocks, Mutexes oder Semaphoren eingesetzt werden? Begründen Sie Ihre Antwort. **1.5 pt**

What measures can be taken to solve the problem without using synchronization primitives such as spinlocks, mutexes, or semaphores? Justify your answer.

Lösung:

Memory fences **(0.5 P)** *can be used to prevent the reordering of the two write accesses* **(0.5 P)**, *so that `shared[100]` is only modified after `shared[0]` has been written* **(0.5 P)**.

Total: 12.0pt

Aufgabe 4: Speicher

Assignment 4: Memory

a) Erläutern Sie das Konzept von Base- und Limitregistern. Wie können damit mehrere Prozesse im Speicher isoliert werden? **1.5 pt**

Explain the concept of base and limit registers. How can they be used to isolate multiple processes in memory?

Lösung:

The base register is added to the virtual address and thus restricts access to physical addresses greater or equal to the base register **(0.5 P)**. *The limit register holds the length of the permissible memory region* **(0.5 P)**. *An access outside this area causes a page fault. To isolate multiple processes, the base and limit registers need to be configured on each context switch to cover the current process's physical memory only* **(0.5 P)**.

Welche Vor- und Nachteile hat dieser Ansatz gegenüber seitenbasiertem virtuellen Speicher? **1.5 pt**

What are the advantages and disadvantages over page-based virtual memory?

Lösung:

- *Pro: Easy to implement and fast to evaluate* **(0.5 P)**.
- *Con: Requires contiguous memory allocation (i.e., hard to grow, fragmentation problems, etc.)* **(0.5 P)**.
- *Con: Sharing data is not practical* **(0.5 P)**.

b) Welche Informationen benötigt das Betriebssystem bei einem Seitenfehler von der CPU? Geben Sie jeweils an, wozu diese Information nötig ist. **1.5 pt**

What information does the operating system need from the CPU on a page fault? Specify the purpose for which this information is required.

Lösung:

Virtual Address *Required to determine page table entry (PTE)* **(0.5 P)**.

Access Type *This might be* `read`, `write` or `instruction fetch`. *Required to decide if page fault can be resolved (e.g., copy-on-write) or the access should be denied* **(0.5 P)**.

Instruction Pointer *Required for re-starting the instruction after the fault has been resolved* **(0.5 P)**.

c) Gegeben sei ein System mit 8 GiB physischem Speicher und 35 bit virtuellen Adressen. Die Seitengröße beträgt 8 KiB, jeder Seiteneintrag benötigt 4 Bytes.

Ein Prozess alloziert 1 GiB Speicher. Berechnen Sie für jede angegebene Tabellenart, wieviel Speicher für die Seitentabelle des Prozesses mindestens nötig ist. **4 pt**

Consider a system with 8 GiB physical memory and 35 bit virtual addresses. The page size is 8 KiB, a page table entry requires 4 bytes.

A process allocates 1 GiB of memory. For each specified page table type, calculate how much memory is at least required for the page table of the process.

Lösung:

Linear Page Table
Linear page table depends only on virtual address space (VAS) size.
Number of pages in VAS: $\frac{2^{35}}{2^{13}} = 2^{22}$
Total: $2^{22} * 2^2\,Byte = \boxed{2^{24}\,Byte}$ **(1.0 P)**

Hierarchical Page Table
Number of pages: $\frac{2^{30}\,Byte}{2^{13}\,Byte} = 2^{17}$
Number of PTEs per page: $\frac{2^{13}\,Byte}{2^{2}\,Byte} = 2^{11}$
Number of page tables: $\frac{2^{17}}{2^{11}} = 2^{6}$
We need 11 bits for each level to address all PTEs in one page table.
Splitting of VA: 13 bit Offset + 11 bit 2-level Index + 11 bit 1-level Index = 35 bits
We thus have a hierarchical page table with two layers.
Total: $(1 + 2^6) * 2^{13}\,Byte = \boxed{2^{13}\,Byte + 2^{19}\,Byte}$ **(2.0 P)**

Inverted Page Table
For the inverted page table, we have one PTE for each physical frame.
Number of frames: $\frac{2^{33}}{2^{13}} = 2^{20}$
Number of page tables: $\frac{2^{20}}{2^{11}} = 2^{9}$
Total: $2^9 * 2^{13}\,Byte = \boxed{2^{22}\,Byte}$ **(1.0 P)**

Erklären Sie für die drei Seitentabellenarten, wie sich der Speicherverbrauch durch Seitentabellen verändert, wenn ein zweiter Prozess gestartet wird, der ebenfalls 1 GiB alloziert. **1 pt**

Explain for each page table type how the memory consumption of page tables changes, if a second process is started, which also allocates 1 GiB.

