15. September 2005

Aufgabe1/*Question* 1

1. „Zählen Sie **drei Plattenarmstrategien** (*disk arm scheduling policies*) auf, welche das **Leistungsmaß mittlere Zugriffszeit optimieren** bzw. **suboptimal unterstützen**!“
   *“Enumerate **three disk arm scheduling policies** which **optimize** respectively **suboptimize** the **performance measure average access time**.”*
   a) SSTF = shortest seek time first
   b) SCAN (~Elevator)
   c) C-SCAN

2. „Zählen Sie **drei verschiedene Anlässe** dafür auf, weswegen ein **reiner Benutzerebenenthread** (*PULT*) auf einen anderen PULT der **gleichen Task** umschaltet (bzw. umgeschaltet wird)!“
   *“Enumerate **three different causes** why a **pure user level thread** (PULT) switches (or is switched) to another PULT of the **same task**.”*
   a) Yield
   b) Conditional wait inside a monitor function
   c) Exit

2. „**Speicherbereinigung** (*garbage collection*) macht **nur** bei einer **Speicherverwaltung** mit **verzögerter Freispeichervereinigung Sinn**.“
   *”**Garbage collection** makes **sense only** for a **memory management** with **lazy reunification**.”*
   korrekt inkorrekt

3. „Ein **unsicherer** (*unsafe*) **Zustand** einer **multi-threaded Anwendung** führt **immer** zu einer **Verklemmung** dieser Anwendung.“
   *”An **unsafe state** of **a multi-threaded application always** leads to a **deadlock** of this application.”*
   korrekt inkorrekt

4. „**Zwei Threads**, die sich mittels asynchronem `a_send()` und synchronem `s_receive()` synchronisieren, können **nicht** verklemmen.“
   *“**Two threads** which synchronize each other via asynchronous `a_send()` and synchronous `s_receive` can **not** deadlock.”*
   korrekt inkorrekt

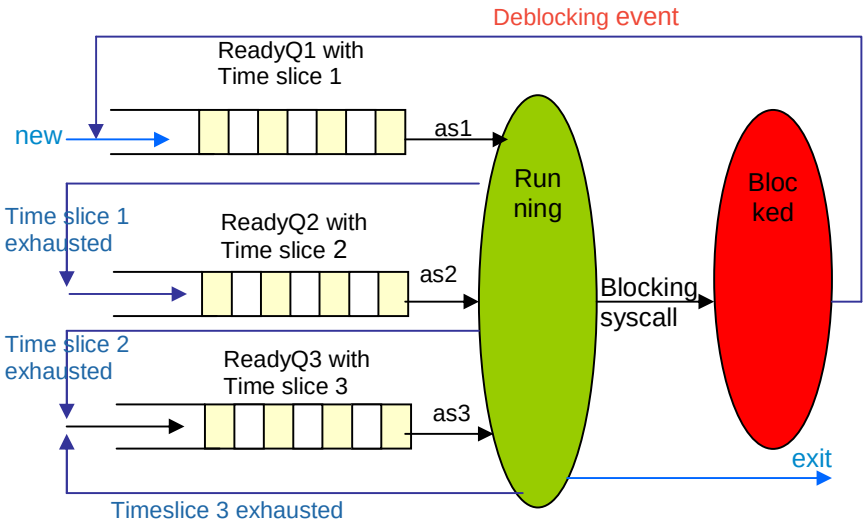
5. „Ein **Linuxsystem** enthält einen **monolithischen Systemkern**.“
   *“A **Linux system** contains a **monolithic kernel**.”*
   korrekt inkorrekt

6. „**Kernel-Level Threads** können **nur** im **privilegiertem Prozessormodus** (*kernel mode*) ausgeführt werden.“
   *“**Kernel-level threads** can **only** be executed in the **privileged CPU-mode** (kernel mode).”*
   korrekt inkorrekt

7. „Die **Arbeitsmenge** (*working set*) $\mathbf{W_P(\Delta t, t)}$ eines Prozesses P zum Zeitpunkt t ist die Menge der Seiten, die in den **letzten realen Δt** Zeiteinheiten referenziert worden sind.“
   *“The **working set** $W_P(\Delta t, t)$ **of a process P at time t** $W_P(\Delta t, t)$ is the set of pages that have been referenced within the **last physical Δt** time units.”*
   korrekt inkorrekt

Aufgabe/*Question* 2

1. „Erläutern Sie die Arbeitsweise der **Multilevel-Feedback Umschaltstrategie** anhand einer Skizze, welche die **drei** Threadzustände „**bereit**“, „**rechnend**“ und „**blockiert**“ enthält! Beschränken Sie sich dabei auf **eine CPU** und auf **drei Level**. Machen Sie dabei deutlich, aus welchen Gründen es zu den **jeweiligen Threadzustandübergängen** kommen kann! Threads, die **nach Beendigung des Wartegrunds** den Threadzustand „**blockiert**“ verlassen, sollen **bevorzugt** um die CPU konkurrieren.“
   *“Illustrate the functionality of the **multilevel feedback CPU scheduling policy** by a drawing which contains the **three** thread states “**ready**”, “**running**”, and “**blocked**”. Restrict it to **one CPU** and to **three levels**. Show for what reasons the **respective thread-state transitions** can happen. Threads leaving the thread state “**blocked**” after the corresponding waiting event had occurred, should be **favored** when competing for the CPU.”*



