Karlsruher Institut für Technologie

Betriebssysteme
(Operating Systems)

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

Musterlösung Hauptklausur
03.03.2016

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 die zwei grundlegenden Aufgaben eines Betriebssystems. **1 pt**

Name the two basic functions of an operating system.

Lösung:

An OS must provide ***abstractions*** **(0.5 P)** *from the hardware and* ***protection*** **(0.5 P)** *from other applications.*

b) Erklären Sie den Unterschied zwischen einem Interrupt und einer Exception. **2 pt**

Explain the difference between an interrupt and an exception.

Lösung:

An Interrupt is generally caused outside the currently executing program. **(1 P)** *Technically, the program could thus go on without handling the interrupt. In contrast, an exception is caused by an instruction in the currently executing program* **(1 P)** *and prevents the program from continuing until the exception is handled.*

c) Erklären Sie den Begriff *multiprogramming*. **1 pt**

Explain the term multiprogramming.

Lösung:

Multiprogramming refers to multiple programs being loaded in memory at the same time, with the OS switching between them as necessary. **(1 P)** *If, for example, a program blocks for I/O, the OS can switch to a different one to keep the CPU busy.*

d) I/O-Instruktionen, mit denen auf Geräte zugegriffen wird, sind normalerweise privilegiert, d.h. diese Instruktionen können nur im Kernelmodus, aber nicht im Benutzermodus ausgeführt werden. Geben Sie einen Grund an, warum diese Instruktionen privilegiert sein sollten. Nennen Sie außerdem einen Grund, warum es sinnvoll sein kann, dem Benutzermodus direkten Zugriff auf bestimmte Geräte zu geben. **2 pt**

I/O-instructions, which are used for accessing hardware devices, are typically privileged (i.e., these instructions can only be executed in kernel mode). Give a reason why these instructions are privileged. Also, give a reason why it may be beneficial to allow user mode direct access to some devices.

Lösung:

I/O instructions should be privileged to guarantee process isolation. **(1 P)** *For example, if user space code had direct access to a DMA-enabled device, that code could use DMA to access the memory of other processes. Making device access privileged allows the kernel to validate DMA requests.*

Entering the kernel on each I/O request induces application overhead. Therefore, it may be beneficial to allow user space code direct device access if performance is critical for the operation of the device. **(1 P)** *Examples of such devices are GPUs or certain*

high-performance network hardware. However, these devices must implement appropriate protection mechanisms themselves to allow user space direct access without breaking isolation.

e) Nennen Sie zwei Möglichkeiten, die Parameter eines Systemaufrufs an den Kernel zu übergeben. Welche dieser Möglichkeiten würden Sie wählen? Begründen Sie Ihre Antwort. **2 pt**

Name two ways of passing the parameters of a system call to the kernel. Which of these ways would you prefer? Explain your answer.

Lösung:

Parameters can be passed either in registers **(0.5 P)** *or on the stack.* **(0.5 P)** *Passing them in registers is preferable,* **(0.5 P)** *because registers can be accessed faster than memory, which may improve system call performance* **(0.5 P)**.

f) Welche der folgenden Aussagen sind richtig?
(falsches Kreuz: -0.5P, kein Kreuz: 0P, korrektes Kreuz: 0.5P) **4 pt**

Which of the following statements are correct?
(incorrectly marked: -0.5P, not marked: 0P, correctly marked: 0.5P)

korrekt/ *correct*	inkorrekt/ *incorrect*	
☐	■	Zu jedem Kernel-Thread gehört ein eigener, geschützter Adressraum. *Each kernel thread has its own, protected address space.*
☐	■	Systemaufrufe sind eine Form von Interrupts. *System calls are a type of interrupt.*
■	☐	Ein Aufruf der Funktion `malloc()` kann eine Vergrößerung des Heap-Segments zur Folge haben. *A call to `malloc()` can increase the size of the heap-segment.*
☐	■	Die maximale Größe des virtuellen Adressraums kann unabhängig von der verwendeten Hardware frei gewählt werden. *The maximum size of a virtual address space can be chosen independently of the hardware used.*
■	☐	Ein Programm kann durch mehrere Prozesse gleichzeitig ausgeführt werden. *A program can be executed by multiple processes simultaneously.*
■	☐	Dieselbe virtuelle Adresse in unterschiedlichen Adressräumen kann auf unterschiedliche physische Adressen verweisen. *The same virtual address in different virtual address spaces can reference different physical addresses.*
☐	■	Um mehrere nebenläufige Prozesse zu unterstützen ist Verdrängung von Prozessen zwingend erforderlich. *Preemption is necessary to support multiple concurrent processes.*
■	☐	Ein Erhöhen der Seitengröße des virtuellen Speichers führt zu mehr interner Fragmentierung. *Increasing the page size of virtual memory increases internal fragmentation.*

