Karlsruher Institut für Technologie

Betriebssysteme
(Operating Systems)

Prof. Dr. Frank Bellosa
Lukas Werling, M.Sc.
Dr. Marc Rittinghaus

Musterlösung Hauptklausur

Theorie
02.03.2020

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 (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 ist nicht bestanden, wenn Sie aktiv oder passiv 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	9	9	9	9	9	45
Erreichte Punkte						
Note						

Aufgabe 1: Grundlagen

Assignment 1: Basics

a) Geben Sie zwei Paare von Hardware-Ressource und passender Abstraktion des Betriebssystems an. **1 pt**

Give two pairs of hardware resource and corresponding operating system abstraction.

Solution:

Hardware Resource	OS Abstraction
CPU Time	Process/Thread
RAM/Memory	Virtual Address Space
Disk	File

(0.5 P) *for each correct pair.*

b) Der folgende Systemaufruf liest `count` Bytes aus der Datei mit dem Dateideskriptor `fd` in den Puffer `buf`. Beschreiben Sie textuell, auf welche Eigenschaften die Parameter bereits vor dem Dateizugriff überprüft werden sollten. Gehen Sie davon aus, dass `buf` $\neq$ `NULL` und `count` ≥ 0 ist. Es dürfen nach der Prüfung Seitenfehler auftreten. **2 pt**

The following system call reads `count` bytes from the file with the file descriptor `fd` into the buffer `buf`. Describe textually for which properties the parameters should be checked before file access. Assume that `buf` $\neq$ `NULL`, and `count` ≥ 0. Page faults may occur after the check.

```
ssize_t read(int fd, void *buf, size_t count);
```

Solution:

- `fd` *is a valid file descriptor* **(0.5 P)** *and allows read access* **(0.5 P)**.
- `buf` **(0.5 P)** *and* `buf + count` **(0.5 P)** *are in user-mode memory.*

c) Warum sollten Interrupt Service Routinen (ISRs) möglichst kurz sein? **1 pt**

Why should interrupt service routines (ISRs) be as short as possible?

Solution:

While the ISR runs, the respective interrupt is usually masked **(0.5 P)**. *Further signals from the device may thus get lost* **(0.5 P)** *if the ISR takes too long.*

d) In einem hypothetischen Betriebssystem werden Interrupt Service Routinen nicht auf einem eigenen Stack ausgeführt. Erläutern Sie kurz, warum dies die Sicherheit und Stabilität des Systems gefährdet. **1 pt**

In a hypothetical operating system, interrupt service routines are not executed on a dedicated stack. Explain briefly why this endangers the security and stability of the system.

Solution:

An adversary could exploit the moment when an ISR runs on a user-mode stack under her control. Data on the stack left over from the ISR could leak sensitive information **(0.5 P)**. *In addition, the attacker could use a parallel thread to manipulate the ISR's stack frame while the routine is still running in order to hijacking the control flow of the ISR* **(0.5 P)**.

e) Nennen Sie zwei Aspekte, in denen sich Interrupts von Exceptions unterscheiden. **1 pt**

Give two aspects in which interrupts differ from exceptions.

Solution:

- *Interrupts are asynchronous, exceptions synchronous* **(0.5 P)**
- *Interrupts notify the OS of external events (e.g., key press), exceptions signal specific (error) conditions (e.g., page fault)* **(0.5 P)**

f) Wofür stehen die folgenden Abkürzungen? **2 pt**

What do the following abbreviations stand for?

Solution:

PCB: Process Control Block **(0.5 P)**
IPI: Inter-Processor Interrupt **(0.5 P)**
ASID: Address Space Identifier **(0.5 P)**
GOT: Global Offset Table **(0.5 P)**

g) Erläutern Sie einen Vorteil und einen Nachteil von statischem gegenüber dynamischem Linking von Bibliotheken. **1 pt**

Explain an advantage and a disadvantage of static versus dynamic linking of libraries.

