Karlsruher Institut für Technologie

Betriebssysteme
(Operating Systems)

Prof. Dr. Frank Bellosa
Dipl.-Inform. Marc Rittinghaus

Musterlösung Hauptklausur
10.03.2017

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 15 Blättern: Einem Deckblatt und 14 Aufgabenblättern mit insgesamt 5 Aufgaben.
  The examination consists of 15 pages: One cover sheet and 14 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) Skizzieren Sie die wichtigsten Schritte bei der Behandlung eines Interrupts durch die CPU und das Betriebssystem. **2 pt**

Outline the most important steps during interrupt handling by the CPU and the operating system.

Lösung:

1. *Lookup: The CPU performs a lookup in the interrupt vector to identify the service routine* **(0.5 P)**
2. *Entry: The CPU calls the service routine. The routine is expected to save the current CPU context* **(0.5 P)**.
3. *Execution: The service routine in the OS handles the interrupt. This may include communicating with devices (e.g., reading device memory)* **(0.5 P)**.
4. *Exit: The CPU returns from the service routine. The routine is expected to restore the previous CPU context* **(0.5 P)**.

b) Nennen Sie zwei privilegierte CPU-Operationen auf x86 und erklären Sie jeweils, warum diese privilegiert sein müssen. Eine Angabe der Assemblermnemonics ist nicht nötig. Inverse Operationen zählen nicht. **2 pt**

Give two privileged CPU operations on x86 and explain why each of them needs to be privileged. You do not need to provide the assembler mnemonics. Inverse operations do not count.

Lösung:

- *Deactivate interrupts (`CLI`)* **(0.5 P)***: Timer triggers via interrupt. If the user could disable interrupts, she could interfere with scheduling* **(0.5 P)**.
- *Set CR3 register* **(0.5 P)***: The CR3 register controls which address space is currently active. Allowing the user to modify the address space breaks memory isolation* **(0.5 P)**.

c) Treten CPU-Ausnahmen synchron zum Programmfluss auf? **0.5 pt**

Do CPU exceptions occur synchronously with the program flow?

Lösung:

■ *Ja / Yes* □ *Nein / No*

d) Betrachten Sie ein Betriebssystem, in dem Prozesse und der Kernel jeweils dedizierte, vollständige Adressräume besitzen. Bei Ausführung der Trap-Instruktion wird automatisch in den Adressraum des Kernels sowie auf den jeweiligen Kernelstack gewechselt.

Consider an operating system in which processes and the kernel each posses a dedicated, full address space. When executing the trap instruction, the system automatically switches to the kernel's address space and to the respective kernel stack.

Ein Systemaufruf erfordert mehr Argumente, als es CPU-Register gibt. Erläutern Sie, wie die Parameterübergabe in diesem System funktionieren kann. **1.5 pt**

A system call requires more arguments than there are CPU registers available. Explain how the parameters could be passed in this system.

Lösung:

Since there are not enough registers, at least some parameters have to be passed via the stack. However, the stack is only accessible in the process's address space. To access the arguments in the kernel we therefore have to:

1. *Save the user stack pointer in a selected register* **(0.5 P)**
2. *Map the user stack into the kernel address space* **(1 P)**
3. *Copy the arguments from the user stack to the kernel stack* **(0.5 P)** *using the saved (adjusted) user stack pointer*

Nennen Sie einen Vor- und einen Nachteil von einem dedizierten Kerneladressraum auf 32 Bit Systemen. Wie bewerten Sie dieses Design auf 64 Bit Systemen? **1.5 pt**

Give an advantage and a disadvantage of a dedicated kernel address space on 32 bit systems. How do you rate this design on 64 bit systems?

Lösung:

Pro *Having dedicated address spaces instead of sharing a single virtual address space between the process and the kernel, provides more space for both parties* **(0.5 P)**.

