

6. „Eine Methode, mit der der Kern das Schedulinggeschehen auf **Anwenderebene** unterstützen kann, nennt man **Aufwärtsruf** (*upcall*).“
 “One method for a kernel to support **user level scheduling** is called *upcall*.”

korrekt

inkorrekt

7. „Auf einem symmetrischen Mehrprozessorsystem (SMP) kann **nur eine multithreaded Task aus reinen User-Level Threads (PULTs) gleichzeitig rechnen**.“
 “On a symmetrical multiprocessor system (SMP) **only one multithreaded task consisting of pure user level threads (PULTs) can run at the same time**.”

korrekt

inkorrekt

8. „In einem **geschichteten (layered) System** dürfen Programme der **Schicht S_{i+1}** nur Funktionen der **nächst tieferen Schicht S_i** aufrufen, aber **nicht Befehle der niedersten Schicht** (hardware) oder **Funktionen** anderer tieferer Schichten.“
 “In a **layered system**, programs of **layer S_{i+1}** may only call functions of the **next lower layer S_i** , but not instructions of the **lowest layer** (hardware) or **functions** of other lower layers.”

korrekt

inkorrekt

9. „In einer **korrekten** Lösung des Produzenten-Verbraucherproblems bestehend aus genau einem Produzenten und einem Verbraucher mit **jeweils vielen synchronen E/A-Aktivitäten**, **können nie beide Prozesse gleichzeitig blockiert** sein.“
 “In a **correct** solution of the producer-consumer problem with exactly one producer and one consumer with **both intensively using synchronous I/O activities**, **both processes can never be blocked at the same time**.”

korrekt

inkorrekt

Aufgabe 2 / Question 2

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

1. „Erläutern Sie das **Prinzip eines erweiterbaren Systemkerns** (*extensible system kernel*) und **diskutieren** Sie dessen **Vorteile** und dessen **wesentlichen Nachteil**!“
 “Explain the **principle** of an **extensible system kernel** and discuss its **advantages** and its **major disadvantage**.”

An extensible kernel can be started with only the absolutely required functionality. If an additional service has to be offered, the kernel can be extended with “kernel-modules” at run time (swapped out if no longer needed). An extensible kernel improves adaptability and of course extensibility. However, it also increases the danger of malicious code being imported into the kernel with the danger of crashing the whole system.

2. „Nennen Sie **ein konkretes Beispiel** eines erweiterbaren Systemkerns!“
 “Name **one real example** of an extensible system kernel.”

Linux

3. „Erläutern Sie die beiden Begriffe **Mehrprogrammssystem** und **Mehrbenutzersystem**!“
 “Explain both terms **multiprogramming system** and **multiuser system**.”

In a multi programming system, multiple applications can run at the same time.

In a multi user system, applications of multiple users can run at the same time.

4. „Zählen Sie **alle Vorteile** des **Kernel-Level Threadmodells** gegenüber dem PULT-Modell stichpunktartig auf!“
*“Enumerate **all advantages** of the **kernel-level thread model** compared with the **PULT model**.”*

No blocking of the complete task if one KLT does a blocking system call

Schedule multiple KLTs of the same task on a SMP

Consistent global scheduling is possible with a more efficient resource control

Efficient gang scheduling is possible for parallel numerical problems

No additional scheduling at user level reduces system complexity

5. „Einige Systeme versuchen den Kern zusätzlich dadurch zu schützen, dass sie einen extra **Kernstapel** (*kernel stack*) anstelle des Benutzerstapels (*user stack*) benutzen, **während** sie einen **Systemaufruf** (*system call*) durchführen. Erläutern Sie, **warum diese Vorgehensweise die Systemintegrität tatsächlich erhöht!**“
*“Some systems try to enhance kernel protection by introducing an **extra kernel stack** instead of using the application’s stack **while performing a system call**. Explain why this technique really improves the integrity of the system.”*

A malicious user could manipulate the (user = kernel) stack pointer just before the system call, thus tricking the kernel into overwriting its own data.

If the user stack is used, the system must be prepared to handle pagefaults during system calls (which can be tricky); using an extra kernel stack, the kernel can guarantee that no page faults occur and thus remove this requirement.

Furthermore, using an extra kernel stack removes a covert channel: During a system call the kernel pushes entities to the stack which would still be accessible to a malicious user-mode programmer if the user-stack was used.

