Karlsruher Institut für Technologie

Betriebssysteme
(Operating Systems)

Prof. Dr. Michael Beigl
Jonas Kiemel, M.Sc.
Mathias Gottschlag, M.Sc.

# Musterlösung Hauptklausur
## 06.03.2019

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 20 Blättern: Einem Deckblatt und 19 Aufgabenblättern mit insgesamt 5 Aufgaben.

  *The examination consists of 20 pages: One cover sheet and 19 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) Nennen Sie ein Beispiel für eine vom Betriebssystem angebotene Abstraktion und nennen Sie den Mehrwert dieser Abstraktion. **1 pt**

*Name an example of an abstraction provided by the operating system and name the added value of this abstraction.*

**Lösung:**

*Correct answers include:*

- *The file system* **(0.5 P)** *makes it easier to find information compared to manually storing data directly on the disk* **(0.5 P)**.
- *Processes and threads* **(0.5 P)** *multiplex the CPU and allow multiple programs to run concurrently* **(0.5 P)**. *Note that virtual addresses themselves are provided by the CPU, not by the OS.*
- *The system call interface* **(0.5 P)** *hides the details of the underlying hardware by providing a device-independent uniform API* **(0.5 P)**. *Note that not all hardware details are hidden, so a valid answer should include an example for an abstraction (syscall interface, files, IPC, . . . )*

b) Erklären Sie den Unterschied zwischen *Parallelität* und *Nebenläufigkeit/Pseudoparallelität*. **1 pt**

*Explain the difference between* parallelism *and* concurrency/pseudoparallelism.

**Lösung:**

*Parallelism means that multiple threads/processes are executed at the same time on multiple CPUs* **(0.5 P)**, *whereas concurrency means that multiple processes are multiplexed onto one CPU via frequent context switching between the processes* **(0.5 P)**.

c) Weshalb sind sehr große Speicherallokationen durch Anwendungen in der Regel selbst dann erfolgreich, wenn das System nicht genügend Speicher besitzt? **1 pt**

*Why do very large memory allocations from applications usually succeed even if the system does not have sufficient memory?*

**Lösung:**

*The virtual address space is larger than physical memory* **(0.5 P)**. *When the program uses more of its address space than can be backed by physical memory, the OS moves some of the pages to a swap file or partiton to make space for additional data* **(0.5 P)**.

*Alternative: Demand paging enables virtual memory allocations bigger than the amount of available physical memory, as pages are only allocated in physical memory after the first access to the page.*

d) Erklären Sie den Begriff *Kontextwechsel*. **1 pt**

*Explain the term* context switch.

**Lösung:**

*During a context switch, the operating system switches from one process/thread to another by saving the CPU state (register state, address space, . . . ) of the previous process and restoring the CPU state of the next process* **(1.0 P)**.

e) Speicherzugriffe zeigen oft *zeitliche* und/oder *örtliche Lokalität*. Erklären Sie, was man jeweils darunter versteht. **2 pt**

*Memory accesses often show* temporal *and/or* spatial locality*. Explain the meaning of both.*

Zeitliche Lokalität / *temporal locality*:

Örtliche Lokalität / *spatial locality*:

**Lösung:**

*Temporal locality means that if the same memory address is accessed repeatedly, the accesses are likely to happen shortly after each other* **(1.0 P)**.

*Spatial locality means that the addresses accessed by consecutive memory accesses are likely close to each other* **(1.0 P)**.

Weshalb bieten Festplatten höhere Performance, wenn die Festplattenzugriffe einem hohen Grad an Lokalität aufweisen? **1 pt**

*Why do hard disk drives provide higher performance if the hard disk accesses feature a high degree of locality?*

**Lösung:**

*Hard disk drives need to position their read/write arm to the accessed sector* **(0.5 P)**. *If sectors are close together, the seek distance and therefore the seek latency is kept low* **(0.5 P)**.

f) Kann es bei rein kooperativem Scheduling passieren, dass nach Bearbeitung eines Interrupts ein anderer Prozess ausgeführt wird als vor Eintreffen des Interrupts? Begründen Sie. **1 pt**

*Assuming purely cooperative scheduling, is it possible that after the processing of an interrupt a different process is executed than before the arrival of the interrupt?*

**Lösung:**

*No* **(0.5 P)**, *as the described situation is essential preemption (involuntary context switch), which is not supported by purely cooperative scheduling* **(0.5 P)**.

*Note that interrupt handlers never directly terminate programs, as the program could be in the middle of a critical section in the kernel. Instead, the interrupt handler can deliver a signal to the process which then "commits suicide".*

g) Weshalb ist die Unterscheidung zwischen privilegierten und nicht privilegierten Instruktionen notwendig? **1 pt**

*Why is the distinction between privileged and non-privileged instructions necessary?*

**Lösung:**

*Some instructions can be used to circumvent the protection mechanisms of the operating system, e.g., to read other processes' memory or to starve other processes by disabling interrupts* **(1.0 P)** *These instructions must not be usable from user space.*

Operationen wie das Auslesen der Systemuhr können sowohl als nicht-privilegierte Instruktionen als auch als System Calls umgesetzt werden. Welchen Vorteil haben nicht-privilegierte Instruktionen? **1 pt**

*Operations such as reading the system clock can be implemented both as non-privileged instructions and as system calls. Which advantage do non-privileged instructions have?*

**Lösung:**

*System calls add overhead due to the required trap instruction and the resulting user-kernel transition, whereas direct execution of the instruction is considerably faster* **(1.0 P)**.