Contra *Jumping between user and kernel mode includes a full address space switch, which is expensive* **(0.5 P)**. *It also makes parameter passing more complex.*

While size limits can be a severe problem on 32 bit systems, the virtual address space on 64 bit systems 'ought to be enough for anybody'™. The advantages of sharing then clearly outweigh the disadvantages **(0.5 P)**.

e) Nennen Sie vier Sektionen, die üblicherweise in ELF-Dateien repräsentiert sind. Erläutern Sie jeweils kurz die Bedeutung. **4 pt**

Give four sections that are usually represented in ELF files. Shortly explain each section's purpose.

Lösung:

Code *The executable program code* **(1 P)**.

RW Data *Pre-initialized data, which can be modified during execution* **(1 P)**.

RO Data *Pre-initialized data, which is read-only and should not be modified* **(1 P)**
The operating system marks these pages as read-only in the page tables.

BSS *Not initialized writable data area. The section will be all zeros at startup* **(1 P)**.
The section does not take up any space in the ELF file.

f) Nennen Sie einen Adressraumbereich, der nicht in ELF-Dateien repräsentiert ist. **0.5 pt**

Give an address space area that is not represented in an ELF file.

Lösung:

Stack or heap **(0.5 P)**. **Total: 12.0pt**

Aufgabe 2: Prozesse und Threads

Assignment 2: Processes and Threads

a) Erklären Sie den Unterschied zwischen einem Programm und einem Prozess. **1 pt**

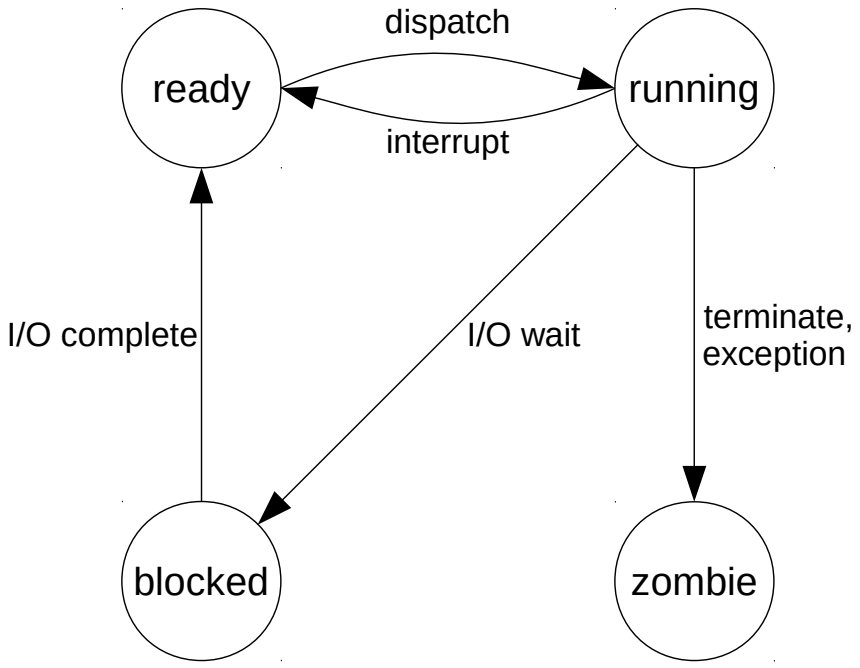
Explain the difference between a program and a process.

Lösung:

A program is a sequence of instructions that accomplish a given task **(0.5 P)**. *A process is an instance of a running program* **(0.5 P)**.

b) Nehmen Sie an, dass ein Betriebssystem die vier Prozesszustände "bereit" ("ready"), "rechnend" ("running"), "blockiert" ("blocked") und "zombie" unterstützt. Stellen Sie grafisch dar, zwischen welchen Zuständen Übergänge möglich sind und beschriften Sie jede Kante mit einem Ereignis, das den jeweiligen Übergang auslöst. **2.5 pt**

Consider an operating system, which supports the four process states "ready", "running", "blocked", and "zombie". Depict the possible state transitions and label each edge with an event causing the transition.