Solution:

+ *No external dependencies (no DLL hell) as all libraries are linked to a single executable*
+ *Link time optimization (LTO) possible as libraries are available at link time of executable*
+ *Potentially smaller memory footprint (at least virtual memory), because only required functionality is included*
+ *Smaller load time, because (a) see previous point, and (b) no dependency resolution and dynamic linking*

- *Higher physical memory consumption due to missing sharing opportunities*
- *Larger executable distribution and storage size (see previous point)*
- *Higher security risks, as all executables that share a vulnerable library have to be relinked and updated*

Total: 9.0pt

Aufgabe 2: Prozesse und Threads

Assignment 2: Processes and Threads

a) Was versteht man unter *kooperativem Scheduling*? **1 pt**

What is cooperative scheduling*?*

Solution:

Cooperative scheduling means that the operating system does not forcibly preempt running threads **(0.5 P)***. Instead, threads must relinquish the CPU voluntarily, for example by blocking for I/O or calling* `yield()` *explicitly* **(0.5 P)***.*

Bei welchem Thread-Modell kommt kooperatives Scheduling üblicherweise zum Einsatz? **0.5 pt**

In which thread model is cooperative scheduling usually used?

Solution:

With the many-to-one thread model (ULT).

b) Im Gegensatz zur Kombination von `fork()` und `exec()` startet `CreateProcess()` unter Windows einen neuen Prozess in nur einem Systemaufruf. Welchen Vorteil neben der reduzierten Anzahl von Systemaufrufen hat dies noch? **1 pt**

In contrast to the combination of `fork()` *and* `exec()`*,* `CreateProcess()` *on Windows creates a new process in a single system call. What advantage besides the reduced number of system calls does this offer?*

Solution:

- `fork()` *has to copy the full address space of the calling process, which is slow for big address spaces. A combined call does not need to copy the address space.*
- `fork()` *duplicates lots of process state that is thrown away immediatly upon calling* `exec()`*. A combined call can initialize the new processes with the correct state directly.*
- `fork()`*ing is problematic in multi-threaded applications. Only the thread calling* `fork()` *is copied, so any locks that other threads might hold are never released. Consequently, it is not possible to allocate memory after forking without risking deadlocks.*

c) Der Linux-Scheduler nutzt auf Multikernprozessoren eine eigene Warteschlange für jeden Kern. Nennen Sie einen Vorteil (+) und einen Nachteil (-) dieses Ansatzes. **1 pt**

On multi-core procesors, the Linux scheduler uses a queue for each core. Name an advantage (+) and a disadvantage (-) of this approach.

Solution:

(+) *No cross-core synchronization necessary in the normal case, as the scheduler works on local data structures.*

(-) *The scheduler needs to do periodic load balancing across cores. (Note: If the local run queue is empty, the processor does not idle, but steals work from other queues.)*

d) Gegeben seien drei Prozesse auf einem Einprozessorsystem.

- Es wird ein Multi-Level Feedback Queue Scheduler mit den unten stehenden Warteschlangen verwendet.
- Alle Prozesse starten in der Warteschlange 2.
- Manche der Prozesse blockieren, nachdem sie eine gewisse Zeit gelaufen sind (relativ zum Prozessstart bzw. nach der letzten Blockierphase).
- Ein Prozess, der seine Zeitscheibe komplett ausnutzt, wird in die nächste Warteschlange mit niedrigerer Priorität verschoben. Ansonsten wird er in die Warteschlange mit nächsthöherer Priorität verschoben.
- Der Scheduler wird nach jeder halben Zeiteinheit ausgeführt.

Tragen Sie in das Ablaufdiagramm ein, welcher Prozess in welchem Zeitraum ausgeführt wird (Zeile P). Geben Sie für jeden noch lebenden Prozess an, in welche Warteschlange er als nächstes eingefügt wird (Zeile Q). **5.5 pt**

Consider three processes on a uniprocessor system.

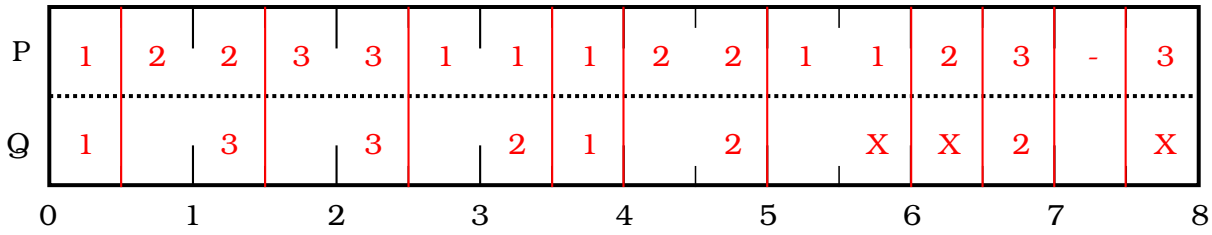
- *The system uses a Multi-Level Feedback Queue scheduler with the queues given below.*
- *Each process starts in queue 2.*
- *Some of the processes block after running for a certain time (relative to the process start or after the last blocking phase, respectively).*
- *A process that uses all of its timeslice is inserted into the next queue with lower priority. Otherwise, it is inserted into the next queue with higher priority.*
- *The scheduler is executed after each half unit of time.*

Complete the scheduling plan by filling in which process runs in each time slot (row P). For each process still alive, specify the queue it will be inserted into next (row Q).

