

University of Karlsruhe
System Architecture Group
Prof. Dr. Jochen Liedtke

Tutors:
Christian Ceelen
Uwe Dannowski
Kevin Elphinstone
Felix Hupfeld
Gerd Liefländer
Espen Skoglund

System Architektur (*Architecture*)

Klausurlösung (*Examination*)

WS 2000/2001, 20. April 2000

- Bitte tragen Sie zuerst auf allen Klausurblättern Ihren Namen, Vornamen und Ihre Matrikelnummer ein, auch auf den Konzeptblättern. *Please enter your last name, first name and matriculation number on each page (including used and unused draft pages).*
- Die Prüfung dauert 60 Minuten und besteht aus 5 Aufgaben auf 11 Seiten und zwei Konzeptblättern. *You have 60 minutes to complete your answers. The examination consists of 5 questions on 11 pages. You have received two additional blank pages for drafts, etc.*
- Die Prüfung ist mit mindestens 20 Punkten von 60 erreichbaren Punkten bestanden. *You pass the examination by obtaining at least 20 marks out of the possible 60 marks.*
- Es sind keinerlei Hilfen erlaubt! *No additional means are 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 more draft pages please notify one of the supervisors.*
- Bitte machen Sie eindeutig klar, was Ihre endgültige Lösung zu den jeweiligen Teilaufgaben ist. Teilaufgaben mit mehreren Lösungen oder mit widersprüchlichen Teilen werden mit 0 Punkten bewertet. *Make sure that it is absolutely clear what your final solution is for each subquestion. Subquestions with multiple solutions or with contradicting parts are void: 0 marks.*

Die folgende Tabelle wird von uns ergänzt! *The below table is completed by us!*

Aufgabe/ <i>Question</i>	1	2	3	4	5	Total
Erreichbare Punkte <i>Possible marks</i>	12	12	12	12	12	60
Erreichte Punkte/ <i>Obtained marks</i>						
Note/ <i>Grade:</i>						

Nachname/ <i>Last name</i>	Vorname/ <i>First name</i>	Matrikelnummer/ <i>Matriculation number</i>
Nobody	Anybody	4711

Aufgabe/Question 1 (Zum Aufwärmen/Warm up, 3 + 3 + 1 + ... + 1 Punkte/marks)

1. „Zählen Sie **drei** verschiedene **Typen** von **Sicherheitsbedrohungen** für Systeme auf und geben **jeweils** Sie **ein konkretes Beispiel** an!“

*“Enumerate **three** different **types of security threats** for systems and give **a concrete example for each** of them.”*

- a) **Interruption, an attack on availability**
e.g. cutting a communication line
- b) **Interception, an attack on confidentiality**
e.g. illicit copying of files
- c) **Modification, an attack on integrity**
e.g. changing records in a data base

2. „Zählen Sie **drei** wesentliche **Abstraktionen** der Systemarchitektur auf!“

*“Enumerate **three** basic **abstractions** of system-architecture*

- a) **Thread**
- b) **Address Space**
- c) **File**

Einige der folgenden Aussagen sind korrekt, einige inkorrekt. Unterstreichen Sie „korrekt“, wenn die Aussage korrekt ist, unterstreichen Sie „inkorrekt“, wenn die Aussage inkorrekt ist.

Some of the following statements are correct, some are incorrect. Underline “korrekt” if the statement is correct; underline “inkorrekt” if the statement is incorrect!

3. „**Java** bietet zur **Synchronisation** von Threads das **Konzept Semaphore** an.“

*“To **synchronize** threads, **Java** offers the **concept of semaphores**”.*

korrekt

inkorrekt

Fortsetzung von Aufgabe 1 / Question 1 continued: (1+1+1+1+1 Punkte/marks)

4. „Ein NFS-Dateidienstgeber (*server*) ist zustandslos (*stateless*).“
“An NFS-file-system server is a stateless server.”
[korrekt](#) inkorrekt
5. „Skalierbarkeit ist ein wesentliches Merkmal von Clusterarchitekturen.“
“Scalability is a essential characteristic of cluster-architectures.”
[korrekt](#) inkorrekt
6. „Auf einem System mit einem **Mikrokern der zweiten Generation** sind abgesehen vom Mikrokern **alle sonstigen Systeminstanzen auslagerbar**.“
“On a system based upon a second generation μ -kernel, apart from the μ -kernel all other system tasks can be swapped out
korrekt [inkorrekt](#)
7. „Ein **wesentlicher Nachteil jeder zentralisierten Lösung** in einem verteilten System ist die **eingeschränkte Leistung**.“
“One important drawback of any centralized solution within an distributed system is its limited performance.”
korrekt [inkorrekt](#)
8. „Ein Thread, der **unendlich lange** auf das Eintreffen einer Nachricht wartet, wobei er eine **synchrone IPC-Operation *receive_message()*** benutzt hat, **ist verklemmt (*deadlocked*)**.“
*“A thread waiting forever for an incoming message having used a synchronous IPC-operation *receive_message()* is deadlocked*
korrekt [inkorrekt](#)

