Karlsruher Institut für Technologie

Betriebssysteme
(Operating Systems)

Prof. Dr. Frank Bellosa
Mathias Gottschlag, M.Sc.
Dipl.-Inform. Marc Rittinghaus

Musterlösung Nachklausur

24.09.2018

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) Erklären Sie den Unterschied zwischen einem *Programm* und einem *Prozess*. **2 pt**

Explain the difference between a program *and a* process.

Lösung:

A program is a specification which is used to construct a process **(1.0 P)**. *It contains code to be loaded and the address space layout. A process is the activity of executing a program* **(1.0 P)**.

b) Nennen Sie ein Beispiel für eine Policy und einen dazugehörigen Mechanismus. **1 pt**

Give an example for a policy and a corresponding mechanism.

Lösung:

Examples:

- *Scheduling algorithm and dispatcher*
- *Moving memory pages to disk and the page replacement algorithm*

(0.5 P) *each for a valid policy and the corresponding mechanism.*

Erklären Sie die Begriffe *Policy* und *Mechanismus*. **2 pt**

Explain the terms policy *and* mechanism.

Lösung:

A mechanism is an implementation which describes how an activity is performed **(1.0 P)**. *A policy describes when a mechanism is used (and how it is used)* **(1.0 P)**.

Welchen Vorteil bietet die Unterteilung von Betriebssystemfunktionalität in Policy und Mechanismus? **1 pt**

Which advantage does dividing OS functionality into policy and mechanism provide?

Lösung:

Possible answers:

- *The mechanism can be reused even if a different policy is required (e.g., because of changing workloads)* **(1.0 P)**.
- *The hardware-independent policy can be reused if the hardware (and therefore the hardware-dependent mechanism) changes* **(1.0 P)**.

c) Warum verfügt ein modernes System über verschiedene Arten von Speicher (z.B. Cache, Arbeitsspeicher, SSD, Festplatte)? **1 pt**

Why does a modern system contain different types of memory (e.g., cache, RAM, SSD, hard disk)?

Lösung:

Because faster memory such as cache is expensive and therefore only available in small quantities, yet modern systems need to be able to store large amounts of data and need to provide high memory performance.

(1.0 P) *if two aspects are discussed (price/amount, volatility or performance).* **(0.5 P)** *if only one aspect is discussed.*

d) Welche Einschränkungen hat der User-Modus im Vergleich zum Kernel-Modus? **1 pt**

Which limitations does user mode have in comparison to kernel mode?

Lösung:

User mode code cannot access memory marked to be only accessible to kernel mode **(0.5 P)** *and cannot execute privileged instructions* **(0.5 P)**.

Lässt sich Schutz von Prozessen untereinander auch auf einem System implementieren, das nicht zwischen User- und Kernel-Modus unterscheidet? Begründen Sie. **1 pt**

Is it possible to implement protection between processes on a system which does not differentiate between user and kernel mode? Justify your answer.

Lösung:

No, because modifying address spaces needs to be privileged **(1.0 P)**. *If processes could make arbitrary changes to address spaces, they could modify their address space to access other processes' memory.*

Answers which argue that managed runtimes are able to enforce memory safety and therefore isolation are also accepted.

e) Weshalb verwenden interaktive Systeme üblicherweise präemptives Scheduling? **1 pt**

Why do interactive systems usually use preemptive scheduling?

Lösung:

Interactive systems use preemptive scheduling to prevent CPU-intensive processes from monopolizing the CPU and increasing the response time of other processes **(1.0 P)**.

f) Die *int*-Instruktion von x86-Prozessoren lässt den Prozessor in den Kernel-Modus wechseln und führt unmittelbar die Ausführung im Betriebssystem fort. In welche der drei Kategorien für Kerneleintritte (System Call/Trap-Instruktion, Interrupt, Exception) fällt dieser Vorgang?

Gehen Sie von den Definitionen aus der Vorlesung/Übung aus und begründen Sie. **1 pt**

The int *instruction of x86 processors causes the processor to change to kernel mode and to immediately continue execution in the OS. Into which of the three categories for kernel entries (system call/trap instruction, interrupt, exception) does this process fall?*

Base your answer on the definitions from the lecture/exercise and justify your answer.