Queue	Scheduler	Timeslice	Priority
1	Round Robin	1	High
2	Round Robin	1	Mid
3	Round Robin	2	Low

Process	Arrival Time	Job Length	Blocks after ...	Blocks for ...
1	0	3	0.5	2
			1.5	1
2	0.1	2.5	—	—
3	1	2	1.5	0.5

Ablaufplan/*scheduling plan*:



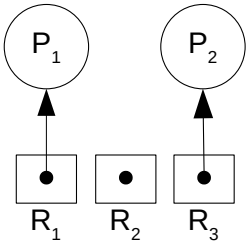
correct timeslice length **(1 P)**, correct job lengths **(0.5 P)**, correct blocking **(1.5 P)**, correct next queues **(1 P)**, correct process selection **(1.5 P)** **Total: 9.0pt**

Aufgabe 3: Koordination und Kommunikation von Prozessen

Assignment 3: Process Coordination and Communication

a) Wie nennt man den Typ des folgenden Diagramms? **0.5 pt**

How is the type of the following diagram called?



Solution:

Resource allocation graph (RAG) **(0.5 P)**

Ausgehend von der abgebildeten Situation führen die Prozesse folgende zusätzliche Operationen in der jeweils angegebenen Reihenfolge aus, um Ressourcen zu allozieren (`acquire()`) bzw. freizugeben (`release()`).

Wie müssen die Prozesse geschedult werden, damit sie ohne Deadlock alle Operationen durchführen können? **1.5 pt**

Starting from the depicted situation, the processes execute the following additional operations in the respective order to acquire or release resources.

How do the processes have to be scheduled so that they can execute all operations without any deadlock?

Prozess P_1 / *process* P_1:

```
1 acquire(R_2);
2 acquire(R_3);
3 release(R_1);
4 release(R_2);
5 release(R_3);
```

Prozess P_2 / *process* P_2:

```
1 release(R_3);
2 acquire(R_3);
3 acquire(R_1);
4 release(R_1);
5 release(R_3);
```

Solution:

Example schedule: First, P_2 executes line 1, then P_1 executes completely, then P_2 executes lines 2 to 5.

Erklären Sie, weshalb sich das System vor diesen zusätzlichen Operationen dennoch nicht in einem sicheren Zustand befindet. **1.5 pt**

Explain why the system is still not in a safe state before these additional operations occur.

Solution:

A safe state is when it is possible to sequentially schedule the processes to completion **(1.0 P)**.

In this case, each process requires resources initially held by the other process, which is why such a schedule is not possible **(0.5 P)**.

b) Wodurch unterscheidet sich ein Futex von einem klassischen Mutex? Welchen Vorteil hat dadurch der Futex? **1.5 pt**

How is a futex different from a classic mutex? Which advantage does the futex have?

Solution:

A futex first tries to use a spinlock in userspace to enter the critical section and only executes a mutex-like syscall to put the thread to sleep if the spinlock is busy **(1.0 P)**.

As a result, futexes cause significantly less syscall overhead if congestion is low **(0.5 P)**.

c) Der folgende Code implementiert eine einfache Barrierensynchronisation für N Threads, die dazu führt, dass die Threads die Funktion `barrier()` erst wieder verlassen, wenn alle N Threads die Funktion aufgerufen haben. Jeder Thread führt `barrier()` insgesamt nur ein einziges Mal aus.

Welches fehlerhafte Verhalten kann beobachtet werden? Beschreiben Sie, wie der Code verändert werden muss, um diesen Fehler zu verhindern. **2 pt**

The following code implements a simple barrier synchronization for N threads, which causes the threads to leave the function barrier() *only when all N threads have called the function. Each thread executes `barrier()` only once in total.*

Which erroneous behavior can be observed? Describe how the code has to be changed to prevent this error.

```
1 int counter = 0;
2
3 void barrier() {
4     counter = counter + 1;
5     while (counter != N) {
6         /* wait until all N threads have called the function */
7     }
8 }
```

Solution:

The threads might never leave the barrier **(0.5 P)** *because an update to the `counter` variable is lost* **(0.5 P)**. *The addition in line 4 has to be made atomic* **(0.5 P)** *by using an atomic instruction or a mutex* **(0.5 P)**.

