Karlsruher Institut für Technologie

Betriebssysteme
(Operating Systems)

Prof. Dr.-Ing. Michael Beigl
Alex Neumann, M.Sc.

2 215000 007470

Musterlösung

18. 03. 2013

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 dem Konzeptblatt) 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 the draft page).
- Die Prüfung besteht aus 15 Blättern: Einem Deckblatt und 14 Aufgabenblättern mit insgesamt 5 Aufgaben.
  The examination consists of 15 pages: One cover sheet and 14 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 contradicting answers are void (0 points).
- Wir werden Punkte abziehen, falls korrekte Antworten auch inkorrekte oder irrelevante Informationen enthalten. Bitte schreiben Sie nicht einfach möglichst viel hin, in der Hoffnung, das richtige Schlagwort zu treffen.
  We will take off points if a correct answer also includes incorrect or irrelevant information. Do not write down everything you know in hopes of saying the correct buzz word.

Die folgende Tabelle wird von uns ausgefüllt! *The following table is completed by us!*

Aufgabe	1	2	3	4	5	Total
Max. Punkte	11	12	13	12	12	60
Erreichte Punkte						
Note						

Aufgabe 1: Zum Aufwärmen/*Assignment 1: Warmup*

a) Nennen Sie zwei plattform-unabhängige ganzzahlige Datentypen aus `stdint.h` und geben Sie für beide die größte darstellbare Zahl an. **2 pt**

Name two platform-independent integer data types defined by `stdint.h` and state their largest representable number.

Lösung:

- *uint8_t, 0xFF*
- *int16_t, 0x7FFF*
- *...*

b) Welche der folgenden Anweisungen sind korrekt, um den Wert `0x00133700` an die Speicheradresse `0x00733100` zu schreiben? **1 pt**

Which of the following are correct statements to write the value of `0x00133700` to memory address `0x00733100`?

Lösung:

☐	`uint32_t * addr = (uint32_t *)0x00733100; addr = 0x00133700;`
■	`* (uint32_t * const)0x00733100 = 0x00133700;`
☐	`* (uint32_t const)0x00733100 = 0x00133700;`

c) **Nennen** Sie drei Möglichkeiten, Parameter an einen Betriebssystem-Aufruf zu übergeben. **1 pt**

***Name** three ways in which to hand parameters to an operating system call.*

Lösung:

- *Via registers of the CPU*
- *Via the user-level program stack*
- *Via dedicated main memory regions*
- *-0.5P für jeden falschen, fehlenden Eintrag*

d) Beschreiben Sie knapp den Unterschied zwischen **Multiprogramming** und **Multitasking**. (Gehen Sie von einem Uniprozessorsystem aus.) **1 pt**

*Briefly describe the difference between **multiprogramming** and **multitasking**. (Assume a uniprocessor system.)*

Lösung:

- *With multiprogramming, the operating system switches between jobs/processes at OS-specific program conditions, e.g. when the currently running job/process is waiting for I/O. (0.5P)*
- *With multitasking, the operating system can switch between jobs/processes independent of any program conditions/design. (0.5P) When switching at a high rate, interactivity of the jobs/processes is fostered.*

e) Welche der folgenden Aussagen sind korrekt, welche sind inkorrekt? **6 pt**
(falsches Kreuz: -1P, kein Kreuz: 0P, korrektes Kreuz: 1P)

Which of the following statements are correct, which are incorrect?
(incorrectly marked: -1P, not marked: 0P, correctly marked: 1P)

