Karlsruher Institut für Technologie

Betriebssysteme
(Operating Systems)

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

Musterlösung Nachklausur
21.09.2016

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 16 Blättern: Einem Deckblatt und 15 Aufgabenblättern mit insgesamt 5 Aufgaben.
  The examination consists of 16 pages: One cover sheet and 15 sheets containing 5 assignments.
- Es sind keinerlei Hilfsmittel erlaubt!
  No additional material is allowed.
- Die Prüfung gilt als nicht bestanden, wenn Sie versuchen, aktiv oder passiv zu betrügen.
  You fail the examination if you try to cheat actively or passively.
- 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 und erläutern Sie einen Vor- und einen Nachteil der Trennung in Benutzer- und Kernelmodus in modernen Betriebssystemen. **2 pt**

Give and explain an advantage and a disadvantage of the separation into user- and kernel-mode in modern operating systems.

Lösung:

Pro: *The separation into user- and kernel-mode dramatically increases the* stability *of the system, because malfunctioning processes cannot inadvertently tamper with kernel code and data structures and thereby crash the system* **(0.5 P)**.

Pro: *The boundary increases the* security *of the system, because only proper protection of the kernel allows an operating system to perform trustworthy security checks (e.g., authentication and authorization)* **(0.5 P)**. *Private information such as login credentials are not leaked.*

Pro: *The system can also enforce* fairness *by preventing processes from hogging the CPU or otherwise misuse resources* **(0.5 P)**.

Contra: *Crossing the boundary comes at a cost, because architectural state needs to be saved and restored, and the CPU must switch to a different privilege level* **(0.5 P)**. *System calls are therefore noticeably slower than calls into a library.*

Other points are accepted, if properly reasoned.

b) Erläutern Sie, wie Systemaufrufe technisch umgesetzt werden. Gehen Sie dabei auf folgende Punkte ein: **4 pt**

Explain how system calls are technically implemented. Elaborate on the following aspects:

Lösung:

Switch to kernel mode: *The CPU usually provides a special (non-privileged)* trap *instruction* **(0.5 P)**, *which performs a switch to kernel mode and jumps to a previously configured kernel code location (the system call dispatcher). The address is set up by the operating system during system initialization* **(0.5 P)**.

System service selection: *When a program performs a system call, it requests a* particular *system service. To express which service it needs, the program provides the* system call number **(0.5 P)** *in a predefined location or register* **(0.5 P)**. *The number is usually used as an index into a function table.*

Parameter passing: *A limited number of parameters can be passed via CPU registers* **(0.5 P)**. *More parameters or data-types such as strings must be passed via the user stack* **(0.5 P)**.

Application Program Interface (API): *Since the calling convention of the system call interface as well as the system call numbers are operating system and potentially even version specific, the system call interface is usually accessed through* system libraries **(1 P)**. *This allows the OS to be updated without having to change applications. The system libraries often also implement additional functionality that eases application development.*

c) Erläutern Sie, warum eine zeitnahe und schnelle Bearbeitung von Interrupts durch das Betriebssystem wichtig ist. **2 pt**

Explain why timely and fast processing of interrupts by the operating system is important.

Lösung:

If the operating system does not promptly react to interrupts, the overall system performance becomes sluggish. This is because the lag **(1 P)** *between the occurrence of an event (e.g., a key press by the user) and the system's reaction is high – resulting for example in high input latency or cracking sound.*

Interrupts might also get lost, or more precisely, device buffers may exhaust and further input (such as network packets or key presses) gets lost **(0.5 P)**, *because during the processing of a particular interrupt, the interrupt is masked and will not be delivered* **(0.5 P)**.

d) Bei der Bearbeitung einer E/A-Anfrage tritt in einem Kerneltreiber ein unerwarteter Fehler auf, der vom Treiber nicht behandelt wird. Der Kernel könnte die E/A-Anfrage abbrechen und einen Fehler zurückgeben. Wieso kann es besser sein, das System zu stoppen, z.B. durch Bluescreen oder Kernel Panic? **2 pt**

During the processing of an I/O operation, an unexpected exception occurs in a kernel driver, which does not handle the exception. The kernel could cancel the I/O operation and return an error. Why might it be better to stop the system, for example by blue screen or kernel panic?