Nachname/ <i>Last name</i>	Vorname/ <i>First name</i>	Matrikelnummer/ <i>Matriculation number</i>

Aufgabe/Question 2**(5 + 2 + 5 Punkte/marks)**

1. „Gegeben sei ein symmetrisches Mehrprozessorsystem (SMP) aus 4 Prozessoren, die alle über den Systembus an den gemeinsamen Hauptspeicher (*main memory*) angeschlossen sind. Pro Prozessor existiere ein L1- und ein L2-Pufferspeicher (*cache*). Auf Prozessor CPU₄ läuft ein Schachprogramm mit einer großen Arbeitsmenge (*working set*), die nicht vollständig in den L2-Cache paßt. Dieses Schachprogramm läuft unabhängig von allen sonstigen Anwendungen. Drei kooperierende Threads mit jeweils mittelgroßer Arbeitsmenge (*working set*), d.h. jede von ihnen paßt vollständig in einen L2-Cache, laufen echt parallel auf den Prozessoren CPU₁ bis CPU₃. Diese drei kooperierenden Threads synchronisieren sich mittels der unten skizzierten „Spinlockimplementierung“:

“Given a symmetric multi-processor system (SMP) consisting of 4 processors operating on a shared main memory via the system-bus. L1 and L2 caches exist per processor. Processor CPU₄ executes a chess program with a large working set that does not fit into its L2 cache. This chess program runs independently of all other applications. Three cooperating threads with medium sized working sets, i.e. each working set fits into an L2-cache, run in parallel on processors CPU₁ through CPU₃. The three cooperating threads synchronize by means of the **spinlock implementation** described below:

DO

```
while SpinLock != 0 do {};
{Reading from main memory only once}
```

```
reg := MyThreadId ;
xchg (SpinLock,reg) {special atomic exchange operation}
                {exchanges mem/cache variable SpinLock and reg}
```

UNTIL reg = 0 OD;

```
...
... critical section ...
...
```

SpinLock := 0

{SpinLock ≠ 0 ⇔ another Thread is in its critical section}

{SpinLock = 0 ⇔ critical section is free }

Das Schachprogramm läuft auf diesem SMP-System **deutlich langsamer** als auf einem Einprozessorsystem. **Modifizieren** Sie die **Implementierung** des Spinlocks, so daß das **Schachprogramm nicht mehr so stark behindert** wird!“

The chess program runs substantially slower in this SMP-system than on a single-processor system. Modify the spinlock implementation so that the chess program is not as hampered as before.”

Fortsetzung von Aufgabe 2 / Question2 continued:**(2 + 5 Punkte/marks)**

2. „Analysieren Sie die **maximale Wartezeit** für einen der drei kooperierenden Threads vor seinem kritischen Abschnitt, wenn die verbesserten Spinlocks aus Teilaufgabe 2.1 verwendet werden.“

*“Analyze the **maximal waiting-time** for one of the three cooperating threads in front of its critical region using the improved spin-locks of question 2.1.”*

Maximale Wartezeit (*maximal waiting time*): ∞ , i.e. unbounded waiting time

Begründung (reasoning): **One of the three threads on its CPU might always be the loser competing for the spinlock. Thus its CPU is completely wasted.**

3. „Diese Teilaufgabe ist unabhängig vom konkreten Beispiel aus Teilaufgabe 2.1. und 2.2. Das Betriebssystem bietet an seiner Anwendungsprogrammierschnittstelle (API) **Spinlocks** und **binäre Semaphore** an. Welche **Probleme** können auftreten, wenn ein Programmierer einer „vielfädigen Applikation“ (*multi-threaded application*) einen Spinlock statt einer binären Semaphore zur Lösung des **wechselseitigen Ausschluß** verwendet.