Lösung:

The instruction causes a synchronous and voluntary entry into the kernel **(0.5 P)**. *It is therefore a trap instruction, used for system calls* **(0.5 P)**.

It is ***not*** *an interrupt, as interrupts do not originate from the code itself.*

g) Weshalb muss das Betriebssystem bei der Behandlung eines Interrupts vom Stack der Anwendung auf einen separaten Kernel Stack wechseln? **1 pt**

Why does the OS have to switch from the application's stack to a separate kernel stack when handling an interrupt?

Lösung:

Examples for correct answers:

- *Because the application's stack might be too small or not valid at all, so accesses to the stack might fail and might crash the system* **(1.0 P)**.
- *Because other threads from the same application could, when running on other cores of an SMP system, examine and change sensitive data which is placed on the stack by the OS during interrupt handling* **(1.0 P)**.

Total: 12.0pt

Aufgabe 2: Prozesse und Threads

Assignment 2: Processes and Threads

a) Parallele Ausführung lässt sich sowohl durch Prozesse als auch durch Threads umsetzen. Nennen und erklären Sie zwei Vorteile von Threads gegenüber parallelen Prozessen. **2 pt**

Parallel execution can be implemented both with processes as well as threads. List and explain two advantages of threads compared to parallel processes.

Lösung:

(1.0 P) *per advantage:*

- *All threads of a process share the same address space which facilitates communication and coordination between the threads*
- *Switching between threads causes less overhead than switching between processes, because the address space does not need to be switched.*

b) Nennen Sie einen Vorteil und einen Nachteil von User-Level Threads gegenüber Kernel-Level Threads. **1 pt**

Give one advantage and one disadvantage of user-level threads compared to kernel-level threads.

Lösung:

Advantages:

- *User-level threads can be implemented on operating systems that do not support kernel-level-threads.*
- *Switching between user-level threads causes less overhead than switching between kernel-level threads. As a consequence user-level-threads tend to be faster and more efficient.*

Disadvantages:

- *If one user-level thread invokes a blocking system call all, other threads of the process are blocked as well.*
- *The process scheduler implemented as part of the kernel lacks knowledge about the number of threads per process which might lead to poor scheduling decisions.*

(1.0 P) *for a valid advantage and* **(1.0 P)** *for a valid disadvantage.*

c) Verfügt jeder Thread eines Prozesses über einen eigenen Stack? Begründen Sie Ihre Antwort. **1.5 pt**

Does each thread of a process have its own stack? Justify your answer.

Lösung:

Yes **(0.5 P)**, *every thread needs its own stack to allow independent function calls* **(1.0 P)**.

d) Erklären Sie den „Convoy-Effekt“, welcher bei der Verwendung eines First Come First Serve (FCFS) Schedulers auftreten kann. **1 pt**

Explain the “convoy effect” which can occur when a first come first serve (FCFS) scheduler is used.

Lösung:

If one long job arrives first, the other jobs have to wait for the first job to finish even if they are very short **(1.0 P)**.

The convoy effect leads to a bad utilization of I/O devices and a poor average turnaround time.

e) Gegeben seien fünf Prozesse auf einem Einprozessorsystem mit den angegebenen Ankunftszeiten (0 = Start), Job-Längen und Prioritäten (hohe Werte werden bevorzugt). Vervollständigen Sie den untenstehenden Ablaufplan unter folgenden Annahmen:

- Es wird ein präemptiver Priority Scheduler mit Priority Donation (= Priority Inheritance) verwendet.
- Ein Prozess kann nicht ausgeführt werden, bis die Prozesse, auf die er wartet, vollständig ausgeführt wurden.
- Ein Kasten im Ablaufplan stellt eine Zeiteinheit dar.
- Der Scheduler wird immer nur nach ganzen Zeiteinheiten ausgeführt. **4 pt**

Consider five processes on a uniprocessor system, with given arrival times (0 = start), job lengths, and priorities (high values are favored). Complete the scheduling plan given below, under the following assumptions:

- *The system uses a preemptive priority scheduler with priority donation (= priority inheritance).*
- *A process cannot be executed until the processes for which it waits have been completely executed.*
- *A box in the scheduling plan represents one unit of time.*
- *The scheduler is only executed after whole units of time.*

