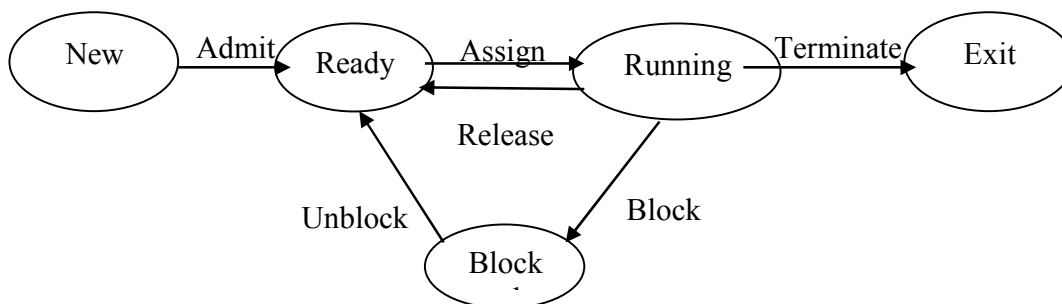


## Musterlösung System Architektur (Architecture) Klausur (Examination) (15.09.04)

### Aufgabe/Question 1 (Zum Aufwärmen/Warm up, 4 + 2 + 1 + ... + 1 Punkte/marks)

1. „Gegeben sei ein Einprozessorsystem mit folgenden **fünf Threadzuständen**. Welcher dieser Threadzustände wird typischerweise durch mehr als **nur eine Liste** implementiert? (Bei **Multiple Choice Zutreffendes** unterstreichen und **kurze Begründung**)“

“Given a single processor system with the following **five thread states**. Which of these thread states typically will be implemented using more than **just one single list**? (Underline the correct answers in multiple choice and give a short reasoning)”



new

ready

running  
Priorities

blocked  
Pro Event extra queue  
⇒ shorter access times

exit

2. „Warum werden erfahrene Kernprogrammierer versuchen, sowohl **Rekursion** als auch **große Arrays von lokalen Variablen** in einer **stack-basierten Kernarchitektur** zu vermeiden?“  
“Why do experienced kernel programmers try to avoid **recursion** and **large arrays of local variables** in a **stack based kernel architecture**?”

**Kernel stack is limited, thus avoiding kernel stack overflow**

3. „Gegeben sei ein Mehrprozessorsystem, bei dem jeder Thread einer **multi-threaded Anwendung** aus Cachelokalitätsüberlegungen stets auf **dem gleichen Prozessor** abläuft. Dann ist es für die Gesamtbearbeitungszeit **unerheblich**, ob man das **PULT-** oder das **KLT-Threadmodell** einsetzt.“

“Given a multi-processor system where every thread of a **multi-threaded application** is always running on the **same CPU** for cache-locality reasons. Then it does **not influence** the total turnaround-time whether you are using the **PULT-** or the **KLT-thread model**.”

korrekt

inkorrekt

4. „Die meisten Architekturen benutzen einen **direkt-abgebildeten Pufferspeicher** (direct mapped cache), um **ihren TLB** zu implementieren.“

“Most architectures use a **direct-mapped cache** to implement their **TLB**.”

korrekt

inkorrekt

5. „Eine **Unterbrechungsbehandlung** (*interrupt handler*) verrichtet ihre Arbeit bezogen auf das **E/A-Gerät**, das die Unterbrechung ausgelöst hat, und wird dann **unmittelbar darauf** den **Anwendungsthread** deblockieren, der zuvor diese E/A initiiert hatte.“  
*“An interrupt handler does its task related with the I/O-device causing the interrupt and immediately afterwards unblocks the application thread that previously had initiated this I/O.”*

korrekt

inkorrekt

6. „Das **Reiser Dateisystem** erhöht **Zuverlässigkeit** durch die Konzepte **Transaktionen** und durch **Protokolldateien** (*journaling*).“  
*“The Reiser file-system increases availability and recovery by the concepts of transactions and journaling.”*

korrekt

inkorrekt

7. „Bei der Plattenzugriffstrategie **„Kürzeste Suchzeit Zuerst“** (*shortest seek time first*) können Plattenaufträge **verhungern**.“  
*“Disk requests of the scheduling policy shortest seek time first may starve.”*

korrekt

inkorrekt

8. „**Plattenblock-Interleaving** wurde eingeführt, um die **Übertragungsrate** moderner Platten an den **Durchsatz** der Plattencontroller (*disk controller*) und des Bussystems anpassen zu können.“  
*“Disk block Interleaving has been introduced to adapt the transfer rate of modern disks to the throughput of disk controllers and the bus system.”*

korrekt

inkorrekt

## Aufgabe/Question 2

(1.5+2+1+ 6 Punkte/marks)

1. „Nennen Sie **drei verschiedene Ereignisse**, die zu einem Übergang vom Threadzustand **„Rechnend“** nach **„Bereit“** mittels der Übergangsfunktion **„Release“** führen!“  
*“Name three different events causing a transition from thread-state “Running” to “Ready” with the transition-function “Release”.”*

