



## Musterlösung zur Klausur

### System Architektur

22. Februar 2000

#### Aufgabe/Exercise 1

(4 + 2 + 1 + ... + 1 Punkte/Points)

1. Zählen Sie vier verschiedene Anlässe für eine Threadumschaltung (*dispatching*) auf!  
*Enumerate four different reasons for a dispatcher activity!*

- a) Ende einer Zeitscheibe/ *end of a time slice*
- b) Kooperatives Scheduling / *cooperative scheduling (by yield)*
- c) Terminierung des Threads / *termination of a thread*
- d) Blockierender Systemaufruf / *blocking system call (e.g. I/O)*
- e) Verdrängung durch wichtigeren Thread / *preemption by a high priority thread*
- f) Seitenfehler / *page fault*

2. Zählen Sie mindesten zwei Situationen auf, in denen der aktuelle Inhalt einer Kachel nicht ausgelagert werden darf.  
*Enumerate at least two situations in that the current content of a page frame cannot be replaced.*

- a) Seite wird als E/A-Puffer benutzt, der von der E/A-Einheit nur physisch adressierbar ist / *page serves as I/O-buffer, which is only physically addressable by the I/O-unit*
- b) Seite ist Bestandteil des Seitentausches selbst / *page belongs to the paging algorithm*
- c) Seite wird gerade eingelagert / *page is currently paged in*
- d) Seite einer Echtzeitaufgabe ist „festgenagelt“ / *pinned page of a real-time task*

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. „Es sei ein strikt prioritätsorientiertes System ohne Verdrängung vorausgesetzt, in dem also alle Warteschlangen (*waiting queues, ready queue* etc.) strikt nach Prioritätswerten geordnet sind, also nicht nach dem FCFS-Prinzip (*first come first served*). Dann ist garantiert, daß alle Threads, die darauf warten, den kritischen Abschnitt zu betreten, gleiche oder kleinere Priorität besitzen, als der Thread, der sich gerade im kritischen Abschnitt befindet.

*“Assume there is a strictly priority oriented system without preemption, i.e. all queues (e.g. waiting queues, ready queue etc.) are ordered strictly by priorities, not according to FCFS (first come first served). Then it is guaranteed that all threads waiting to enter the critical section have less or equal priority than the thread inside the critical section.”*

korrekt

**inkorrekt**

4. „Kooperatives Scheduling (mittels *yield*) garantiert faires Scheduling in einem Multiprogramming-System.“

*“Cooperative scheduling (through yield) guarantees fair scheduling in a multiprogramming-system.”*

korrekt

**inkorrekt**

5. „Die in der *Inode* abgelegten Zugriffsrechte bei Unixdateien sind *Capabilities*.“

*“The access rights within the inode of a Unix file are capabilities.”*

korrekt

**inkorrekt**

6. „Unsichere Zustände (*unsafe*) führen immer zu einer Verklemmung (*deadlock*).“

*“Unsafe states lead always to a deadlock.”*

korrekt

**inkorrekt**

7. „Ein symbolischer Verweis (*symbolic link*) in Unix kann auf ein Verzeichnis (*directory*) in einem nicht montierten Subdateisystem verweisen.“

*“In Unix a symbolic link may reference a directory in a non-mounted sub file-system.”*

**korrekt**

inkorrekt

8. „Ein Eintrag in einem Unixverzeichnis (*directory*) enthält – neben einem Längelfeld – nur den Dateinamen und die *Inodenummer*.“

*“An entry in a Unix directory contains – besides a length field – only the file name and the inode number.”*

**korrekt**

inkorrekt

## Aufgabe/Exercise 2

( 3 + 3 + 6 Punkte/Points)

Wechselseitiger Ausschluß (*mutual exclusion*) ist eng verknüpft mit kritischen Abschnitten (*critical sections*).

*Mutual exclusion is closely related to critical sections.*

1. Zählen Sie drei unterschiedliche Lösungstypen (*types of solutions*) zur Realisierung des wechselseitigen Ausschlusses (*mutual exclusion*) auf! Geben Sie zu jeder Lösung jeweils eine konkrete Methode an!

*Enumerate three different types of solutions for implementing mutual exclusion and give a concrete method per solution type!*

**Reine Softwarelösung auf Anwendungsebene, z.B. Peterson's Algorithmus o.a. /  
*pure software solution, e.g. bakery algorithm etc.***

**Hardware unterstützte Lösungen, z.B. Teste und Setze-Befehl etc. /  
*Hardware supported solutions, e.g. Disable Interrupts, CompareAndSwap***

**Sprach-/Betriebssystemunterstützte Lösung, z.B. Monitore oder Semaphore /  
*Programming-language or OS supported solutions, e.g. monitors or semaphore***

2. Gegeben sei ein Einprozessorsystem, das Spinlocks zum wechselseitigen Ausschluß kurzer kritischer Abschnitte für Anwenderprogrammierer anbietet. U.a. gibt es eine Applikation aus drei hochprioren Threads, die über Monate hinweg problemlos ausgeführt wurde. Nach einem Wechsel zu einer neuen – korrekt implementierten – Betriebssystemversion (mit gleichem API) wird diese Applikation nicht mehr korrekt ausgeführt. Geben Sie eine mögliche Erklärung für dieses Systemverhalten an!