Process	Arrival Time	Job Length	Priority	Waits for process
1	0	4	2	-
2	0,5	2	1	-
3	1,5	3	3	2
4	8,5	2	5	-
5	4,5	3	4	1

Lösung:

Scheduling plan:

1	*1*	*2*	*2*	*3*	*1*	*1*	*5*	*5*	*4*	*4*	*5*	*3*	*3*

*-**0.5 P** for each incorrect scheduling decision*

Berechnen Sie die Umlaufzeit (turnaround time) von Prozess 4, die Wartezeit (waiting time) von Prozess 1 und die Antwortzeit (response time) von Prozess 5. **1.5 pt**

Calculate the turnaround time of process 4, the waiting time of process 1 and the response time of process 5.

Umlaufzeit von Prozess 4 / *turnaround time of process 4*:

Lösung:

$t_{turnaround} = t_{finish} - t_{arrival} = 11 - 8{,}5 = 2{,}5$ **(0.5 P)**

Wartezeit von Prozess 1 / *waiting time of process 1*:

Lösung:

$t_{wait} = t_{finish} - t_{arrival} - t_{length} = 7 - 0 - 4 = 3$ **(0.5 P)**

Antwortzeit von Prozess 5 / *response time of process 5*:

Lösung:

$t_{response} = t_{firstdispatch} - t_{arrival} = 7 - 4{,}5 = 2{,}5$ **(0.5 P)**

f) Betrachten Sie ein Scheduling-Verfahren, welches immer den Prozess mit dem kürzesten nächsten CPU-Burst zur Ausführung auswählt, um die durchschnittliche Wartezeit zu minimieren. Weshalb führt dieses Verfahren jedoch nicht zwangsläufig zur minimalen durchschnittlichen Umlaufzeit? **1 pt**

Consider a scheduling policy which selects for execution the process with the shortest next CPU burst, in order to minimize average waiting time. Why does this policy, however, not necessarily result in the minimal average turnaround time?

Lösung:

If a long I/O intensive job with many short CPU bursts **(0.5 P)** *is executed, it is given priority over shorter jobs with longer CPU bursts* **(0.5 P)**. *Due to their lower priority, those shorter jobs have to wait, which might increase the average turnaround time.* **Total: 12.0pt**

Aufgabe 3: Koordination und Kommunikation von Prozessen

Assignment 3: Process Coordination and Communication

a) Erklären Sie den Begriff *Mailbox* im Kontext von Message Passing. **1 pt**

Explain the term mailbox *in the context of message passing.*

Lösung:

A mailbox is a indirection mechanism: With mailboxes, communicating processes do not name the communication partner directly but rather deposit messages to a mailbox or receive them from the mailbox **(1.0 P)**.

Note that the mailbox is not a buffer, but only an addressing mechanism. There are, for example, mailbox implementations with synchronous unbuffered message passing such as L4Re IPC gates.

b) Warum müssen Sendeoperationen in der Praxis manchmal auf den Empfänger warten, selbst wenn asynchrones Message Passing mit Puffern verwendet wird? **1 pt**

In practice, why do send operations sometimes need to wait for the receiver even if asynchronous message passing with buffers is used?

Lösung:

If the buffer is full **(0.5 P)**, *the sender needs to wait for the receiver to remove messages from the buffer until enough space is available* **(0.5 P)**.

c) Nennen Sie die vier notwendigen Bedingungen für einen Deadlock. **2 pt**

Name the four necessary deadlock conditions.

Lösung:

(0.5 P) *per condition: Mutual exclusion, hold and wait, no preemption, circular wait.*

d) Wie lässt sich mit einem Wait-for-Graphen bestimmen, ob ein Deadlock existiert? **1 pt**

How can a wait-for graph be used to determine whether a deadlock exists?

Lösung:

Any cycle in the wait-for graph is a deadlock **(1.0 P)**.

Ein Resource-Allocation-Graph beschreibt für einen einzelnen Zeitpunkt, welcher Prozess auf welche Resource wartet.

