Karlsruher Institut für Technologie

Betriebssysteme
(Operating Systems)

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

Musterlösung

17. 03. 2015

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 12 Blättern: Einem Deckblatt und 11 Aufgabenblättern mit insgesamt 5 Aufgaben.
  The examination consists of 12 pages: One cover sheet and 11 sheets containing 5 assignments.
- Es sind keinerlei Hilfsmittel erlaubt!
  No additional material is allowed.
- Die Prüfung gilt als nicht bestanden, wenn Sie versuchen, aktiv oder passiv zu betrügen.
  You fail the examination if you try to cheat actively or passively.
- Wenn Sie zusätzliches Konzeptpapier benötigen, verständigen Sie bitte die Klausuraufsicht.
  If you need additional draft paper, please notify one of the supervisors.
- Bitte machen Sie eindeutig klar, was Ihre endgültige Lösung zu den jeweiligen Teilaufgaben ist. Teilaufgaben mit widersprüchlichen Lösungen werden mit 0 Punkten bewertet.
  Make sure to clearly mark your final solution to each question. Questions with multiple, contradicting answers are void (0 points).

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

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

Aufgabe 1: Grundlagen

Assignment 1: Basics

a) Nennen Sie zwei Beispiele für Abstraktionen in einem Betriebssystem. **1 pt**

Give two examples for abstractions in an operating system.

Lösung:

- *Files as abstraction for storage devices such as hard disks*
- *Threads as abstraction for the CPU*
- *Virtual address spaces as abstraction of physical memory*

(0.5 P) *each. General "hardware abstraction" was rated too vague. System calls provide an interface, but are not an abstraction.*

b) Durch welche drei Umstände kann es zur Ausführung von Kernel-Mode Code kommen? Geben Sie jeweils eine kurze Erläuterung. **3 pt**

What three circumstances can lead to the execution of kernel-mode code? Give a short explanation for each.

Lösung:

- *System Calls - Voluntary, synchronous invocation by a program to request a privileged OS service*
- *Interrupts - Involuntary, asynchronous signal by a device that requires the OS's attention, for example the completion of an I/O operation*
- *Exceptions - Involuntary, synchronous invocation as reaction to performing an invalid operation in the CPU*

(1 P) *each*

c) Welche Varianten zur Behandlung von Ausnahmen gibt es? Geben Sie für jede Variante ein Beispiel an, in welchem Sie zum Einsatz kommt. **2 pt**

What variants for exception handling exist? For each variant, give an example where it is used.

Lösung:

- *Continue execution by retrying the faulting instruction (e.g., page fault).*
- *Continue execution by skipping the faulting instruction (e.g., virtualization with trap and emulate).*
- *Abort execution (e.g., division by zero, invalid opcode).*
  Also accepted: Hand exception to process to perform custom exception handling

d) Welche der folgenden Segmente einer ELF-Datei **können** schreibgeschützt in den Speicher eingeblendet werden? Welche Segmente **müssen** schreibbar eingeblendet werden?
(falsches Kreuz: -0.5P, kein Kreuz: 0P, korrektes Kreuz: 0.5P) **2 pt**

*Which of the following segments from an ELF-file **can** be mapped read-only? Which segments **must** be mapped writable?*
(incorrectly marked: -0.5P, not marked: 0P, correctly marked: 0.5P)

schreibgeschützt/ *read-only*	schreibbar/ *writable*	
■	□	.text
□	■	.data
■	□	.rodata
□	■	.bss

e) Nennen Sie je zwei Vorteile von statischem und dynamischem Linken. **2 pt**

Give two advantages of both dynamic and static linking.

Lösung:

Dynamic linking:

- *Smaller memory footprint (library code may be shared between processes)*
- *Avoids loading unnecessary code (library code can be loaded when actually used)*
- *Easy global updating of shared libraries*

Static linking:

- *Faster code (no indirection via import table, not position-independent)*
- *Faster program startup (no need to wait for dynamic linker)*

(0.5 P) *each. Plain "faster"-like answers were rated as too vague.*

f) Beim Kopieren einer großen Datei von einer SSD auf eine Festplatte liegt die Kopiergeschwindigkeit für einige Zeit deutlich über der maximalen Schreibgeschwindigkeit der Festplatte und fällt dann abrupt auf das Maximum ab. Wie erklären Sie diese Beaobachtung? **2 pt**

While copying a large file from an SSD to an HDD the copy speed greatly exceeds the HDD's maximum write speed and then suddenly drops to the maximum. How do you explain this?