Lösung:

After an unexpected error the system is in an unknown condition as the error could have effects that are not directly visible, but ultimately lead to erratic system behavior such as data corruption or loss **(1 P)**. *The reliability of the system should therefore be considered compromised and an early stop might prevent additional damage* **(1 P)**.

e) Welche der folgenden Aussagen sind richtig? **2 pt**
Which of the following statements are correct?

korrekt/ *correct*	inkorrekt/ *incorrect*	
☐	■	Systemaufrufe werden ausschließlich zur Ausführung von privilegierten CPU Instruktionen benötigt. *System calls are only needed to execute privileged CPU instructions.*
☐	■	Der Heap wird üblicherweise an hohe Adressen gelegt und wächst in Richtung niedrigerer Adressen. *The heap is usually placed at high addresses and grows in the direction of lower addresses.*
☐	■	Größere physische Seiten bedingen mehr externe Fragmentierung. *Larger page frames cause more external fragmentation.*
■	☐	Port-basierte E/A verwendet einen eigenen Adressraum. *Port-mapped I/O uses a dedicated address space.*

Total: 12.0pt

Aufgabe 2: Prozesse und Threads

Assignment 2: Processes and Threads

a) Gegeben seien vier Prozesse auf einem Einprozessorsystem mit den angegebenen Ankunftszeiten (0 = Start) und Burst-Zeiten. Vervollständigen Sie die untenstehenden Ablaufpläne für die Strategie *Shortest Job First (SJF)* sowie die Strategie *Round Robin (RR)* mit der Zeitscheibenlänge von einer Zeiteinheit. Neue Prozesse werden ans Ende der Warteschlange angehängt. Ein Kasten im Zeitplan stellt eine Zeiteinheit dar. **4 pt**

Consider four processes on a uni-processor system, with given arrival times (0 = start) and burst times. Complete the scheduling plans given below for the policy Shortest Job First (SJF) *and the policy* Round Robin (RR) *with a timeslice length of one unit of time. New processes are added to the tail of the ready queue. A box in the scheduling plan represents one unit of time.*

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

Lösung:

Shortest Job First (SJF)

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

Round Robin (RR)

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

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

b) Berechnen Sie für den obigen *SJF-Ablaufplan* die Turnaround-Zeit aller Prozesse. **2 pt**

For the above SJF 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	*15*	*7,5*	*7,5*
2	*12*	*1,5*	*10,5*
3	*19*	*6,5*	*12,5*
4	*6*	*0*	*6*

c) Berechnen Sie für den obigen *RR-Ablaufplan* die Wartezeit aller Prozesse. **2 pt**

For the above RR 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	*18*	*7,5*	*3*	*7,5*
2	*17*	*1,5*	*6*	*9,5*
3	*19*	*6,5*	*4*	*8,5*
4	*12*	*0*	*6*	*6*

d) Aktuelle Scheduler verwenden auf SMP-Systemen eine eigene Warteschlange für jede CPU, um Synchronisierungsoverhead einzusparen. Beschreiben Sie ein Problem dieses Ansatzes gegenüber einer einzigen Warteschlange für alle CPUs, und schlagen Sie eine Lösung für dieses Problem. **2 pt**

On SMP systems, current schedulers use one ready queue per CPU to avoid synchronization overhead when accessing the queues. Describe a problem of this approach compared to a single ready queue shared by all CPUs, and suggest a solution for the problem.

Lösung:

If multiple ready queues are used, it is possible for one queue to run empty while others still contain threads ready to run. This results in poor system utilization since the empty queue causes one CPU to idle even though there is plenty of work to be done. **(1 P)**

A possible solution to this problem is for the idle CPU to check if the ready queues of other CPUs contain waiting threads, and if so, to pull some of these waiting threads into its own ready queue. **(1 P)** *This scheme is also known as work stealing.*

e) *Kooperatives Round-Robin-Scheduling* ist eine Variante von Round-Robin-Scheduling, bei der die Tasks nicht nach Ablauf einer festen Zeitscheibe verdrängt werden. Stattdessen wählt der Scheduler erst dann einen neuen Task aus, wenn der vorherige Task entweder blockiert (z.B. aufgrund von E/A) oder die CPU freiwillig abgibt. Die Auswahl des nächsten Tasks funktioniert dann wie beim nicht-kooperativem Round-Robin-Scheduling, das in der Vorlesung vorgestellt wurde.