Aufgabe 3 / Question 3

(2 + 3 + 1 + 2 + 4 Punkte/marks)

1. „**Unterstreichen** Sie diejenigen der unten angegebenen **Kombinationen** aus **Task-** und **Threadzustand**, die bei einer **mehrfädigen** (*multi-threaded*) **Applikation auftreten können!**“
*“Of the **combinations** of **task-** and **thread-state** mentioned below, underline those that **can occur** within a **multi-threaded application**.”*

Taskzustand=bereit /KLT-Zustand=rechnend (*task state=ready/KLT state=running*)

Taskzustand=blockiert/KLT-Zustand=bereit (*task state=waiting/KLT state=ready*)

Taskzustand=bereit/PULT-Zustand=rechnend (*task state=ready/PULT state=running*)

Taskzustand=blockiert/PULT-Zustand=bereit (*task state=waiting/PULT state=ready*)

2. „Zählen Sie alle aufwändigen Arbeiten auf, die - im Vergleich zum allgemeinen Fall - wegfallen können, wenn der **Kernablaufplaner** (*kernel scheduler*) von einem **KLT** auf einen anderen **KLT** der **gleichen Task umschaltet**.“
*“Enumerate all overhead that – compared to the general case - can be avoided if the **kernel scheduler switches from a KLT to another KLT of the same task.**”*

TLB-Flush and refill in case you do not have a tagged TLB
Switch the address space context, i.e. install the new page table environment

3. „Diskutieren Sie **den wesentlichen Nachteil** eines Kommunikationsobjekts **im Kern**, das an der Kernschnittstelle **asynchrones Senden** anbietet!“
*“Discuss **the major disadvantage** of a communication object **inside the kernel** that offers **asynchronous send** at the kernel interface.”*

You must provide ((un-)bounded) buffers in the kernel with the danger of a denial of service attack.

4. „Eine mehrfädige Applikation bestehe aus $n \gg 1$ KLTs und drei kritischen Regionen CR_1, CR_2, CR_3 . Jeder der n KLTs betritt in unterschiedlicher Reihenfolge und mehrfach die zugehörigen kritischen Abschnitte, aber kein kritischer Abschnitt wird von einem KLT betreten, während er sich noch in einem anderen kritischen Abschnitt befindet. Kann es bei korrekter Synchronisation in dieser Applikation **wegen der kritischen Abschnitte zu Verklemmungen** kommen? **Begründen** Sie Ihre Antwort!“
*“A multi-threaded application consists of $n \gg 1$ KLTs and three critical regions CR_1, CR_2, CR_3 . Each of these n KLTs enters the corresponding critical sections in varying order and multiple times. But no critical section is entered by a KLT as long as it is still in another critical section. Given a correct synchronization of the critical sections, is it still possible that the application runs into a **deadlock due to these critical sections**? **Explain** your answer.”*

No deadlock is possible because the necessary deadlock condition “hold-and-wait” is violated: No KLT enters another CS before leaving the previous one.

5. „Das Betriebssystem bietet an seiner Kernschnittstelle zur **Synchronisation** nur folgende **synchrone Kernfunktionen** an: **send()**, **receive()** und **send_and_receive()**. Skizzieren Sie, wie die Applikation aus Teilaufgabe 3.4 möglichst effizient mit diesen IPC Operationen implementiert werden kann!“
*“For **synchronization** the operating system offers only the following synchronous kernel-functions at its kernel-interface: **send()**, **receive()**, and **send_and_receive()**. Draft how the application of question 3.4 could be implemented as efficiently as possible with these IPC operations.”*

Install an additional thread T_i per critical region CR_i .
Each of the n KLT_k does a **send_and_receive(T_i, s) when it wants to enter critical section CS_s of the critical region CR_i .**
Each of the three threads T_i executes a loop with a switch: First it does a **receive(ANY), then it switches to the code s for the requested critical section CS_s , and finally it does a **send(KLT_k)**.**

Aufgabe 4 / Question 4**(1 + 2 + 3 + 1 + 3 + 2 Punkte/marks)**

1. „Unterstreichen Sie diejenigen der folgenden Aktivitätsformen, die prinzipiell als Einheit in einem Mehrprozessorsystem migriert werden können!“
 “Underline those of the following activity forms that can in principle be migrated as a single entity in a multi-processor system.”