Hinweis: Diese Applikation soll auf Ein- und Mehrprozessorsystemen lauffähig sein; ferner soll sie unabhängig von einer spezifischen Schedulingstrategie sein. “

*“This subquestion is independent of the specific example in subquestions 2.1. and 2.2. The operating system offers **spinlocks** as well as **binary semaphores** at its **API**. What **problems** might arise, when a programmer of a multi-threaded application uses a spinlock instead of a binary semaphore in order to solve **mutual exclusion**?”*

Hint: *This application has to run on single- and multi-processor systems; furthermore it should be independent of a specific scheduling policy.”*

Single-Processorsystem:

With pure priority scheduling (without time slices) a thread with a high priority may wait forever in front of the closed spinlock, thus the system is lifelocked forever.

Multi-Processorsystem:

1. Starvation situation due to unfair arbitration of processors of 2.2 (may be induced by bus arbitration)
2. You cannot prevent application programmers to use spinlocks for long critical sections and/or with many concurrent threads. Thus the overall system performance may decrease significantly, CPUs are wasted and bus traffic is heavy with solution 2.1.

Nachname/ <i>Last name</i>	Vorname/ <i>First name</i>	Matrikelnummer/ <i>Matriculation number</i>

Aufgabe 3 / Question 3**(3 + 3 + 6 Punkte/marks)**

1. „Analysieren Sie das folgende Programm! Stellt es eine **korrekte Lösung** für das **Produzent-/Konsumentproblem** mit einem **zyklischen Puffer** der **Kapazität b** dar? Begründen Sie Ihre Meinung!

Hinweis: Der Puffer sei als **zusammenhängendes Feld (array)** realisiert, zwei Zeiger unterstützen das Einfügen und Entfernen von Datenelementen (*items*) in den und aus dem Puffer. “

“*Analyze the following program. Is it a correct solution for the bounded producer-consumer problem using a b -slot cyclic buffer? Clarify your conclusion.*”

```
lock: semaphore := 0;      {mutual exclusion on the buffer}
empty_slots: semaphore := P;
full_slots: semaphore := 0;
```

```
{producer code}
```

```
{consumer code}
```

```
repeat
```

```
  produce item;
  P(lock);
  P(empty_slots);
  insert item;
  V(full_slots);
  V(lock);
```

```
forever
```

```
repeat
```

```
  P(lock);
  P(full_slots);
  remove item;
  V(empty_slots);
  V(lock);
  consume item;
```

```
forever
```

This is not a correct solution, the system may result in a deadlock. Suppose the producer has already produced P items and tries to produce another one. It passes $P(\text{lock})$ but will wait within $P(\text{empty slots})$ forever, because the consumer can never pass its $P(\text{lock})$.

2. „Wenn man in obiger Programmskizze die Semaphore **lock** nebst ihren Schnittstellenoperationen **P(lock)** und **V(lock)** ganz wegläßt, wie sieht dann das Analyseergebnis aus? (Gleiche Pufferimplementierung wie in 3.1.)“

“*What’s the result of the analysis, if you leave out the semaphore **lock** as well as its interface-operations **P(lock)** and **V(lock)** in the above program? (Same buffer implementation as in 3.1.)*”

Analyseergebnis (*result of the analysis*):

It’s correct because the producer is always one step in front of a consumer, so there is no conflict on the buffer.

Fortsetzung von Aufgabe 3 / Question 3 continued:

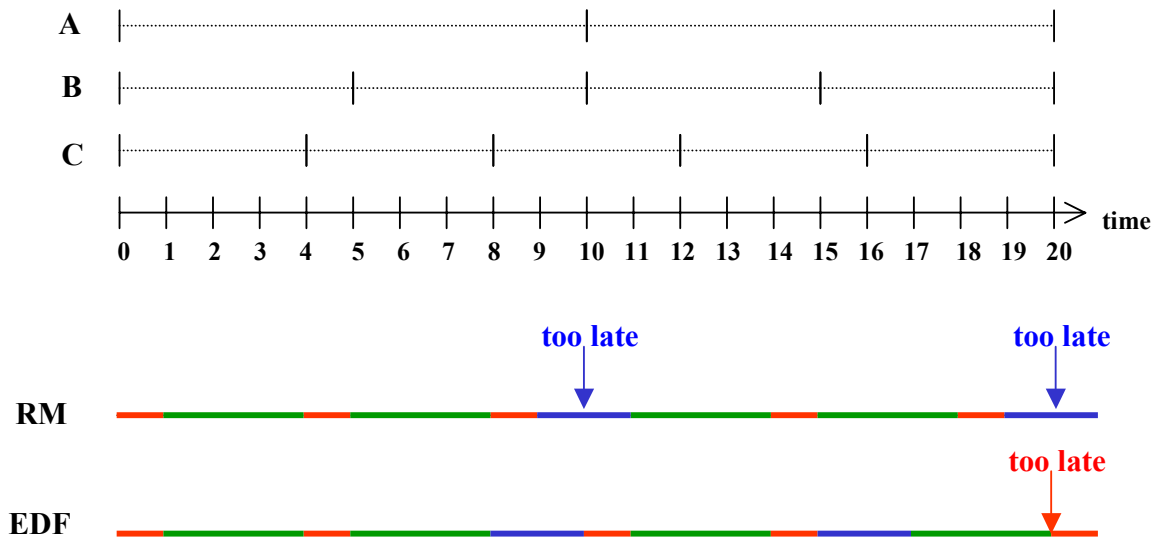
(6 Punkte/marks)