Total: 12.0pt

Aufgabe 2: Prozesse und Threads

Assignment 2: Processes and Threads

a) Ist es sinnvoll, auf einem System mit nur einer CPU mehrere Threads zu verwenden? Begründen Sie Ihre Antwort. **1 pt**

Does it make sense to use multiple threads on a computer with only one CPU? Explain your answer.

Lösung:

Yes, it makes sense. **(0.5 P)** *Threads tend to block for I/O from time to time. In that case, the operating system can just switch to another ready thread if multiple threads are used. Multiple threads thus result in a higher CPU utilization.* **(0.5 P)**

b) Gegeben seien fünf Prozesse auf einem Einprozessorsystem mit den angegebenen Ankunftszeiten (0 = Start), Burst-Zeiten und Prioritäten (hohe Werte werden bevorzugt). Vervollständigen Sie die untenstehenden Ablaufpläne für die Strategie *Preemptive Shortest Job First (PSJF)* sowie die Strategie *Preemptive Strict Priority (PSP)*. Ein Kasten im Zeitplan stellt eine Zeiteinheit dar. **4 pt**

Consider five processes on a uniprocessor system, with given arrival times (0 = start), burst times, and priorities (high values are favored). Complete the scheduling plans given below for the policy Preemptive Shortest Job First (PSJF) *and the policy* Preemptive Strict Priority (PSP). *A box in the scheduling plan represents one unit of time.*

Process	Arrival Time	Burst Time	Priority
1	4	1	1
2	0	7	3
3	2	4	2
4	5	3	5
5	6	6	4

Lösung:

Preemptive Shortest Job First (PSJF)

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

Preemptive Strict Priority (PSP)

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

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

c) Berechnen Sie für den obigen *PSJF-Ablaufplan* die Turnaround-Zeit aller Prozesse. **2.5 pt**

For the above PSJF scheduling plan, *calculate the turnaround time of each process.*

Lösung:

The turnaround time for each process can be calculated as $t_{tournaround} = t_{finish} - t_{arrival}$

Process	*Finish time*	*Arrival time*	*Turnaround time*
1	*5*	*4*	*1*
2	*15*	*0*	*15*
3	*7*	*2*	*5*
4	*10*	*5*	*5*
5	*21*	*6*	*15*

d) Berechnen Sie für den obigen *PSP-Ablaufplan* die Wartezeit aller Prozesse. **2.5 pt**

For the above PSP scheduling plan, *calculate the waiting time of each process.*

Lösung:

The waiting time for each process can be calculated as $t_{wait} = t_{finish} - t_{arrival} - t_{burst}$

Process	*Finish time*	*Arrival time*	*Burst time*	*Waiting time*
1	*21*	*4*	*1*	*16*
2	*16*	*0*	*7*	*9*
3	*20*	*2*	*4*	*14*
4	*8*	*5*	*3*	*0*
5	*14*	*6*	*6*	*2*

e) Betrachten Sie einen Drucker, der ununterbrochen neue Aufträge von vielen Nutzern erhält. Immer wenn ein Auftrag fertig bearbeitet ist, verwendet der Drucker folgenden Scheduling-Algorithmus, um den nächsten Auftrag zum Drucken auszuwählen: Zunächst werden alle Aufträge aufsteigend nach Ihrer Seitenzahl sortiert. Anschließend wird der Auftrag mit der geringsten Seitenzahl ausgeführt.

Ist der beschriebene Scheduling-Algorithmus in diesem Szenario eine gute Wahl? Begründen Sie Ihre Antwort. **2 pt**

Consider a printer which continuously receives print jobs from many users. Whenever the printer has finished printing a job, it uses the following scheduling algorithm to select the next job for execution: First, it sorts all queued print jobs in ascending order of their page count. Then, it prints the job containing the smallest number of pages.

Is the scheduling algorithm described above a good choice in the above scenario? Explain your answer.