*Assume there is a single processor system offering spin locks to application programmers to implement short critical sections! There is an application task consisting of three high priority threads. The application executes without problems for months. After switching to a new – correctly implemented – OS-version (same API), the application no longer works correctly. Try to find an explanation for this system behavior!*

**Die neue Schedulingstrategie benutzt ein starres Prioritätsschema ohne Zeitscheibenmechanismus, so daß ein am Spinlock aktiv wartender hochpriorer Thread alle anderen Threads davon abhält, den Prozessor zu bekommen (oder diese hochprioren Threads werden durch einen niederprioren Thread, der ebenfalls diesen Spinlock benutzt, ausgebremst, Prioritätsumkehr)**

*The new scheduling policy is strictly priority oriented without time slicing, i.e. one of the three high priority threads waiting in front of a "spin locked" critical section prevents any other thread to be executed (or these high priority threads sharing this spin lock with a low level priority thread may suffer from priority inversion)*

3. Gegeben sei eine Applikation aus 3 zyklischen Threads  $T_0$ ,  $T_1$  und  $T_2$ , in der diese 3 Threads turnusmäßig ablaufen sollen, d.h.  $a_0, a_1, a_2, a_0, a_1, a_2$  usw., wobei Thread  $T_i$  folgenden Code ausführen möge:

*Suppose you have an application consisting of three cyclic threads  $T_0$ ,  $T_1$  and  $T_2$ , running in turns, i.e. repeatedly one after the other,  $a_0, a_1, a_2, a_0, a_1, a_2$  etc. Thread  $T_i$  executes the following code:*

```
while (true) {  
    WaitForTurn(i);          /* ??? */  
    ai;  
    FinishedTurn(i);        /* ??? */  
};
```

Hinweis:

Benutzen Sie z.B. Semaphore und entwerfen Sie den Code (nebst aller benötigten Variablen) für die Operationen `WaitForTurn(i)` und `FinishedTurn(i)`. Sie können also die Semaphoreoperationen `P()` oder `V()` verwenden, ohne ihre Implementierung zu beschreiben oder zu codieren!

Hint:

*Use, e.g., semaphores and design the code (including all required variables) for the operations `WaitForTurn(i)` and `FinishedTurn(i)`. You do not have to code the semaphore-operations `P()` and `V()`!*

```
semaphore s[2] = {1,0,0};  
/* initialized array of semaphores */
```

```
WaitForTurn(i) {  
    P(s[i]);  
}
```

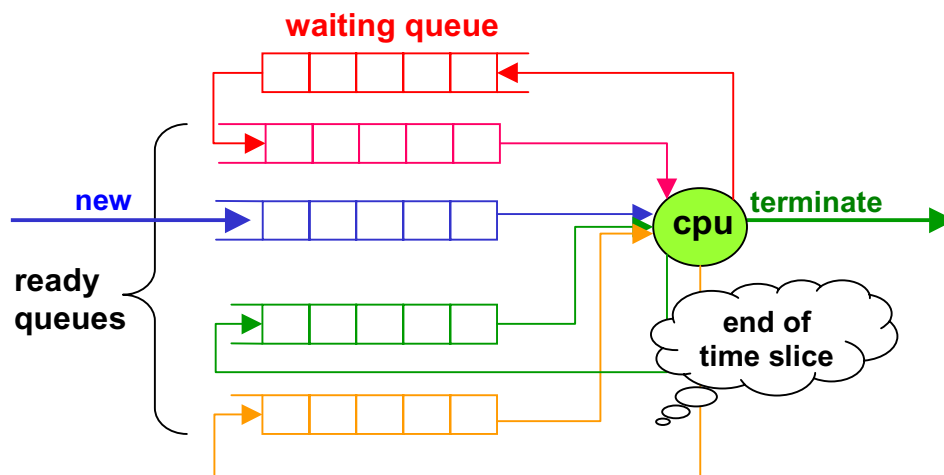
```
FinishedTurn(i) {  
    V(s[(i+1)%3]);  
}
```

### Aufgabe/Exercise 3

(6 + 4 + 2 Punkte/Points)

1. Erläutern Sie die Arbeitsweise der Multilevel-Feedback Umschaltstrategie anhand einer Skizze! Beschränken Sie sich dabei auf drei Levels und eine CPU! Berücksichtigen Sie dabei auch den Threadzustand „blockiert“ bzw. „wartend“, aus dem heraus Threads nach Beendigung des Wartegrunds bevorzugt um die CPU konkurrieren sollen.