Note that the questions explicitly asks for visible behaviour. Not all race conditions become visible and develop into an actual problem.

Die vorliegende Implementierung der Barriere erfordert Shared Memory und ist damit nicht in jedem Fall verwendbar. Beschreiben Sie, wie sich Barrierensynchronisation mittels direktem asynchronem Message Passing implementieren lässt. Sie können dabei davon ausgehen, dass jeder beteiligte Prozess weiß, welche anderen Prozesse an der Barrierensynchronisation teilnehmen und dass jeder Prozess durch eine eindeutige bekannte Zahl (Prozess-ID) adressiert werden kann. **2 pt**

The presented implementation of the barrier requires shared memory and is therefore not usable in all cases. Describe how barrier synchronization can be implemented with direct asynchronous message passing. You can assume that every involved process knows which other processes participate in the barrier synchronization and that each process can be addressed by a unique known number (process ID).

Solution:

Example implementation:

- *The process with the lowest process ID is designated as the "leader"* **(0.5 P)**.
- *When the other processes reach the barrier, they send a message to the leader and wait for a response before continuing to execute the code after the barrier* **(1.0 P)**.
- *When the leader reaches the barrier, it counts the incoming messages and when it has received $N-1$ messages, it sends responses to the processes before continuing* **(0.5 P)**.

Alternatively, the barrier can be implemented without the notion of a "leader" process:

- *All processes send messages to all other processes* **(1.0 P)**.
- *Each process then waits until $N-1$ messages were received before it continues* **(1.0 P)**.

In any case, no mailboxes or message queues may be used in the solution, as those are indirect methods for message passing, whereas the question specifically asks for direct message passing.

Total: 9.0pt

Aufgabe 4: Speicher

Assignment 4: Memory

a) Um sich vor der Meltdown-Attacke zu schützen, wird der Kerneladressbereich im Benutzermodus in großen Teilen ausgeblendet und es findet ein expliziter Adressraumwechsel bei Eintritt in den Kernel statt. Erläutern Sie welches Hardwarefeature die Kosten für diesen Wechsel reduzieren kann und wie dieses funktioniert. **1.5 pt**

To protect against the meltdown attack, the kernel address space is largely hidden from user mode and an explicit address space switch takes place when entering the kernel. Explain which hardware feature can reduce the cost of this switch and how it works.

Solution:

The cost can be mitigated by using tagged TLBs **(0.5 P)**. *With tagged TLBs, every address space is assigned a tag and only TLB entries with the current tag are used for translation* **(0.5 P)**, *thus eliminating the need to flush the TLB on kernel entries and exits* **(0.5 P)**.

b) Gegeben sei ein System mit vier physischen Seiten, welches den Clock-Algorithmus zur Seitenersetzung verwendet. Die Uhr schreitet in aufsteigender Reihenfolge der Rahmennummern fort, wobei die Hand initial auf Rahmen 2 (unterstrichen) zeigt und das Referenzbit für alle Seiten gesetzt (■) ist.

Vervollständigen Sie die Übersetzungstabelle für die unten angebenene Zugriffsfolge auf virtuelle Seiten. Geben Sie dabei jeweils auch die Position der Hand sowie den Zustand der Referenzbits an. **3 pt**

Consider is a system with four physical page frames, which uses the clock algorithm for page replacement. The clock advances in ascending order of frame numbers, with the hand initially pointing to frame 2 (underlined) and the reference bit for all pages being set (■).

Complete the translation table for the virtual page access sequence given below. Also specify the position of the hand and the state of the reference bits.

Solution:

Frames	*Pages*(t_0)		*Pages*(t_1)		*Pages*(t_2)		*Pages*(t_3)		*Pages*(t_4)		*Pages*(t_5)	
0	*1*	■	*1*	□	*1*	□	*2*	■	*2*	■	*2*	■
1	*5*	■	*5*	□	*5*	□	*5*	□	*1*	■	*1*	■
2	*2*	■	*0*	■	*0*	■	*0*	■	*0*	■	*0*	□
3	*3*	■	*3*	□	*3*	■	*3*	□	*3*	□	*5*	■

Zugriffsfolge/*Access Sequence*: **0 3 2 1 5**

c) Durch welche Eigenschaft einer vorwärtsgerichteten Seitentabelle wird die maximale Größe des physischen Adressraums bestimmt? **0.5 pt**

Which property of a forward page table determines the maximum size of the physical address space?