Lösung:

The algorithm is a bad choice. **(1 P)** *Since new jobs arrive continuously, there is a good chance that there will always be jobs containing a small number of pages in the queue. Large jobs can therefore starve.* **(1 P)** *Back in the days, this used to be a huge problem with the central SCC printers. By now, the SCC has made adjustments to its scheduling which alleviate the problem somewhat.*

Total: 12.0pt

Aufgabe 3: Koordination und Kommunikation von Prozessen

Assignment 3: Process Coordination and Communication

a) Nennen und erläutern Sie kurz die drei notwendigen Bedingungen für eine gültige Lösung des Problems kritischer Abschnitte. **3 pt**

Enumerate and briefly explain the three requirements for a valid solution of the critical section problem.

Lösung:

*(a) Mutual exclusion: Only one thread can be in the CS at a time. (must not be more than 1! If you only limit the number of threads that can **enter** the CS at any given time, you may end up with several threads in the CS at the same time)*

(b) Progress:

- *If no process is in the CS one of the processes trying to enter will eventually get in.*
- *Processes that are not trying to enter do not hinder processes that try to enter from getting in.*

If no process is executing in its critical section and there exist some processes that wish to enter their critical sections, then only those processes that are not executing in their remainder sections can participate in deciding which will enter its critical section next, and this selection cannot be postponed indefinitely.

(c) Bounded waiting: Once a thread starts trying to enter the critical section, there is a bound on the number of times other threads get in. A bound must exist on the number of times that other processes are allowed to enter the critical sections after a process has made a request to enter its critical section and before that request is granted.

***0.5pt** for each notion, **0.5pt** for each correct and matching explanaton.*

b) Welche Informationen über die Ressourcennutzung von Programmen benötigt ein System zur Laufzeit, um Deadlockvermeidung *(deadlock avoidance)* zu implementieren? **2 pt**

Which information about resource usage of processes do you need in an operating system at runtime for implementing deadlock avoidance?

Lösung:

(a) current allocation of resources to processes **(0.5 P)**,

(b) available resources **(0.5 P)**, *and*

(c) most importantly: maximum resource usage of each process needs to be known in advance. **(1 P)**

c) Nennen Sie die zwei grundlegenden Arten des Wartens in Synchronisationsprimitiven. Warum ist auf einem Einprozessorsystem nur eine davon sinnvoll, und welche ist dies? **2 pt**

State the two fundamental forms of waiting within synchronization primitives. Why does only one of them make sense on a single-processor system, and which one is that?

Lösung:

Busy waiting **(0.5 P)** *and sleep + wakeup (or blocking)* **(0.5 P)**.

Only sleep/wakeup makes sense on a single-processor system. While busy waiting, the lock holder cannot make progress towards releasing the lock because the waiting thread occupies the single CPU. Thus, this time spent busy waiting is wasted. **(1 P)**

d) Betrachten Sie den Pseudocode für Interprozesskommunikation (IPC) in Listing 1. Klassifizieren Sie die hier eingesetzte IPC möglichst präzise nach den Begriffen der Vorlesung. **2 pt**

Consider the pseudocode for interprocess communication (IPC) in Listing 1. Describe the IPC mechanism used here as specific as possible using the classification terms from the lecture.

```
char * msg = "Hello World!";
mqd_t mq = mq_open("my message queue", O_NONBLOCK | O_CREAT | O_RDWR);
mq_setattr(mq, maxmsg=32 /* queue capacity */, msgsize=512);
mq_send(mq, msg, strlen(msg), 0);
```

Listing 1

Lösung:

Indirect **(0.5 P)** *message passing* **(0.5 P)** *(a.k.a. indirect messaging), used in a non-blocking / asynchronous* **(0.5 P)** *fashion, with bounded buffers* **(0.5 P)**.

e) Betrachten Sie das in Abbildung 1 dargestellte System und die zukünftigen Systemaufrufe der drei Prozesse in Listing 2. Befindet sich das System in einem sicheren Zustand *(safe state)*? Begründen Sie Ihre Antwort. **3 pt**

Consider the system depicted in Figure 1 and the three processes' future system calls in Listing 2. Is the system in a safe state? Explain why.