Weshalb kann das Betriebssystem in der Praxis nicht immer einen vollständigen Resource-Allocation-Graphen für das gesamte System aufstellen? **1 pt**

A resource allocation graph describes, for a single point in time, which process waits for which resource.

Why is the OS in practice not always able to create a complete resource allocation graph for the whole system?

Lösung:

Not all resources are known to the OS **(0.5 P)**. *For example, processes can implement spin locks in userspace without knowledge of the OS, to wait for resources in shared memory (***(0.5 P)** *for any valid example/explanation).*

e) Weshalb sollten User-Level-Threads (Many-to-One-Modell) keine Spinlocks zur Synchronisierung innerhalb des Prozesses verwenden? **1 pt**

Why should user-level threads (many-to-one model) not use spinlocks for synchronization within the process?

Lösung:

Spinlocks perform busy waiting **(0.5 P)**. *In a many-to-one model, all other threads have to wait for the running thread* **(0.5 P)**, *which hurts performance if another ULT from the same process holds the lock.*

Note that cooperative scheduling would cause this scenario to result in a permanent deadlock.

f) Warum muss beim Reader-Writer-Problem der Eintritt von schreibenden Threads in den kritischen Abschnitt gegenüber lesenden Threads priorisiert werden, wenn Bounded Waiting erfüllt werden soll? **1 pt**

Why does the reader-writer problem require the entry of writer threads into the critical section to be prioritized over the entry of reader threads if bounded waiting shall be fulfilled?

Lösung:

If reader threads are admitted into the critical section whenever they could enter it, a large number of reader threads can starve writer threads. If additional reader threads arrive at the section before the previous reader threads have left the section, there always is a reader thread in the section, so writer threads can never get in **(1.0 P)**.

g) Die folgenden beiden Codeabschnitte werden gleichzeitig ausgeführt. `x` ist dabei eine von den Threads gemeinsam genutzte Integervariable. Erklären Sie, weshalb das Programm manchmal abstürzt. **1 pt**

The following two code paths are executed at the same time. `x` is a integer variable shared between the threads. Explain why the program sometimes crashes.

```
1 /* first thread */
2 while (true) {
3     x = 0;
4     x = 1;
5 }
```

```
1 /* second thread */
2 if (x != 0) {
3     printf("%d\n", 42 / x);
4 }
```

Lösung:

The race condition happens if the check in line 2 of the second code fragment is executed while `x` has the value 1, whereas the division by `x` happens while `x` is 0 **(1.0 P)**.

Therefore, the program can crash with a division-by-zero exception.

h) Gegeben sei eine Compare-and-Swap-Instruktion, die atomar den Wert einer Variable im Speicher mit dem Wert eines Registers vergleicht und, falls die Werte gleich sind, den Wert im Speicher wechselseitig mit dem eines anderen Registers austauscht.

Erklären Sie, wie sich mithilfe dieser Instruktion eine atomare Addition auf einer einzelnen Variable implementieren lässt. Verwenden Sie dabei keine weiteren Speicherstellen für Spinlocks oder andere Synchronisierungsmechanismen. **3 pt**

Assume a compare-and-swap instruction which atomically compares the value of a variable in memory with the value of a register and, if the values are equal, reciprocally exchanges the value in memory with the value of another register.

Explain how this instruction can be used to implement an atomic addition onto a single variable. Do not use additional memory locations for spinlocks or other synchronization mechanisms.

Lösung:

The atomic addition can be implemented with the following algorithm:

(a) Fetch the value of the variable and store it in a temporary local variable.

(b) Perform the addition on that copy (non-atomically).

(c) Use compare-and-swap to check whether the value of the variable was changed, and if it wasn't, replace its value with the value of the temporary variable.

(d) If the variable was changed by other threads (i.e., the exchange of the compare-and-swap instruction returned a different value than the expected previous value), start again from step 1.

(1.0 P) *for modifying a copy of the variable,* **(1.0 P)** *for every solution which uses compare-and-exchange to detect concurrent modifications,* **(1.0 P)** *for repeating the operation until no concurrent modifications took place.*

Total: 12.0pt

Aufgabe 4: Speicher

Assignment 4: Memory

a) Erläutern Sie den Begriff *PFN*. **0.5 pt**

Explain the term PFN*.*

Lösung:

PFN is the abbreviation for Page Frame Number and denotes the index of a physical page frame **(0.5 P)**.

b) Welche üblicherweise vorhandenen Bereiche im Benutzeradressraum werden nicht von dem Betriebssystem mit Inhalt aus einer Datei initialisiert? **1.5 pt**

Which areas usually existing in the user address space of a process are not initialized by the OS with content from a file?