h) Systeme bestehend aus mehreren gleichartigen Prozessoren können üblicherweise Interrupts an einen beliebigen Prozessor zustellen. Geben Sie zwei Gründe an, weshalb die Wahl des Zielprozessors Auswirkungen auf die Gesamtperformance des Systems hat. **2 pt**

*Systems consisting of multiple equal processors can usually deliver interrupts to an arbitrary processor. Give two reasons why the choice of the target processor affects the overall performance of the system.*

**Lösung:**

*Valid reasons include the following:*

- *If a busy processor is targeted by an interrupt even if an idle processor is available, the running process is unnecessarily interrupted and delayed* **(1.0 P)**.
- *Each processor has its own cache. How efficiently the interrupt can be handled depends on the data in the cache* **(1.0 P)**. *Processors which have recently accessed the data required to process the interrupt will perform significantly better than processors with a "cold" cache.*
- *If one processor processes all interrupts, the chance is high that interrupts are not handled immediately by the OS, which increases the response time to user input* **(1.0 P)**.

*Note that a correctly designed system does not "lose" interrupts, no matter the interrupt rate, as the incoming interrupts are stored in data structures such as a bit mask with enough space to store the state of all possible interrupts.* **Total: 12.0pt**

## Aufgabe 2: Prozesse und Threads

*Assignment 2: Processes and Threads*

a) Was ist der Unterschied zwischen *Kernel-Level Threads* und *Kernel-Mode Threads*? **1 pt**

*Explain the difference between* kernel-level threads *and* kernel-mode threads*?*

**Lösung:**

*Kernel-mode threads perform background tasks* **(0.5 P)** *such as garbage collection, freeing page frames and swapping them out to disk, calculating checksums, mirroring disks in a software RAID, or distributing load to other processors. As the name suggests, kernel-mode threads are completely running in kernel-mode, meaning that privileged instructions can be executed without an additional system call. Kernel-level threads are threads running in user-mode and kernel-mode* **(0.5 P)**, *which are managed and scheduled by the kernel – as opposed to user-level threads which are managed in user-mode.*

b) Was versteht man unter einem *Upcall*? **1 pt**

*Explain the term* upcall*?*

**Lösung:**

*An upcall is a call from a lower-level subsystem, such as a kernel to a higher-level subsystem, such as user-code* **(1 P)**. *Note: A system call is not an upcall but a downcall from user code to the kernel.*

Erklären Sie die Bedeutung von Upcalls bei Verwendung eines M-to-N-Threadmodells. **2 pt**

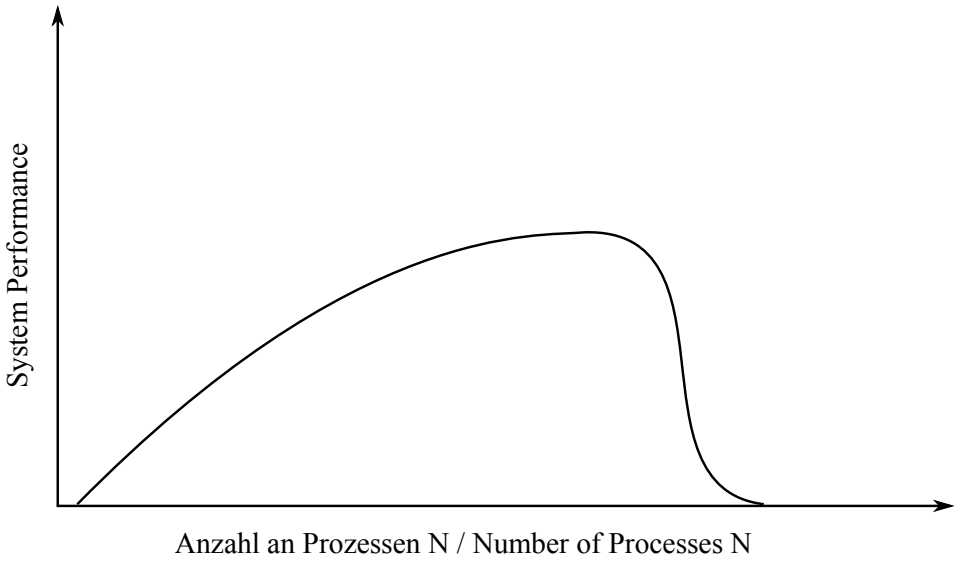
*Explain the importance of upcalls when using a M-to-N threading model?*

**Lösung:**

*In the many-to-many model there must be some way for the kernel to communicate with the user-level thread manager to maintain an appropriate number of kernel-level threads allocated to the process* **(1 P)**. *This mechanism is called scheduler activations. The kernel notifies the user-level thread manager of important kernel events using upcalls from the kernel to the user-level thread manager. Examples of such events includes a thread making a blocking system call and the kernel allocating a new kernel-level thread to the process* **(1 P)**.
*(***(1 P)** *is given if upcalls are confused with system calls providing that the explanation is detailed and correct.)*

c) Die folgenden Abbildung zeigt die Performance eines Rechnersystems in Abhängigkeit der Anzahl an nebenläufig ausgeführten Prozessen.