korrekt *correct*	inkorrekt *incorrect*	
☐	■	DMA bietet **immer** eine geringere Latenz als unterbrechungsgetriebene E/A. *DMA **always** outperforms interrupt-driven I/O in terms of latency.*
☐	■	Nach der write-back-Strategie werden veränderte Daten im Cache **nur in Folge eines write misses** zurückgeschrieben. *According to the write-back policy, modified data in a cache are written back **only in consequence of a write miss**.*
☐	■	Ringförmige Zwischenspeicher **lösen** das Erzeuger-Verbraucher-Problem zwischen E/A-Geräten und Anwendungen. *Circular-buffering **solves** the producer-consumer problem between I/O devices and applications.*
☐	■	Wenn das `SUID`-Bit eines Verzeichnisses gesetzt ist, können Programme innerhalb des Verzeichnisses nur vom **Besitzer des Verzeichnisses** ausgeführt werden. *If the `SUID` bit is set on a directory, only the **owner of that directory** is allowed to execute programs contained in it.*
■	☐	Unter Unix enthält nur der Eintrag in der system-weiten Tabelle geöffneter Dateien die aktuelle Position des Zugriffs (*seek position*) einer geöffneten Datei. Daher stören sich geforkte Prozesse gegenseitig, wenn sie gleichzeitig auf einer Datei über demselben globalen Eintrag arbeiten. *In Unix, only the entry in the system-wide table of open files contains the current seek position of an open file. Therefore, forked processes interfere with each other when working in parallel on a file via the same global entry.*
■	☐	Auf C99-konformen Systemen entspricht die Länge (in Bytes) eines Zeigers (z.B. `unsigned integer *`) **nicht** immer der des `unsigned integer`-Datentypens. *On C99-compliant systems, the size (in bytes) of a pointer (e.g. `unsigned integer *`) is **not** always equivalent to the size of the `unsigned integer` datatype.*

Total: 11.0pt

Aufgabe 2: Prozesse und Ablaufplanung/*Assignment 2: Processes and Scheduling*

a) Nennen Sie **vier** in der Vorlesung besprochene Bestandteile eines Prozesskontrollblocks. **1 pt**

*Name **four** elements of a Process Control Block discussed in the lecture.*

Lösung:

- *Process state*
- *Process identifier*
- *Program counter*
- *CPU registers*
- *CPU scheduling information*
- *Memory-management information*
- *List of open files*
- *...*
- *-0.5P für jeden falschen, fehlenden Eintrag*

b) Erklären Sie kurz, **wann** der Kontrollblock eines Prozesses ausschließlich gültige Informationen enthält und **wann nicht**. **1 pt**

*Explain briefly **when** the control block of a process contains entirely valid information and **when it does not**.*

Lösung:

- *When a process is not in execution, its PCB contains valid information only. (0.5P)*
- *When a process is in execution, its PCB may contain invalid information, e.g. CPU registers. (0.5P)*

c) Nennen Sie die drei **Prozesstypen**, beschreiben Sie deren typische **Reaktionsfreudigkeit** und **Gebundenheit** (E/A, CPU). **3 pt**

*Name the three **process types** and describe their typical **responsiveness** and **boundness** (I/O, CPU).*

Lösung:

Typ Type	*Gebundenheit* Boundness	*Reaktionsfreudigkeit* Responsiveness
Interactive processes	*IO-bound*	*Highly responsive to comfort users (low variance)*
Batch processes	*CPU-bound*	*No need to be responsive (no user interaction)*
Real-time processes	*IO-bound or CPU-bound*	*Requirements/contraints on responsiveness (bound variance) to achive QoS*

- *-0.5P für jeden falschen, fehlenden Eintrag*

d) Gegeben seien die untenstehenden 5 Prozesse auf einem Uniprozessorsystem mit den angegebenen Ankunftszeiten (Start-Zeit = 1.0) und Burst-Zeiten. Vervollständigen Sie den untenstehenden Ablaufplan für die Strategie **Virtual Round Robin** (VRR). Ein Kasten im Zeitplan entspricht einer Zeitscheibe.

Beachten Sie: Der Prozess P_5 blockiert nach der Hälfte des Bursts (nach einer halben Zeitscheibe) und ist nach einer Dauer von 2.0 Zeitscheiben wieder bereit. **3 pt**

*Consider the 5 processes given below on a uniprocessor system, with the given arrival times (start time = 1.0) and burst times. Complete the schedule given below according to the policy **Virtual Round Robin** (VRR). A box in the schedule represents one time slice/quantum.*

***Attention**: the process P_5 blocks after half its burst (after half a time slice) and is ready again after a duration of 2.0 time slices.*

Prozess *Process*	Ankunft-Zeit *Arrival time*	Burst-Zeit *Burst time*
P_1	1.9	7.0
P_2	1.0	4.0
P_3	5.9	1.0
P_4	5.8	6.0
P_5	6.1	1.0

Lösung:

- *The first column numbers the time slices.*
- *The second column holds the schedule. Each cell indicates the process running in that time slice. In parentheses the remaining time of the process' burst after the current time slice is given.*
- *The third column holds the state of the ready queue when the scheduling decision is taken for that time slice.*
- *The fourth column is analogous for the auxiliary queue.*

Zeit *Time*	*Ablaufplan* *Schedule*	*Ready Queue* Ready queue	*Auxiliary Queue* *Auxiliary queue*
1	*2*	*2*	
2	*1*	*1 2*	
3	*2*	*2 1*	
4	*1*	*1 2*	
5	*2*	*2 1*	
6	*1*	*1 4 3 2*	
7	*4*	*4 3 2 5 1*	
8	*3*	*3 2 5 1 4*	
9	*2*	*2 5 1 4*	
10	*5*	*5 1 4*	
11	*1*	*1 4*	
12	*4*	*4 1*	
13	*5*	*1 4*	*5*
14	*1*	*1 4*	
15	*4*	*4 1*	

- *-0.5P für jeden falschen, fehlenden Eintrag*
- *Wenn der komplette Schedule teilweise korrekt ist, ein Fehler erfolgte und er davon ausgehend plausibel ausgefüllt wurde, gibt es Folgefehler.*

Berechnen Sie die **durchschnittliche Wartezeit** T^W_{avg} für den obigen Ablaufplan. (Ergebnisse in Bruchform sind ausreichend.) **1 pt**

*Calculate the **average waiting time** T^W_{avg} for the above schedule. (Results in fraction form are sufficient.)*

Lösung:

$$T^W_{P_1} = 0.1{+}9 = 9.1, T^W_{P_2} = 5.0, T^W_{P_3} = 0.1{+}2.0 = 2.1, T^W_{P_4} = 0.2{+}7.0 = 7.2, T^W_{P_5} = 0.9{+}3.0{+}0.5 = 4.4$$

$$T^W_{avg} = \frac{T^W_{P_1} + T^W_{P_2} + T^W_{P_3} + T^W_{P_4} + T^W_{P_5}}{5} = \frac{9.1 + 5.0 + 2.1 + 7.2 + 4.4}{5} = \frac{278}{50} = 5.56$$

Berechnen Sie die **durchschnittliche Turnaround-Zeit** T^T_{avg} für den obigen Ablaufplan. (Ergebnisse in Bruchform sind ausreichend.) **1 pt**

*Calculate the **average turnaround time** T^T_{avg} for the above schedule. (Results in fraction form are sufficient.)*

Lösung:

Process	*Arrival time*	*Finish time*	*Turnaround time*
P_2	*1.0*	*10.0*	*9.0*
P_3	*5.9*	*9.0*	*3.1*
P_5	*6.1*	*13.5*	*7.4*

$$T^T_{P_2} = 9.0, T^T_{P_3} = 3.1, T^T_{P_5} = 7.4$$

$$T^T_{avg} = \frac{T^T_{P_2} + T^T_{P_3} + T^T_{P_5}}{3} = \frac{9.0 + 3.1 + 7.4}{3} = \frac{195}{30} = 6,5$$

e) Beschreiben Sie knapp die Fairness-Strategie nach welcher der **Linux Completely Fair Scheduler** versucht, immer wenn er aufgerufen wird, die Interaktivität von Prozessen zu gewährleisten? **2 pt**

*Explain briefly according to which fairness strategy the **Linux Completely Fair Scheduler** tries to provide interactivity of processes each time it is invoked?*

Lösung:

- *On invocation, whatever time has passed, to establish fair scheduling, each process must have had a proportional share of the CPU over its entire lifetime. If a process did not get its share during this time, it has* unfairly waited*, and is to be executed next. (1P)*
- *This behaviour fosters interactiveness of I/O-bound processes, as CPU-bound processes have a generally lower priority. (1P)*

Total: 12.0pt

Aufgabe 3: Prozesskoordination und -kommunikation/*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**

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

Lösung:

- *Mutual exclusion: only one thread can be in a critical section at a time.*
- *Progress:*
  - *If no process is in a critical section, one of the processes trying to enter it will eventually get in.*
  - *Processes that are not trying to enter a critical section do not hinder processes that try to enter it from getting in.*
- *Bounded waiting: once a thread starts trying to enter a critical section, there is a bound on the number of times other threads get in.*

b) Erklären Sie warum es auf Multiprozessor-Systemen im Allgemeinen nicht ausreicht, Unterbrechungen zu deaktivieren um das Problem kritischer Abschnitte zu lösen. Welche Anforderung an eine gültige Lösung wird nicht erfüllt? **2 pt**

Explain why disabling interrupts does not generally suffice to solve the critical section problem on multiprocessor systems. Which requirement for a valid solution is not satisfied?