(0.5 P) per edge if labeled correctly **(-0.5 P)** per incorrect edge.

c) Erläutern Sie die Funktion des *PCB* und des *TCB*. Geben Sie dabei je zwei Einträge an, die in der jeweiligen Struktur gespeichert werden. **2 pt**

Explain the purpose of the PCB *and the* TCB*. For each structure, give two pieces of information that are stored in it.*

Lösung:

Process Control Block (PCB) *The PCB is the kernel data structure to represent a process* **(0.5 P)***. Two typical fields are: list of open files, structure to describe the address space* **(0.5 P)***.*

Thread Control Block (TCB) *The TCB is used by the kernel to represent a thread* **(0.5 P)***. Two typical fields are: pointers to stacks, thread execution state* **(0.5 P)***.*

d) Was versteht man unter einem *User-Level Thread (ULT)*? Welche aus der Vorlesung bekannten Threadmodelle verwenden ULTs? **1.5 pt**

What is a User-Level Thread (ULT)*? Which thread models presented in the lecture employ ULTs?*

Lösung:

A user-level thread is a thread that is entirely implemented (i.e., control structures, scheduling, dispatching, etc.) in user level **(0.5 P)***. Depending on the thread model, the kernel is not aware of user-level threads* **(0.5 P)***.*

ULTs are used in the many-to-one and the M-to-N thread models **(0.5 P)***.*

e) Ein Webserver verwendet einen statischen Pool von Prozessen, um Clientverbindungen parallel zu verarbeiten. Nennen Sie einen Vor- und einen Nachteil dieses Ansatzes gegenüber der Verwendung einer gleichen Anzahl von Threads. **2 pt**

A webserver uses a fixed pool of processes to handle connections from multiple clients in parallel. Give an advantage and a disadvantage of this scheme compared to using the same number of threads.

Lösung:

Using multiple processes, each worker has its own address space. As a result, a segmentation fault in one worker does not affect the others. When using multiple threads in the same address space, a segmentation fault in one thread crashes the entire server **(1 P)***. In case of a security vulnerability, a separate process for each connection also prevents leaking data from different clients.*

On the downside, communication and coordination between worker processes is more complex than doing the same between threads in the same address space **(1 P)***.*

f) Linux enthält neben dem präemptiven MLFQ-basierten Scheduler einen nicht-präemptiven FCFS-Scheduler mit der Eigenschaft, dass FCFS-Threads immer Vorrang vor MLFQ-Threads haben. Prozesse mit Administratorrechten können entscheiden, unter welchem Scheduler ihre Threads laufen. Prozesse ohne Administratorrechte verwenden immer den MLFQ-Scheduler.

Besides the preemptive MLFQ-based scheduler, the Linux kernel also contains a non-preemptive FCFS scheduler. Threads scheduled by the FCFS scheduler always have priority over threads under the MLFQ scheduler. Processes with administrator privileges can choose which scheduler their threads use. Processes without administrator privileges always use the MLFQ scheduler.

Für welche Art von Anwendungen ist es sinnvoll, den FCFS-Scheduler zu verwenden? Begründen Sie Ihre Antwort. **2 pt**

For which type of application is the FCFS scheduler beneficial? Justify your answer.

Lösung:

The non-preemptive FCFS scheduler is suited for applications that execute mostly in short bursts **(0.5 P)** *and that must be handled with high urgency* **(0.5 P)**. *These are typically real-time applications.*

Short Bursts: If the application does not execute in short bursts, there is a danger of these applications starving the system **(0.5 P)**.

Urgency: Since applications under the FCFS scheduler are always preferred, the application should be time-critical **(0.5 P)**.

Warum darf der FCFS-Scheduler nur von Anwendungen mit Administratorrechten verwendet werden? **1 pt**

Why are only applications with administrator privileges allowed to use the FCFS scheduler?

Lösung:

It is easy for applications to abuse the non-preemptive FCFS scheduler: Since FCFS processes are never preempted, an application can easily hog the CPU by putting a CPU-bound thread into the FCFS scheduler **(1 P)**. *Therefore, untrustworthy users should not be allowed to use the FCFS scheduler without approval from an administrator.*

Total: 12.0pt

Aufgabe 3: Koordination und Kommunikation von Prozessen

Assignment 3: Process Coordination and Communication

a) Erläutern Sie kurz den Begriff *kritischer Abschnitt*. Warum müssen diese geschützt werden? **2 pt**

Briefly explain the term critical section. *Why do they need to be protected?*

Lösung:

Sections of programs where concurrent activities or threads **(0.5 P)** *access shared data* **(0.5 P)**.