*The following figure shows the performance of a computing system in dependence on the number of concurrently executed processes.*




Erklären Sie, warum die Performance im linken Teil der Abbildung mit steigender Anzahl an Prozessen zunimmt. **1 pt**

*Explain why the system performance rises in the left part of the figure when the number of processes is increased.*

**Lösung:**

*Multiprogramming lets processes use the CPU when it would otherwise become idle.* **(0.5 P)** *Suppose the CPU phases of a process are as long as the I/O phases. In a single-programming system there would be no overlapping on the CPU. So, the CPU would be idle up to 50 % of the time while accessing the I/O device. As the number of processes is increased, the probability that at least one process wants to use the CPU rises.* **(0.5 P)** *Note: Processes are executed concurrently (not in parallel). Answers claiming that the speed up is caused by assigning processes to multiple CPUs are not accepted.*

Warum bricht die Performance im rechten Teil der Abbildung ein? **1 pt**

*Why does the system performance collapse in the right part of the figure?*

**Lösung:**

*When the number of processes is to high, thrashing occurs.* **(0.5 P)** *The system does not have enough physical memory to keep the working sets of all running processes in memory. This causes constant swapping of pages and severely inhibits progress.* **(0.5 P)** *Note: Answers claiming that the decrease in performance is caused by the overhead of the scheduler are not accepted for two reasons. Firstly, the collapse shown in the figure is to sharp to be explained by additional scheduling overhead. Secondly, modern systems typically use O(1) (constant time) or O(n) scheduler.*

Wie kann das Betriebssystem den Einbruch der Performance verhindern? **1 pt**

*How can the collapse of the system performance be prevented by the operating system?*

**Lösung:**

*If the working sets of all running processes do not fit into memory, the long-term scheduler may select some processes and temporally suspend them, so that other processes can make progress.* **(1 P)**

d) Nehmen Sie an, dass zwei Prozesse A und B zusammen mit weiteren Prozessen nebenläufig auf einem System mit einem Round-Robin Scheduler ausgeführt werden. Die Zeitscheibenlänge des Round-Robin Schedulers wird in den folgenden Experimenten verändert. Dabei werden jeweils die Umlaufzeiten der Prozesse A und B gemessen.

Wird die Zeitscheibenlänge von $T_0$ auf $T_1 = 0.1 \cdot T_0$ reduziert, so erhöhen sich die Umlaufzeiten von Prozess A und von Prozess B deutlich. Erklären Sie die Ursache. **1 pt**

*Assume that two processes A and B are executed alongside other processes on a system with a round-robin scheduler in a concurrent way. For the following experiments, the length of the timeslice of the round-robin scheduler is changed and the turnaround times of process A and process B are measured for each experiment.*

*If the length of the time slice is reduced from $T_0$ to $T_1 = 0.1 \cdot T_0$, the turnaround times of process A and process B are considerably increased. Explain the cause.*

**Lösung:**

*The number of context switches is increased by a factor of 10 which leads to additional computational overhead due to kernel invocation, storing and loading of the process control blocks, flushing of caches etc.* **(1 P)**

Wird die Zeitscheibenlänge von $T_0$ auf $T_2 = 2 \cdot T_0$ erhöht, so nimmt die Umlaufzeit von Prozess A leicht ab, wohingegen die Umlaufzeit von Prozess B deutlich zunimmt. Erklären Sie dieses Verhalten. **2 pt**

*If the length of the time slice is raised from $T_0$ to $T_2 = 2 \cdot T_0$, the turnaround time of process A is slightly reduced whereas the turnaround time of process B is considerably increased. Explain this behavior.*

**Lösung:**

*As the time slice is increased, the computational overhead for context switches is reduced which is beneficial for CPU-bound processes like process A.* **(1 P)** *Process B, however, is IO-bound and isn't able to use its full time quantum if the time slice is doubled due to frequent waiting for I/O. Whenever process B invokes the kernel to access IO, the scheduler selects the next process for execution regardless of whether its time quantum is fully used or not* **(1 P)**.

*Also accepted: Assuming that the runtime of process A is between $T_0$ and $2 \cdot T_0$ it can be completed within one time slice after increasing the length of the time slice. The turnaround time might decrease if process A is the first process that is selected for execution* **(1 P)**. *If we assume that process B can be completed within $T_0$ but is positioned at the end of the queue at initialization, the turnaround time of process B might increase due to the longer time slice assigned to other processes in between* **(1 P)**. *(***(0.5 P)** *are given for answers that explain the behavior of one process without being able to explain the behavior of the other process.)*

e) Erklären Sie das Funktionsprinzip einer Multilevel Feedback Queue. Worin unterscheiden sich die verschiedenen Level der Multilevel Feedback Queue? **2 pt**

*Explain the functional principle of a multilevel feedback queue. How do the levels of a multilevel feedback queue differ from each other?*

**Lösung:**

*When using a multilevel feedback queue, the operating system tries to achieve a good trade-off between interactivity and computational overhead caused by context switches. A Multi-Level Feedback Queue consists of multiple scheduling queues with different priorities. Processes that do not use up their quantum repeatedly are promoted into a higher priority queue while CPU-bound processes are demoted. IO-bound processes end up in a high priority queue and are scheduled more frequently but with a shorter time slice, thereby improving their interactivity. CPU-bound processes are given a lower priority but run for longer at a time, which reduces the computational overhead due to context switches.* **(1 P)** *Apart from priorities* **(0.5 P)**, *the queues can differ in their time quantum* **(0.5 P)** *and in the the scheduling algorithm (e.g. Round Robin or FCFS).*

**Total: 12.0pt**

## Aufgabe 3: Koordination und Kommunikation von Prozessen

*Assignment 3: Process Coordination and Communication*

a) Nennen Sie einen Vorteil von indirektem gegenüber direktem Message Passing. **1 pt**

*Name an advantage of indirect message passing compared to direct message passing.*