Lösung:

*The file is copied using the operating system's **file system cache**, caching the destination file. During the first period there is still free RAM to let the cache grow and use the full read speed of the SSD. However, the slow HDD cannot keep up with the high rate of incoming data and eventually the cache grows to its maximum size, capping the read speed to the HDD's write speed (i.e., the speed with which memory in the cache can be replaced).*

Total: 12.0pt

Aufgabe 2: Prozesse und Threads

Assignment 2: Processes and Threads

a) Welche der folgenden Aussagen zum Thema Threads sind richtig? (falsches Kreuz: -0.5P, kein Kreuz: 0P, korrektes Kreuz: 0.5P) **2 pt**

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

korrekt/ *correct*	inkorrekt/ *incorrect*	
☐	■	Hybride Threads blockieren sich bei blockierenden Systemaufrufen gegenseitig. *Hybrid threads block each other during blocking system calls.*
■	☐	`fork()` kopiert nur den ausführenden Thread. *`fork()` copies the calling thread only.*
☐	■	Der Wechsel von Threads erfolgt stets im Systemkern (privilegierter Maschinenbefehl). *Switching threads is always done in the kernel (privileged instruction).*
☐	■	Gewöhnlich verwenden User-Level Threads präemptive Ablaufplanung. *User-level threads commonly use preemptive scheduling.*

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

Consider four processes on a uniprocessor system, with given arrival times (0 = start), burst times and priorities (high values are favored). Complete the scheduling plans given below for the policy preemptive shortest job first (PSJF) *and the policy* non-preemptive static priority (NPSP)*. A box in the scheduling plan represents one unit of time.*

Process	Arrival Time	Burst-Time	Priority
1	2.5	1	3
2	3.5	3	4
3	0	6	2
4	1.5	3	1
5	8	5	5

Lösung:

Preemptive Shortest Job First (PSJF)

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

Non-Preemptive Static Priority (NPSP)

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

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

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

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

Lösung:

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

Process	*Arrival time*	*Start time*	*Response time*
1	*2.5*	*3*	*0.5*
2	*3.5*	*6*	*2.5*
3	*0*	*0*	*0*
4	*1.5*	*2*	*0.5*
5	*8*	*13*	*5*

d) Berechnen Sie für den obigen *NPSP-Ablaufplan* die Tournaround-Zeit aller Prozesse. **2 pt**

For the above NPSP scheduling plan, calculate the tournaround 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*	*2.5*	*12.5*
2	*9*	*3.5*	*5.5*
3	*6*	*0*	*6*
4	*18*	*1.5*	*16.5*
5	*14*	*8*	*6*

e) Welches Problem tritt im Zusammenhang mit Static Priority Scheduling häufig auf? Welcher Mechanismus wird üblicherweise eingesetzt, um dieses Problem zu lösen? Beschreiben Sie das Problem und die Lösung jeweils kurz. **2 pt**

Which problem frequently occurs in conjunction with static priority scheduling? What mechanism can be used to solve that problem? Briefly describe both the problem and the solution.

Lösung:

Starvation* (0.5 P)** *is a frequent problem with static priority scheduling: Low priority processes may never execute if there is a number of high-priority processes in the system.* **(0.5 P)** *A solution to this problem is* ***aging*,* **(0.5 P)** *i.e. increasing the priority of waiting processes as time progresses.* **(0.5 P)**

Alternative solution: ***Priority inversion*** **(0.5 P)** *is when a high-priority process waits for another process with a lower priority. The high-priority process effectively has the lower-priority process' priority during the wait time.* **(0.5 P)** *A solution is* ***Priority donation*** *or* ***Priority inheritance****:* **(0.5 P)** *The process being waited for is given the waiting process' priority during the wait time.* **(0.5 P)**

Total: 12.0pt

Aufgabe 3: Koordination und Kommunikation von Prozessen

Assignment 3: Process Coordination and Communication

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

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

Lösung:

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

(b) Progress:

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

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

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

(d) 0.5pt for each notion, 0.5pt for each correct and matching explanaton.

b) Nennen Sie die zwei grundsätzlichen Kommunikationsmodelle der Interprozesskommunikation. Hinweis: Gefragt sind nicht die verschiedenen Designparameter, sondern die grundlegenden Modelle. **1 pt**

Enumerate the two fundamental models of interprocess communication (IPC). Note: We do not ask for the various design parameters, but for the fundamental models.