Concurrent execution of critical sections by different threads can lead to wrong results and crashes **(0.5 P)**, *a failure called race condition* **(0.5 P)**.

b) Gegeben sei ein Einprozessorsystem, das alle Threads präemptiv im Kernelmodus ausführt. Welche der notwendigen Bedingungen für eine gültige Lösung des Problems kritischer Abschnitte werden erfüllt, wenn zum Schutz von kritischen Abschnitten Interrupts deaktiviert werden? Welche nicht? Erläutern Sie jeweils warum. **4.5 pt**

Consider a uni-processor system in which all threads are executing with preemption in kernel mode. Which of the requirements for a valid solution of the critical section problem does deactivating interrupts fulfill, which not? For each requirement explain why / why not.

Lösung:

Mutual Exclusion *Yes* **(0.5 P)**. *While interrupts are disabled, other threads are not scheduled (since the timer interrupt – which drives the scheduler – is disabled) and therefore cannot enter the critical section* **(1 P)**.

Progress *No* **(0.5 P)**. *Other threads are barred from entering the critical section if any thread deactivates interrupts for another reason (including entering a different critical section)* **(1 P)**.

Bounded Waiting *No* **(0.5 P)**. *Once interrupts are enabled again, there is no guarantee about which thread is allowed to enter the critical section next* **(1 P)**.

c) Betrachten Sie die folgende Implementierung eines Spinlocks. Funktioniert diese wie gewünscht? Falls ja, begründen Sie, warum. Falls nicht, geben Sie einen Ablauf an, der zu unerwünschtem Verhalten führt. **2.5 pt**

Consider the following implementation of a spinlock. Does it work as intended? If so, explain why. If not, describe a flow of execution that leads to undesired behavior.

```
1  bool locked = false
2
3  function acquire() {
4      retry:
5      if (locked == true)
6          goto retry
7
8      locked = true
9  }
10 function release() {
11     locked = false
12 }
```

Lösung:

No **(1 P)**. *Assume two threads calling* `acquire()`. *The first passes the check in line 5 and is preempted in line 7* **(0.5 P)**. *Then, the second thread also passes the check, because* `locked` *is not set to* `true`, *yet* **(0.5 P)**. *Now, both threads will successfully acquire the lock and enter the critical section* **(0.5 P)**.

d) Unter welcher Vorraussetzung ist die Verwendung von Spinlocks zur Synchronisation zweier Threads sinnvoll? **1 pt**

Under which condition does the use of spinlocks to synchronize two threads makes sense?

Lösung:

The threads are running on a multi-processor system **(0.5 P)** *and blocking is not a (viable) option* **(0.5 P)** *– e.g., short critical section or blocking in current context not possible.*

e) Gegeben sei ein System, das blockierende (d. h. synchrone) Interprozesskommunikation (IPC) unterstützt. Wie kann auf einem solchen System ein blockierendes Lock für beliebig viele parallele Threads mit IPC realisiert werden? **2 pt**

Consider a system that supports blocking (i.e., synchronous) inter-process communication (IPC). How can you realize a blocking lock with IPC for an arbitrary number of parallel threads?

Lösung:

The lock must be managed by a dedicated lock manager **(1 P)**. *Threads wanting to use the lock send an IPC to the lock manager* **(0.5 P)** *and wait for a reply. The lock manager then decides who gets to proceed, and sends an IPC back to that thread* **(0.5 P)**.

Total: 12.0pt

Aufgabe 4: Speicher

Assignment 4: Memory

a) Nennen Sie zwei Situationen, in denen das Betriebssystem einen Prozess bei einem Seitenfehler in der Regel beendet. **1 pt**

Give two situations in which the operating system usually terminates a process on a page fault.