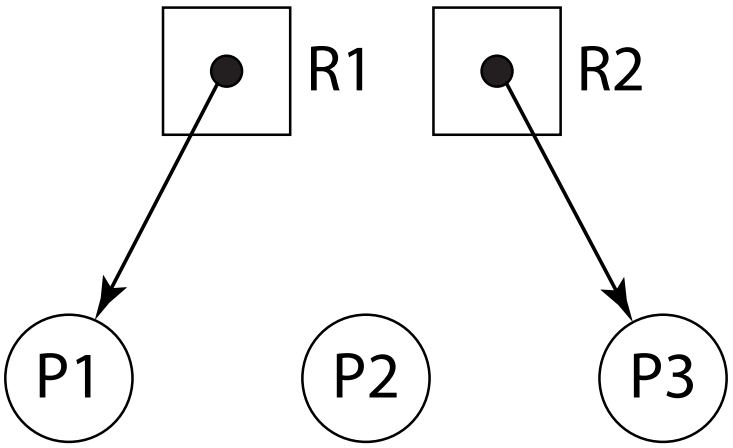


Abbildung 1 / *Figure 1*

```
P1: acquire(R2)      P2: acquire(R2)      P3: release(R2)
    release(R1)          acquire(R1)
    release(R2)          release(R1)
                         release(R2)
```

Listing 2

Lösung:

Yes, the system is in a safe state **(1 P)** *because all processes can run to completion* **(0.5 P)** *in the order P3* **(0.5 P)** *(releases R2), P1* **(0.5 P)** *(acquires R2 then releases R1 and R2), and finally P2* **(0.5 P)** *(acquires and then releases both resources).* **Total: 12.0pt**

Aufgabe 4: Speicher

Assignment 4: Memory

a) Welche Art von TLB wird benötigt, um die Verwendung eines betriebssystemspezifischen Seitentabellenformats zur Übersetzung von virtuellen zu physischen Adressen zu erlauben? Begründen Sie Ihre Antwort. **1 pt**

What type of TLB is required to support an operating system specific page table format for the translation of virtual to physical addresses. Explain your answer.

Lösung:

A software-TLB **(0.5 P)** *is required. TLB-misses are handed over to the operating system, which has to walk the page tables in software* **(0.5 P)**. *The page table format can thus be chosen by the operating system.*

b) Betrachten Sie ein System, das mittels Seitentabellen 24 bit virtuelle in 48 bit physische Adressen übersetzen soll. Jede Seite umfasst 4 KiB. Bewerten Sie folgende Seitentabellentypen nach ihrer Eignung für das gegebene System: (1) einstufige Seitentabelle, (2) mehrstufige Seitentabelle, (3) invertierte Seitentabelle. **3 pt**

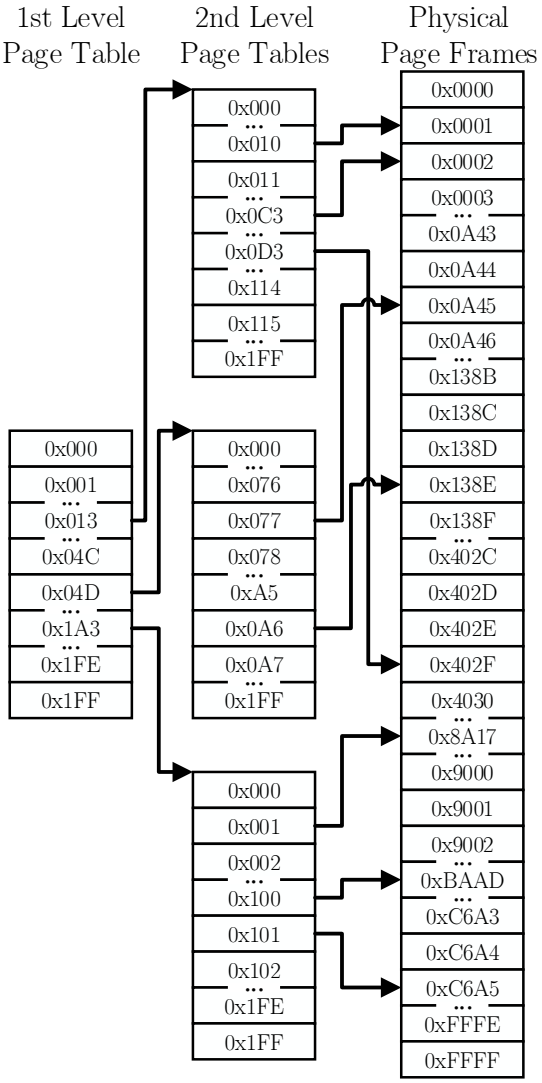
Consider a system that should translate 24 bit virtual addresses to 48 bit physical addresses using page tables. Each page is 4 KiB in size. Evaluate the following types of page tables for use in the given system: (1) single-level page table, (2) multi-level page table, (3) inverted page table.

