

Solutions

Examination System Architecture

April 14 2004

Aufgabe/Question 1 (Zum Aufwärmen/Warm up, 3 + 3 + 1 + ... + 1 Punkte/marks)

1. „Zählen Sie die **drei prinzipiell unterscheidbaren Dateizugriffsmethoden auf**, und geben Sie für jede eine typische Anwendung an!“
*“Enumerate the **three principal file-access methods**, and give a **typical application for each of them**.”*

a) **Sequential, editing of text**

b) **Index-sequential, managing employees**

c) **Direct (hashed), online dictionary**

2. „Zählen Sie **drei** der vier **Betriebsmittelzuordnungsprotokolle** zur Verbesserung des **Prioritätsinversionsproblems** auf und nennen Sie jeweils deren Eigenschaft hinsichtlich möglicher **Verklemmungen!**“
*“Enumerate **three** of the **four resource-allocation protocols** improving the **priority inversion problem** and mention their characteristics concerning potential **deadlocks**.”*

Non preemption protocol, deadlock free

Priority inheritance, non deadlock free

Priority ceiling protocol, deadlock free

3. „Auf einem **Einprozessorsystem** mit **Mehrprogrammbetrieb** (*multi programming*) können **gleichzeitig mehrere „Reine User-Level-Threads“** (*PULTs*) im **User-Level-Thread-zustand „Rechnend“** (*running*) sein.“
*“On a **single-processor** with **multi programming** **multiple pure-user-level-threads** may be in the **user-level- thread state “running” at the same time**.”*

korrekt

inkorrekt

2

4. „Das **Buddysystem** mit **sofortiger Verschmelzung** bei der Freigabe von Speicherbereichen kann sowohl zu **internem** als auch zu **externem Verschnitt** (*fragmentation*) führen.“
“*The **buddy-system with immediate reunification** when releasing a memory portion may lead to **internal** as well as to **external fragmentation.***”

korrekt

inkorrekt

5. „Der Kindprozess in einem Linux-System erhält als Rückgabewert des Systemaufrufs **fork()** die Prozess-ID des Elterprozess zurück.“
“*The child process in a Linux system gets the process-ID of the parent process as the result of the system call **fork()**.*”

korrekt

inkorrekt

6. „Bei gleicher Anwendungslast (*application load*) ist die **Anzahl benötigter Adressräume** in einem **Mikrokern-basierten** System **nicht kleiner** als bei einem **Makrokern-basierten** System.“
“*With the same application load the **number of needed address-spaces** in a **micro-kernel based** system is **not smaller** than the one in a **macro-kernel based** system*”

korrekt

inkorrekt

7. „In einem Linux-System kann nur der Systemadministrator **Hardlinks** angelegen.“
“*In a Linux system only the system administrator may install a **hardlink.***”

korrekt

inkorrekt

8. „Je **kleiner** die **Kachelgröße** (*size of page frame*) bei einem seitenorientierten virtuellen Speicher gewählt wird, **desto kleiner der interne Verschnitt** (*fragmentation*)“.
“*The **smaller** the size of a **page frame**, the **smaller** the **internal fragmentation.***”

korrekt

inkorrekt

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

1. „Erläutern Sie das Systemarchitekturprinzip: „**Trennung von Strategie und Mechanismus**“ und geben Sie hierfür ein ganz **konkretes Beispiel** aus der Systemarchitektur an!“
*“Explain the system architecture principle: “**Separation of policy and mechanism**” and give a **clear example** within a system architecture.”*

Separation of policy and mechanism allows you to be orthogonal concerning changing one of them, i.e. different policies may be implemented upon one base mechanism. On the other hand a policy might be improved by a better underlying mechanism (to be developed in the future).

Example 1: GUI = policy, X-Windows = mechanism,

Example 2: Scheduling = policy (SJF, FCFS, ... and Dispatching = mechanism (context switch, timer slice)

2. „Erläutern Sie den Begriff: „**orthogonale Entwurfsparameter**“ und geben Sie hierfür ein **Beispiel** aus der Systemarchitektur an!“
*“Explain the notion: “**orthogonal design parameter**” and give a **clear example** within a system architecture.”*

Orthogonal design parameters are independent from each other, at least in principle, e.g. the following parameters for a memory management system are orthogonal:

Sequence of allocate/release operations

Size of memory portion

3. „Was sind die Vorteile einer **transparenten Systemunterstützung** für „Superseiten“, wie sie im gleichnamigen Artikel der Systemgruppe der Rice Universität beschrieben sind?“
*“What are the advantages of a **transparent operating system support** for **superpages** as described in the article with the same name of the system group at Rice University?”*

