



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

Tutors:
Uwe Dannowski
Gerd Liefänder
Espen Skoglund

System Architektur

Freiwillige Zwischenprüfung

WS 1999/2000

10. 12. 1999

Nachname	Vorname	Matrikelnummer
Musterperson		4711

- Bitte tragen Sie zuerst auf allen Klausurblättern Ihren Namen, Vornamen und Ihre Matrikelnummer ein, auch auf dem Konzeptblatt.
- Die Prüfung dauert 60 Minuten und besteht aus 5 Aufgaben auf 6 Seiten und einem Konzeptblatt.
- Sie hätten die Prüfung mit mindestens 20 Punkten von maximal 60 erreichbaren Punkten bestanden.
- Es sind keinerlei Hilfen erlaubt!
- Die Prüfung gilt als nicht bestanden, wenn Sie bei einem aktivem oder passiven Betrugsversuch erwischt werden.
- Wenn Sie zusätzliches Konzeptpapier benötigen, verständigen Sie bitte die Klausuraufsicht.
- Bitte machen Sie eindeutig klar, was Ihre endgültige Lösung zu den jeweiligen Teilaufgaben ist. Wir bewerten keine Teilaufgabe, zu der mehr als eine Lösung vorgeschlagen wird.

Die folgende Tabelle wird von uns bei der Korrektur ausgefüllt.

Aufgabe	1	2	3	4	5	Total
Erreichbare Punkte/Aufgabe						60
Erreichte Punkte/Aufgabe						
					Note:	

Nachname	Vorname	Matrikelnummer

Aufgabe 1

(Zum Aufwärmen, 12 Punkte)

Einige der folgenden Aussagen sind wahr, einige falsch. Markieren Sie den Wahrheitswert mit einem Kreuz an der richtigen Stelle.

Einige der folgenden Aussagen sind unvollständig, ergänzen Sie die fehlenden Stellen!

1. Unabhängig von der Schedulingstrategie können *spin locks* in einem Einprozessorsystem zum wechselseitigen Ausschluß (*mutual exclusion*) für sehr kurze kritische Abschnitte (*short critical sections*) verwendet werden.

Wahr:

Falsch: **X**

2. Eines der Ziele beim Zeitscheibenverfahren (*time-slice mechanism*) ist es zu verhindern, daß ein Thread den Prozessor monopolisieren kann.

Wahr: **X**

Falsch:

3. Das Transaktionskonzept sollte die ACID-Eigenschaften erfüllen, wobei die Abkürzungen ACID folgende Bedeutung im Englischen haben:

A = **Atomicity**

C = **Consistency**

I = **Isolation**

D = **Durability**

4. Das normale Zweiphasensperrprotokoll erzeugt maximale Nebenläufigkeit (*concurrency*), „konflikt serialisierbare“ (*conflict serializable*) und wiederaufsetzbare (*recoverable*) Pläne (*schedules*), kann aber zu Verklemmungen (*deadlocks*) führen.

Wahr:

Falsch: **X**

5. Tragen Sie die vier notwendigen Bedingungen für eine Verklemmung (*deadlock*) ein!

1. **Exklusivität (*Exclusivness*)**

2. **Blockierendes Belegen(Hold and Wait)**

3. **Zirkulares Warten(*Circular wait*)**

4. **Keine Verdrängung(*No preemption*)**

6. Das Buddysystem kann sowohl zum internen (*internal*) als auch zum externen (*external*) Verschnitt (*fragmentation*) führen.

Wahr: **X**

Falsch:

7. Nur in Speicherverwaltungssystemen mit verzögerter Freispeichervereinigung (*lazy reunification*) macht eine Speicherbereinigung (*garbage collection*) Sinn.

Wahr: **X**

Falsch:

8. Sie geben einen Speicherblock *rb* der Größe 2^k in einem Buddysystem frei. Um herauszufinden, ob der Buddy von *rb* auch frei ist, müssen Sie die Freiliste der Speicherblöcke der Größe 2^k durchsuchen.