Lösung:

- *shared memory* **(0.5 P)**
- *message passing* **(0.5 P)**

c) Welches Kommunikationsmodell können die Threads eines Prozesses immer implizit nutzen? Erläutern Sie warum. **2 pt**

Which is the IPC model that threads of a process can always use implicitly? Explain why.

Lösung:

*Threads can always use **shared memory** (1 P) for communication within a process. Being part of the same process, they are running in the same vrtual address space and thus they implicitly share all virtual memory visible to them.* **(1 P)**

We expect you to use more formal notions and be more precise than just "threads share the memory". Further, you should be able to discern between runtime/library constructs such as the heap and the OS-provided common virtual AS.

d) Was wird benötigt, um in einem System gleichzeitig **nicht-blockierendes Senden und nicht-blockierendes Empfangen** für die Interprozess-Kommunikation zu unterstützen? Erklären Sie, warum.
Nennen Sie einen hierbei zu wählenden Design-Parameter und erläutern Sie, warum dieser kritisch ist. **3 pt**

*What is required to support **both non-blocking send and non-blocking receive** in interprocess communication? Explain why.*
State one of the design parameters that is introduced thereby. Explain why you need to choose that parameter carefully.

Lösung:

*Supporting both non-blocking send and receive requires **buffering (queueing)**.* **(0.5 P)**

Messages have to be stored in a buffer because an arbitrary amount of time could pass between the non-blocking send and the corresponding receive operation. We cannot assume the initial copy of the message (in the sender's AS) to be available at receive time because the sender may destroy it once the non-blocking send operation succeeds. **(1 P)**

An additional helper thread is not a valid solution in this scenario!

*Queueing introduces the parameter of **buffer capacity**, which needs to be chosen carefully.* **(0.5 P)**

*With a chosen **bounded capacity,** a sender has to wait once all buffers are occupied, thereby degrading the non-blocking send to a blocking send operation. With **unbounded capacity** for undelivered messages, the sender will never have to wait. Buffer space may exceed the amount of available memory, though.* **(1 P)** *(You may alleviate this issue by having the sender or the receiver provide buffer space)*

*Another design parameter is **the location of buffers*** **(0.5 P)**

Message buffers can be located in

receiver AS: *If the number of clients rises, the number of messages sent to this receiver might increase, thus requiring an ever larger receive area. If the receive area is not large enough, many IPCs will either fail and be repeated (doubling the IPC overhead) or block the sender, turning the asynchronous send undesirably into a synchronous one.*

kernel: *Malicious or bogus threads might flood the kernel with messages, thus we would need to impose limits on the message buffer size with consequences similar to the receiver-based buffers. Kernel-based buffers can provide opportunities for denial-of-service attacks if not implemented properly.*

sender AS: *The sender has stored the to-be-sent message anyways. If we use the original message memory as message buffer, the application needs to be notified once the message has been consumed by the receiver so that the memory can be released or reused.*

Alternatively, we might copy the message to a separate sent-message buffer iff an asynchronous send cannot deliver the message instantaneously. This approach would allow the application to overwrite the original message memory at the cost of requiring an (additional) copy of the message.

Sender-based buffering scales well, as only the sending processes pay for their communication requirements; applications that seldom communicate with other threads will never block due to insufficient buffer space. **(1 P)**

e) Nennen Sie drei Nachteile von einfachen Spinlocks bzw. Semaphoren und erläutern Sie jeweils, wie der jeweils andere Mechanismus diesen Nachteil behebt. **3 pt**

Give three disadvantages of simple spinlocks and/or semaphores. Explain how the respective other mechanism improves on the problem.

