

Nachname/Last name	Vorname/First name	Matrikelnummer/ Matriculation number
Musterfrau	Marlene	999999

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

1. „Zählen Sie drei gebräuchlichen Kriterien auf, auf welche sich praktikable Seitenersetzungsverfahren (*page replacement*) abstützen, und geben Sie jeweils ein konkretes Verfahren an!“
“Enumerate the three commonly used criteria being used by practical page replacement algorithms, and give a concrete example for each of these algorithms.”
 - a) **Einlagerungszeitpunkt der Seite für FIFO**
Timestamp of swap-in for FIFO
 - b) **Zeitpunkt des letzten Seitenzugriffs für LRU**
Time of last reference on that page for LRU
 - c) **Anzahl der Referenzen pro Seite für LFU**
Number of references on that page for LFU

2. „Zählen Sie drei typische Kernschnittstellenoperationen (*kernel interface function*) auf!“
“Enumerate at least three typical kernel interface functions.”
 - a) **Sende_Nachricht()**
Send_Message()
 - b) **Erzeuge_Thread()**
Create_Thead()
 - c) **Yield()**
Yield()

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. „In einem Einprozessorsystem, in dem bis zu 1000 Threads gleichzeitig aktiviert sind, ist es effizienter, den Threadzustand „bereit“ (*ready*) durch eine explizite Datenstruktur zu implementieren als nur durch ein entsprechendes TCB-Attribut.“
“In a single-processor system with up to 1000 currently activated threads, it is more efficient to implement the thread-state “ready” by a separate data structure than by a corresponding TCB attribute.”

korrekt

inkorrekt

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

4. „Beim **Speicherbereinigungsverfahren (garbage collection)** werden belegte Stücke so lange verschoben, bis ein genügend großes Speicherstück frei geworden ist, mit dem die aktuelle Speicheranfrage erfüllt werden kann.

*“The **garbage collection** algorithm shifts allocated memory regions until there is a free memory region large enough to satisfy the current storage request.”*

korrekt

inkorrekt

5. „Das strikte Zwei-Phasen Protokoll kann zu Verklemmungen (*deadlocks*) führen.“

“The strict two-phase protocol may lead to deadlocks.”

korrekt

inkorrekt

6. „Unter allen nicht verdrängenden Schedulingverfahren ist SJF (*shortest job first*) **optimal** bezüglich der **mittleren Bearbeitungszeit (turnaround-time)**.“

*“When only considering non-preemptive scheduling algorithms, SJF (shortest job first) is **optimal** with respect to the **average turnaound-time**.”*

korrekt

inkorrekt

7. „Semaphore sind ein komfortables und robustes Mittel, um auf Applikationsebene wechselseitigen Ausschluß (*mutual exclusion*) zu etablieren.“

“Semaphores are a comfortable and robust tool to esrablish mutual exclusion at application level.”

korrekt

inkorrekt

8. „Nebenläufige (*concurrent*) Threads treten nur in Einprozessorsystemen auf.“

“Concurrent threads only occur within single-processor systems.”

korrekt

inkorrekt

Nachname/ <i>Last name</i>	Vorname/ <i>First name</i>	Matrikelnummer/ <i>Matriculation number</i>
Musterfrau	Marlene	999999

Aufgabe/Question 2**(6 + 4 + 2 Punkte/marks)**

1. „Geben Sie in folgender Programmskizze eine Implementierung für eine generalisierte Synchronisation für n Threads an! Gehen Sie davon aus, daß die Schnittstellenoperation `n_synchronize` atomar ist.“

*“Implement in the following program template a generalized synchronization module to synchronize n threads. Assume that the interface operation `n_synchronize` is **atomic**.”*

```

module generalized synchronization
  export n_synchronize
  import UnblockThread, BlockThread
  type n_sync = record

    NUMBER: integer = n; {# allowed threads}

    COUNT: integer := 0; {# current waiting threads}

    SWT: waiting thread := empty {waiting queue}
  end

  procedure n_synchronize(SY:n_sync)
    begin

    SY.COUNT := SY.Count + 1;

    if SY.COUNT < SY.NUMBER
    then begin          {I am not the last thread}
      BlockThread(SY.SWT)  {wait for my partner}
    end
    else begin        {I am the last thread and must}
      while SY.SWT != empty do
        UnblockThread(SY.WT); {release my partners}
        SY.COUNT := 0;        {resetting for reuse}
      end
    end
  end
end module

{queuing and dequeing will be done within BlockThread and UnblockThread}