Lösung:

- *The process accessed a page outside a valid virtual memory area* **(0.5 P)**.
- *The process accessed a page inside a valid virtual memory area, but the required access permissions are not granted (e.g., a write access to a read-only page)* **(0.5 P)**.

Note: Addresses outside of the address space do not exist, because the address space is inherently created by all representable addresses.

b) Erklären Sie, warum sich die LRU-Seitenersetzungsstrategie auf x86 nicht effizient umsetzen lässt. Durch welche Strategie kann LRU approximiert werden? Begründen Sie Ihre Antwort. **3 pt**

Explain why LRU page replacement cannot be efficiently done on x86. Which strategy can be used to approximate LRU? Explain your answer.

Lösung:

The least recently used (LRU) replacement strategy preserves recently accessed pages and selects the page that has not been accessed for the longest time **(0.5 P)**. *This approach requires some form of timestamp in page table entries, which must be updated by the MMU on each access. This feature is not present in x86* **(0.5 P)**. *Instead, there is only a single reference bit* **(0.5 P)**, *which can be used with the clock strategy* **(0.5 P)**. *With this strategy the OS periodically checks each page's reference bit. If the page has not been accessed (bit is not set), the algorithm picks the page for replacement. Otherwise, the bit is cleared and the clock inspects the next page, thereby preserving recently accessed pages* **(1 P)**.

c) Gegeben sei ein 4-fach satzassoziativer, virtuell-indizierter, physisch-getaggter (VIPT) Cache mit 16 KiB Kapazität und 64 Bytes pro Cachezeile. Das System verwendet 4 KiB Speicherseiten. Die Satznummer wird nach folgendem Prinzip aus der virtuellen Adresse ermittelt:

Consider a 4-way set-associative, virtually-indexed, physically-tagged (VIPT) cache with a capacity of 16 KiB and 64 bytes per cache line. The system uses 4 KiB pages. The set number is determined from the virtual address according to the following principle:

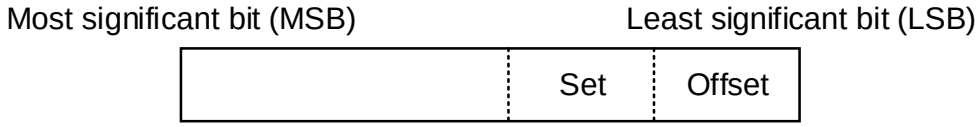


Abbildung 4.1 / *Figure 4.1*

Erläutern Sie warum der Cache auf diesem System und in dieser Konfiguration auch als physisch-indiziert, physisch-getaggt (PIPT) bezeichnet werden kann. **2.5 pt**

Explain why the cache can also be called physically-indexed, physically-tagged on this system and in this configuration.

Lösung:

The lowest 6 bits (64 bytes = 2^6) of the cache are used for the offset within a cache line **(0.5 P)**. *The cache has $\frac{16\,KiB}{64bytes} = \frac{2^{14}}{2^6} = 2^8$ cache lines. In each set there are 4 cache lines. We therefore have $\frac{2^8}{2^2} = 2^6$ sets and need 6 bits to address these sets* **(0.5 P)**. *The system uses $4\,KiB = 2^{12}bytes$ pages. The offset within the page will thus be expressed with the first 12 bits (from LSB)* **(0.5 P)**. *Exactly these 12 bits will also be used for addressing in the cache.*

Since the offset remains unchanged during the translation from the virtual to the physical address, the cache can also be called a PIPT cache **(1 P)**.

Wie kann die Kapazität des Caches erhöht werden, ohne diese Eigenschaft zu zerstören? Die Seitengröße soll dabei unverändert bleiben. Begründen Sie Ihre Antwort. **1 pt**

How can the capacity of the cache be increased without losing this property? The page size should remain unchanged. Explain your answer.

Lösung:

The capacity can only be increased by increasing the associativity of the cache. This way, the number of sets and the width of the offset is not altered, which preserves the property.

Leidet der Cache unter dem Problem der Mehrdeutigkeit (Ambiguity)? **0.5 pt**

Does the cache suffer from the ambiguity problem?