Lösung:

Heap **(0.5 P)**, *Stack* **(0.5 P)**, *and BSS* **(0.5 P)**.

Ein Prozess schreibt zum ersten Mal auf eine virtuelle Seite eines solchen Speicherbereichs. Das System verwendet Demand Paging. Beschreiben Sie eine Möglichkeit, die Latenz des Seitenfehlers klein zu halten. **1 pt**

A process writes to a virtual page of such a memory area for the first time. The system uses demand paging. Describe a way to keep the latency of the page fault low.

Lösung:

When a process accesses anonymous memory for the first time, the operating system must (1) find a free physical page (this may include evicting other pages), and (2) clear the page by filling it with zeros **(0.5 P)**. *The operating system can keep the latency low, by maintaining a pool of zero pages from which the allocation can directly be satisfied* **(0.5 P)**.

c) Erläutern Sie, wie Wissen über die Working Sets eines Systems vom Betriebssystem genutzt werden kann, um Thrashing zu verhindern. Gehen Sie davon aus, dass LRU-Seitenersetzung verwendet wird und diese nicht modifiziert werden soll. **1 pt**

Explain how knowledge on the working sets of a system can be used by the OS to avoid thrashing. Assume that LRU page replacement is used and that it shall not be modified.

Lösung:

Thrashing occurs when the system does not have enough physical memory to satisfy the working sets of the running processes. This causes constant swapping of pages and severely inhibits progress. The working set of a process is the set of pages accessed during the last measurement interval and can be used as indicator for the true memory demand of a process for the next measurement interval.

If the working sets of all running processes do not fit into memory, the operating system may select a process with a large working set and temporally suspend it, so other processes can make progress **(1.0 P)**.

d) Gegeben sei ein System, das virtuelle Adressen mit einer vierstufigen Seitentabelle übersetzt. Die Tabelle verfügt über 16 Einträge in der obersten Stufe und 512 Einträge in den folgenden Stufen. Die Seitengröße beträgt 4 KiB. Jeder Tabelleneintrag ist 8 Bytes groß.

Bei gleichem Speicherverbrauch für die Seitentabelle, wieviel größer wäre der maximal adressierbare virtuelle Speicher bei einer linearen Seitentabelle? **2 pt**

Consider a system which translates virtual addresses with a four-level page table. The table has 16 entries in the top-most level and 512 entries in each following level. The page size is 4 KiB. Each table entry is 8 bytes in size.

With the same memory consumption for the page table, how much larger would the maximum addressable virtual memory be for a linear page table?

Lösung:

The fourth level in the hierarchical page table is also required in the linear page table **(0.5 P)***. We thus only consider the higher levels. This gives us a memory consumption of* 1 *page for the top-most level,* 2^4 *pages for the second level, and* $2^4 * 2^9$ *pages for the third level* **(0.5 P)***. In a linear page table each of this pages could be used to address additional 512 pages of 4 KiB* **(0.5 P)***.*

This gives us a total of $(2^0 + 2^4 + 2^{13}) \cdot 2^9 \cdot 2^{12}$ *bytes* **(0.5 P)***.*

Nennen Sie einen weiteren Vorteil sowie einen Nachteil einer linearen Seitentabelle. **1 pt**

Give another advantage as well as a disadvantage of a linear page table.

Lösung:

Advantage **(0.5 P)***: Small latency and less memory accesses during translation.*
Disadvantage **(0.5 P)***: Not suited for sparse address spaces of large size due to high memory consumption.*

Inwieweit wird in diesem System die Größe des adressierbaren physischen Speichers durch die Seitentabelle bestimmt? Eine Berechnung ist nicht nötig. **1 pt**

How is the size of the addressable physical memory determined by the page table in the given system? A calculation is not required.