```

Fortsetzung von Aufgabe 2 / *Question 2 continued:*

(4 + 2 Punkte/marks)

2. „Geben Sie in Form einer knappen Programmskizze ein Anwendungsbeispiel für obiges generalisierte Synchronisationsmodul an!“

“Try to illustrate in a brief program template an example for an application of the above generalized synchronization module.”

```
{some numerical problem solved via difference
equations}
```

```
.
```

```
.
```

```
while true do
```

```
  begin
```

```
    for all i,j
```

```
      begin
```

```
        temp[i,j] := old[i-1,j] + old[i+1,j]
```

```
      end
```

```
  n_synchronize(S)
```

```
  for all i,j
```

```
    begin
```

```
      old[i,j] := temp[i,j]
```

```
    end
```

```
  n_synchronize(S)
```

```
end
```

```
.
```

```
.
```

3. „Welche Hardwareunterstützung würden Sie verwenden, um ein atomares `n_synchronize` in einem Ein- bzw. Mehrprozessorsystem zu implementieren?“

“Which hardware support would you use to implement an atomic `n_synchronize` in a single- and in a multi-processor system?”

Single-processor system:

`Disable_Interrupt` and `Enable_Interrupt`

Multi-processor system:

`Spin_Lock` with a `TEST_AND_SET` instruction

Nachname/ <i>Last name</i>	Vorname/ <i>First name</i>	Matrikelnummer/ <i>Matriculation number</i>
Musterfrau	Marlene	999999

Aufgabe 3 / Question 3 :**(5 + 3 + 2 + 2 Punkte/marks)**

1. „Analysieren Sie die folgende Pogrammskizze, ob dadurch alle Anforderungsbedingungen für einen korrekten wechselseitigen Ausschluß auf Applikationsebene erfüllt werden!“
“*Analyze whether the following program draft fulfills all requirements for correct mutual exclusion at application level.*”

2.

Process 0

```

.
.
.
flag[0] := true;
while flag[1] do
begin
flag[0] := false;
{ delay for a while }
flag[0] := true
end;

{ critical section }

flag[0] := false;
.
.
.

```

Process 1

```

.
.
.
flag[1] := true;
while flag[0] do
begin
flag[0] := false;
{ delay for a while }
flag[1] := true
end;

{ critical section }

flag[1] := false;
.
.
.

```

Obiger Vorschlag kommt einer korrekten Lösung recht nahe, aber bei ungünstigen Zeitverhältnissen kann es zu fortwährendem Warten beider Threads vor dem kritischen Abschnitt kommen. Somit verletzt diese „Scheinlösung“ eine wesentliche Anforderung nach einer begrenzten Wartezeit (*bounded waiting*) vor dem kritischen Abschnitt und ist somit unbrauchbar. Die anderen Anforderungen wären dagegen erfüllt.

This proposal is close to a solution, but it is still flawed. With certain timing conditions both threads may wait indefinitely in front of their critical sections. Thus in the above approach there is no bounded waiting in front of a critical section, so we can't use it. However, the other requirements are fulfilled.

Fortsetzung von Aufgabe 3 / Question 3 continued:**(3 + 2+ 2 Punkte/marks)**

3. „Gegeben seien zwei nebenläufige Transaktionen T_1 und T_2 , von denen jede aus eine Reihe von Lese- bzw. Schreiboperationen auf eine gemeinsame Datenbank aufgebaut sei. Die Datenbank bestehe aus $d \gg 1$ Datenobjekten do_i , von denen jedes separat durch einen Read_Lock bzw. Write_Lock vor gleichzeitigem Zugriff geschützt werden kann. Welche Bedingungen müssen gegeben sein, damit obige Transaktionen im Konflikt stehen?“

“Assume there are two concurrent transactions T_1 and T_2 each of which consists of a sequence of read- and write-operations to a shared database. This database consists of $d \gg 1$ data objects each of which can be protected against concurrent accesses by a separate Read_Lock and/or by a separate Write_Lock. Which conditions may lead to conflicting transactions?”