Wahr:

Falsch: **X**

Nachname	Vorname	Matrikelnummer

Aufgabe 2 (Wechselseitiger Ausschluß = mutual exclusion, 4+4+4 Punkte)

Wechselseitiger Ausschluß ist verknüpft mit kritischen Abschnitten (*critical sections*) innerhalb von Threads. Nehmen Sie an, daß die betrachteten Threads zu einer Applikation mit eigenem Adreßraum (*address space*) gehören.

1. Beschreiben Sie so genau wie möglich, was Sie unter einem kritischen Abschnitt auf Applikationsebene (*application level*) verstehen!

Ein kritischer Abschnitt in einem Applikationsthread ist exklusiv auszuführen, da er auf gemeinsame Daten zugreift, d.h. in anderen Applikationsthreads werden in anderen kritischen Abschnitten auf die gleichen Daten zugegriffen. Würde das unkontrolliert nebenläufig erfolgen, dann ist die Konsistenz dieser Daten gefährdet.

2. Welche 3 wesentlichen Anforderungen (*requirements*) sollte eine Lösung zum Problem des wechselseitigen Ausschlusses anbieten?

Wechselseitiger Ausschluß (*mutual exclusion*)

Sofortige Eintrittsmöglichkeit (*progress*)

Kein endloses Warten vor dem Abschnitt (*bounded waiting*)

3. Analysieren Sie, ob die folgende Softwarelösung auf Anwendungsebene (*application level solution*) für die zwei Threads eine korrekte Lösung darstellt, die alle Anforderungen (*requirements*) der Teilaufgabe 2 erfüllt!

```

var flag: array[0..1] of boolean;      {initially flag[0]=flag[1]=false}
{thread 0}                             {thread 1}
repeat                                  repeat
  while flag[1] = true do {nothing};    while flag[0] = true do {nothing};
  flag[0] := true;                      flag[1] := true;
  {critical section}                    {critical section}
  flag[0] := false;                    flag[1] := false;
  {remainder section}                  {remainder section}
forever                                 forever

```

Sofortige Eintrittsmöglichkeit ist gegeben, wenn der andere Thread im nicht kritischen Abschnitt abbricht.

Wechselseitiger Ausschluß ist jedoch nicht gegeben, siehe folgende Sequenz:

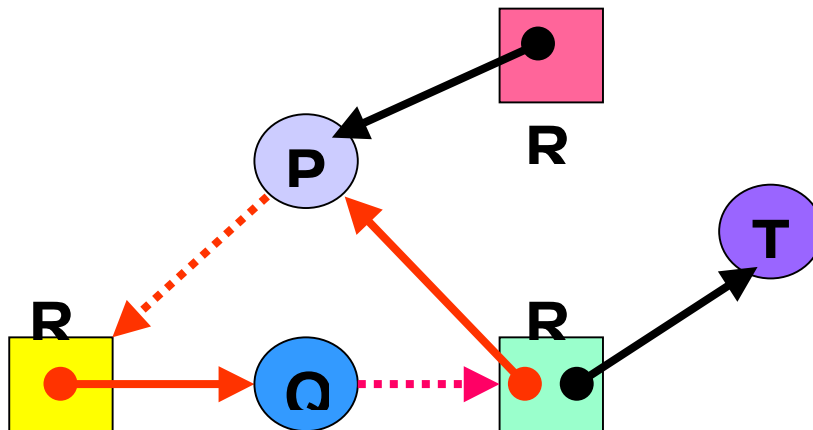
Thread 0 führt die while-Anweisung aus und findet, daß `flag[1] = false` ist. Nun führt Thread 1 diese while-Anweisung aus und findet, daß `flag[0] = false` ist. Beide Threads setzen daraufhin ihre Flags auf 0 und betreten den kritischen Abschnitt.

Nachname	Vorname	Matrikelnummer

Aufgabe 3

(Verklemmung = deadlock, 4 + 4 + 8 Punkte)

1. Zeigen Sie, warum die zirkulare Wartebedingung (circular wait) nur eine notwendige, aber keine hinreichende Bedingung für das Eintreten einer Verklemmung darstellt.