As long as there is one TCB in the ready queue RQ1, the first one in this queue will be assigned to the processor via as1. Having exhausted its small time slice TS1 on the CPU, the TCB of the running thread will be put back to the ready queue RQ2. A ready thread of RQ2 will be assigned via as2, only if there is no other TCB in RQ1. If a running thread (coming from RQ2) again exhausts its larger time slice TS2, its TCB will be put back to RQ3. A thread waiting in RQ3 can only be assigned via as3, if both RQ1 as well as RQ2 are empty. RQ3 is scheduled according to round robin. However, each time a running thread does a blocking system call, its TCB will be put to one of the blocked states. There this thread is waiting until the corresponding deblocking event will occur. Afterwards its TCB is put back to RQ1. When a thread finishes it leaves the subsystem via exit.

2. „Unterstreichen Sie von den folgenden vier angegeben Umschaltstrategien diejenige, welche die **mittlere Verweilzeit** (*average turnaround time*) **optimiert**!“
   *“From the following four scheduling policies underline the one which **optimizes** the **average turnaround time**.”*

   Priority **Shortest Process Time First**

   Round Robin **Multi-Level Feedback**

3. „Welche der **vier** in der Vorlesung behandelten **Betriebsmittelbelegungsprotokolle** *(resource allocation protocols) sind **verklemmungsfrei**?“*
   *“Which of the **four resource allocation protocols** mentioned in the lecture are **free of deadlocks**?”*

   Non Preemptive Scheduling
   Priority Ceiling
   Stack Based Priority Ceiling

Aufgabe 3 / *Question 3* :

1. „Welche **Probleme** können aus Sicht der **Kommunikationspartner** auftauchen, wenn Sie **Nachrichten unterschiedlicher Länge** unterstützen sollen? Schlagen Sie für jedes angegebene Problem eine geeignete Lösung vor!“
   *“From the viewpoint of the **communicating partners**, what **problems** might arise if you should support **messages of variable size**? Propose an **appropriate solution** for each mentioned problem.”*
   Problem: The receiver’s buffer can be too small.
   Solution: Enhance the receive operation to inform the receiver that message was not yet complete, so the application programmer can perform a “receive-remaining-operation” to get the missing part(s) of the message later on. Or you just inform the receiver, that the original message had been bigger and that x bytes have been lost.
   Problem: If an asynchronous **a_send** is used, the message pool within the kernel might be exhausted. You need a complicated message buffer memory management.
   Solution: Before copying a message from sender buffer to the kernel, check whether there is sufficient contiguous space in this message buffer-pool to contain the complete message. If not, block the **a_send** until another receive has happened. Then try again.
   Instead of variable sized messages you can offer messages of fixed sized length. A larger message is cut down in n*fixed sized message portions augmented by a message ID and a sub-message ID to enable the receive operation to get the full message.
   In any case, in variabkle sized messages either its exact length is a parameter or the end of the message must have a specific character, i.e. an ‘\0.

2. „Welche der **vier notwendigen** Anforderungen an eine **gültige** Lösung des kritischen Abschnittproblems werden durch **Spin-Locks** auf **Einprozessorsystemen nicht** erfüllt?“
   *„When using **spin-locks** on **single-processor systems**, which of the **four necessary** requirements for a **valid** solution of a critical section problem are **not** satisfied?”*
   Portability
   Bounded Waiting

3. „Als **Implementierungssprache** Ihres **multi-threaded Anwendersystems** wird **Java** vorgeschrieben. Geben Sie **drei Argumente** an, warum Sie zur Synchronisation Ihrer Anwenderthreads **Semaphore nicht** verwenden werden.“
   *“To **implement** your **multi-threaded application system** you have to use **Java**. Give **three arguments** why you would **not use semaphores** to synchronize your application threads.”*
   Java already offers the higher concept *monitor* which is easier to use
   Semaphores are only synchronization primitives ⇒
   more programming *effort* and danger of more programming *errors*

4. „Diskutieren und **erläutern** Sie den Begriff **Skalierbarkeit**! Geben Sie ein **Beispiel** an, wie die Skalierbarkeit einer Systemarchitektur erreicht werden kann!“
   *“Discuss and **explain** the term **scalability**. Give one **example** how to achieve scalability of a system-architecture.”*
   Scalability indicates the capability of a system to increase performance under an increased load when resources (typically hardware) are added, i.e. a system characteristic that allows good performance despite of increasing application load or increasing system components.
   SMP-Scheduling supporting many threads and many CPUs or
   Multithreaded server