Lösung:

For the first and second type of tables, a full new set of page tables is required, the memory consumption is thus doubled **(0.5 P)**. *Since there is only a single inverted page table for the entire physical memory in the system, the memory consumption stays constant for inverted page tables* **(0.5 P)**.

d) Beschreiben Sie einen Ansatz mit dem die Anzahl der TLB-Misses nach einem Adressraumwechsel reduziert werden kann. **1 pt**

Describe an approach that reduces the number of TLB misses after an address space switch.

Lösung:

The TLB can be extended with address space identifiers **(0.5 P)**. *Each TLB entry is tagged with an identifier that maps it to the virtual address space in which the TLB entry is valid. This makes flushing the TLB on context switches unnecessary* **(0.5 P)**.

e) Welche Seiten bilden das Working-Set eines Prozesses? **0.5 pt**

Which pages make up the working-set of a process?

Lösung:

The pages that have been accessed within the last measurement interval **(0.5 P)**.

f) Beschreiben Sie eine Situation, bei der es mit dem Buddy Allokator zu externer Fragmentierung kommt. **1 pt**

Describe a situation in which the buddy allocator suffers from external fragmentation.

Lösung:

After all chunks of a size n have been allocated, every second chunk (i.e., one of each pair of buddies) is released. Since the released chunks are not contiguous they cannot be merged and used for requests larger than n, even if enough total free space is available **(1.0 P)**.

Total: 12.0pt

Aufgabe 5: I/O, Hintergrundspeicher und Dateisysteme

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

a) Auf den relative Pfad `../../asdf/./jkl` wird vom Verzeichnis `/a/b/c/` ausgehend zugegriffen. Geben Sie den absoluten und so weit wie möglich gekürzten Pfad an. **0.5 pt**

The relative path `../../asdf/./jkl` is accessed from within the directory `/a/b/c/`. Give the absolute path, without any unnecessary elements.

Lösung:

`/a/asdf/jkl` **(0.5 P)**

b) Erläutern Sie den Unterschied zwischen *Shared File Locks* und *Exclusive File Locks*. **1 pt**

Explain the difference between shared file locks *and* exclusive file locks.

Lösung:

Exclusive file locks can only be acquired by one reader **(0.5 P)**. *Shared file locks can be acquired by multiple readers (or one writer)* **(0.5 P)**.

c) Welchen Einfluss auf die Performance kann es bei einer klassischen Festplatte haben, sehr häufig genutzte Daten (zum Beispiel Inode-Tabellen) auf den mittleren Zylindern zu speichern anstatt auf den inneren oder äußeren Zylindern? Erklären Sie kurz, wodurch dieser Effekt entsteht. **1.5 pt**

On conventional hard disks, what impact on performance can placing very commonly required data (e.g., inode tables) on the center cylinders of the disk have (as opposed to placing the data on inner or outer cylinders)? Explain briefly how this effect is caused.

Lösung:

The average seek time is reduced **(0.5 P)***: The seek time depends on the distance traveled by the head* **(0.5 P)** *This distance is short because the center cylinders are on average closer to all other cylinders of the disk* **(0.5 P)** *than e.g. the inner or outer cylinders.*

Alternative: The head is parked near the outer edge of the platters **(0.5 P)**. *Therefore, it needs to travel longer distances whenever the disk is first accessed after a period of inactivity, which increases the seek time* **(1 P)**.

d) Was ist *Spooling*? Geben Sie ein Beispielgerät an, für das Spooling verwendet wird. **1 pt**

What is spooling*? Give an example device for which spooling is used.*

Lösung:

Spooling means that the system holds back output for a device while that device is busy executing another request **(0.5 P)**. *Spooling is necessary for devices which can only serve one request at a time (e.g., a printer)* **(0.5 P)**.

e) Ein System verarbeitet den Datenstrom eines I/O-Geräts, bestehend aus vielen gleich großen Datenpaketen. Das System benötigt 5 ms, um ein Datenpaket von dem I/O-Gerät in den Puffer einer Anwendung zu kopieren und anschließend 3 ms, um das Datenpaket in der Anwendung zu verarbeiten.