3. „Die Algorithmen „**Rate-Monotonic (RM)**“ und „**Earliest Deadline First**“ (EDF) sind zwei im Echtzeitbereich populäre nicht verdrängende (*nonpreemptive*) Schedulingstrategien. Beschreiben Sie beide Verfahren! Illustrieren Sie deren Wirkungsweise am folgenden Schedulingproblem aus drei periodischen Echtzeithreads! Sie können annehmen, daß Threadumschaltungen (*thread switches*) unendlich schnell sind.“
- “Rate monotonic (RM) and earliest deadline first (EDF) are two popular nonpreemptive scheduling policies for real-time systems. Describe both algorithms. Illustrate your answer by showing how each of them would schedule the following set of threads. You may assume that thread switches are instantaneous.”*

Thread	Exakte Bearbeitungszeit (Requires exactly)	Periodenlänge (every)
A	2 ms	10 ms
B	3 ms	5 ms
C	1 ms	4 ms

Rate monotonic scheduling favors threads with shorter periods. Deadline violations tend to appear to threads with longer periods.

EDF-scheduling favors always those threads with the closest deadline thus minimizing maximum lateness.



In the above example RM produces 2 deadline violations, but it's always the low priority "blue" thread A.

On the other hand EDF produces only one deadline violation, however, this happens to the "red" thread C with the highest priority.

Nachname/ <i>Last name</i>	Vorname/ <i>First name</i>	Matrikelnummer/ <i>Matriculation number</i>

Aufgabe/Question 4**(3 + 3 + 3 + 3 Punkte/marks)**

1. „Zählen Sie die wesentlichen Speichermanagementfunktionen eines Betriebssystems auf!“
“Outline the basic memory management functions of an operating system.”

Create_AS, Delete_AS, Extend_AS

Allocate_Mem, Free_Mem, (Un)Pin_Mem, (Un)Lock_Mem

Swap_IN, Swap_OUT, Adjust_WorkingSet

Set_UP_MMU, Handle_PageFaults, change_PT_Entries

2. „Geben Sie jeweils mindestens eine konkrete Anwendung an, für welche Sie das jeweilige System **mit virtuellem Speicher** und **ohne virtuellem Speicher** ausstatten würden!“
“Give at least one concrete application in each case where you would install a system **with virtual memory** and **without virtual memory**.”

A static embedded system where all the programs fit into main memory does not need virtual memory.

However, all systems with dynamic load characteristic and/or where the applications can not trust each other should operate with virtual memory.

Fortsetzung von Aufgabe 4 / Question 4 continued:

(3 + 3 Punkte/marks)

3. „In einem System mit virtuellem, seitenorientierten Speicher ist ein **Seitenfehler (page-fault)** aufgetreten, obwohl die **Seite im Hauptspeicher** vorhanden ist. Nehmen Sie an, daß **kein Programmierfehler** vorliegt. Beschreiben Sie Szenarien, in denen obige Situation auftreten kann.“

“A page-fault has occurred in a system with paged virtual memory, even though the page is in main memory. Assume there is no programming error. Describe scenarios where the above situation can happen.”

Shared page having been swapped in by another thread

Shared page with copy_on_write

Due to a page-demon the page may be part of the free-list. If no other thread has needed its underlying page-frame and if this page is referenced again, the pager does not have to page-in, but it can reuse the page adjusting its corresponding page-table entry and removing from free-list.

4. „Diskutieren Sie **Vor- und Nachteile** einer **einstufigen** versus einer **mehrstufigen Seitentabelle (multi-level page table)**!“

“Discuss the pros and cons of a one-level versus a multi-level page table.”

A one-level page-table (e.g. of a 64 byte address system) is too huge for most main memory of a realistic size.

Even worse, small address-spaces don't need such a huge page-table.