Aufgabe/*Question* 4

1. „Gegeben sei ein System mit 4GB virtuellem Speicher, der nur 4 KB große Seiten unterstützt. Im System ist entweder das Seitentabellenmodell A oder B implementiert (s.u.). Eine kleine **Applikation** benötigt für Code und Daten Platz an den Adressen 8K, 12K und 16 K; ihre Stapelseite (*stack page*) beginnt bei 3G. Wieviel **Speicherplatz** wird für die Implementierung der **Seitentabelle(n) benötigt**, um den Adressraum **dieser Applikation** für beide unten angegebenen Seitentabellenmodelle zu repräsentieren. Jeder **Seitentabellen-eintrag** benötigt **4 Bytes**.“
   “Consider a System with 4 GB virtual memory space that only supports 4 KB pages. The system uses one of the two page table schemes A or B (s.b.). A small **application** only requires code and data at addresses 8 K, 12 K, and 16 K; and a stack page at address 3G. How much **memory** is **required** for implementing the **page table(s)** to **represent** the address space of **this application** for both page table models (see below)? Each **page table entry** requires **4 bytes**.”
   A: Einstufige Seitentabelle *Single-level page table*
   3 MB (4 MB)
   B: Zweistufige Seitentabelle, wobei die untere Seitentabelle 1024 Einträge enthält *A two-level page table where the lower page table contains 1024 entries*
   12 KB

2. **Warum** benutzt man in einem modernen leistungsfähigen System einen **TLB**? Aus welchen **typischen Bestandteilen** besteht ein TLB-Eintrag?“
   *“**Why** do you use a **TLB** in a modern efficient system? What **typical parts** does a TLB entry consist of?”*
   To accelerate the address translation from virtual to physical addresses
   Control bits, i.e. valid, reference, modified etc.
   Virtual page number, page frame number

3. „**Erläutern** Sie einen **Ansatz** aus der **Systemarchitektur**, mit dem man die **Trefferrate** des TLBs **erhöhen** kann!“
   *“**Explain** an **approach** from **system architecture** that **increases** the TLB **hit rate**.”*
   Superpages, or larger page sizes

4. „Obwohl Anwendung A eine bestimmte Seite S zuvor noch nicht referenziert hatte, muss beim ersten Zugriff von A auf S diese Seite **nicht** vom Hintergrundspeicher geladen werden. Geben Sie **verschiedene Gründe** dafür an, wie es dazu kommen kann!“
   *“Even though application A has never before referenced page S, this page S does **not** need to be loaded from background memory when A first references S. Give a couple of **different reasons** why this can happen.”*
   S is part of a shared library which has been used by another application.
   S is part of a shared memory region ~

S is part of a memory mapped file ~
S has already been loaded due to a prepaging algorithm
S is a copy-on-write page

5. „Diskutieren Sie **Vor-** und **Nachteile** einer **invertierten Seitentabelle**!“
*“Discuss **advantages** and **disadvantages** of an **inverted page table.**”*
IPT contains only m entries (m = number of page frames), much less than a normal page table
You have to take care when collisions of equally hashed virtual page numbers occur
You must implement a region-table containing all start addresses of unmapped regions

Aufgabe 5/ *Question 5*: **(3 + 1 + 2 +6 Punkte/*marks*)**

1. „Zählen Sie sechs typische **Dateiattribute** auf!”
*“Enumerate six typical file **attributes.**”*

read, write, execute
file size
file type
file name
date of creation, updating, …
file owner

2. „Erläutern Sie den Zweck des Systemaufrufs `lseek()`!“
“Explain the purpose of the system call `lseek()`.”

To place the file pointer to the desired position within the file

3. „Erläutern Sie Vor- und Nachteile von **Memory-Mapped Dateien** (*memory mapped files*)!“
*“Discuss the advantages and disadvantages of **memory mapped files.**”*
You can access a MM-file the same way as you access any data segment, i.e. you no longer need special read_file or write_file operations, that may lead to at least one additional copy between disk and main memory.
In smaller systems the address space might be too small to host large files.
`fork()` implies that the child inherits open files, i.e. however, an `execve()` overwrites the address space.
Complicated when using a MM-file concurrently in the classical manner.

4. „Welche **Größen** bestimmen die **maximale Zugriffszeit** bei der **B*-Dateiorganisation**, wenn der Zugriff über einen **Schlüssel** erfolgt?“
“What **entities** determine the **maximum access time** in a **B*-file organization** when the access is done via a **key**.”
Height of B*-tree, size of inner node, size of container (i.e. leave node)

5. „Diskutieren Sie Vor- und Nachteile einer **zusammenhängenden Belegung** von Platten-blöcken zur **Abspeicherung einer Datei**!“
“Discuss advantages and disadvantages of a **contiguous allocation of disk blocks** to **store** a **file.**”
Contiguous allocation of file blocks can speed up sequential accesses.
However, if a file grows more than previously specified, you must find a larger free area on the disk to store the bigger file, i.e. you need a more complicated disk space management