*Illustrate the functionality of the multilevel-feedback scheduling policy by a rough drawing! Restrict it to one CPU and to three levels! Include also the thread state “blocked” respectively “waiting”. After leaving this “blocked/waiting” state such threads are favored when competing for the CPU.*



2. Geben Sie vier verschiedene Gründe dafür an, warum man Threads blockieren will!  
*Give four different reasons why you may want to block threads!*

**Warten auf ein Synchronisationssignal von einem Partnerthread /**  
*Waiting for a signal from some other thread*

**Warten auf eine Nachricht (message) von einem Server /**  
*Waiting for a message (e.g. an acknowledge) from a server*

**Warten auf das Ende einer E/A-Aktivität /**  
*Waiting for the end of an I/O*

**Seitenfehler mit Ein-/Auslagern /**  
*pagefault with swappin/out*

**Warten auf bestimmten Zeitpunkt /**  
*waiting for a certain time stamp*

**Warten auf das Freiwerden eines Betriebsmittels (resource) /**  
*Waiting for the release of a resource*

3. Erklären Sie, warum wir innerhalb der Kernfunktion *thread-switch* den Befehlszähler IP (*instruction pointer*) des laufenden Threads nicht retten müssen!  
*Explain why we do not have to save the instruction pointer IP of the current thread in the kernel-function thread-switch!*

**Innerhalb von *thread-switch* wird stets an der gleichen Programmadresse der alte Stapel gegen den neuen Stapel ausgetauscht, der Userlevel-Befehlszähler IP des alten Threads wird bereits beim Eintritt in den Kern auf den Kernstapel gerettet /**

***Within the function we exchange the old stack against the new one always at exactly the same instruction pointer, whereas the user level instruction pointer is saved within the kernel entrance utility.***

#### **Aufgabe/Exercise 4**

**(2 + 2 + 2 + 2 + 4 Punkte/Points)**

1. Geben Sie zwei wesentliche Vorteile des virtuellen Speichers an!  
*Give two major advantages of a virtual memory!*

**Nutzung von Adreßräumen, die größer als der physische Speicher sind /  
*You can use address spaces larger than main memory***

**Beim virtuellen Speicher müssen nur die gerade benötigten Adreßraumbereiche eingelagert sein, auf die der oder die Threads gerade zugreifen, so daß der Rest des Hauptspeichers für andere Aktivitäten genutzt werden kann /  
*You do not have to load a complete address space into main memory.***

**Schutz vor anderen Programmen / *protection against other programs***

2. Belady stellte fest, daß der FIFO-Seitenersetzungsalgorithmus (*page replacement*) eine Anomalie besitzt. Beschreiben Sie diese Anomalie!  
*Belady stated that the FIFO page replacement method has an anomaly. Describe this anomaly!*

**Die Belady-Anomalie besagt, daß ein größerer Hauptspeicher u.U. beim FIFO-Verfahren zu mehr Seitenfehler führt als ein kleinerer Hauptspeicher. /  
*A larger main memory may induce more page faults than a smaller one.***

3. Geben Sie mindestens zwei Gründe dafür an, warum kleinere Seitengrößen die Speicherauslastung verbessern können!  
*Give at least two reasons why smaller page sizes can lead to a better storage utilization!*

**Der interne Verschnitt ist kleiner, ferner ist auch die Information, die mit eingelagert, aber auf die u.U. nie zugegriffen wird, kleiner /  
*Internal fragmentation is smaller, furthermore the surrounding information which may never be used is smaller, too.***

4. Nennen Sie einen Grund, weswegen kleinere Seitengrößen die Systemleistung verschlechtern können!

*Give a reason why smaller page sizes might decrease system performance!*

**Kleinere Seitengrößen vergrößern den räumlichen Overhead, z.B. größere Seitentabellen, mehr TLB-Einträge /**  
*Smaller page sizes induce larger page tables and more TLB entries, thus they may reduce the multi programming degree.*

**Häufigere Transporte von kleineren Transporteinheiten zwischen Platte u. Hauptspeicher sind ineffizienter (z.B. bei sequentiellen Zugriffen) /**  
*More transfers of smaller sizes between disk and main memory are less efficient (e.g. for strictly sequential accesses)*

5. Unser System habe 3 Kacheln (*page frames*) und 8 Seiten (*pages*). Initial sind alle Kacheln leer, d.h. enthalten keine Seite. Wir referenzieren die Seiten **0, 1, 7, 2, 3, 2, 7, 1, 0, 3** in genau dieser Reihenfolge. Wieviel Seitenfehler treten auf, wenn wir die LRU Ersetzungsstrategie benutzen? (Sie können die folgende Tabelle verwenden!)