Lösung:

- *On multiprocessor systems, disabling interrupts of a processor is not sufficient to synchronize and provide mutual exclusion (1P) between them in entering critical sections. As threads are running truely in parallel (on different cores), at any time, threads can be in a critical section at the same time (with interrupts disabled though). (1P)*

c) Ein Spinlock kann wie folgt auf Basis einer atomaren `swap`-Instruktion implementiert sein.

A spinlock might be implemented based on an atomic `swap` instruction as follows.

```
1  int volatile spinlock;
2
3  void lock(int volatile * spinlock) {
4      int key = 1;
5
6      while ( *spinlock != 0 ) ;
7      while( key )
8          Swap(spinlock, &key);
9  }
10
11 void unlock(int volatile * spinlock) { *spinlock = 0; }
```

Erklären Sie knapp, warum die Variable `spinlock` als `volatile` deklariert ist. **1 pt**

Explain briefly why the variable `spinlock` is declared `volatile`.

Lösung:

- *It is volatile to make the running code fetch the current value from memory (which might be cached) on every read. (1P)*

Unter welchem Umständen **performt** diese Spinlock-Implementierung schlecht wenn mehrere Threads folgenden Code ausführen? **2 pt**

*Under which circumstances does this spinlock implementation exhibit poor **performance** if multiple threads run the following code?*

```
void thread() {
    while (1) {
        lock(&spinlock);
        // critical section
        unlock(&spinlock);
        // remainder section
    }
}
```

Lösung:

- *In case all cores of a multi-core have dedicated caches on the shared memory (0.5P) and the spinlock is held by one thread (running on one core) and more than one more thread (running on other cores) try to acquire the lock (0.5P), continuous cache-line invalidations can be caused. (1P)*

Erklären Sie welche einfache Änderung im Code das beschriebene Performanz-Problem löst, und **warum**. **2 pt**

*Explain briefly, which minor change in the code solves the illustrated performance issue, and **why**.*

Lösung:

```
1  int volatile spinlock = 0; /* initialize to free */
2
3  void lock(int volatile * spinlock) {
4      int key = 1; /* occupied state */
5
6      while (key) { /* while lock is held */
7          while ( *spinlock != 0 ) ; /* while lock is held */
8
9          Swap(spinlock, &key); /* try to acquire lock */
10     }
11 }
12
13 void unlock(int volatile * spinlock) {
14     *spinlock = 0; /* release lock */
15 }
```

- *Situation revisited: in case the spinlock is held by one thread (running on one core) and more than one more thread (running on other cores) try to acquire the lock.*
- *To fix the performance problem: exchange the* `while`*-loops. (1P)*
- *Problem is gone now*
  - *When these thread try to acquire the lock, in the first place, they are busy waiting. As the threads only read the* `spinlock` *value, no performance issue arise so far.*
  - *When the current thread releases the lock by re-setting* `spinlock`*, again the other threads may step forward truely in parallel. One thread will acquire the lock, and the others will fall back to busy waiting. (1P)*

d) Erweitern Sie die in der vorherigen Teilaufgabe gegebenen `lock`- und `unlock`-Funktionen, so dass sie im Kontext des folgenden Beispiels alle drei notwendigen Bedingungen aus Teilaufgabe (a) erfüllen. **3 pt**

Extend the `lock` *and* `unlock` *functions given in the previous sub-task, such that they fullfil all three necessary conditions of sub-task (a) in the context of the following example.*

```
int is_waiting(thread_id_t i);
int set_waiting(thread_id_t i, int wait_state);
thread_id_t get_next_waiting(thread_id_t i);

void thread(thread_id_t id) {
    while (1) {
        lock(&spinlock, id);
        // critical section
        unlock(&spinlock, id);
        // remainder section
    }
}

void lock(int volatile * spinlock, int id) {
    int key = 1;

    set_waiting(id, 1);
```

```
    set_waiting(id, 0);
}

void unlock(int volatile * spinlock, int id) {
    thread_id_t next = get_next_waiting(id);

    if (next == id)
```

```
}
```