Lösung:

The physically addressable memory is only determined by the width of the PFN field in a page table entry **(1.0 P)***.*

e) Was beschreibt der *TLB Reach*? Nennen Sie zwei Möglichkeiten, diesen zu erhöhen. **2 pt**

What does the TLB reach *describe? Give two ways to increase it.*

Lösung:

The TLB reach describes the amount of virtual memory for which a TLB can store translations at any point in time **(1.0 P)***.*

The TLB reach can be increased by using a larger TLB **(0.5 P)** *or increasing the page size* **(0.5 P)***.*

f) Kann bei einem softwaregesteuerten TLB das Format eines Seitentabelleneintrags frei gewählt werden? Begründen Sie Ihre Antwort. **1 pt**

Does a software-managed TLB allow a custom format for page table entries? Justify your answer.

Lösung:

Yes **(0.5 P)**. *The page table is walked in software, and that software can convert arbitrary page table entries into the format expected by the TLB* **(0.5 P)**.

g) Ein Programm alloziert stetig Speicher, ohne diesen freizugeben. Wie kann es trotzdem zu Fragmentierung kommen? Geben Sie ein Beispiel an. **1 pt**

A program constantly allocates memory without freeing it. How can fragmentation still occur? Give an example.

Lösung:

Whether fragmentation occurs depends on the memory allocation policy. A buddy allocator for example produces internal fragmentation for allocations of size other than 2^n*, irrespective of the allocation order or deallocation of memory* **(1.0 P)**. **Total: 12.0pt**

Aufgabe 5: I/O, Hintergrundspeicher und Dateisysteme

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

a) Sie besitzen ein Verzeichnis mit geheimen Daten. Auf die Dateien innerhalb des Verzeichnisses sollen Sie selbst Lese- und Schreibzugriff haben, während die Mitglieder Ihrer Benutzergruppe ausschließlich lesenden Zugriff haben sollen. Andere Benutzer sollen keinen Zugriff haben.

Welche Zugriffsrechte müssen Sie auf einem UNIX-Dateisystem jeweils für Dateien und Verzeichnisse innerhalb dieses Verzeichnisses setzen? **2 pt**

You own a folder containing secret data. You should have read and write access to the files inside this folder, while members of your user group should have only read access to those files. Everyone else should have no access.

On a UNIX file system, which access rights do you have to set for files and directories inside this directory?

Lösung:

Files: 640 or rw-r-----
Folders: 750 or rwxr-x---
(0.5 P) *each for correct user, group and other access rights for files.* **(0.5 P)** *for recognizing that folders need the executable bit set.*

Wie ist es möglich, einem einzelnen Benutzer außerhalb Ihrer Benutzergruppe Zugriff auf das Verzeichnis zu geben, ohne die oben genannten Anforderungen zu verletzen und ohne die Gruppe des Benutzers zu ändern? **1 pt**

How is it possible to give a single user outside your user group access to the directory without violating the requirements described above and without changing the user's group?

Lösung:

ACLs **(1.0 P)** *can be used to grant access only to the single user.*

b) Wieviele Festplattenzugriffe sind nötig, um den Inhalt der zuvor ungeöffneten Datei `/home/reiser/klausur/filesystems.tex` aus einem inode-basierten Dateisystem zu lesen? Nehmen Sie an, dass keinerlei Festplatteninhalte im RAM gecached sind und dass sowohl die Datei als auch alle Verzeichnisse maximal einen Block groß sind. Gehen Sie zusätzlich davon aus, dass Metadaten und Daten getrennt voneinander gespeichert werden. **1 pt**

How many disk accesses are necessary to read the contents of the not-yet-opened file `/home/reiser/klausur/filesystems.tex` from an inode-based file system? Assume that no disk content is cached in RAM and that the file as well as all directories each fit in one disk block. In addition, assume that metadata and data are stored separately.