*Our system has 3 page frames and 8 pages. Initially, all page frames are empty, i.e. no page frame contains a page. We reference pages 0, 1, 7, 2, 3, 2, 7, 1, 0, 3 – in exactly that order. How many page faults will occur if we use LRU as page replacement policy? (You can use the following table.)*

	0	1	7	2	3	2	7	1	0	3
Kachel 0	0	0	0	2	2	2	2	2	0	0
Kachel 1		1	1	1	3	3	3	1	1	1
Kachel 2			7	7	7	7	7	7	7	3
Seitenfehler	X	X	X	X	X			X	X	X

Anzahl Seitenfehler/Number of page faults: **8**

## Aufgabe 5

(3 + 5 + 4 Punkte/Points)

1. Ein Angreifer (*attacker*) kann ein System beschädigen, in dem er bösartige Programme bzw. Programmteile ins System einschleust. Zählen Sie solche Typen bösartiger Programme bzw. Programmteile auf, welche sich selbst replizieren können.

*An attacker can damage a system by inserting some malicious programs respectively parts of programs into the system. Enumerate those types of malicious programs respectively program parts that can replicate themselves!*

**Viren / viruses**

**Bakterien / bacteria**

**Würmer / worms**

2. Sie sollen ein möglichst sicheres Passwortschema einrichten, in dem die Passwortlänge auf acht Zeichen limitiert ist. Welche zusätzlichen Schutzmaßnahmen schlagen Sie vor?

*You have to establish an as-secure-as-possible password scheme but password length is restricted to 8 characters. Which additional protection measures do you propose?*

**Akzeptiere nur Passwörter, die weder im Wörterbuch stehen, noch voreingestellte Passwörter sind, noch mit persönlichen Daten des Benutzers zusammenhängen (z.B. Name eines Kinds)**

**Verlange eine Mindestanzahl von Sonderzeichen im Passwort**

**Setze ein regelmäßiges Update des Passworts durch (ohne Zulassung von bereits benutzten Passwörtern)**

**Das neue Passwort muß sich in einer Mindestanzahl von Stellen vom Vorgängerpasswort unterscheiden**

**Einsatz von Passwortschlüsselungsprogrammen, die bei Erfolg, den Benutzer davon unterrichten, bzw. dazu zwingen, sein Passwort zu ändern**

**Benutze das „Salt“-Konzept**

**Schlage rechnerproduzierte Passwörter vor**

**Protokolldatei aller Anmeldevorgänge**

**Verlange vor/während einer Sitzung zusätzliche persönliche Daten**

**Bei abweichendem Nutzerprofil eingreifen**

**Zusatzhardware, z.B. Smart-Cards, Scanner für Biometrie (z.B. retina)**

*Accept only passwords that are neither in a dictionary, nor that they are default ones, nor that they are related to personal data of the user (e.g. name of a child)*

*Demand at least some special non letter characters in the password*

*Enforce a periodic update of the password not allowing to reuse earlier passwords*

*The new password must differ in at least some characters from its predecessor*

*Use password check programs, in case of a success inform the user and enforce him to alter its password*

*Use the salt concept*

*Propose automatic generated password.*

*Auditing of all logins*

*Demand before and during a session personal data*

*React on unusual user profile*

*Additional hardware, e.g. smart cards, biometrical scanner (e.g. retina)*

3. Erläutern Sie den Begriff einer mandatorischen Sicherheitsstrategie und geben Sie ein Beispiel dafür an!

*Explain the term mandatory security policy and give an example!*

**Durch eine mandatorische Sicherheitsstrategie wird eine für das gesamte System verbindlich geltende Restriktion eingeführt, wie Subjekte einer bestimmten Sicherheitsstufe auf Objekte einer anderen oder der gleichen Sicherheitsstufe zugreifen dürfen. Nutzer haben somit keine Möglichkeit, diese verbindlichen Vorschriften zu verändern.**

*A mandatory security policy describes the means of restricting access to objects on the sensitivity (as represented by a label) of the information contained in the objects and the formal authorization (i.e. clearance) of subjects to access information of such sensitivity.*

**Beispiel ist das Bell-LaPadula- oder das Clark-Wilson (angewendet auf ein militärisches Informationssystem, bei dem nach “oben zwar geschrieben”, aber nur von “unten gelesen” werden darf, wenn oben beispielsweise streng geheim, darunter geheim, vertraulich und schließlich öffentlich verstanden wird).**

*An example is the Bell-LaPadula- or the Clark-Wilson-model (applied to a military information system with levels according to top secret, secret, ..., public, in that subjects may write to objects of higher levels, but may read from lower levels, only).*

**Legende:**

**Die blau angegebenen Musterlösungen sind originelle Studentenlösungen.**