Beschreiben Sie einen Vor- und einen Nachteil von kooperativem Round-Robin-Scheduling gegenüber nicht-kooperativem Round-Robin-Scheduling. Gehen Sie davon aus, dass alle Tasks nach endlicher Rechenzeit eine E/A-Operation ausführen.

2 pt

Cooperative Round-Robin-Scheduling *is a variant of Round-Robin-Scheduling, in which tasks are not preempted at the end of a fixed timeslice. Instead, the scheduler selects a new task to run only after the previous task has either blocked (e.g., due to I/O) or yielded the CPU voluntarily. Selection of a new task then works the same way as in non-cooperative Round-Robin-Scheduling, presented in the lecture.*

Describe an advantage and a disadvantage of cooperative Round-Robin-Scheduling compared to non-cooperative Round-Robin-Scheduling. Assume that all tasks perform I/O after a finite amount of computation.

Lösung:

The main advantage of non-cooperative Round-Robin over cooperative Round-Robin is that the former distributes the available CPU time fairly: Each task receives the same number of time slices on average, with each time slice having the same length. With cooperative Round-Robin, tasks run for as long as they can without blocking, resulting in tasks blocking less often receiving more CPU time.

Even assuming that all tasks block for I/O after a finite amount of computation, these tasks may still take a long time to do so. This may lead to a convoy effect, where many tasks must wait for one long-running task at the head of the queue. Therefore, cooperative round-robin potentially leads to long waiting times and thus poor interactivity.

*(***(1 P)***, one answer is sufficient)*

On the other hand, a non-cooperative Round-Robin scheduler frequently preempts all running tasks, possibly at inconvenient times. In contrast, a cooperative Round-Robin scheduler switches away from a running task only when it is convenient for that task to do so (e.g., when that task must wait for I/O anyway). Therefore, the main advantage of a cooperative Round-Robin scheduler is a much lower scheduler overhead. **(1 P)**

Total: 12.0pt

Aufgabe 3: Koordination und Kommunikation von Prozessen

Assignment 3: Process Coordination and Communication

a) Betrachten Sie das in Abbildung 1 dargestellte System und die zukünftigen Systemaufrufe der zwei Prozesse in Listing 1. Kann in diesem Szenario ein Deadlock auftreten? Falls nein, begründen Sie warum. Falls ja, geben Sie einen denkbaren Ablauf an, wie von der dargestellten Situation ein Deadlock erreicht werden kann.

2 pt

Consider the system depicted in Figure 1 and the two processes' future system calls in Listing 1. Can a deadlock arise from this scenario? If not, explain why. Otherwise, describe how the system might reach a deadlock from the depicted scenario.

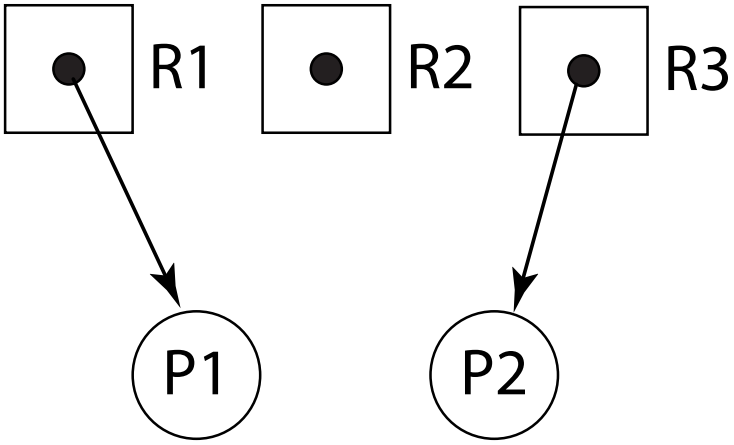


Abbildung 1 / *Figure 1*

```
P1: acquire(R2)      P2: acquire(R2)
    release(R1)          release(R3)
    acquire(R3)          release(R2)
    release(R2)          acquire(R1)
    acquire(R3)          release(R1)
```

Listing 1

Lösung:

Yes. **(0.5 P)**.