However, parsing a linear page-table is less time consuming than parsing several hierarchical page-tables.

A multi-level page-table for small address-spaces needs far less main memory (even though an address space of maximal order would require more memory). In general an address-space has some temporal locality, i.e. we do not have to keep all page-tables in memory at the same time.

However, parsing takes more time (hopefully a TLB will help).

Nachname/ <i>Last name</i>	Vorname/ <i>First name</i>	Matrikelnummer/
----------------------------	----------------------------	-----------------

		Matriculation number

Aufgabe 5/ Question 5:**(3 + 3 + 6 Punkte/marks)**

1. „Welche **Metadaten** einer Unix-Datei kennen Sie? In welchen **Datenstrukturen** sind diese Metadaten implementiert?“
 “What **meta-data** are associated with a Unix-file? In what **data-structures** are these meta-data implemented?”

File name, length of file name, inode (in the directory)

File type, owner, access rights, dates, link counter, pointers to data blocks (in the inode)

2. „Beschreiben Sie zumindest zwei verschiedene Formen der Semantik für gemeinsam benutzte Dateien.“
 “Describe at least two different semantics for file sharing.”

One-copy semantics (à la Unix):

Updates are written to the single copy and are available immediately

Session semantics:

Copy the file on open, work on your local copy, and copy back on close

No updates on files:

Any update of a file causes the creation of a new file

Fortsetzung von Aufgabe 5/ Question 5 continued:

(6 Punkte/marks)

3. „Beschreiben Sie so ausführlich wie nötig, aber dennoch so knapp wie möglich, was eine B*-strukturierte index-sequentielle Datei ist! Geben Sie ein konkretes Beispiel an, wofür Sie eine B*-strukturierte indexsequentielle Datei verwenden würden (ein anderes als die chinesische Sozialversicherung)“

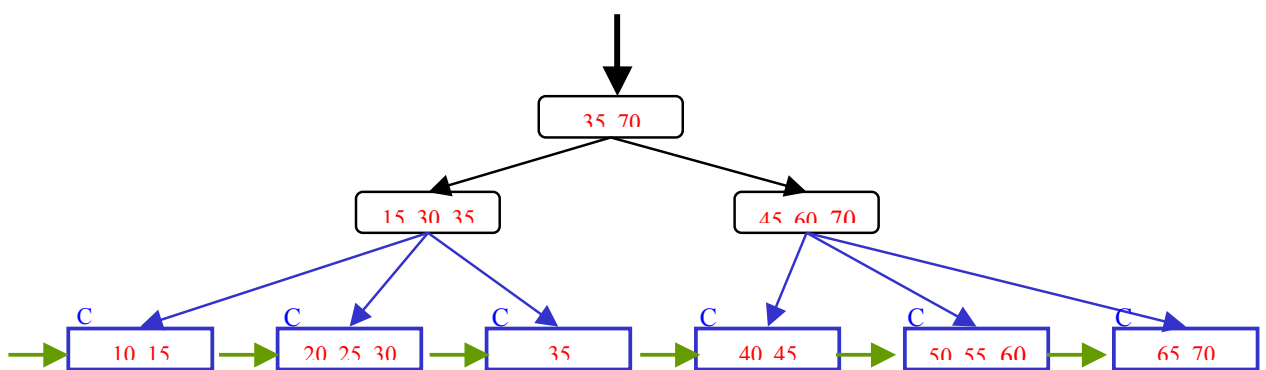
“Describe as detailed as necessary, but nevertheless as short as possible, what a “B-structured index-sequential file is. Give a concrete example when you would use a B*-structured index-sequential file (another one than the Chines social insurance).”*

Meta-data for random-accesses for a B*-indexed-file form a B*-tree, each inner-node may have up to t -successors ($t = (\text{size of node}/\text{size of an entry})$). Each entry in an inner-node contains a pointer to the next lower-level inner-node (or to the container for the records) and the maximal key in the corresponding sub-tree.

For any random access to a specific record the search always begins at the root of the B*-tree. Depending on the value of the key of this record the search-process continues in one of the next lower level sub-trees. The search within the container is a linear search.

Insertion of new items may affect only the involved containers (i.e. one of the leaves of the B*-tree, but it could also affect upper inner nodes. In the worst case you even have to insert another level in the B*-tree, i.e. the height of the tree increases by one. Updates of items don't affect the overall structure, as long as the record size is not changed.

The other entry for this file-type is used for sequential accesses, i.e. all containers are linked together to build up a linked list.



Example: Data base for the staff members or for the students of a university