Lösung:

- *Spinlocks do not provide* ***bounded waiting*** *because busy waiting does not guarantee an order in which threads acquire the lock.* **(0.5 P)**
  Strong semaphores wake up threads in the order they called wait, and thereby provide bounded waiting. **(0.5 P)** *(Ticket spinlocks are also ok.)*
- *Semaphores require a system call even when the thread may proceed immediately.* **(0.5 P)**
  Spinlocks on the other hand can be implemented completely in userspace and do not require syscalls. They are very efficient for short wait times. **(0.5 P)**
- *Spinlocks waste resources in busy waiting when the wait time is long. No other thread can do useful work on the CPU where a thread is busy waiting.* **(0.5 P)**
  In contrast, semaphores block waiting threads and thereby make the CPU available for other threads immediately. **(0.5 P)**
- *Non-issues: Semaphores can count and admit several threads. However, that is not viable for critical sections. All practical architectures with multiprocessor implementations offer some form of atomic read-modify-write operation. Thus, spinlocks requiring atomic operations is not a disadvantage (Further, you need them for implementing semaphores on a multiprocessor system as well).*

Total: 12.0pt

Aufgabe 4: Speicher

Assignment 4: Memory

a) Zerlegen Sie die virtuelle Adresse 0x58D4E004 wie es für eine Adressübersetzung mit einer zweistufigen, hierarchischen Seitentabelle notwendig ist. Geben Sie dabei für jeden Teil seinen Wert an, wozu er dient und wie viele Bits er umfasst. Gehen Sie davon aus, dass jede Stufe aus 1024 Einträgen besteht und eine Seite 4096 Bytes umfasst. **3 pt**

Split the virtual address 0x58D4E004 into the parts that are necessary for an address translation with a two-level, hierarchical page table. For every part, give its value, denote its purpose and state its size in bits. Every level comprises 1024 entries. The page size is 4096 bytes.

Lösung:

Page Directory Index (10 bits) *Index into the page directory, to select the page table, which contains the entry for address translation. Value: 0x163*

Page Table Index (10 bits) *Index into the page table, which selects the right page table index to get the page frame number. Value: 0x14E*

Offset (12 bits) *Offset into the page, which is concatenated to the page frame number after translation. Value: 0x004*

(1 P) *each. Half-points were deducted for mistakes.*

b) Nennen Sie je einen Vorteil von hardware- und softwaregesteuerten TLBs. **2 pt**

Give one advantage for both hardware-managed and software-managed TLBs.

Lösung:

Hardware-managed TLB:

- *Faster **TLB-miss** handling, by walking the page tables and inserting the correct entry into the TLB in hardware (this includes a replacement policy in hardware)*

Software-managed TLB:

- *Full flexibility in the page-table format*
- *Full flexibility in the replacement policy*

(1 P) *each. Note that in a software-managed TLB, entries are compared in hardware, too! Plain "fast"- or "more flexible"-answers were generally rated as too vague.*

c) Wann eignen sich einstufige Seitentabellen zur Übersetzung von virtuellen zu physischen Adressen? Erläutern Sie Ihre Antwort. **1 pt**

When are single-level page tables suitable for the translation of virtual to physical addresses? Explain your answer.

Lösung:

When the width of the virtual address space is small (e.g., less than 32 bits), the page size is large or the address space is extensively used (i.e. not sparsely as usually). In these cases, the single-level page table does not take up too much memory or even saves memory compared to hierarchical page tables.

(0.5 P) *for a scenario,* **(0.5 P)** *for the reason*

d) Erläutern Sie das Konzept einer invertierten Seitentabelle. Warum sind sie beim Einsatz von gemeinsam genutztem Speicher (Shared Memory) von Vorteil? **3 pt**

Explain the concept of inverted page tables. Why are they advantageous when using shared memory?

Lösung:

Inverted page tables map physical frames to virtual pages instead of the other way around. The system thus has only a single inverted page table for all processes with one page table entry for each physical page frame. Note that inverted page tables do not limit a frame to be mapped to a only single virtual page. A page table entry in the inverted page table may hold multiple <address space; virtual page number>tuples. **(1.5 P)**

When using shared memory, the OS often (e.g., for page replacement) needs to know for a particular page frame whether it is shared, and if so, which virtual pages map the frame. This information can be easily stored in the inverted page table. **(1.5 P)**

e) Welche Eigenschaft wird bei der FIFO, LRU und optimalen Seitenersetzung zur Auswahl der zu ersetzenden Seite jeweils herangezogen? Erläutern Sie für jede Eigenschaft, ob, und wenn ja, wie sie in einem Betriebssystem auf x86 (näherungsweise) gewonnen werden kann. **3 pt**

What attribute is used for the FIFO, LRU and optimal page replacement to select the victim page? For each attribute, state whether, and if so, how it can be collected or approximated in an operating system running on x86 hardware.