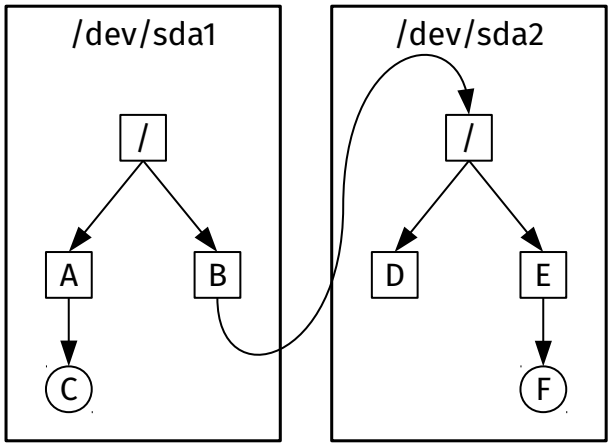
Lösung:

Ten **(1.0 P)** *disk operations:*

1. *Read the inode of the root directory (stored in a known location)*
2. *Read the contents of the root directory to get the inode number of* `/home`
3. *Read the inode of* `/home` *to get the disk block number of* `/home`*'s contents. Note that the block number is stored directly in the inode (no indirection for small files)*
4. *Read the contents of* `/home`
5. *Read the inode of* `/home/reiser`
6. *Read the contents of* `/home/reiser`
7. *Read the inode of* `/home/reiser/klausur`
8. *Read the contents of* `/home/reiser/klausur`
9. *Read the inode of* `/home/reiser/klausur/filesystems.tex`
10. *Read the contents of* `/home/reiser/klausur/filesystems.tex`

c) Betrachten Sie die beiden unten dargestellten Dateisysteme. Quadrate bezeichnen Verzeichnisse, Kreise Dateien. `/dev/sda2` ist im leeren Verzeichnis `B` von `/dev/sda1` eingehängt (mounted).

Consider the two file systems depicted below. Squares denote folders and circles denote files. `/dev/sda2` *is mounted in the empty directory* `B` *of* `/dev/sda1`*.*



Ist es in diesem Szenario möglich, im Verzeichnis `A` einen Softlink auf `F` zu erstellen? Begründen Sie Ihre Antwort! **1.5 pt**

Is it possible to create a soft link to `F` *inside the directory* `A`*? Justify your answer!*

Lösung:

Yes **(0.5 P)***. Soft links only point to a path name* **(0.5 P)***. Therefore, it does not matter where the target is stored* **(0.5 P)** *(or if it even exists).*

Ist es in diesem Szenario möglich, im Verzeichnis `A` einen Hardlink auf `F` zu erstellen? Begründen Sie Ihre Antwort! **1.5 pt**

Is it possible to create a hard link to `F` inside the directory `A`? Justify your answer!

Lösung:

No **(0.5 P)**. *Hard links are multiple directory entries pointing to the same inode number* **(0.5 P)**. *However, inode numbers are only valid for the device where the inode is stored (i.e., different devices may use the same inode number for different files)* **(0.5 P)**. *It is therefore not possible to have a hard link refer to a file on a different device.*

Wie lautet der **relativen** Pfad von `F` ausgehend von `A`? **1 pt**

*What is the **relative** path of `F` starting from `A`?*

Lösung:

`../B/E/F`

d) Wie wirkt sich ein RAID 1 von zwei identischen Festplatten im Vergleich zu einer einzelnen Festplatte auf die Bandbreite und Latenz von Lese- und Schreibzugriffen aus? Begründen Sie jeweils Ihre Antwort!

Nehmen Sie an, dass sämtliche Bewegungen der Lese-/Schreibköpfe identische Zeit benötigen. **4 pt**

How does a RAID 1 consisting of two identical hard disks change the bandwidth and latency of read and write accesses compared to using a single disk? Justify each answer.

Assume that all movements of the read/write heads require identical time.

Lesebandbreite / *read bandwidth*:

Leselatenz / *read latency*:

Schreibbandbreite / *write bandwidth*:

Schreiblatenz / *write latency*:

Lösung:

Write: Bandwidth is unchanged **(0.5 P)** *since all data must be written to both disks* **(0.5 P)**. *Latency is also unchanged* **(0.5 P)** *since the same seeks take place on both disks* **(0.5 P)**.

Read: Bandwidth is doubled **(0.5 P)**: *Since all data exists on both disks, the two can read different blocks in parallel* **(0.5 P)**. *Latency is unchanged* **(0.5 P)** *since one of the disks must still perform a seek* **(0.5 P)**. **Total: 12.0pt**