**Lösung:**

*Indirect message passing...*

- *... allows the communication partners to be replaced (e.g., after a software update).*
- *... allows a message to be received by any process of a group of processes (e.g., for a multi-process server).*

**(1.0 P)** *for any correct advantage. Note that mailboxes do not always have to be buffered (although in practice they often are). Indirection is all about* addressing*, so it is possible to implement buffered direct message passing just as easily as unbuffered indirect message passing (example: L4Re call gates).*

b) Welche Kombinationen von asynchronem bzw. synchronem Senden und Empfangen benötigen grundsätzlich gepuffertes Message Passing? Begründen Sie. **1.5 pt**

*For which combinations of asynchronous or synchronous send and receive is buffered message passing fundamentally required? Justify.*

**Lösung:**

*Buffers are necessary whenever sending is asynchronous* **(0.5 P)***. Asynchronous send operations return immediately and potentially continues to overwrite the message, so the message has to be placed in a buffer* **(0.5 P)***. Synchronous send, in contrast, requires the sender to wait for the receiver, so rendezvous-style message passing can be implemented and the message can be copied directly to the receiver* **(0.5 P)***.*

c) Beschreiben Sie das *Producer-Consumer-Problem*. Nennen Sie hierbei die Fälle, in denen einer der Kommunikationspartner blockieren muss. **2 pt**

*Describe the producer-consumer problem. Name the cases in which one of the communication partners has to block.*

**Lösung:**

*The producer-consumer problem consists of senders (producer) and receivers (consumer)* **(0.5 P)** *communicating via a shared bounded buffer* **(0.5 P)***. Producers put messages into the buffer and the receivers remove messages from the buffer. A producer has to block while the buffer is full, and a sender has to block while the buffer is empty* **(1 P)***.*

d) Im folgenden Codebeispiel möchte der erste Thread fortfahren, nachdem der zweite Thread eine Bedingung erfüllt. `wakeup` sei dabei ein System Call, der einen Thread fortfahren lässt, falls dieser gerade den `wait`-Systemaufruf ausführt.

Welches unerwünschte Verhalten kann auftreten? In welcher Situation tritt es auf? **2 pt**

*In the following code example the first thread wants to proceed after the second thread fulfills a condition.* `wakeup` *is a system call which unblocks a thread and lets it continue if that thread currently executes the* `wait` *system call.*

*Which unwanted behaviour can occur? In which situation does it occur?*

```
   int condition = 0;

1  /* first thread */
2  while (condition == 0) {
3      /* wait until wakeup
4         is called */
5      wait();
6  }
```

```
1  /* second thread */
2  if (condition == 0) {
3      /* if false, set to
4         true and wake up
5         first thread */
6      condition = 1;
7      wakeup(first_thread);
8  }
```

**Lösung:**

*The first thread might block forever* **(1.0 P)**.

*This situation happens if the first thread checks the condition (which is still 0), then the second thread sets the condition and tries to wake up the first thread* **(1.0 P)**. *At this point, the first thread is not in the* `wait` *system call yet, so it is not woken up. Once it is scheduled, executes the* `wait` *system call, but there is no further* `wakeup` *call, so the thread waits forever.*

*Note that in this case it is not possible that lines 6 and 7 of the second thread are reordered either by the compiler or the CPU. The compiler treats system calls as calls to unknown functions which might access arbitrary memory and therefore tries to provide a consistent view of memory at this point in time. The CPU will not reorder the lines as any correct implementation of* `wakeup()` *needs to contain a memory barrier to serialize memory accesses, as otherwise* `wakeup()` *could not reliably wake up a thread on a second processor (after all, the thread to be woken up needs to be specified by writing a value to memory).*

Nennen Sie ein Synchronisierungsprimitiv, das geeignet ist, um andere Threads zu benachrichtigen, und dieses Problem nicht besitzt. **0.5 pt**

*Name a synchronization primitive which is suitable to notify other threads and which does not have this problem.*

**Lösung:**

*Semaphore or condition variable* **(0.5 P)**.

*While a condition variable is the best way to solve any such problem, counting semaphores can be used as well in some situations. Non-counting primitives such as mutexes or spinlocks are not suitable for notification problems in general as they can only be used once at a time, and subsequent attempts to wait for a condition either result in race conditions or excessive blocking.*

e) Erklären Sie den Unterschied zwischen *Deadlock Prevention* und *Deadlock Avoidance*. **2 pt**

*Explain the difference between* deadlock prevention *and* deadlock avoidance.

**Lösung:**

*Deadlock prevention means that the system is restructured at design time so that arbitrary execution orders cannot lead to deadlocks, e.g., by negating one of the deadlock conditions* **(1.0 P)**. *Deadlock avoidance, instead, restricts the allowed resource requests at runtime so that the system stays in a safe state* **(1.0 P)**, *on the basis on knowledge about potential future resource requests by all involved processes.*

Beschreiben Sie ein Verfahren zur Erkennung von Deadlocks. **1 pt**

*Describe a method to detect deadlocks.*

**Lösung:**

*The system can be tested for deadlocks by creating the wait-for-graph* **(0.5 P)** *of all resource acquisitions and by searching for cycles in that graph* **(0.5 P)**.

*If a resource allocation graph is used, the OS needs to check whether the cycle can be resolved by continuing processes outside of the cycle.*

f) Gegeben sei ein Einprozessorsystem mit nicht unterbrechbaren Interrupt-Service-Routinen. Eine durch einen Spinlock geschützte Resource wird sowohl in einer Interrupt-Service-Routine als auch in unterbrechbarem Kernelcode verwendet.