*(a) P1 acquires R2 (***(0.5 P)** *needs to happen first)*

(b) P2 tries to acquire R2, is blocked waiting for that resource. **(0.5 P)**

(c) P1 releases R1 (does not matter)

(d) P1 tries to acquire R3, is blocked waiting. **(0.5 P)**

After R2 has been acquired by P1, P2 began waiting for R2, and P1 began waiting for R3, the system is in deadlock. P1 and P2 are both blocked waiting for a resource held by the other process, resulting in a circular dependency.

b) Nennen Sie zwei der Gegenmaßnahmen für Deadlocks aus der Vorlesung. Beschreiben Sie kurz die Funktion der von Ihnen genannten Mechanismen. Was könnten diese im obigen Szenario beispielhaft bewirken? **4 pt**

Name two of the deadlock countermeasures that were discussed in the lecture. Briefly describe how these two mechanisms work. Give an example of what they might do in the scenario above.

Lösung:

- ***deadlock prevention*** **(0.5 P)**
  Negate at least one of the required deadlock conditions, mutual exclusion, hold and wait, no preemption, or circular wait, **(0.5 P)** *so that deadlocks cannot occur.* **(0.5 P)**
  In the given scenario, we could break circular wait by ordering the resources and permitting allocation only in that order. As a consequence, P2 would have to be rewritten to acquire R2 before R3 because requesting R2 while holding R3 would be an illegal operation **(0.5 P)**.
- ***deadlock avoidance*** **(0.5 P)**
  Track available resources and resource allocations (e.g., using a resource allocation graph (RAG)). Decide whether the system is in a safe state each time a process requests a resource. **(0.5 P)** *Only grant the request if the system remains in a safe state after the allocation.* **(0.5 P)**
  In this scenario, deadlock avoidance would block P1's request to acquire R2 because it would lead to an unsafe state. Instead, P2 would continue running and thus acquire R2 first., keeping the system in a safe state **(0.5 P)**.
- ***deadlock detection (and recovery)*** **(0.5 P)**
  Maintain wait-for graph (WFG) for the system. Periodically search for cycles in the graph, indicating a deadlock. **(0.5 P)** *Once a deadlock has been found, resolve it by aborting processes (and thereby forcefully releasing their resources) until the deadlock is resolved.* **(0.5 P)**
  In the given scenario, the system would continually update the RAG, derive the WFG, and eventually detect a cycle between P1 and P2. Then, it might abort P2, detect that the deadlock has been resolved, and let P1 continue (or vice versa). **(0.5 P)**

(two of the above, you cannot get more points by introducing a third mechanism)

c) Wie können Semaphoren für die Synchronisation kritischer Abschnitte eingesetzt werden? Geben Sie an, welche Operationen ein Thread hierzu beim Ein- und Austritt eines kritischen Abschnitts auf der Semaphore aufrufen muss. Warum erfüllt der Ansatz wechselseitigen Ausschluss *(mutual exclusion)*? **2 pt**

How can you use semaphores for synchronizing critical sections? What operations will threads have to call on the semaphore when entering and leaving a critical section? Why does this approach fulfill mutual exclusion?

Lösung:

When we initialize a semaphore to the value 1 **(0.5 P)**, *we can use it for providing mutual exclusion for a critical section. Such a semaphore is often called* ***binary semaphore*** *or* ***mutex.***

Threads trying to enter a critical section will call `wait` *on the semaphore* **(0.5 P)**, *while threads leaving a critical section will call* `signal` **(0.5 P)**.

Threads can only enter when the semaphore has the value 1. When there is already a thread in the critical section, having decremented the semaphore to 0, threads subsequently calling `wait` *will have to sleep* **(0.5 P)**.

d) Erklären Sie, welcher Typ von Semaphore für die Synchronisation eines kritischen Abschnitts zu wählen ist, damit auch die Eigenschaft beschränktes Warten *(bounded waiting)* erfüllt wird. **1 pt**

Explain which type of semaphore you have to choose for synchronizing critical sections to also achieve the property of bounded waiting.

Lösung:

A strong semaphore **(0.5 P)** *will wake up threads in the order in which they were put to sleep and thereby provide bounded waiting: Once a thread called* `wait` *to enter a critical section, only the threads already queued sleeping in the semaphore may enter the critical section before it. No thread arriving later my bypass the sleeping threads* **(0.5 P)**.

e) Unter welchen Voraussetzungen funktioniert das Deaktivieren von Interrupts als Synchronisationsmechanismus? Begründen Sie Ihre Antwort. **1 pt**

In which circumstances may disabling interrupts work as a synchronization mechanism? Explain why.