Lösung:

☐ *Ja / Yes* ■ *Nein / No*

d) Erläutern Sie die Begriffe *interne* und *externe Fragmentierung*. **2 pt**

Explain the terms internal *and* external fragmentation.

Lösung:

Internal Fragmentation *Internal fragmentation occurs, when the memory allocator returns more memory than is actually needed. The remaining memory is wasted* **(1 P)**. *This usually happens with allocators, which use allocation blocks of fixed size.*

External Fragmentation *External fragmentation can develop due to different life-times of memory allocations. It describes a situation where enough memory is available to satisfy a request, but it is not contiguous and therefore cannot be used* **(1 P)**.

e) Erläutern Sie, wie sich die Seitengröße auf die Trefferquote des TLB auswirkt. **1 pt**

Explain how the page size affects the hit rate of the TLB.

Lösung:

The TLB holds address translations on a per-page level. Since the number of TLB entries is limited, larger pages increase the reach of the TLB in the virtual address space. It is thus more likely that an access can be serviced from the TLB and the hit rate grows **(1 P)**.

f) Beschreiben Sie, warum es beim Demand-Paging zu einer erhöhten Anzahl von Seitenfehlern kommen kann. **1 pt**

Describe why demand paging may lead to an increased number of page faults.

Lösung:

With demand paging, pages are only loaded when they are actually needed, that is, when the process first accesses them **(0.5 P)**. *In consequence, every first access to a page will cause a page fault* **(0.5 P)**. *Pre-paging, in contrast, loads pages in advance and thereby can reduce the number of page faults.*

Total: 12.0pt

Aufgabe 5: I/O, Hintergrundspeicher und Dateisysteme

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

a) Nennen Sie die drei in der Vorlesung genannten Orte im Dateisystem, an denen der Typ von Dateien kodiert werden kann. **1 pt**

List the three places in the file system, where, according to the lecture, it is possible to encode file types.

Lösung:

- *File system structures (e.g., field in i-node)*
- *Name (e.g., extension)*
- *Content (e.g., magic number)*

(1 P) *if all three are mentioned,* **(0.5 P)** *if one option is missing or incorrect.*

b) Erklären Sie den Unterschied zwischen Mandatory und Advisory File Locks. **1 pt**

Explain the difference between mandatory and advisory file locks.

Lösung:

When a file is locked and a second process requests a lock, the behavior differs: Mandatory locks cause the second process's access to be denied **(0.5 P)**. *Advisory file locks only tell the second process that the file is locked, and the process can decide for itself whether to continue the access to the file* **(0.5 P)**.

c) Während Sektoren oft nur 512 Bytes umfassen, verwalten viele Dateisysteme den Speicherplatz in wesentlich größeren Blöcken (z. B. 4 KiB bei FAT32). Nennen Sie einen Vor- und einen Nachteil von größeren Blöcken fester Größe gegenüber kleineren Blöcken. **1 pt**

Sectors are usually 512 bytes large, whereas many file systems use blocks of significantly larger size (e.g., 4 KiB in FAT32). Give an advantage and a disadvantage of such larger blocks of fixed size compared to smaller blocks.

Lösung:

*Advantages (***(0.5 P)** *if one is mentioned):*

- *Fewer fragments result in fewer disk seeks.*
- *Fewer blocks result in smaller tables required to manage free space.*
- *Larger fragments result in more efficient disk operations.*

*Disadvantages (***(0.5 P)** *if one is mentioned):*

- *Larger blocks result in more internal fragmentation.*
- *Larger pages can result in larger transfers than necessary for small read/write operations.*

d) Gegeben sei ein I-Node mit vier Plätzen zur direkten Blockadressierung, einem Platz zur einfach-indirekten Blockadressierung und einem Platz zur zweifach-indirekten Blockadressierung. Jeder Block umfasst 128 Bytes, eine Blockadresse 8 Bytes.

Consider an i-node with four entries for direct block addressing, one entry for single-indirect block addressing, and one entry for double-indirect block addressing. A block is 128 bytes in size, a block address is 8 bytes long.