In welcher Situation kommt es zu einem Deadlock? Identifizieren Sie die beiden Resourcen, auf die zyklisch gewartet wird. **2 pt**

*Assume a single processor system with non-interruptible interrupt service routines. A resource which is protected by a spinlock is used in an interrupt service routine as well as in interruptible kernel code.*

*In which situation does a deadlock occur? Identify the two resources involved in cyclic waiting.*

**Lösung:**

*Deadlocks occur if the spinlock is held by the interruptible kernel code, that code is interrupted and the interrupt handler tries to acquire the spinlock* **(0.5 P)**. *The interrupt handler prevents the previous code from proceeding but cannot proceed itself as the spinlock is locked* **(0.5 P)**. *The second resource (besides the resource mentioned in the task) is the CPU (which is "locked" by the interrupt handler whereas the interruptible kernel code tries to acquire it)* **(1.0 P)**. **Total: 12.0pt**

## Aufgabe 4: Speicher

*Assignment 4: Memory*

a) Erklären Sie die Begriffe *externe Fragmentierung* und *interne Fragmentierung*. **1 pt**

*Explain the terms* external fragmentation *and* internal fragmentation.

**Lösung:**

*Internal fragmentation:*
*If a memory manager only offers memory blocks of fixed sizes, the difference between requested size and block size cannot be used and is called internal fragmentation.* **(0.5 P)**
*External fragmentation:*
*The free space in physical memory may be too scattered to allow the allocation of a segment of a certain length, even if enough total free space is available. The sum of the currently not usable memory is called external fragmentation.* **(0.5 P)**

b) Geben Sie eine Möglichkeit an, wie interne Fragmentierung bei Paging verringert werden kann und zwei Nachteile, welche sich durch diese Maßnahme ergeben. **1.5 pt**

*State one way to reduce internal fragmentation when using paging and give two disadvantages which are caused by this measure.*

**Lösung:**

*Internal fragmentation can be reduced by choosing a smaller page size.* **(0.5 P)**
*Correct disadvantages include:*

- *A system with a smaller page size uses more pages so the size of the page table is increased*
- *The TLB reach (amount of memory that the TLB can keep track of) is reduced*
- *More virtual-to-physical translations are required, which leads to additional overhead*
- *Spatial locality might be decreased. This might decrease the performance of caches and result in higher access times when using a HDD.*

**(0.5 P)** *if one valid disadvantage is given.* **(1 P)** *if two correct disadvantages are given.*

*Also accepted: Using multiple page sizes allows a lower degree of internal fragmentation, since we can allocate a page just large enough to handle whatever memory we are allocating. Disadvantages:*

- *The hardware can no longer immediately discern the page number from the virtual address, since each page will be a different size.*
- *Software must select a proper page size assignment policy to take advantage of multiple page sizes.*
- *MMU page replacement policies must accommodate multiple pagesizes.*
- *External fragmentation is now possible*

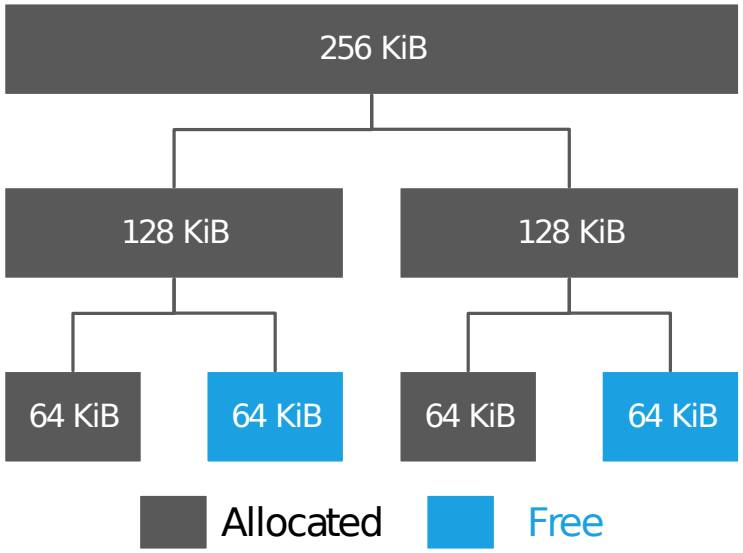
c) Wie kann es bei einem Buddy-Allokator zu externer Fragmentierung kommen?
Hinweis: Eine Skizze kann die Erklärung erleichtern. **2 pt**