Solution:

The size of the physical address space is determined by the width of the page frame number field.

d) Erläutern Sie für welche Szenarien die folgenden Seitentabellentypen geeignet sind. Begründen Sie Ihre Antwort. **2 pt**

Explain for which scenarios the following page table types are suitable. Justify your answer.

Solution:

Linear page table *A linear page table is particularly suited for systems with small or mostly filled virtual address spaces* **(0.5 P)**. *In these cases, there is no benefit in using multiple levels for translation due to the small size and missing sparseness. Instead, the additional levels only increase translation costs and waste memory* **(0.5 P)**.

Inverted page table *An inverted page table makes sense if the physical address space is considerably smaller than the virtual address space. This way, less memory is wasted for page tables* **(0.5 P)**.

Alternative: An inverted table is also useful as an extension to a regular forward page table **(0.5 P)** *to enable fast reverse mapping (e.g., to find virtual mappings of a PFN)* **(0.5 P)**.

e) Welche drei Bedingungen müssen gelten, damit es für eine Menge von Allokationen zu externer Fragmentierung kommen kann? **1.5 pt**

Which three conditions must apply so that external fragmentation can occur for a set of allocations?

Solution:

- *Different allocation lifetimes*
- *Different allocation sizes*
- *No relocation of previous allocations*

f) Nennen Sie ein Adressraumsegment, welches niemals Teil einer ELF-Datei ist. **0.5 pt**

Name an address space segment that is never part of an ELF file.

Solution:

- *Heap*
- *Stack*
- *Dynamically created virtual memory areas (e.g., anonymous memory or memory mapped files)*

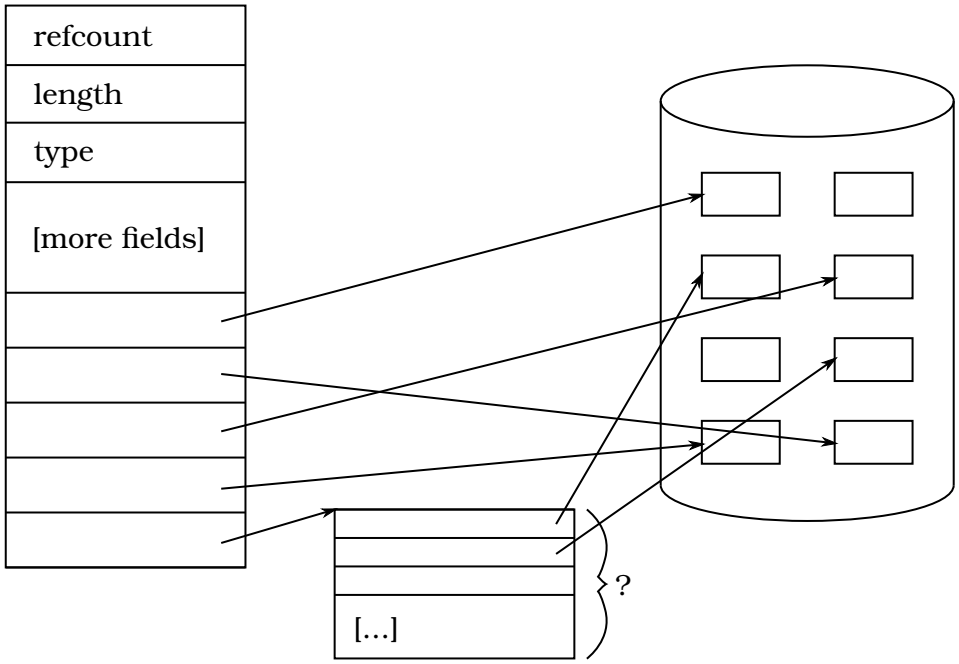
Total: 9.0pt

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 Datenstrukturen eines Dateisystems. Welche Blockallokationsstrategie kommt in dem abgebildeten Dateisystem zum Einsatz? **1 pt**

The figure below shows a secondary storage system with the data structures of a file system. Which block allocation strategy does the depicted file system use?



Solution:

Indexed Allocation

Benennen Sie die mit einem „?“ markierte Datenstruktur. **0.5 pt**

Name the data structure marked with a '?'.