Berechnen Sie näherungsweise den Speedup, wenn man einen zusätzlichen Systempuffer hinzufügt (*Single Buffering*). Die Daten gelangen dabei zuerst per DMA vom I/O-Gerät in den Systempuffer (5 ms) und werden in den Puffer der Anwendung kopiert (1 ms), sobald diese das vorherige Datenpaket verarbeitet hat (3 ms).

2 pt

A system processes the data stream of an I/O device, consisting of many data packets of equal size. The system requires 5 ms to copy a data packet from the I/O device into the buffer of an application and then 3 ms to process the packet in the application.

*Calculate the approximate speedup when an additional system buffer is added (*Single Buffering*). The data is first transferred from the I/O device into the system buffer via DMA (5 ms), and is copied into the application's buffer (1 ms), as soon as the application finished processing the previous packet (3 ms).*

Lösung:

The transfer from the I/O device to the system buffer (DMA) and the processing by the application can run in parallel **(0.5 P)**. *Therefore, the speedup is:*

$$\frac{T+P}{\max\{T,P\}+C} = \frac{5+3}{\max\{5,3\}+1} = \frac{4}{3}$$

(1.0 P) *for the formula,* **(0.5 P)** *for the result.*

f) Welches Problem kann durch DMA beim Ersetzen von Seiten auftreten, und wie kann diese Situation verhindert werden?

1.5 pt

Which problem can be caused by DMA during page replacement, and how can this situation be prevented?

Lösung:

DMA operates on physical addresses, so the DMA controller will happily write into frames even after the corresponding page table entries have been invalidated **(1.0 P)**. *This situation can be prevented by pinning all DMA targets into physical memory* **(0.5 P)**.

g) Ein Programm hängt Daten an eine Datei, auf welche mehrere Hardlinks zeigen. Warum ist es hierbei sinnvoll, dass Attribute wie die Dateigröße nicht im Verzeichniseintrag, sondern im Inode gespeichert werden? **1 pt**

A program appends data to file, which has multiple hard links. Why is it advantageous to store attributes like the file size not in the directory entry, but instead in the inode?

Lösung:

If the file size was stored in the directory entry, then all directory entries of all hard links would have to be visited and modified, which is more expensive than changing the single inode of the file **(1.0 P)**.

h) Beschreiben Sie für jede der drei Allokationsstrategien *Contiguous Allocation*, *Chained Allocation* und *Indexed Allocation* ein Szenario, für welches die Strategie besonders geeignet ist. Begründen Sie Ihre Antwort. **3 pt**

For each of the three allocation strategies contiguous allocation, chained allocation, *and* indexed allocation, *describe a scenario for which the strategy is particularly well suited. Justify your answer.*

Lösung:

Contiguous Allocation *Suited if data is only written once (e.g., when creating read-only media such as DVDs)* **(0.5 P)**. *Strategy is prone to fragmentation and cannot cope well with changing file sizes. Files should therefore be static (i.e., read-only)* **(0.5 P)**.

Chained Allocation *: Suitable if data is only linearly accessed (e.g., for video or audio files)* **(0.5 P)**. *Random access is very slow, because the reader must walk the chain to find a certain offset in the file* **(0.5 P)**.

Indexed Allocation *Suitable whenever good random access performance is required* **(0.5 P)**. *The index blocks allow very fast mapping of file offsets to blocks on disk* **(0.5 P)**.

i) Welche Schreibstrategie wird üblicherweise in einem Dateicache verwendet? **0.5 pt**

Which write policy is typically used in a file cache?

Lösung:

☐ *Write-Through* ■ *Write-Behind*

Total: 12.0pt