*How can external fragmentation occur when using a buddy allocator.*

*Hint: A sketch might simplify the explanation.*

**Lösung:**

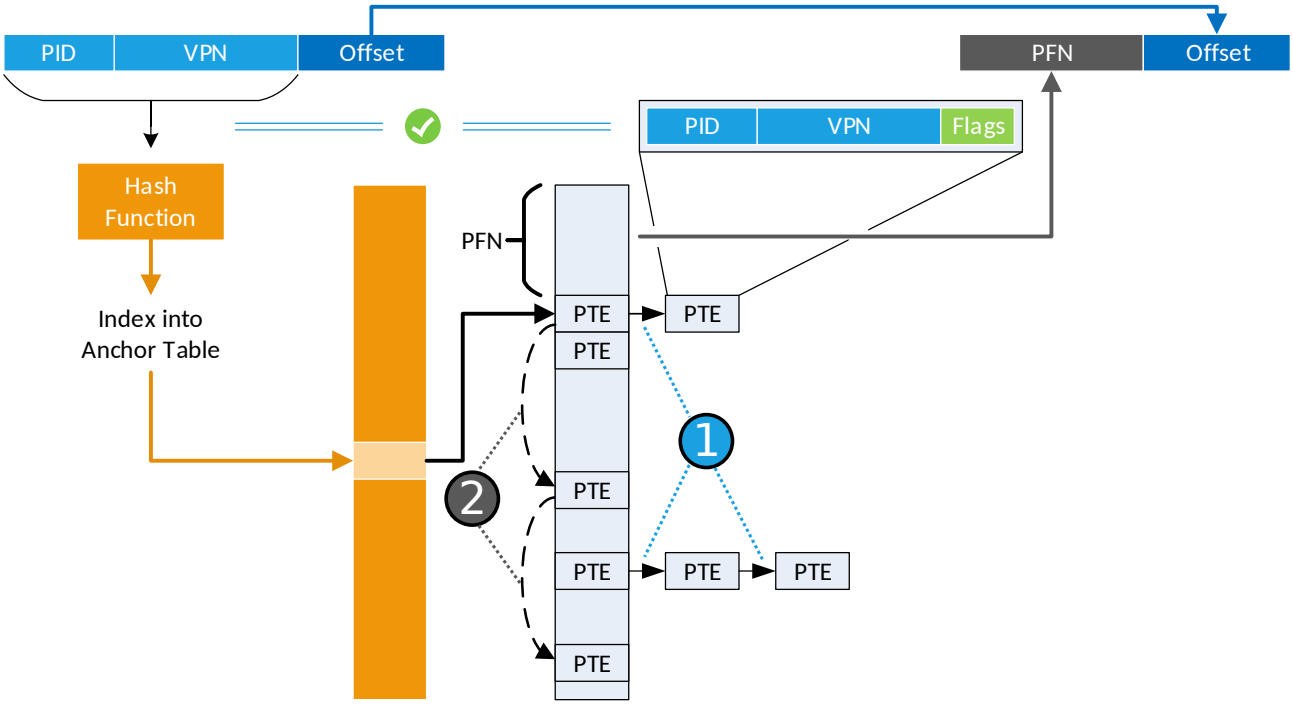
*External fragmentation can happen for example if multiple blocks of a certain level have been freed but cannot be merged* **(1 P)** *because their respective buddies still contain valid allocations – you can only merge blocks that are buddies. In that case the total free memory might suffice to fulfill a memory request, but the memory is not contiguous* **(1 P)**.




*In the sketch above, the total free memory is 128 KiB but memory requests higher than 64 KiB cannot be fulfilled due to external fragmentation.*

d) Geben Sie den Namen der in der folgenden Abbildung dargestellten Struktur an. **0.5 pt**

*Give the name of the structure depicted in the following figure?*




**Lösung:**

*Hashed inverted page table* **(0.5 P)**
*Note: Only the correct and complete name is considered as correct.*

Erklären sie die Bedeutung der mit „1“ und „2“ beschrifteten Pfeile in der obigen Abbildung.
Hinweis: Die Pfeile stellen Pointer dar. **2 pt**

*Explain the meaning of the arrows marked with ”1” and ”2” in the figure above.*
*Hint: The arrows represent pointers.*

**Lösung:**

*Pointer 1:*
*A physical address can be mapped by multiple virtual addresses in order to support shared memory* **(0.5 P)**. *All virtual addresses that point to the same physical address are stored in a linked list. The first entry of the list is pointed to by the entry of the hash anchor table or by pointers likes those marked with „2“. All other entries are linked with pointers like those marked with „1“.* **(0.5 P)**

*Pointer 2:*
*With a normal inverted page table, in order to map a virtual address to a physical address, the table must be searched until the correct entry is found. A hash table can be used to reduce the lookup costs. However, as with all hash tables, collisions may occur* **(0.5 P)**. *In case of a collision, entries in the inverted page table can be linked with pointer like those marked with „2“* **(0.5 P)**.

e) Erklären Sie, warum das Betriebssystem die Struktur der TLB-Einträge auf einem System mit einem Software-Managed TLB nicht frei wählen kann. **1 pt**

*Explain, why the operating system cannot choose an arbitrary format for TLB entries on a system with a software-managed TLB.*

**Lösung:**

*The TLB entry format is specified in the instruction set architecture (ISA) of the respective CPU. Even with a software-managed TLB, the TLB is a hardware component, which is specified by the cpu manufacturer* **(1 P)**. *The format of the page table entries, however, can be chosen by the operating system.*

f) Nennen und erläutern sie einen Vorteil und einen Nachteil von physischen Tags gegenüber virtuellen Tags. **2 pt**

*Name and explain one advantage and one disadvantage of physical tagging compared to virtual tagging.*

**Lösung:**

*Correct answers for advantages include (***(1 P)** *for a valid advantage):*

- *No ambiguity (homonym problem): A virtual address might point to different physical addresses at different points in time. When using physical tags, homonyms can be detected which means that the cache does not have to be flushed at each context switch.*
- *Synonyms (multiple virtual addresses point to the same physical address) can be detected by checking all entries of the cache*
- *Easy write-back. If the cache entry needs to be written back to main memory, no additional TLB lookup is required.*

*Disadvantage (***(1 P)** *for a valid disadvantage):*
*Modern systems use virtual memory abstraction. A TLB look-up is required to translate the virtual address to the physical address which is used for tagging. The virtual to physical translation takes some time and might increase the latency of the cache.*

g) Um die Latenz des L1-Caches zu reduzieren, kann man einen virtuell indizierten Cache verwenden.