Lösung:

FIFO *The order in which pages have been added to the system (i.e., the time a particular page has been fetched). Can be stored explicitly.*

LRU *The order in which pages have been accessed (i.e., the time a particular page has been accessed the last time). Not directly available, but can be approximated by periodically scanning the reference bit.*

Optimal *The order in which pages will be accessed in the future (i.e., the time a particular page will be accessed the next time). Not available.*

(1 P) *each. Solutions that described the respective policies and indicated how they can be implemented on x86 were also accepted.*

Total: 12.0pt

Aufgabe 5: Hintergrundspeicher und Dateisysteme

Assignment 5: Secondary Storage and File Systems

a) Welche UNIX Dateizugriffsrechte wurden in der Vorlesung besprochen und wofür stehen diese bei Dateien und Verzeichnissen? **3 pt**

Which UNIX file permissions where discussed in the lecture and what do these permissions stand for when set for files and directories?

Lösung:

Files: 1 = execute **(0.5 P)** *2 = write* **(0.5 P)** *4 = read* **(0.5 P)** *Directories: 1 = directory open* **(0.5 P)** *2 = create, delete, rename files* **(0.5 P)** *4 = list files* **(0.5 P)**

b) Die meisten Betriebssysteme stellen einen Buffer Cache für Dateisysteme bereit. Beschreiben Sie einen Nachteil dieses Caches und schlagen Sie eine mögliche Lösung für dieses Problem vor! **2 pt**

Most operating systems provide a buffer cache for file systems. Describe one disadvantage of this cache and propose a possible solution to this problem!

Lösung:

Write-behind policy might lead to data losses in case of system crash and/or inconsistent state of the FS **(1 P)** *Solution: no write-behind policy or a journaling file system* **(1 P)**

or

FS Cache wiping if sequentially reading a very large file from end to end and not accessing it again **(1 P)** *Solution: by-passing the buffer cache, either by direct I/O or hints like sequential access* **(1 P)**

c) Nennen und erklären Sie die Unterschiede zwischen, den in der Vorlesung behandelten File Locking Ansätzen! **2 pt**

Name and explain the difference between the two file locking approaches presented in the lecture!

Lösung:

Mandatory: The operating system ensures that no process can open a file that another process holds a lock on. **(1 P)**

Advisory: The operating system permits the simultaneous access of multiple processes to the same file. If required, the processes have to perform correct synchronization on their own. **(1 P)**

d) Welchen RAID Level würden Sie für ein Datacenter vorschlagen? Es soll sowohl eine bessere Performance als auch eine höhere Zuverlässigkeit als eine einzelne Festplatte bieten. Beachten Sie auch die Kosten. Warum haben Sie den Level gewählt und wie viele Festplatten brauchen Sie mindestens für die Umsetzung? **2 pt**

Which RAID level would you suggest for a datacenter? It should provide both higher performance and better reliability than a single hard disk. Keep the costs in mind. Why have you chosen the level and how many disks are at least required?

Lösung:

RAID5, requires at least 3 disks. **(1 P)** *RAID5 provides higher performance and better reliability without the need of a single bottleneck disk (RAID4) and lower costs compared to RAID01 (4 disks).* **(1 P)**

e) Wir haben in der Vorlesung vier Dateiallokationsstrategien kennengelernt. Nennen Sie diese Strategien.

Die Metadaten des Dateisystems wurden beschädigt. Die Dateidaten sollen wiederhergestellt werden. Nehmen Sie an, dass wenigstens der Beginn einer Datei durch eine Dateitypanalyse identifiziert werden kann. Welche Allokationsstrategien sind für dieses Szenario am wenigsten geeignet und warum? **3 pt**

We have described four file allocation strategies in the lecture. Name these strategies.

The metadata of the file system has been corrupted. The file data should be recovered. Assume that at least the beginning of a file can be determined by a file type analysis. Which of the allocation strategies are least suitable for this scenario? Explain why.

Lösung:

(a) Indexed Allocation **(0.5 P)** *, (b) File Allocation Table* **(0.5 P)** *, (c) Chained Allocation* **(0.5 P)** *, and (d) Contiguous Allocation.* **(0.5 P)**

Indexed allocation (with high fragmentation) and FAT, because to reconstruct the file data the beginning of a file is not sufficient (in the worst case, you only get the first block of the file). The index tables are also needed. This is not the case with chained or contiguous allocation. **(1.0 P)** *However, it is not always possible to get the exact file size, due to internal fragmentation.*

Total: 12.0pt