Zwischen den Threads P und Q existiert eine zirkulare Wartebedingung, da P eine Betriebsmittleinheit des grünen Betriebsmittels besitzt und auf das gelbe Betriebsmittel wartet, das gerade Q besitzt, während Q auf eine BM-Einheit des grünen Betriebsmittels wartet, weil keine der beiden grünen BM-Einheiten gerade frei ist. Gleichwohl ist obiges System nicht verklemmt, denn wenn T terminiert, dann kann Q fortsetzen.

2. Warum führt eine unsichere (*unsafe*) Betriebsmittelsituation nicht notwendigerweise zu einer Verklemmung?

Beim Übergang von einem sicheren in den anderen sicheren Systemzustand achtet man darauf, daß selbst wenn alle Threads auf einmal ihre maximalen Betriebsmittelanforderungen stellen würden, es noch eine Ausführungsfolge gibt, in der diese Threads terminieren könnten. Wird von einem dieser n Threads nun eine aktuelle Betriebsmittelanforderung gestellt, die das System unsicher werden ließe, dann könnte zwar eine Verklemmung auftreten, aber es könnte auch sein, daß einige dieser Threads bereits ihre maximalen Betriebsmittelanforderungsphasen hinter sich haben, und quasi nur noch Betriebsmittel sukzessive freigeben, so daß es nicht zu einer Verklemmung kommen muß.

3. In einem System mit 5 Threads $\{T_1, T_2, T_3, T_4, T_5\}$ und 4 Betriebsmitteltypen (resource types) $\{r_1, r_2, r_3, r_4\}$ liege die folgende Betriebsmittelsituation vor. Stellen Sie fest, ob das System noch sicher oder bereits unsicher ist. Geben Sie nach jedem Algorithmenschritt den Vektor der nicht belegten Betriebsmitteln sowie die Menge der noch nicht terminierten Threads an!

$R = (3, 2, 4, 1)$ ist der Vektor der pro Betriebsmitteltyp im System vorhandenen Betriebsmittel.
 C ist die Matrix der Gesamtanforderungen (*maximal resource requirement*) und A ist die Matrix der gerade belegten Betriebsmittel (*currently allocated resources*).

$$\begin{array}{cccc|cccc}
 3 & 2 & 0 & 0 & 0 & 0 & 0 & 0 \\
 2 & 0 & 4 & 0 & 1 & 0 & 0 & 0 \\
 C = 1 & 1 & 2 & 1 & A = 0 & 1 & 1 & 1 \\
 2 & 2 & 2 & 0 & 1 & 0 & 2 & 0 \\
 1 & 1 & 1 & 0 & 0 & 0 & 1 & 0
 \end{array}$$

Mit „ $R - A$ “ \Rightarrow $F = (1, 1, 0, 0)$

Gesucht: T_i mit kleineren Restanforderungen als F , also ein T_i : “ $C_i - A_i$ “ $\leq F$, das ist $T_5!$ \Rightarrow nach Terminierung von T_5 ist der neue

$F' = (1, 1, 1, 0)$, d.h. $NT' = \{T_1, T_2, T_3, T_4\}$, nun kann T_3 fortsetzen und terminieren, denn $(1, 0, 1, 0) \leq F' \Rightarrow$

$F'' = (1, 2, 2, 1)$, d.h. $NT'' = \{T_1, T_2, T_4\}$, nun kann T_4 fortsetzen und terminieren, denn $(1, 2, 0, 0) \leq F'' \Rightarrow$

$F''' = (2, 2, 4, 1)$, d.h. $NT''' = \{T_1, T_2\}$, nun kann T_2 fortsetzen und terminieren, denn $(1, 0, 4, 0) \leq F''' \Rightarrow$

$F'''' = (3, 2, 4, 1)$, d.h. $NT'''' = \{T_1\}$, nun kann auch T_1 fortsetzen und terminieren, denn $(3, 2, 0, 0) \leq F'''' \Rightarrow NT'''''' = \{\}$,

\Rightarrow Das System ist sicher

Nachname	Vorname	Matrikelnummer
----------	---------	----------------

--	--	--

Aufgabe 4

(Threadumschaltung = thread switch, 4 +4+2 Punkte)

1. Erläutern Sie die wesentlichen Unterschiede zwischen „Userlevel Threads“ und „Kernellevel Threads“ sowie der jeweils entsprechenden Threadumschaltung (dispatching).