Welches Kohärenzproblem tritt by Verwendung eines virtuell indizierten, physisch getaggten Caches auf? **0.5 pt**

*In order to reduce the latency of the L1 cache, a virtually indexed cache can be used.*

*Which coherence problem arises when using a virtually indexed, physically tagged cache?*

**Lösung:**

*Aliasing* **(0.5 P)** *(coherence problem with synonyms). Virtual addresses that point to the same physical address might be mapped to different cache sets.*

Erklären Sie, warum dieses Kohärenzproblem nicht auftritt, wenn die Größe des L1-Caches durch $W \times P$ begrenzt ist. $W$ bezeichnet hierbei die Assoziativität des Caches und $P$ die Größe der Seiten. **1.5 pt**

*Explain, why this coherence problem does not arise if the size of the L1 cache is limited to $W \times P$, where $W$ is the associativity of the cache and $P$ is the page size.*

**Lösung:**

*By limiting the size of the L1-cache, aliasing (coherence problem with synonyms) can be prevented. Physically indexed caches (like the L2 or L3 cache) do not suffer from aliasing. The index of the L1-cache, however, is determined in parallel with the TLB translation to reduce the latency of the L1-cache. To avoid aliasing, only the untranslated offset bits of the virtual address* **(0.5 P)** *can be used in the index, which limits the number of cache sets to the page size $P$.* **(0.5 P)** *The offset bits are shared between the virtual and physical address. Two virtual addresses pointing to the same physical address will therefore be mapped to the same cache set. Within each cache set, synonyms can be detected by checking all entries.* **(0.5 P)** *Each cache set has $W$ entries, which leads to a maximum cache size of $W \times P$.*

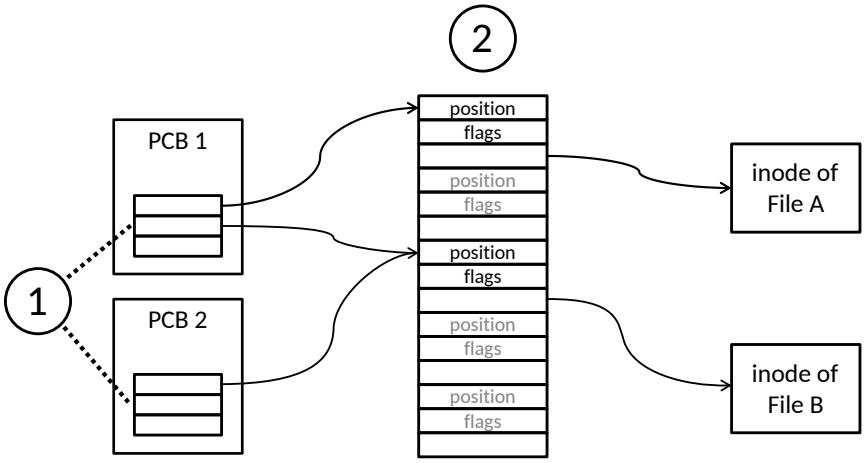
**Total: 12.0pt**

## Aufgabe 5: I/O, Hintergrundspeicher und Dateisysteme

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

a) Die unten stehende Abbildung zeigt zwei Prozesse, die zwei Dateien geöffnet haben. Benennen Sie die beiden markierten Datenstrukturen, die zur Verwaltung offener Dateien benötigt werden. **1 pt**

*The following figure shows two processes which have opened two files. Name the two marked data structures which are necessary for managing open files.*




(1) local open file table **(0.5 P)**

(2) global open file table **(0.5 P)**

Nennen Sie eine Information, die in dem „flags"-Feld in (2) gespeichert wird. **0.5 pt**

*Name some information that is stored in the "flags" field in (2).*

**Lösung:**

*The "flags" field stores the file access mode (read or write) as well as file creation flags (e.g., truncate) passed to* `open()`*.*

*Common wrong answer: access permission. The access permission (read/write/execute bits) is stored in the inode.*

Die beiden oben dargestellten Prozesse teilen sich einen Eintrag in (2). Wie kann es auf einem Unix-System dazu kommen? **1 pt**

*The two processes pictured in the figure above share an entry in (2). How can this happen on a Unix system?*

**Lösung:**

*A call to* `fork()` **(1 P)** *copies the local open file table, but not the entries in the global open file table. Consequently, two processes share an open file.*

*Common wrong answers:*

*(a) Two processes have opened the same file (possibly via symlinks or hardlinks). In that case, each process can read/write the file independently. Here, the two processes share the access mode (read/write) and position and thus might need extra synchronization to do useful work with the file.*

*(b) The entry is shared only temporarily and Copy-on-Write will kick in once a process writes to the file. After a `fork()`, the operating system will indeed use CoW for the user address space of the processes. However, the data structures in the picture are all in kernel memory and thus not subject to that mechanism. There is no special CoW mechanism for files either. (***(0.5 P)** *for `fork()` + CoW)*

*(Alternative answer: sending/receiving file descriptors over unix sockets)*

Ist es im abgebildeten System möglich, ausgehend von einem Eintrag in (1) den Dateinamen der geöffneten Datei herauszufinden? Begründen Sie. **1 pt**

*In the system pictured above, is it possible to get the filename of an opened file starting from an entry in (1)? Explain.*

**Lösung:**

*No* **(0.5 P)**, *because there is no association from an inode to a directory entry/filename, but only the other way around. Even with the association, there might be multiple hardlinks or the opened file might have been renamed or deleted.* **(0.5 P)**

*(Note: Some operating systems save the path of the file at the time it was opened. This path becomes invalid if the file is moved or deleted, so it does not give a correct path in general.)*

*Common wrong explanations:*

*(a) The filename is stored in the inode.*

*(b) The filename is not stored in the local file table / file descriptors (but no mention of the other data structures in the picture).*

b) Anwendungen nutzen häufig Code ähnlich dem unten gegebenen, um ihre Dateien zu speichern.