Lösung:

```
1  void lock(int volatile * spinlock, int id) {
2      int key = 1;
3
4      set_waiting(id, 1);
5
6      while (is_waiting(id) && key) {
7          while ( is_waiting(id) && *spinlock != 0 ) {};
8
9          Swap(spinlock, &key);
10     }
11
12     set_waiting(id, 0);
13 }
14
15 void unlock(int volatile * spinlock, int id) {
16     thread_id_t next = get_next_waiting(id);
17
18     if (next == id) {
19         spinlock = 0;
20     } else {
21         set_waiting(next, 0);
22     }
23 }
```

Total: 13.0pt

Aufgabe 4: Speicher und Caches/*Assignment 4: Memory and Caches*

a) Ausgehend von einem typischen Speicherlayout für Prozesse: Für welche der folgenden Sektionen eines Prozesses wird **anonymer Speicher** alloziert? Wofür kann ein Betriebssystem beim Swapping das Wissen ausnutzen, dass es sich bei einer Seite um **nicht-anonymen Speicher** handelt? **2 pt**

*Assume a common memory layout for processes. For which of the following sections of a process is **anonymous memory** allocated? When swapping, how can an operating system take advantage of the knowledge that a page contains **non-anonymous memory**?*

Lösung:

■	*Stack*
■	*Heap*
□	*Text*
■	*BSS*

- *Anonymous memory is not associated with a specific file. It is e.g. used for process stacks or heap pages. When a page of anonymous memory is evicted, the page has to be written to disk-based swap space for later retrieval.*
  Therefore, when a page of non-anonymous memory is chosen to be evicted, that page can be discarded and later be read from the appropriate file again. By doing so, unnecessary writes to the swapping area can be avoided. (1P)

- *-0.5P für jeden falschen, fehlenden Eintrag*

b) Erklären Sie knapp, was man unter **interner** und **externer Fragmentierung** im Kontext von Speicherallokation versteht. Beschreiben Sie **jeweils eine Situation**, in der dies auftreten kann. **2 pt**

*Briefly explain the terms **internal** and **external fragmentation** in the context of memory allocation. For **each one**, describe **a situation** in which it may occur.*

Lösung:

- *Internal Fragmentation (0.5P): Not all of the allocated memory is used, the remaining space is wasted.*
  Allocated memory may be slightly larger than requested memory because of alignment requirements or the use of fixed partition sizes. The unused fragement is inside the allocated partition.
  - *Examples (0.5P):*
    - *Allocation with page granularity in an address space, e.g. Paging (only for the last page)*
- *External Fragmentation (0.5P): Total memory space exists to satisfy a request, but it is not contiguous while the allocation scheme only support contiguous allocation. The unused fragments are between allocated blocks of memory.*
  - *Examples (0.5P):*
    - *Contiguous allocation, e.g. Segmentation (two non-consecutive free blocks of memory that in sum could satisfy a request, but cannot because the memory is not contiguous).*

Erklären Sie knapp, was man unter **Kompaktierung** im Zusammenhang mit **externer Fragmentierung** versteht, und unter welcher **Bedingung** Kompaktierung anwendbar ist. **1 pt**

*Briefly explain the term **compaction** in the context of **external fragmentation**, and under which **condition** compaction is applicable.*

Lösung:

- *Reduce external fragmentation by compaction. Shuffle memory contents to place all free memory together in one large block. (0.5P)*
- *Compaction is possible only if relocation is dynamic (i.e. the memory references must remain valid when moving a program from one region to another), and is done at execution time. (0.5P)*

c) Beschreiben Sie die nötigen Schritte, um einen Seitenfehler im Addressraum einer Anwendung zu behandeln (typischerweise 6, inkl. Reaktivierung). **3 pt**

Describe the steps necessary to handle a page fault in an application's address space (typically 6, incl. reactivation).