Nur Kernellevel Threads sind dem zentralen Dispatcher im Kern überhaupt bekannt, d.h. haben als Repräsentanten im Kern einen Threadkontrollblock (TCB). Der zentrale Dispatcher schaltet demnach nur zwischen Kernellevel Threads um, bei dieser Umschaltung wird zunächst vom nicht privilegierten Prozessormodus (user mode) in den Kernmodus (kernel mode) umgeschaltet, ehe dann im Kern die zentrale Prozedur Thread_Switch aufgerufen wird.

Ein Userlevel Thread kann auf einen Kernellevel Thread abgebildet werden, dann wird er wie oben beschrieben abgehandelt. Es ist aber auch möglich, alle Userlevel Threads einer Applikation auf einen einzigen Kernellevel Thread abzubilden, d.h. der zentrale Umschalter kennt dann nur einen TCB für die gesamte Applikation, somit wird gemäß der Strategie des zentralen Dispatcher diese Applikation als Ganzes umgeschaltet. Welcher Userlevel Thread bei einer Umschaltung dann gerade rechnet, muß von einem applikationsspezifischen Umschalter bestimmt werden.

2. Warum muß der Befehlszeiger (*instruction pointer*) innerhalb der Prozedur Thread_Switch nicht gerettet und geladen werden?

Weil der Wechsel vom alten zum neuen Kernstapel immer an der gleichen Programmadresse, mittendrin im Thread_Switch erfolgt.

3. Zählen Sie mindestens 4 unterschiedliche Anlässe bzw. Ereignisse (*events*) auf, die Anlaß für einen Threadumschaltung (thread_switch) darstellen.

Terminierung eines Threads

Ende einer Zeitscheibe (Time Slice Interrupt)

Blockierung nach der Initiierung einer synchronen E/A

Erzeugung bzw. Aktivierung eines höherprioren Threads

Deblockierung eines höherprioren Threads...

Nachname	Vorname	Matrikelnummer

Aufgabe 5 (Speicherverwaltung = memory management, 4+4+2 Punkte)

1. Charakterisieren Sie das Randkennzeichnungsverfahren (boundary tag) hinsichtlich der in der Vorlesung angegebenen orthogonalen Entwurfparameter!

Beliebige Reihenfolge an Belege- bzw. Freigabeoperationen (*arbitrary allocate-/release operations*)

Beliebige Anforderungsportionen (*arbitrary sized portions*)

Integrierte Speicherverwaltungsstrukturen

Externer Verschnitt (*external fragmentation*)

Verschiedene Vergabestrategien sind möglich (z.B. Best-Fit, First-Fit,...)

Sowohl sofortige als auch verzögerte Wiedervereinigung sind möglich

2. Zählen Sie möglichst viele Vorteile des virtuellen Speichers (*virtual memory management*) gegenüber einer nicht virtuellen Speicherverwaltung auf!

Der logische Adreßraum kann größer als der physische Adreßraum, d.h. Hauptspeicher sein, ohne daß sich der Anwender um die Zuordnung benötigter Portionen seines logischen Adreßraums auf Ausschnitte des Hauptspeicher kümmern muß.

Flexiblere Ausnutzung des Betriebsmittels Hauptspeichers.

Elegante Ausnutzung des Prinzips der Lokalität, d.h. gleichzeitig werden mehr oder weniger genau gerade die benötigten Ausschnitte des logischen Adreßraums im Hauptspeicher gehalten.

Erhöhung des Multiprogrammingsgrades und somit eine mögliche Erhöhung der Systemleistung.

Eleganter und relativ fein granularer Schutz gegen willkürlichen bzw. beabsichtigten Zugriff durch Threads aus anderen Adreßräume.

3. Zählen Sie Nachteile des virtuellen Speichers auf.

Zusätzliche Hardwarekosten (MMU, TLB etc.)

Erhöhter Zeitaufwand bei den Zugriffen, teilweise muß über mehrere Abbildungstabellen (z.B. Seitentabellen) gegangen werden.

Erhöhter Raumaufwand durch zusätzliche Abbildungstabellen.