**There is a data object do_i being accessed by both Transactions T_1 und T_2
At least one of them, e.g. T_1 does a write operation on this data object do_i .
If the other transaction T_2 does an additional operation on this data object
-either another write or a read- then both transaction are conflicting.**

4. „Welche der vier Eigenschaften des ACID-Prinzips wird durch einen serialisierbaren (*serializable*) Ablaufplan herbeigeführt?“

“Which of the four requirements of the ACID-principle is achieved by a serializable schedule?”

Isolation

5. „Was sind die Vor- und Nachteile, wenn in einer gemeinsamen Datenbank jedes Datenobjekt separate Sperren (*locks*) besitzt?“

“What are the pros and cons of having separate locks per data object in a shared data base?”

Pros: Additional concurrency may lead to a better throughput, you are only locking regions of the data base you really need

**Cons: Increased overhead for locking and relasing (+ additional space for all these locks)
Increased danger for deadlocks**

Nachname/Last name	Vorname/First name	Matrikelnummer/ Matriculation number
Musterfrau	Marlene	999999

Aufgabe/Question 4**(4 + 2 + 2 + 4 Punkte/marks)**

1. „In einem System mit virtuellem, seitenorientierten Speicher ist ein Seitenfehler (*page-fault*) aufgetreten. Welche Systemaktivitäten, die zur Beseitigung des Seitenfehlers notwendig sind, werden dabei innerhalb und welche werden außerhalb der Unterbrechungsbehandlungs-routine ausgeführt? Geben Sie eine kurze Begründung für Ihre Wahl!“

“A page-fault has occurred in a system with paged virtual memory. Which of the system activities necessary to handle this page-fault will be executed within the page-fault interrupt handler, and which will be executed outside of this interrupt handler. Give a short explanation for your choice.”

Seitenfehlerbehandler:**Festhalten, welcher Thread den Seitenfehler produziert hat****Benachrichtigen des Seitenersetzer (*pager*)****Seitenersetzer:****Kachel zum Beheben des Seitenfehlers gemäß der Ersetzungsstrategie besorgen (u.U. 2 mal den Plattentreiber aufrufen für's Ein- bzw. Auslagern der alten bzw. neuen Seite)*****Page fault handler:******Save the thread-id of the thread issuing the page-fault******Send an appropriate page-fault message to the pager******Pager:******Get a page-frame to handle the page-fault according to a paging policy******(i.e. this pager might invoke two times the disk driver to swap out the old page respectively to swap in the new one)***

2. „Wieviel Bytes benötigt eine flache Seitentabelle in einem System mit 32-bit logischen Adressen, einem Seitentableneintrag von 4 Bytes und einer Seitengröße von 1 KB?“

“How many bytes would a flat page table need in a system with 32-bit logical addresses, a page entry size of 4 bytes, and a 1 KB page size?”

Seitentabellengröße in bytes/Page table size in bytes: 16 MB

Fortsetzung von Aufgabe 4 / Question 4 continued: (2 + 4 Punkte/marks)

3. „ Gegeben sei ein Seitenreferenzstring für einen Applikationsthread mit einer Hauptspeichermenge (*resident set*) von m Kacheln, die initial alle leer seien. Der Seitenreferenzstring habe die Länge p , wobei n verschiedene Seitennummern enthalten sind. Wie lauten für jeden Seitenersetzungsalgorithmus

- a) die untere Schranke für die Seitenfehleranzahl,
- b) die obere Schranke für die Seitenfehleranzahl?“

“*Consider a page reference string for an application thread with a resident set of m frames, initially all empty. The page reference string is of length p with n distinct page numbers in it. For any page replacement algorithm,*

- a) *what is the lower bound on the number of page faults and*
- b) *what is the upper bound on the number of page faults.*”

a) n

b) p

4. „Skizzieren oder beschreiben Sie die Wirkungsweise von **Copy on Write**! In welchem Zusammenhang wird **Copy on Write** eingesetzt?“

“Describe the method **Copy on Write**. In which content **Copy on Write** is used?”

Copy on Write erlaubt es, daß Threads aus verschiedenen Adreßräumen Daten möglichst platzsparend gemeinsam sowohl lesend als auch schreibend nutzen können. Der erste Thread, der auf diese gemeinsamen Daten modifizierend zugreifen möchte, erhält eine private Kopie der entsprechenden Datenseite(des Datensegments).