1. Increasing TLB Coverage

2. Decreasing TLB misses

3. Increasing application performance of larger working sets

4. Incremental promotion of super page size

5. Incremental denotion of super pages

4. „Ein Mehrprozessor-Scheduler kann entweder eine **zentrale Bereitwarteschlange** (ready queue) oder eine **per Prozessor** verwenden. Diskutieren sie die **Vor- und Nachteile** der zentralen Bereitwarteschlange!“
*“A multi-processor scheduler can choose to use either a **central or per-processor ready queue**. Discuss the **pros and cons** of a central ready queue.”*

Advantages:

Automatic load balancing of all processors ⇒ easier to establish a consistent scheduling policy

Disadvantages:

No scalability, central ready queue might become a bottleneck

Avoidable synchronization overhead at central ready queue

Dispatching without regarding processor affinity may lead to an increase in cache loading times

Accessing the central ready queue and modifying some entries may have additional side effects to cache lines of other running processes

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

1. „Gegeben sei ein mehrprogrammfähiges Mehrprozessorsystem mit $m \gg 1$ Prozessoren mit prozessorlokalen „L1-caches“. Sie sollen eine **Anwendung** entwerfen, die nach dem **Fließbandprinzip** (*pipelining*) funktioniert, wobei das **Zwischenergebnis** jedes Fließbandglied im Mittel größer als der L1-Cache sei. Welches der **beiden Threadmodelle** – „PULTs oder **Kernel-Level Threads**“ – würden Sie verwenden, wenn Sie die **Leistung** dieser Anwendung optimieren wollen? **Begründen** Sie Ihre Ansicht so ausführlich wie möglich!“
*“Assume a multi-programming multi-processor system with m processors ($m \gg 1$), and with processor local L1-caches. You have to design a **pipelined application**, whereby the average **intermediate result** of each pipeline stage is larger than the L1-cache. Which of the **two thread models** – “PULTs or kernel level threads” – would you use, if you have to optimize the **performance** of this application? Provide a clear and complete **reasoning** for your choice.”*

Kernel-level threads because otherwise you do not use the $m \gg 1$ processors.

No performance loss when all pipeline-entities are scheduled on different processors, because their intermediate results are larger than their L1-cache.

2. „Messungen haben ergeben, dass die **Fließbandanwendung aus 3.1.** im **zweiten Fließbandglied** einen **Engpass** hat. Wie würden Sie dieses Problem **möglichst elegant lösen**?“
*“Measurements have shown that the **pipelined application of 3.1** has a **bottleneck** within the **second pipeline stage**. How would you resolve this problem **as elegantly as possible**?”*

Replicate pipeline-entity 2 as often as needed.

3. „Spezifizieren Sie die **benötigten Kommunikationsobjekte** zwischen den **Fließbandgliedern von 3.2** vollständig hinsichtlich der möglichen **Entwurfsparameter!**“
“Specify the required communication objects between the pipeline stages of 3.2 for each of the potential design parameters.”

3.1 Between pipe 1 and pipes 2 (suppose n pipes of stage 2 are used)

1 : n

Asynchronous : Synchronous

Every receiver can consume the “message”

Every intermediate message contains original request number

3.2 Between pipes 2 and pipe 3

n:1/see above

4. „Nennen Sie die Gründe, warum **Spinlocks** auf **Anwenderebene** in **Einprozessorsystemen** **nicht** verwendet werden sollten!“
“Enumerate the reasons why spinlocks should not be used at application level in a single-processor system.”

Spin locks are no good idea, because its effectiveness relies on scheduling:

a) System may starve if a low-priority thread owns the lock whilst the high priority thread is spinning in front of this lock

b) Spinning for the rest of a time-slice might be too inefficient

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

1. „Zählen Sie die **vier Anforderungen** für eine **gültige Lösung** des **kritischen Abschnitt-problems** auf!“
*“Enumerate the **four requirements** for a **valid solution** of the **critical section problem.**”*

a) **exclusiveness**b) **portability**c) **bounded waiting**d) **progress**

2. „Zählen Sie je zwei konkrete Lösungen für das **kritische Abschnittproblem** aus den drei unten angegebenen Lösungsklassen auf! Diskutieren Sie jeweils deren **Vor- und Nachteile!**“
*“In each of the three solution classes below enumerate two specific solutions for the **critical section problem.** Discuss the **advantages and disadvantages** of each of them.”*