Lösung:

You may implement synchronization by disabling interrupts only on single processor systems **(0.5 P)**. *On a multi processor system, disabling interrupts (even globally) will not keep other CPUs from entering a critical section and therefore cannot prevent race conditions* **(0.5 P)**.

f) Für die Implementierung von Synchronisationsprimitiven werden in der Regel das Deaktivieren von Interrupts oder atomare Instruktionen eingesetzt. Darf ein Betriebssystem Anwendungen im *Benutzermodus* gestatten, diese beiden Mechanismen zu verwenden? Begründen Sie jeweils Ihre Antwort. **2 pt**

Deactivating interrupts and atomic instructions are commonly used for implementing synchronization primitives. May an operating system allow user space applications to use these two mechanisms? Explain your answer.

Lösung:

Disabling interrupts: *No* **(0.5 P)**. *Applications might never reactivate them and thereby hog the CPU, circumventing the operating system's scheduling policy.*

Atomic instructions: *Yes* **(0.5 P)**. *Atomic instructions can only modify a thread's register contents and its process's virtual address space, which is accessible to the application anyway. While an atomic instruction might take longer than a regular memory access, it cannot keep the OS from regaining control of the CPU* **(0.5 P)**. **Total: 12.0pt**

Aufgabe 4: Speicher

Assignment 4: Memory

a) Zwei Prozesse teilen sich globale Daten über Copy-On-Write (COW). Müssen sich die Prozesse beim Zugriff auf diese Daten synchronisieren? **0.5 pt**

Two processes share global data with copy-on-write (COW). Do the processes have to synchronize their access to the data?

Lösung:

☐ *Ja / Yes* ■ *Nein / No*

b) Ein Prozess ruft `fork()` auf. Beschreiben Sie für den Eltern- und den Kindprozess, wie der jeweilige Adressraum konfiguriert werden muss, um Copy-On-Write (COW) zu ermöglichen. Begründen Sie Ihre Antwort. **1.5 pt**

A process calls `fork()`. Describe for the parent and the child processes, how each address space has to be configured to allow copy-on-write (COW). Explain you answer.

Lösung:

Both, the parent and the child processes, map to the same page frames. Therefore, all shared parts of the address spaces need to be write-protected **(0.5 P)**. *This way, a write access triggers a page-fault and the operating system is invoked, which can create a private copy* **(0.5 P)**. *Since modifications from both processes should not be mutually visible, both address spaces need to be configured in the same way* **(0.5 P)**.

c) Erläutern Sie inwieweit die Auslagerung von Seitentabellen durch das Betriebssystem möglich ist. Gehen Sie davon aus, dass Seitentabellen in Hardware durchlaufen werden. **3 pt**

Explain to what extent the operating system can page out page tables. Assume hardware page table walking.

Lösung:

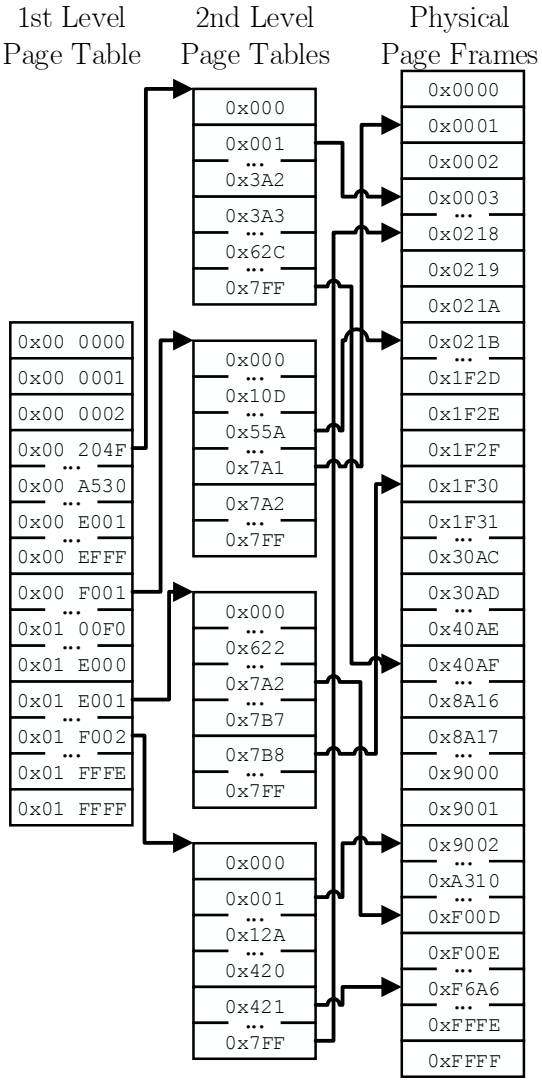
If a process runs, its entire page table hierarchy can be paged out, except