Copy on write enables that threads of different address spaces may use common data (read or write) without wasting main memory. The first thraed trying to write on these common data will get a private copy of the corresponding data page (or segment)

Nachname/Last name	Vorname/First name	Matrikelnummer/ Matriculation number
Musterfrau	Marlene	999999

Aufgabe 5/ Question 5:**(7 + 5 Punkte/marks)**

1. „Beschreiben Sie so ausführlich wie nötig, aber dennoch so knapp wie möglich, was ein Mikrokern ist und welche Vor- und Nachteile mit einer Mikrokernarchitektur verknüpft sind!“

“Describe as detailed as necessary, but nevertheless as short as possible what a micro-kernel is and which advantages and disadvantages are connected with a micro-kernel based architecture.”

Ein Mikrokern sollte nur die absolut wichtigen Basisfunktionen eines Betriebssystems enthalten, die eh im privilegierten Prozessormodus ausgeführt werden müssen bzw. die so essentiell sind, daß ohne sie ein System überhaupt nicht organisiert werden kann. Hierzu zählen zum einen Funktionen, die zur Etablierung und zur Kontrolle von Threads und Adreßräumen beitragen, u.a. `create_thread()`, `create_address_space()`, `schedule_thread()` etc.

Damit Threads (im allg. aus verschiedenen Adreßräumen) miteinander interagieren können, indem sie z.B. Nachrichten austauschen, oder sich an kritischen Abschnitten synchronisieren, müssen vom Mikrokern geeignete, d.h. insbesondere schnelle Interaktionsoperationen (IPCs) angeboten werden.

Alle sonstigen Systemfunktionen werden als Dienste (*server*) implementiert, die wie jede Applikation im Benutzermodus ablaufen. Ob jeder Dienst in einem eigenem Adreßraum ablaufen muß, ist eine Entscheidung, die vom Mikrokernansatz unabhängig ist, aber sehr wohl die Leistungsfähigkeit des Systems entscheidend prägt. Insbesondere an die Effizienz der IPC-Operationen werden Ansprüche gestellt, denn wenn dieser Serveransatz überhaupt praktikabel sein soll, müssen die Funktionen zum Interagieren zwischen Anwendungs- und/oder Serverthreads extrem schnell sein. Bei den heutigen Rechnerarchitekturen mit ein- bis mehrstufigen Caches muß deswegen der Mikrokern auch so klein wie möglich sein, damit seine IPC-Funktionen schnell sein können, da der Mikrokern ansonsten den Applikationen zuviel Platz im Cache wegnimmt bzw. zu viele Cachemisses auftreten können.

Der Mikrokernansatz ermöglicht Systemarchitekturen, welche unterschiedliche Varianten eines Dienstes anbieten können, so daß somit flexibel auf neue Hardwarefähigkeiten bzw. bisher nicht unterstützbare Applikationen reagiert werden kann. Ferner kann man damit robustere System erstellen, da sich im Gegensatz zu monolithischen Systemen Fehler eines neuen Dienstes nur innerhalb dieses Dienstes auswirken können.

Fortsetzung von Aufgabe 5 / Question 5 continued:**(5 Punkte/marks)**

2. „Gegeben sei ein symmetrisches Multiprozessorsystem (*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 ZP_4 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 ZP_1 bis ZP_3 . Diese drei kooperierenden Threads synchronisieren sich mittels des unten skizzierten „**Spinlocks**“:

Warum läuft das Schachprogramm deutlich langsamer als auf einem Einprozessorsystem?“

“Given a symmetric multi-processors system (*SMP*) consisting of 4 processors operating on a common main memory via a system-bus. L1 and L2 caches exist per processor. Processor ZP_4 executes a chess program with a large working set that in general 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 ZP_1 through ZP_3 . The three cooperating threads synchronize with each other by means of the following **spinlock**:

Why does the chess program run substantially slower than on a single-processor system?“

DO

```

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 = 1 ⇔ another Thread is in its critical section}
{SpinLock = 0 ⇔ critical section is free }

```

Only one of the three cooperating threads may be in its critical section holding the spin lock SpinLock. The other two threads meanwhile have to compete for acquiring this spin lock. Because both competing threads are reading and writing the variable SpinLock, the contents of SpinLock will be ping-ponged between their related caches, accessing the variable SpinLock in main memory, too. This will generate lots of bus traffic, causing other memory intensive programs in the system to run slower. Due to its large working set the chess program is quite memory intensive.