Lösung:

- *When a page fault occurs, the page fault handler first checks whether an **access** to the appropriate page is **legal or not**. (0.5P)*
- *If it is not, the application is probably aborted. If it is legal, the handler has to **find and allocate a free frame**. (0.5P)*
- *If there is none, the page replacement algorithm is invoked, which **chooses a frame whose contents are written to disk and which can afterwards be reused** for something else. (0.5P)*
- *The page fault handler then has to **load the correct contents into the frame**: In some cases (e.g., a new frame for a growing stack or heap) it suffices to fill the frame with zeros; in other cases, the correct contents have to be retrieved from disk (e.g., when the page was previously swapped out). (0.5P)*
- *Once the frame is filled with the correct data, the page fault handler needs to **adapt the page tables** by inserting the new mapping and setting the valid bit. (0.5P)*
- *The faulting application can now **resume its execution**. (0.5P)*

d) Wie groß ist der **maximal adressierbare virtuelle Speicher** bei einer dreistufigen Seitentabelle mit 1024 Einträgen pro Stufe? Gehen Sie davon aus, dass jede Seite 8 KiB umfasst und der Speicher byte-weise addressierbar sein soll.

Wieviele Bits müssen virtuelle Adressen in diesem Fall **mindestens lang** sein, damit der Adressraum den **gesamten** Speicher abdeckt? **1 pt**

*How large is the **maximally addressable virtual memory** when using a three-level page table with each level comprising 1024 entries? Assume that a page is 8 KiB in size and that the memory should be byte-addressable.*

*What is the **minimal length** (in bits) for virtual addresses such that the address space covers the **entire** memory?*

Lösung:

$$MaxAddressableVirtualMemory = 8\ KiB \times 1024 \times 1024 \times 1024 = 8\ TiB\ (0.5P)$$

$$MinVirtualAddressLength = 13\ Bits + 10\ Bits + 10\ Bits + 10\ Bits = 43\ Bits\ (0.5P)$$

Skizzieren Sie ein geeignetes **minimales** Format für virtuelle Adressen (für das System mit einer dreistufigen Seitentabelle). **1 pt**

*Sketch a suitable **minimal** format for virtual addresses (for the system with a three-level page table.).*

Lösung:

42 *33*	*32* *23*	*22* *13*	*12* *0*
1st level index	*2nd level index*	*3rd level index*	*Page offset*

Der TLB mit 4096 Einträgen basiert auf einem 4-Wege Satz-assoziativem Cache. Welche Bits dienen als **Set-Index** und welche als **Tag**, wenn eine Teilfolge der Adressbits als Set-Index genutzt wird. **1 pt**

*The TLB contains 4096 entries and is based on a 4-way set associative cache. Which bits serve as the **set index** and which as the **tag**, if a sub-sequence of the address bits is used as set index.*

Lösung:

42 *23*	*22* *13*
Tag	*Index*

- *Index: identifies the set in the TLB*
- *Tag: identifies the entry in the set*

e) Was sind **Aliase** im Zusammenhang mit virtuell-indizierten Daten-Caches? Welche Bedingung muss ein Betriebssystem erfüllen, damit diese im Daten-Cache auftreten können? **1 pt**

*What are **aliases** in the context of virtually-indexed data caches? Which condition has to be satisfied by an operating system such that these can occur in the data cache?*

Lösung:

- *Alias: different virtual addresses point to the same physical memory location. (0.5P)*
- *The operating system must allow to map a physical memory location to more than one virtual memory location: shared memory (0.5P)*

Total: 12.0pt

Aufgabe 5: Hintergrundspeicher und Dateisysteme/*Assignment 5: Secondary Storage and File Systems*

a) In Linux besteht die Oktalnotation der Zugriffsrechte aus 4 Ziffern. Zählen Sie auf, wofür die 4 Ziffern jeweils stehen (nicht welche Werte sie annehmen können). **2 pt**

In Linux, the octal notation for access rights consists of 4 digits. List what these 4 digits stand for (not which values they can hold).