Higher priority thread is activated or becomes ready again

End of time slice of current thread

Yield

2. „Vergleichen Sie **kooperatives** mit **verdrängendem** (*preemptive*) **Scheduling**!“  
*“Compare cooperative with preemptive scheduling.”*

With cooperative scheduling skilled programmers may avoid unnecessary thread-switches which may have a serious impact on system overhead. However, cooperative scheduling is not a general solution, i.e. it can be used within the set of threads of one application or within a static system (e.g. an embedded system),

where the application programmer has the needed survey of all currently activated threads. This method also requires cooperative programmers otherwise starvation may become a major problem.

Preemptive scheduling –either supported via time-slicing or via priorities- is a more general approach, but may lead to avoidable thread-switches. Preemptive scheduling is more a technique for hard-real time systems, whereas cooperative scheduling may be used within soft real-time systems.

3. „Warum würde ein hypothetischer Ablaufplaner (*scheduler*) bei der Auswahl des **nächsten bereiten Threads** einen aus **demselben Adressraum** gegenüber einem Thread **aus einem anderen Adressraum** bevorzugen? Ist dies immer eine **gute Idee**? **Begründen** Sie Ihre Ansicht.“

*“Why would a hypothetical scheduler favour a thread from the same address space over a thread from a different address space? Is this always a good idea? Reason your answer.”*

Switching to a thread within the same address may save a lot of overhead, i.e. you do not have to sweep your TLB, you do not need to reestablish the needed address space information for the MMU.

However, this idea is sometimes bad. Suppose one application with a lot of KLT-threads is currently running this may lead to the starvation of all other applications.

4. „Welches Problem versucht man mit dem „**Fair-Share Scheduler**“ in einem **Mehrbenutzer-system** zu lösen?“

*“What is the problem that **fair-share scheduling** tries to solve in a **multi-user system**?”*

To enable fair resource allocation of system capacity to all active users, i.e. a user with currently 20 active jobs does not get more capacity than another one with only one job.

5. „Abstrahieren Sie von Hardwaredetails, beschreiben Sie gleichwohl so knapp wie möglich und ausführlich wie nötig, welche Folge von **Kernprozeduraufrufen** in einem **Unix-ähnlichen Kern** benötigt wird, wenn im Rahmen einer **Zeitgeberunterbrechung** (*timer interrupt*) ein Threadwechsel (*thread switch*) vorgenommen werden muss!“

*“Abstract from hardware details, describe as concisely as possible, but as completely as necessary the sequence of **kernel-procedure calls** performed in a **Unix-like kernel** when a **timer-interrupt** leads to a thread switch.”*

Save rest of registers

Release(CT)

NT=schedule()

if AS(NT) ≠ AS(CT) AS\_SWITCH

Thread\_switch(NT)

Assign(CT') /\* CT' =NT \*/

Set Timer(CT')

Load rest of register

RTI



2. „Beschreiben Sie **knapp** und **präzise**, was passieren würde, wenn die **Speicheranforderung D** aus Teilaufgabe 3.1 eine Größe von **410 KB** gehabt hätte!“  
*“Describe as **concisely** and **precisely** as possible what would have happened if the **memory request D** from question 3.1 had a size of **410 KB**.”*

**This request can not be done, out of free memory.**

3. „Gegeben sei ein zu verwaltender Speicher der **Größe  $2^k$** , der nach dem Halbierungsverfahren (*buddy system*) verwaltet wird. Wie viele Einträge (*entries*) stehen **minimal** und **maximal** in der **Freispeicherliste** zur **Anforderungsgröße  $2^m$** ,  $m < k$ ?“  
*“Given a memory of size  $2^k$  managed according to the buddy system. How many entries are there in the free list for a memory request size of  $2^m$  maximally and minimally with  $m < k$ ?”*

**Minimal: 0**

**Maximal:  $\frac{1}{2} * 2^{k-m}$**

#### **Aufgabe/Question 4**

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

1. „In der Systemarchitektur wird das **Prinzip des „verzögerten Tuns“** (*lazy doing*) mehrfach angewendet. Zählen Sie hierfür **zwei typische Anwendungen** auf!“  
*“In system architecture, we have used the **principle of “lazy doing”** multiple times. Enumerate **two typical applications** of this principle.”*

**Lazy filling of a new address space**

**Lazy saving of file blocks to the disk**

**Lazy saving of dirty pages to swap area ...**

2. „Vergleichen Sie **Seitentauschen auf Verlangen** (*demand paging*) mit **vorausschauendem Seitentauschen** (*pre-paging*)! Nennen Sie jeweils deren Vor- und Nachteile!“  
*“Compare **demand paging** with **pre-paging**. Mention their advantages and disadvantages.”*

**With demand paging you exactly page in those parts of an address space or a mapped file that are currently needed. However, each fault requires some delay of the application.**

**With pre-paging you speculate what parts of the address space respectively mapped file will be used in the future. If you're right your application does fewer page faults, if you're wrong, then the needed main memory frames are just wasted (i.e. space overhead) and the previous page-ins have been done in vain (i.e. time overhead).**

3. „Zählen Sie die **Hauptaufgaben** einer allgemeinen **Speicherverwaltung** auf“  
 “Enumerate the *main tasks* of a general *memory management* system..”