Solution:

(single) indirect block

Nennen Sie ein weiteres Feld, das an der mit [more fields] markierten Stelle häufig gespeichert wird. **0.5 pt**

Name another field that is often present at the row marked [more fields].

Solution:

- *access rights / mode*
- *owner, group*
- *time stamps (creation, modification, access)*

Berechnen Sie die maximale Dateigröße, die in dem abgebildeten Dateisystem möglich ist. Gehen Sie von einer Blockgröße von 4 KiB und von 8 B großen Pointern aus. **1.5 pt**

Calculate the maximum file size for the pictured file system. Assume a 4 KiB block size and a 8 B pointer size.

Solution:

Each inode has 4 direct pointers and one pointer to a single indirect block. The single indirect block holds $2^{12}/2^3 = 2^9$ *pointers* **(0.5 P)**.
Thus, the maximum file size is $2^9 \times 2^{12} + 4 \times 2^{12} = 2^{21} + 2^{14} = 2064$ *KiB* **(1.0 P)**.

Wofür wird das `refcount`-Feld benötigt? Wann wird das Feld vom Dateisystem inkrementiert bzw. dekrementiert? **1 pt**

What is the `refcount` *field needed for? When does the file system increment or decrement this field?*

Solution:

The file system needs the `refcount` *field to know when to delete the inode in the presence of hard links* **(0.5 P)**. *The file system increments the field when a new hard link to the inode is created and decrements it when a hard link is deleted.* **(0.5 P)**

b) Moderne SSDs können auf Daten mit hoher Bandbreite bei geringer Latenz zugreifen. Erklären Sie, welche Rolle der Dateisystem-Cache beim Zugriff auf ein Dateisystem spielt und warum dieser Cache für solche SSDs dennoch sinnvoll ist. **2 pt**

Modern solid-state drives can access data with high bandwidth and low latency. Explain how the file system cache works and why such a cache is still desirable for fast drives.

Solution:

The file system cache buffers all accesses to secondary storage in DRAM **(0.5 P)**. *When writing to files, the modifications are written back to storage asynchronously* **(0.5 P)**. *This is still important for fast SSDs: SSDs have a limited amount of write cycles and wear out over time* **(0.5 P)**. *The file system cache can buffer many small (byte-granularity) writes to a single block and only writes the full block to the SSD once applications have stopped writing to that block. It thus reduces wear on the SSD* **(0.5 P)**.

Note: *Although modern SSDs are still slower than DRAM, they are now fast enough that managing the file system cache adds significant latency when reading a block for the first time. Consequently, higher DRAM speeds do not automatically make the file system cache desirable.*

Beschreiben Sie einen Nachteil des Dateisystem-Caches. **1 pt**

Describe one disadvantage of the file system cache.

Solution:

Write-behind policy might lead to data losses and/or inconsistent state of the FS in case of a system crash.

c) Moderne Dateisysteme wie btrfs können transparent Dateien über mehrere Laufwerke replizieren. Lesefehler kann btrfs dabei mit einer Checksumme über die gesamte Datei erkennen.

Modern file systems such as btrfs can transparently replicate files across multiple disks. Btrfs can detect read errors by calculating a checksum over the whole file.

Welchem RAID-Level entspricht dieses Vorgehen? **0.5 pt**

Which RAID level does this approach correspond to?

Solution:

RAID 1

***Note:** A checksum is distinct from the parity used in other RAID levels. The checksum only allows detecting errors, whereas parity provides redundancy and thus also allows restoring data. File replication means storing identical copies of the data on multiple drives, which is exactly what RAID 1 is doing. RAID does not have any concept of a file, so there cannot be any equivalent to a file checksum in any RAID level.*

Geben Sie einen Vorteil an, den Replikation wie in btrfs im Vergleich mit einem entsprechenden RAID-System hat. **1 pt**

Give one advantage of replication like in btrfs compared to a corresponding RAID system.

Solution:

- *RAID 1 relies on the disk to report read errors, it cannot detect errors on its own. Btrfs can detect invalid data with its checksum.*
- *With a file-level checksum, there is no need to fetch data from both disks to detect discrepancies like with RAID 1.*
- *File-level replication is more flexible than the block-level replication used in RAID. For example, it is possible to replicate data across differently-sized disks, or to disable replication for individual files.*
- *File system-level replication knows about unused blocks and does not need to initialize the whole drive like RAID 1.*

Total: 9.0pt