PULT

KLTProzess

2. „Führen Sie mindestens **drei Begründungen** an, warum man eine **bestimmte Aktivität** trotz prinzipieller Migrierbarkeit nicht migrieren würde!“
 “Enumerate at least **three reasons** for not migrating a **certain activity** even though it could be migrated in principle.”

Migrating could induce too much overhead, so that it would not pay off even if the target CPU is underloaded.

Migrating a KLT of a task away from its cooperating KLTs to another CPU might not pay off if the number of cache misses increases significantly.

Sometimes gang scheduling requires to reserve a certain number of CPUs for the next “gang”, i.e. you better leave a CPU idle for a while.

3. „Zählen Sie **drei Maßnahmen** auf, mit denen man **auf einem Einprozessorsystem** die **Verweilzeit** (turn around time) einer **E/A-intensiven** (I/O intensive) **Applikation prinzipiell reduzieren kann!**“
 “Enumerate **three measures** that in principle **reduce the turnaround time** of an **I/O-intensive application on a uniprocessor system.**”

Offer asynchronous I/O whenever possible

Offer read-ahead in case of sequential files

Use the KLT-model if application can profit from parallelism

4. „Welche **Systemkomponente schreibt** die **physische Adresse** (physical address) in den zugehörigen **Seitentableneintrag** der Seite?“
 “Which **system component writes** the **physical address** into the corresponding **page table entry** of the page?”

Page fault handler

5. „Unter **welchen Umständen** werden „**TLB-miss-Handler**“ bzw. „**Pagefault-Handler**“ aufgerufen?“
 “Under **what circumstances** are **TLB-miss handlers** respectively **page-fault handlers** invoked?”

A TLB-miss handler is invoked whenever no valid TLB entry allows the requested memory access and the TLB is controlled by software.

A page fault handler is invoked whenever no valid page-table entry allows the requested memory access and the TLB is controlled by hardware.

6. „**Warum** ist es **vorteilhaft**, wenn ein über DMA betriebenes Gerät über einen **separaten E/A-Bus** benutzt wird?“
 “**Why** is it **preferable** when a **DMA driven device** is connected via a **dedicated I/O bus**?”

No cycle stealing from the CPU, both DMA and CPU can work in parallel

Aufgabe 5/Question 5

(1 + 2 + 3 + 3 + 3 Punkte)

1. „Welchen **gravierenden Nachteil** haben **klassische index-sequentielle Dateien**?“
 “Which major disadvantage do *classical index-sequential files* have?”

Bad access time whenever the overflow container is completely filled, causing the complete file to be reorganized on disk.

2. „Durch welche **effizientere Dateiorganisationsform** wurden diese index-sequentiellen Dateien **abgelöst** und warum tritt dieser ehemalige Nachteil nicht mehr auf?“
 “By which *more efficient file organization form* have these index-sequential files been *replaced* and why does the former disadvantage no longer occur?”

B*-sequential files only have local container overflow that is resolved locally; multiple levels of the B*-tree are affected only occasionally.

3. „Unterstreichen Sie jede der **folgenden Dateneinheiten**, die in der **inode** einer **ext2-Datei** gespeichert wird! Für **jede Dateneinheit**, die **nicht** in der inode gespeichert wird, geben Sie den **Ort an, an dem sie gespeichert wird**.“
 “Underline each of the following data entities that is stored in an *inode* of an *ext2* file. For each item that is *not* stored in the *inode* state where this information is stored.”

- Filename **directory**
- Name of containing directory **parent directory**
- File size
- File Type
- Number of symbolic links nowhere
- Number of hard links

4. „Obwohl der Unix/Linux Systemaufruf **open ()** nicht **notwendig** wäre, bringt er doch einige **Vorteile** mit sich. **Welche**?“
 “Although the Unix/Linux systemcall *open()* is not *necessary*, it has some *advantages*. Which ones?”

You must resolve the pathname only once
You must check its access rights only once
You do not have to add the filename for each read or write

5. „Erklären Sie, unter **welchen Umständen** eine **Bitmap** eine **vorteilhafte Datenstruktur** ist!“
 “Explain under *what circumstances* a *bitmap* is an *advantageous data structure*.”

In a memory manager, if the managed blocks are big enough (e.g., 1 KB) and if you need to quickly check whether a block is free or not, you better use a bitmap than a linked list etc. (you can use fast bit-operations on many CPUs).