Lösung:

With 24 bit virtual addresses and 4 KiB pages, a single address space is only 16 MiB in size, comprising 4096 pages. Assuming a PTE size of 8 bytes, we can store 512 PTEs per page. This gives a total of only 8 pages to map the whole address space. The virtual address space is thus small enough to justify a single-level page table without wasting huge amounts of memory **(1 P)**. *In contrast to the multi-level page table, the single-level page table requires only one memory access to resolve a TLB miss* **(1 P)**. *The inverted page table is not well suited to perform virtual to physical address translation as a linear search is involved* **(1 P)**. *Especially, if we consider that the system is obviously designed to support large amounts of physical memory (across many small address spaces).*

c) Gegeben sei folgende zweistufige, hierarchische Seitentabelle. Die Größe einer Seite betrage 8 KiB. Geben Sie die Größe des virtuellen Adressraumes (VAS) in Bytes an. Zerlegen und Übersetzen Sie anschließend die virtuelle Adresse `0x04DA61FA`. **5 pt**

Consider the following two-level, hierarchical page table. The page size is 8 KiB. Give the size of the virtual address space (VAS) in bytes. Then, split and translate the virtual address `0x04DA61FA`.



Lösung:

The 1st and 2nd level page tables contain $512 = 2^9$ *(*$0x1FF = 511$*) entries each. This gives* $512 * 512 = 2^9 * 2^9 = 2^{18}$ *pages. Each page is* $2^{13} = 8$ *KiB in size. The address space is thus* $2^{18} * 2^{13} = 2^{31}$ *bytes large* **(1 P)**.

For 8 KiB pages, we need 13 bits for the offset. The given page table requires 9 bits for each of the two indices. By splitting the virtual address accordingly, we get:
1st Level Page Table Index: `0x013` **(1 P)**
2nd Level Page Table Index: `0x0D3` **(1 P)**
Offset: `0x01FA` **(1 P)**

To get the physical address, we follow the page tables to the PFN: `0x402F` **(0.5 P)**.
The PFN is than concatenated on the bit-level with the offset: `0x0805E1FA` **(0.5 P)**

d) Welche Information muss ein Seitentabelleneintrag mindestens enthalten, damit das Working-Set eines Prozesses bestimmt werden kann? **1 pt**

What information must a page table entry contain at least to allow determining a process' working-set?

Lösung:

Reference bit **(1 P)**

e) Wieso kann eine beliebige Zuweisung von physischen zu virtuellen Seiten zu einer verminderten Ausführungsgeschwindigkeit führen, wenn ein physically-indexed, physically-tagged Cache verwendet wird? Nennen Sie eine Strategie zur Allokation von physischen Seiten, die diesem Problem entgegenwirkt. **2 pt**

An arbitrary assignment of physical pages to virtual pages can lead to reduced execution performance when working with a physically-indexed, physically-tagged cache. What might be the reason? What strategy for the allocation of physical pages can be used to mitigate the problem?

Lösung:

A physically-indexed, physically-tagged cache uses the physical address to determine the position of data in the cache. With an arbitrary assignment, the virtual pages in the working set of a process are more likely to map to physical pages that, when accessed together, produce cache conflicts or lead to only partial cache use **(1 P)**. *There is also a high chance that with every run of the program, the OS allocates different physical pages, which can result in less deterministic execution performance. Page coloring can be used to mitigate the problem* **(1 P)**.

Total: 12.0pt

Aufgabe 5: I/O, Hintergrundspeicher und Dateisysteme

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

a) Nennen und erläutern Sie zwei wesentliche Aufgaben eines I/O Subsystems in einem Betriebssystem. **2 pt**

Give and explain two essential tasks of an operating system's I/O subsystem.

Lösung:

(0.5 P) *for each given task (two at most) plus* **(0.5 P)** *for a corresponding explanation. Other tasks are accepted if they are plausible.*

Abstraction *The I/O subsystem should provide a uniform interface and abstract from the physical details of devices*

Serialization *In a system with asynchronous I/O or multiprogramming, multiple open I/O requests can exist simultaneously. However, most devices support only a single I/O operation at a time. It is the I/O subsystem's task to serialize I/O requests.*

Scheduling *On a system often certain jobs are more important than others (e.g., interactive application vs. background virus scan). The I/O subsystem should reflect this by scheduling I/O requests according to the jobs' priority. Other point: achieve fairness.*

Error Handling *The I/O subsystem should cope with device malfunction such as failed disk reads or writes by gracefully completing associated I/O requests and report the error to the issuer.*

Buffering *Depending on the I/O interface, it can be the I/O subsystem's task to buffer data between a process and a device, because of device transfer size limits or to maintain a copy semantic for asynchronous I/O.*

b) Erläutern Sie die Unterschiede zwischen Programmed I/O und Interrupt-getriebener I/O. **2 pt**

Explain the differences between programmed I/O and interrupt-driven I/O.