- *the page directory (i.e., the first level)* **(0.5 P)**. *The page directory is needed so that the MMU is given a valid first page table* **(0.5 P)**. *As most PTEs will not have the valid bit set, a page-fault will be triggered that allows the OS to page in further page tables as needed.*
- *(most) shared kernel page tables* **(0.5 P)**. *At least a certain set of shared kernel pages must be accessible without triggering a page-fault for paging in further page tables and pages (e.g., page-fault handler, I/O code to load data, certain drivers, etc.)* **(0.5 P)**.

If the process is not running, its page directory can also be paged out **(0.5 P)**, *because the MMU will not access it until the process is activated again* **(0.5 P)**.

d) Gegeben sei folgende zweistufige, hierarchische Seitentabelle. Die Größe einer Seite betrage 1 MiB. Geben Sie die Größe des virtuellen Adressraumes (VAS) in Bytes an. Zerlegen und Übersetzen Sie anschließend die virtuelle Adresse `0xF000FA24FC42`. **5 pt**

Consider the following two-level, hierarchical page table. The page size is 1 MiB. Give the size of the virtual address space (VAS) in bytes. Then, split and translate the virtual address `0xF000FA24FC42`.



Lösung:

The 1st page table contains 2^{17} *(*$0x1FFFF + 1$*) entries. The 2nd level page tables contain* $2048 = 2^{11}$ *(*$0x7FF + 1$*) entries each. This gives* $2^17 * 2^11 = 2^{28}$ *pages. Each page is* $2^{20} = 1$ *MiB in size. The address space is thus* $2^{28} * 2^{20} = 2^{48}$ *bytes large* **(1 P)**.

For 1 MiB pages, we need 20 bits for the offset. The given page table requires 17 bits for the first and 11 bits for the second index. By splitting the virtual address accordingly, we get:
1st Level Page Table Index: `0x1E001` **(1 P)**
2nd Level Page Table Index: `0x7A2` **(1 P)**
Offset: `0x4FC42` **(1 P)**

To get the physical address, we follow the page tables to the PFN: `0xF00D` **(0.5 P)**.
The PFN is than concatenated on the bit-level with the offset: `0xF00D4FC42` **(0.5 P)**

e) Wofür steht die Abkürzung TLB? Erläutern Sie, warum ein TLB für eine schnelle Programmausführung essentiell ist. **1 pt**

What does the abbreviation TLB stand for? Explain why a TLB is essential for fast program execution.

Lösung:

Translation-Lookaside-Buffer **(0.5 P)**. *A TLB caches virtual-to-physical address translations and thereby drastically reduced the number of times that the MMU has to do costly page table walks* **(0.5 P)**.

f) Nennen Sie eine Lösung für das Problem der Mehrdeutigkeit bei virtually-indexed, virtually-tagged (VIVT) Caches. Begründen Sie Ihre Antwort. **1 pt**

Give a solution for the ambiguity problem in virtually-indexed, virtually-tagged (VIVT) caches. Explain your answer.

Lösung:

The ambiguity problem arises, when the same virtual address points to two different physical addresses at two points in time; the cache, however, does not recognize this, because it works with virtual addresses **(0.5 P)**. *The operating system then has to invalidate respective cache lines, whenever the mapping changes* **(0.5 P)**. **Total: 12.0pt**

Aufgabe 5: I/O, Hintergrundspeicher und Dateisysteme

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

a) Welche drei Benutzerklassen kennt das traditionelle Modell der UNIX-Zugriffskontrolle? Was ist der Vorteil von ACLs gegenüber diesem Modell im Bezug auf die Differenzierung zwischen einzelnen Nutzern? **2 pt**

Which three classes of users are present in the traditional UNIX access control model? What advantage do ACLs have compared to this model in terms of differentiating between users?