Reine Softwarelösung auf Anwenderebene/*Solutions solely based on application level software:*

Bakery algorithm**Peterson's solution****In both cases:****Pro: portable****Con: Inefficient active waiting**

Lösungen mit Programmiersprachen- bzw. Systemunterstützung/*Solutions based on programming language or operating system:*

Monitor**Semaphores****Pro: Simple, Con: -****Pro: Elementary, Con: Errorprone**

Lösungen mittels spezieller Prozessorbefehle/*Solutions based on special processor instructions:*

TestAndSet**CompareAndSwap****In both cases:****Pro: Fast****Con: Side effects on Bus traffic**

3. „Analysieren Sie gemäß Aufgabe 4.1 folgendes Programm für ein **kritisches Abschnittsproblem!**“

“Analyze the following program for a *critical section problem* according to question 4.1.”

```

/* program critical section */

binary flag[2] = {false, false};

void thread1()      {
...
  while (flag[1]) /* do nothing */;
  flag[0] = true; /* enter critical section */
...
  /* critical section */
...
  flag[0] = false;
...
}

void thread2()      {
...
  while (flag[0]) /* do nothing */;
  flag[1] = true; /* enter critical section */
...
  /* critical section */
...
  flag[1] = false;
...
}

main(){
  costart(thread1, thread2); /* start threads concurrently */
}

```

Analyse/analysis:

- a) **exclusiveness:** **No, both threads may enter the CS**
- b) **portability** **Yes**
- c) **bounded waiting** **No, it's possible that always the same thread is entering the section**
- d) **progress** **Yes**

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

1. „Globale Daten sollen in einem Unix-ähnlichen System mittels „**Copy-on-Write**“ zwischen zwei Prozessen gemeinsam benutzt werden. Müssen sich beide Prozesse **synchronisieren** beim gleichzeitigen Zugriff auf diese globale Daten?“

*“In a Unix-alike system global data will be used by two related processes using the “**copy-on-write**”. Do these two processes have to **synchronize** when accessing concurrently these global data?”*

Ja/yes:

Nein/no:

2. „Beschreiben Sie so ausführlich wie nötig, gleichwohl so knapp wie möglich, was das System während eines **fork()** an den **Seitentableneinträgen** der beiden Prozesse **modifizieren** muss, um ein „Copy-on-Write“ zu unterstützen!“

*“Describe as complete as necessary, but as short as possible, what the system has to **modify** during a **fork()** in the **page-table entries** of both processes to support the copy-on-write.”*

Copy-On-Write pages need a cow-Bit. or the surrounding region-data-structure has this cow-Bit.

During fork all data-structure describing the address-space are copied, e.g. region-descriptors and page-tables.

In all cow-pages the read-only bit will be set.

3. „Welche **Modifikationen** müssen vorgenommen werden, wenn einer der beiden Prozesse anschließend **modifizierend** auf eine dieser globalen Datenseiten zugreift?“

*“What **modifications** have to be done when one of the above two processes will **modify** one of these global data pages?”*

Make a copy of the cow-page

Change the frame-address in the corresponding page-table and change its access bits again to read/write

4. „Welche **Verantwortlichkeit** ergibt sich für die CPU beim Datentransfer von einem Gerät in den Hauptspeicher bei jedem der **drei folgenden E/A-Modelle**?“
 ”What is the **responsibility** of the CPU for device to memory **data transfer** in each of the following **three primary I/O models**?“

Programmierte E/A, *Programmed I/O*:

CPU has to do all I/O-management activities, thus even waiting actively until I/O has completed. CPU cannot be used elsewhere

Unterbrechungsgesteuerte E/A, *Interrupt-driven I/O*:

I/O command is issued in the CPU, then either the CPU switches to another thread in case of synchronous I/O or continues with the old thread in case of asynchronous I/O. The I/O invoking CPU later has to deal with the interrupt coming from the device.

DMA, *DMA*:

CPU only signals the DMA what to do in the very next future, then either it switches control to another thread or resumes the invoking thread.

5. „Welche **Hauptideen** stecken hinter dem **Entwurf** des **Reiser-Dateisystems**?“
 “What are the **main design ideas** behind the **Reiser file-system**?“

Speed and efficiency (especially for very small files)
Layout of files as a B+ tree

Reliability (via journaling of accesses)
Every file system-modifications is a transaction

Compatibility + extensibility (via sophisticated plug-ins)