*Applications often use code similar to that in the snippet below for saving files.*

```
1 /* open new temporary file */
2 fd = open("config.tmp", O_WRONLY | O_CREAT);
3 /* write new file contents */
4 write(fd, data, data_size);
5 /* close the temporary file */
6 close(fd);
7 /* replace the original file */
8 rename("config.tmp", "config.txt");
```

Nehmen Sie an, ein zweites Programm liest die Datei `config.txt` während des Speichervorgangs. Wie unterscheidet sich die Sicht auf den Dateiinhalt, wenn die Speicherstrategie oben angewandt wird, anstatt direkt in die Ursprungsdatei `config.txt` zu schreiben? **2 pt**

*Assume that a second program reads the file* `config.txt` *during the save operation. Compare the resulting view of the file contents with an alternative saving implementation that writes directly to the original file* `config.txt`.

**Lösung:**

*The changes appear atomically in the new file. Applications reading the file concurrently will either see the complete old or the complete new file* **(1 P)**. *In contrast, when overwriting the file, it is possible to observe partial updates* **(1 P)**.

Wegen eines Stromausfalls stürzt der Computer während des `rename`-Systemaufrufs ab, wodurch das Dateisystem in einen inkonsistenten Zustand gerät. Geben Sie zwei Möglichkeiten an, wie diese Inkonsistenzen aussehen könnten.

Hinweis: Überlegen Sie, welche Änderungen `rename` an Ordnereinträgen und I-nodes durchführt. **2 pt**

*The computer crashes during the* `rename` *system call due to a power failure, leaving the file system in an inconsistent state. Give two examples of possible inconsistencies.*

*Hint: Consider how* `rename` *changes directory entries and inodes.*

**Lösung:**

*The* `rename` *operation would do the following things in any order:*

- *remove the directory entry of "config.tmp"* ***or*** *of "config.txt"*
- *overwrite the directory entry of "config.txt" so that it points to the new inode* ***or*** *rename the "config.tmp" directory entry to "config.txt"*
- *decrement the reference count of the old inode*

*There are two valid interpretations of "inconsistent file system state" here:*

*(a) A file system state that is not reachable at all during normal operation.*

*(b) A file system state that is not reachable with the* `rename` *operation, but would be possible with other file system operations (e.g.,* `link` *and* `unlink`*).*

*Possible inconsistencies for interpretation (a):*

- *there are two directory entries with the same name "config.txt"*
- *the new inode has a reference count of 1 but is not reachable from any directory*
- *the old inode has a reference count of 1 but is not reachable from any directory*
- *both "config.tmp" and "config.txt" point to the new inode, but that inode has a reference count of 1*

*Possible inconsistencies for interpretation (b):*

- *directory entry "config.tmp" deleted, but "config.txt" still points to the old inode*
- *both directory entries "config.tmp" and "config.txt" point to the new inode*
- *both directory entries "config.tmp" and "config.txt" deleted*

**(1 P)** *for each non-duplicate item.*

*Common invalid answers:*

*(a) No rename happened. (Not inconsistent, but a valid result also with journaling.)*

*(b) Directory entry or inode corrupted due to partial write. (HDDs can write at least 512 bytes atomically which is large enough for updating these structures.)*

*(c) Invalid modifications to the inode to fields such as length, file block pointers, file name. (This data does not need any changes for `rename` or is not part of the inode.)*

Mit welchem Mechanismus können solche Inkonsistenzen verhindert werden? Erklären Sie den Mechanismus kurz. **2 pt**

*Which mechanism could prevent these inconsistencies? Briefly explain the mechanism.*

**Lösung:**

*The file system could implement journaling to prevent such inconsistencies* **(1 P)**. *With journaling, changes such as the `rename` operation are written to a log* **(0.5 P)** *first. After a crash, the full operation can be replayed* **(0.5 P)** *from the log.*

*(Mentioning transactions instead of log is also good.)*

Vor der Speicheroperation wird ein Hardlink `config2.txt` angelegt, der auf `config.txt` zeigt. Welchen Dateiinhalt erhält ein Programm beim Auslesen von `config2.txt` nach der Speicheroperation? Erklären Sie. **1 pt**

*Before the save operation, the user creates a hardlink `config2.txt` pointing to `config.txt`. Which file contents would a program reading `config2.txt` obtain after the save operation? Explain.*

**Lösung:**

*The program will read the old file contents* **(0.5 P)**. *The rename operation overwrites the original directory entry, but other hard links to that inode stay valid* **(0.5 P)**.

c) Wofür steht die Abkürzung „DMA“? **0.5 pt**

*What does the abbreviation “DMA” stand for?*

**Lösung:**

*Direct Memory Access*

Ein Betriebssystem programmiert DMA-Geräte so, dass sie in den Speicher von Benutzerprogrammen schreiben. Begründen Sie, warum es dennoch möglich ist, den Adressraum während einer blockierenden I/O-Operation zu wechseln. **1 pt**

*An operating system programs DMA devices so that they write to the memory of user processes. Explain why it is still possible to switch the address space during a blocking I/O operation.*

**Lösung:**

*Changing the address space with a pending DMA operation is possible because the DMA device is programmed with physical addresses* **(1 P)** *that stay valid even if they are not mapped in the virtual address space.*

**Total: 12.0pt**