**Free memory management and book keeping of allocated memory with the two main interface functions:**

**allocate\_memory and release\_memory**

**Modern memory management systems additionally offer:**

**Sharing of memory**

**Copy on write**

**Avoiding of defragmentation**

4. „Beschreiben Sie das Konzept und die benötigten Datenstrukturen eines ganz **konkreten Speicherverwaltungsverfahrens** (≠ Halbierungsverfahren) Ihrer Wahl!“  
 “Describe the concept and the needed data structures of one *concrete memory management system* (≠ buddy system) of your own choice.”

**Stack based memory management, i.e. always the last allocated memory portion will be released first (LIFO-principle)**

**You need two pointers, one to the bottom and one to the top of the stack.**

**Aufgabe 5/ Question 5:**

**(1 + 3 + 2 + 6 Punkte/marks)**

1. „Warum erlaubt UNIX einem **Normalbenutzer nicht**, **Hardlinks** auf seine Verzeichnisse (*directories*) anzulegen?“  
 “Why does UNIX *not allow normal user* to establish *hard links* to his *directories*?”

**Unskilled user might produce a cycle within the file system, ⇒ ls -R might loop forever.**

2. „Beschreiben Sie so ausführlich wie nötig, gleichwohl so knapp wie möglich, welche **Vor- und Nachteile** mit dem **virtuellen Dateisystem** VFS in Unix verknüpft sind!“  
 “Describe as completely as necessary, but as concisely as possible, the *advantages and disadvantages* of the Unix *virtual file system (VFS)*.”

**Advantages: (Ease of programming)**

**Files from very different file-systems can be accessed in the same way, i.e. just using the interface of the VFS, the application programmer no longer has to distinguish between file of different file-systems**

**Disadvantages:**

**Some file-systems may offer very nice specific features that are not supported by VFS**

**Furthermore, the additional interface induces some overhead in time and space.**

3. „Beschreiben Sie so ausführlich wie nötig, gleichwohl so knapp wie möglich, welche Vor- und Nachteile mit einem **zustandsbehafteten** bzw. **zustandslosen Dateiserver** (*file server*) verknüpft sind.“

*“Describe as complete as necessary, but as concisely as possible the **advantages** and **disadvantages** of **stateful** and **stateless** file servers?”*

#### Stateless file server

##### Advantages:

**Fault-tolerance, i.e. if file server has crashed the state of the file-server easily can be reestablished. Some updates on the client sides may be lost, but all file-blocks having been transferred to the file server before the crash are still valid.**

##### Disadvantages:

**Each file block transfer to the file server requires that the client side is adding all the needed file-information.**

#### Stateful file server:

##### Advantages:

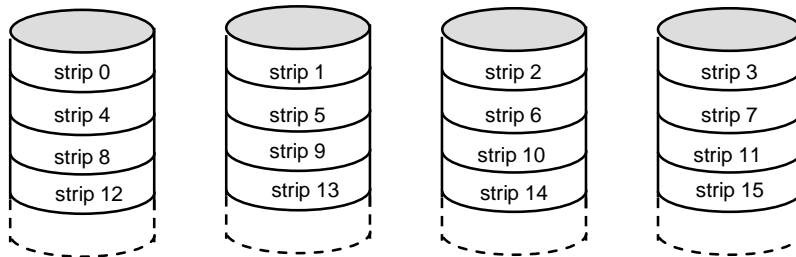
**Enhanced caching on both sides is possible and no need for additional transfer information from the client to the server, once the file has been opened.**

##### Disadvantages:

**Problems with reestablishing the state of the file server, once the file server has crashed.**

4. „Beschreiben Sie, wie bei einem **RAID 0 System** eine **große Datei** auf die **Plattenblöcke abgebildet wird.**“

*“Describe how a **huge file** will be **mapped** to **disk blocks** using a **RAID 0 System.**”*



5. „Analysieren Sie die **Leistung** (*performance*) eines **RAID 0 Systems** aus **vier Platten** in Relation zu einer **einzigsten gleichschnellen Platte** mit **vierfacher Kapazität.**“

*“Analyze the **performance** of a **RAID 0 system** consisting of **four disks** in relation to a **single disk** with **four times capacity** and with **comparable access speed.**”*

#### Performance per application and per file:

**Reading (and writing) up to four times faster (if file is large)**

#### Performance per system and many applications:

**Performance up to four times better, because each application may need file-transfer from a different disk.**