Lösung:

User classes: User/owner, group, others **(1 P)** *(***(0.5 P)** *if one is missing/incorrect)*

ACLs allow an arbitrary number of users to have different access rights, compared to only three classes of users in the traditional model **(1 P)**.

b) Woran erkennt das Betriebssystem nach dem Löschen eines Hard Links, dass der zugehörige Inode freigegeben werden kann? **1 pt**

When a hard link is removed, how does the operating system detect that the corresponding inode can be deleted?

Lösung:

Each inode has a reference counter **(0.5 P)**, *which counts the number of hard links to the file. When the counter reaches zero, the inode can be deleted* **(0.5 P)**.

c) Gegeben sei ein Dateisystem, das nur jene Blöcke einer Datei physisch alloziert, welche auch beschrieben wurden; unabhängig von der Größe der Datei im Dateisystem. Welche in der Vorlesung vorgestellte Allokationsstrategie eignet sich für dieses Dateisystem am besten? Begründen Sie Ihre Antwort. **2 pt**

Consider a file system, which physically allocates only those blocks of a file that have been written; irrespective of the file's size in the file system. Which of the allocation policies presented in the lecture is most suited for this file system? Explain your answer.

Lösung:

Indexed allocation is the most suited allocation policy for sparse files **(1 P)**. *The corresponding index entries are simply set to an illegal value to show that the corresponding block has not been physically allocated yet* **(1 P)**.

d) Erläutern Sie, warum das abrupte Ausschalten eines Computers zum Verlust kürzlich geschriebener Dateiinhalte führen kann, obwohl die zugehörigen Systemaufrufe zum Schreiben erfolgreich abgeschlossen wurden. Wie verhindert Journaling in so einem Fall die Beschädigung der Dateisystemstruktur? **3 pt**

Explain why a sudden computer switch-off can lead to the loss of recently written file data, although the corresponding write system calls have completed successfully. How does journaling prevent the corruption of the file system in such a case?

Lösung:

Buffer cache/page cache/file system cache **(0.5 P)***: The write system call only places data in the cache, and the data is then written to disk asynchronously. A sudden switch-off can prevent recent changes from being written to disk* **(0.5 P)***.*

The file system can be corrupted if updates are only partially written to disk. Journaling file systems prevent partial updates by implementing updates as transactions **(0.5 P)***: The updates are first atomically written to a log, and then later applied to the file system* **(0.5 P)***. Transactions are removed from the log again when they are completely applied. After a sudden switch-off, all transactions in the log are performed again to ensure that they are completely applied to the file system* **(1 P)***.*

e) Was ist der wichtigste Vorteil einer File Allocation Table (FAT) gegenüber Chained Allocation? **1 pt**

What is the main advantage of a file allocation table (FAT) compared to chained allocation?

Lösung:

Compared to the linked list used in chained allocation, the FAT is more compact and can thus be cached more easily **(0.5 P)***. This potentially reduces the number of disk accesses when a block in the middle of a file is requested* **(0.5 P)***.*

*Alternative: With a FAT, finding a block in the middle of a file does not require traversing all file blocks (***(0.5 P)***). Instead, the file system driver only reads the corresponding FAT entries. The FAT is more compact, so these read operations cause fewer disk seeks (***(0.5 P)***).*

f) Wie viele Bits muss ein Eintrag in einer FAT mindestens haben, wenn die FAT ein 1 MiB großes Dateisystem in 256 Bytes großen Blöcken verwaltet? Begründen Sie Ihre Antwort. **2 pt**

How many bits are required for every FAT entry if the FAT is used to manage a 1 MiB file system with 256 byte blocks? Explain your answer.

Lösung:

Each entry of the FAT is the index of a block in the file system. The specified file system contains $\frac{2^{20}}{2^8} = 2^{12}$ *blocks* **(1 P)***, so every FAT entry needs 12 bits* **(1 P)***.*

g) Warum führt Programmed I/O üblicherweise zu hoher Prozessorauslastung, wenn das Betriebssystem schnell auf Eingaben von Geräten reagieren soll? **1 pt**

Why does programmed I/O cause high CPU utilization if the OS should quickly react to device input?

Lösung:

To be able to quickly react to input from devices, the driver needs to poll **(0.5 P)** *the device frequently to check its state: The CPU is busy-waiting and, in a loop, reading the status of the I/O device or operation* **(0.5 P)**, *therefore CPU utilization is high.*

Total: 12.0pt