Wie groß können Dateien in dem gegebenen Dateisystem maximal werden? Begründen Sie Ihre Antwort. **4 pt**

What is the maximum file size in the specified file system? Explain your answer.

Lösung:

Each indirection block contains $\frac{128}{8} = 16$ **(0.5 P)** *references to other blocks. Each entry for single-indirect block adressing can therefore reference* 16 *blocks, and each entry for double-indirect block adressing can reference* $16 * 16 = 256$ *blocks. In total, the i-node can therefore reference* $4 + 16 + 256 = 276$ *blocks. The resulting maximum file size is:* **(0.5 P)** *for any correct explanation which shows the origin of the numbers below.*

$4 * 128\ bytes$ **(1 P)** $+ 16 * 128\ bytes$ **(1 P)** $+ 256 * 128\ bytes$ **(1 P)**
$= 2^9\ bytes + 2^{11}\ bytes + 2^{15}\ bytes$
$= 512\ bytes + 2\ KiB + 32\ KiB$
$= 35328\ bytes$

Wie kann das gegebene Dateisystem geändert werden, um die maximale Dateigröße zu erhöhen, wenn die Gesamtgröße des I-Nodes unverändert bleiben soll? **1 pt**

How can the specified file system be changed to increase the maximum file size, if the total size of the i-node must not be changed?

Lösung:

*Possible solutions (***(1 P)** *if a valid solution is given):*

- *Increase the block size.*
- *Increase the degree of indirection (e.g., by replacing one of the entries with an entry for triple-indirect block addressing).*
- *Reduce the size of block addresses to increase the number of references per indirect block.*

e) Nennen Sie eine Situation, in der es notwendig ist, dass bei einer Schreiboperation in eine Datei zuvor existierende Dateiinhalte der selben Datei vom Blockgerät gelesen werden müssen. Begründen Sie Ihre Antwort. **2 pt**

Describe a situation, where a write operation into a file requires the OS to first read existing file contents of the same file from the block device. Justify your answer.

Lösung:

If data is appended or replaced in the middle of a disk block **(1 P)**, *the existing contents of the block have to be read from the disk: Because disks are block devices, the whole updated block (combination of old and new content) has to be written back to the disk* **(1 P)**.

Alternative solutions:

- *Chained allocation* **(1 P)** *requires all preceding data blocks to be read to find the location where the written data is placed* **(1 P)**.
- *Flash memory requires whole blocks to be read, erased and rewritten* **(1 P)**, *even if only parts of the block have changed* **(1 P)**.
- *Efficient writes to a RAID 4/5 require the old block content to be read* **(1 P)** *in order to efficiently compute the new parity* **(1 P)**.
- *Write-allocate file system caches* **(1 P)** *require the data to be read into the file system cache before it is modified* **(1 P)**.
- *Copy-on-write* **(1 P)** *requires file content to be read on write operations because (parts of) the file are copied to a new location before the write operation* **(1 P)**.

f) Warum kann es dazu kommen, dass eine Leseoperation keine Zugriffe auf das darunter liegende Blockgerät verursacht? **1 pt**

Why do some file read operations trigger no accesses to the underlying block device?

Lösung:

*Most operating systems maintain a file system cache (buffer cache, page cache) which caches the content of the underlying block device. If operations hit the cache, they usually do not cause any block device access***(1 P)**.

Alternative solutions:

- *The I/O buffer of a previous read operation contains the data and is still in memory* **(1 P)**.
- *The caller does not have the appropriate access rights* **(0.5 P)** *to the file* **(0.5 P)**.

g) Muss in einem RAID 3 beim Schreiben eines einzelnen Blocks zwangsweise auf alle Festplatten des Verbunds zugegriffen werden? Begründen Sie Ihre Antwort. **1 pt**

When writing a single block in a RAID 3, do all disks in the array need to be accessed? Justify your answer.

Lösung:

Yes **(0.5 P)**, *due to byte interleaving: Each block is spread over all disks in the RAID, so all disks hold a part of the requested block* **(0.5 P)**. **Total: 12.0pt**