Lösung:

- *Special (0.5P)*
- *User (0.5P)*
- *Group (0.5P)*
- *Other (0.5P)*

Wofür stehen die Werte, die die hinteren drei Ziffern annehmen können, in Bezug auf **reguläre Dateien** und in Bezug auf **Verzeichnisse**? (Konzentrieren Sie sich auf die Zweierpotenzen.) **3 pt**

*What do the possible values of the last three digits represent, for both, **regular files** and for **directories**? (Focus on the powers of two.)*

Lösung:

- *Files:*
  - *execute (0.5P)*
  - *write (0.5P)*
  - *read (0.5P)*
- *Directories:*
  - *enter directory (0.5P)*
  - *create, delete, rename files (0.5P)*
  - *list files (0.5P)*

b) `rsnapshot` sei eine Anwendung, um über die Zeit mehrere vollständige Abbilder von Verzeichnisstrukturen zum instantanen Zugriff zu erstellen (z.B. in Ordern mit "gestern", "letzte_woche"). Beschreiben Sie einen Vorteil für die Nutzung von Hardlinks gegenüber Softlinks, um dies umzusetzen. **1 pt**

Let `rsnapshot` be a filesystem snapshot utility for making it possible to keep multiple backups of directory structures instantly available (e.g., in folders like "yesterday", "last week"). Explain briefly one advantage of using hardlinks instead of softlinks for implementing this.

Lösung:

- *First and consequtive backups can be safely deleted/renamed . . . (1P)*
- *. . .*

Welche Strukturen müssen auf Dateisystemebene zusätzlich angelegt werden, um ein neues Abbild anzulegen? Gehen Sie davon aus, dass es keine Änderungen seit dem letztem inkrementellen Abbild gab. **1 pt**

Which structures have to be created on file system level to take a new snapshot? Assume that there where no changes since the last incremental backup.

Lösung:

- *As hard links are simply directory entries that map a name to an inode, only the directory structure has to be duplicated and file entries are hard links to the files in the previous backup. (1P)*

c) `rsnapshot` basiert auf `rsync`. `rsync` identifiziert identische Teile zwischen Quell- und Zieldatei und versucht nur die modifizierten Teile zu kopieren. Dies basiert auf einer rollenden Prüfsumme. Beschreiben Sie die Eigenschaft einer rollenden Prüfsumme. **1 pt**

`rsnapshot` is based on `rsync` identifies identical parts between source and destination file, and tries to copy only the modified parts. This is based on a rolling checksum. Describe the property of a rolling checksum.

Lösung:

- *The checksum can be incrementally computed. (1P)*

d) Welches sind die zwei Ziele von RAID, und durch welche Konzepte werden diese umgesetzt? **2 pt**

What are the two goals of RAID, and what are the concepts to implement them?

Lösung:

- *Improve performance (0.5P) and reliability (0.5P) of Single Large Expensive Disks (SLED) using Redundant Arrays of Inexpensive/Independent Disks (RAID)*
- *RAID provides I/O parallelism (striping) (0.5P) and redundancy (mirroring, parity) (0.5P)*

e) Welche der folgenden Aussagen sind korrekt, welche sind inkorrekt? **2 pt**
(falsches Kreuz: -0.5P, kein Kreuz: 0P, korrektes Kreuz: 0.5P)

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

korrekt *correct*	inkorrekt *incorrect*	
■	□	Ein RAID 3, mit 4 Datenplatten, 1 Paritätsplatte, und einer logischen Blockgröße von 512 Bytes, bedient E/A-Anfragen optimalerweise mit der vierfachen Transferrate einer einfachen Festplatte mit 512 Byte großen Sektoren. *A RAID 3 featuring 4 data disks, 1 parity disk, and a logical blocksize of 512 bytes, optimally serves I/O requests at four times the transfer rate of a regular individual disk with 512 byte sectors would.*
□	■	Das skizzierte RAID 3 kann mehrere Lesezugriffe echt parallel bedienen. *The sketched RAID can truely serve multiple reads in parallel.*
■	□	Die vorherige Aussage gilt für ein äquivalentes RAID 4 System. Allerdings gilt dies **nicht** für mehrere Schreibzugriffe, da diese alle durch die Paritätsplatte gehen müssen. *The previous statement holds true for an equivalent RAID 4 system. But this does **not** hold for multiple writes, as all writes have to go through the parity disk.*
■	□	Das Schreiben auf ein solches RAID 4 ist allgemein langsamer als auf ein vergleichbares RAID 0 bei gleicher Lesegeschwindigkeit. *Writing on such a RAID 4 is commonly slower than a comparible RAID 0 at the same reading performance.*

Total: 12.0pt