Lösung:

With programmed I/O the CPU issues a device command and polls on a device status register for the completion of the operation. The CPU may also decide to perform other work, however, it will not be signaled when the device is ready and ultimately has to check the status register. Data transfer will be one word at a time. **(1 P)**
With interrupt-driven I/O the CPU does not need to poll on a device status register. Instead, the device raises an interrupt when the operation finishes. Consequently, the CPU can perform other work while waiting for the device. Data transfer, however, still happens one word at a time. **(1 P)**

c) Durch welche Technik kann der Datendurchsatz von Interrupt-getriebener I/O deutlich erhöht werden? Erläutern Sie kurz, wie diese Technik funktioniert. **2 pt**

What technique can noticeably increase the data throughput of interrupt-driven I/O? Briefly explain how this technique works.

Lösung:

Direct memory access (DMA) **(1 P)**. *With DMA the CPU is relieved from having to transfer data on a per-word basis from the device to main memory. Instead, the device directly (or indirectly via the DMA controller) writes to main memory and only a single interrupt is fired, when the entire transaction (e.g., reading a sector from secondary storage) completes* **(1 P)**.

d) Erläutern Sie, weshalb moderne Speichergeräte in der Regel keine Aussage über die tatsächliche Speicherposition von Daten auf dem physischen Medium erlauben. Wieso ist dies insbesondere bei SSDs der Fall? **2 pt**

Explain why modern storage devices usually do not allow conclusions on the actual storage position of data on the physical medium. Why is this particularly true for SSDs?

Lösung:

Modern devices use logical block addressing (LBA), which hides the storage layout of the physical medium **(1 P)**. *This eases the development of device independent driver software and enables the device to perform an arbitrary mapping between block addresses and physical storage locations. While this feature has long been used on HDDs to remap broken sectors to spare sectors, SSDs particularly depend on extensive remapping. This way, SSDs can implement transparent wear leveling* **(0.5 P)** *to mitigate the effects of the limited write count of NAND cells. Furthermore, remapping enables SSDs to work with prematurely erased sectors, which reduces write latency* **(0.5 P)**.

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 512 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 512 bytes in size, a block address is 8 bytes long.

e) Um welche Art der Dateiblockallokation handelt es sich? **1 pt**

What kind of file block allocation is used?

Lösung:

Indexed Allocation **(1 P)**

f) Berechnen Sie den nötigen Speicherbedarf in Bytes zur vollständigen Adressierung einer 1 MiB großen Datei. **3 pt**

Calculate the amount of storage space in bytes required to fully address a 1 MiB file.

Lösung:

To address $1\ MiB = 2^{20}\ bytes$ *we need* $\frac{2^{20}\ bytes}{2^{9}\ bytes} = 2^{11}$ *blocks. A single-indirect block can address* $\frac{2^{9}\ bytes}{2^{3}\ bytes} = 2^{6}$ *blocks. We thus need* $\frac{2^{11}}{2^{6}} = 2^{5}$ *single-indirect blocks (*$512\ bytes$ *each). Because we have only a one single-indirect block in the i-node, we have to use the double-indirect addressing level. This level is effectively only an array of additional* 2^{6} *single-indirect blocks at the cost of a single* $512\ bytes$ *block. The four direct block addresses in the i-node do not save a full single-indirect block, but cost extra* $4 * 8\ bytes = 32\ bytes$*. Furthermore, we have one pointer to the single-indirect and one pointer to the double-indirect blocks (*$2 * 8\ bytes = 16\ bytes$*). This gives a total space requirement of:*

$2^5 * 512\ bytes\ (\mathbf{1\,P}) + 512\ bytes\ (\mathbf{1\,P}) + 48\ bytes\ (\mathbf{1\,P})$
$= 2^{14}\ bytes + 512\ bytes + 48\ bytes$
$= 16\ KiB + 512\ bytes + 48\ bytes$
$= 16944\ bytes$

Total: 12.0pt