Fragemorgen zur Vorlesung TIK II

Mittwoch, 22. Juli 2009.

Diese Notizen erheben weder Anspruch auf Vollständigkeit noch auf Korrektheit!

Allgemeines

In den Prüfungen nach 2005 ist jeweils definiert, in welcher Form die Antwort erwartet wird.

In den Prüfungen aus dem SPOD kann es Fehler drin haben.

Beachte Einheiten genau, z.B. ist 1KiB = 1024 Bytes. Siehe Folien.

Prozesse/IPC

Vorteile und Nachteile von Shared Memory und Message Passing?

Beim Message-Passing haben die Nachrichten eine definierte Grösse, z.B. 1.5kB. Um mit Message Passing grössere Datenmengen zu übertragen braucht man viele Nachrichten, was zu grossem Overhead führt. Beim Shared Memory kann die Grösse des gemeinsamen Speichers mehr oder weniger frei bestimt werden, es könne also grosse Blöcke auf einmal übertragen werden.

Message Passing hat dafür in gewissem Sinne Synchronisation bereits „eingebaut", d.h. metaphorisch: für den Briefkasten braucht es einen Schlüssel. Beim Shared Memory muss man sich selbst um die Synchronisation kümmern.

Prüfung Frühling 2005, Aufgabe 2

Realisierung des Monitoren mit Sempahoren:

```
Semaphore v;
Initialisierung: signal(v);

b()  {
     wait(v);
     body;
     signal(v);
}
```

Monitor in Java mit „synchronized", d.h. nur ein Thread kann eintreten.

```
class MyClass
{
     int myRingBuffer;
     final int objectID;
     synchronized void producerFunction( int b ) {...}
     int getObjectID() {...}
     synchronized int consumerFunction() {...}
     ...
}
```

Monitor auf Objektebene: Ist ein Thread in einer synchoronized Methode des Objekts drin, so gelten **alle** synchronized Methoden des Objekts als belegt.

Kann es bei der Verwendung von einem/mehreren Monitoren zu Deadlocks kommen?

JA, auch bei einem einzelnen, denn Deadlocks entstehen durch Programmierfehler, und Programmierfehler können immer entstehen! Beispielsweise könnte man die gemonitorte Funktion aus dem eigenen Body wieder selbst aufrufen, was zu einem Deadlock führt.

Übung 2, Aufgabe 2d

```
int 2Darray [1024][4];
int i,j = 0;
for (i=0; i<1024; i++) {
      for (j=0; j<4; j++)
            2Darray[i][j] = i+j;
}
```

Ausgeführt auf OS mit Demand Paging und Seitengrösse 4KiB. Seitenersetzungsstrategie sei LRU. Dem Prozess werden für Code und Daten (ausgenommen page table) 3 Frames statisch zugewiesen. Die Seitentabelle befindet isch vollständig im Hauptspeicher (memroy resident). Wieviele page faults wird die Ausführung dieses Prozesses verursachen, falls (a) das 2D-array in row-major Reihenfolge gespeichert und (b) in column-major Riehenfolge gespeichert wrid?

(a) Das Array hat 1024 Zeilen à 4 Spalten. Da pro Spalte ein Integer à 32 Bit (4 Bytes) gespeichert wird, hat jede Zeile die Länge 4*4 Bytes = 16 Bytes. In einem Frame à 4096 Bytes bringt man folglich 256 Zeilen unter. Daraus folgt, dass bei row-major nur 1024/256 = 4 Pagefaults entstehen, nämlich jeweils dann, wenn die nächste Zeile nicht mehr in den aktuellen Frame passt.

(b) Bei column-major befindet sich jeweils eine Spalte à 1024 Integers (à 4 Bytes) in einem Frame. Daher greift jede Iteration der inneren Schleife auf Elemente in unterschiedlichen Frames zu und erzeugt daher wegen der LRU-Seitenersatzstrategie 4096 Pagefaults.

Prüfung HS2005, Aufgabe 3.1

In dieser Aufabe wird betrachtet, unter welchen Umständen Zugriff auf eine Variable b, die als Zähler genutzt wird, mittels Synchronisationsmechanismus geschützt werden muss. Rahmenbedingungen:

- *Prozesse dürfen nur die Werte 0 oder 1000 schreiben oder b um genau 1 erhöhen.*
- *Werden auf die Variable zwei oder mehrere der Operationen aus parallel ausgeführt, muss das Ergebnise einer der möglichen sequentiellen Ausführungen der Operationen entsprechen.*
- *Die Variable b wird durch das System immer atomar gelesen oder geschrieben.*

In welchen der Fälle (a) mehrere Prozesse lesen nur von b, einer schreibt nur in b, (b) ein Prozesse liest von b, mehrere schreiben in b und (c) zwei Prozesse schreiben und lesen b, muss b durch Synchronisation geschützt werden?

(a) Keine Synchronisation nötig, da nur einer schreibt und daher nichts verloren gehen kann.

(b), (c) Hier muss synchronisiert werden, weil mehrere Prozesse schreiben und es zu Problemen wie dem Lost Update Problem (zwei Prozesse inkrementieren die Variable, aber nur eine Änderung wird tatsächlich durchgeführt, also b+1 statt b+2, vgl. VL) kommen kann.

Prüfung Herbst 2005, Aufgabe 3.2

Initalisierung von S1 ist wichtig, sonst startet keiner der Threads. Und aus der Aufgabenstellung ist ersichtlich, das Thread 1 als erstes Startet, daher initialisieren S1=1.

Prüfung Herbst 2005, Aufgabe 3.3

Aufgabenstellung siehe in der Prüfung.

```
void inc_prio(int * wait_prio ) {
     if ( *wait_prio < 1000000 ) {
          *wait_prio = *wait_prio + 1;
     }
}
```

Theoretisch würde es ab dem Moment nicht mehr funktionieren, wenn mehr als eine Million Prozesse in der Queue sind, weil dann z.B. zwei mit der maximalen Priorität exisiteren.

Lösungsvorschlage: Man könnte in diesem Fall den Eintritt in die Warteschlange verhindern, z.B. mit FIFO. Oder: Alle Prozesse, die Priorität 1000000 erreichen, werden in eine FIFO gestellt.

(In der Praxis ergibt sich zusätzlich das Problem, dass der Integer das Maximum seines Wertebereichs erreichen kann und dann die Prioritäten nicht mehr unterschieden werden können. *Achtung:* Hier ist Wertebereich für int vorgegeben, 0 bis 1000000, daher existiert dieses Problem nicht.)

Frühling 2004, Aufgabe 2

Wenn steht, dass eine Aufgabe mit Semaphoren gelöst werden soll, kann man einfach wait(s) und signal(s) verwenden, und muss diese nicht noch implementieren.

Wie **könnte** man sie implementieren? Wenn die CPU eine Instruktion „getAndSet“ unterstützt, geht es damit. Andere Variante: Wenn man die Semaphorvariable verändert, schaltet man Interrupts und/oder den Scheduler ab (das geht als OS-Prozess).

Generelle Fragen zu Seitentabellen

Was ist ein Pageframe?

Kommt wohl von der DE-EN Übersetzung von Seitenrahmen: **Ignorieren!**

In der VL verwendete Begriffe: Eine **Page** ist ein Memoryblock im virtuellen Adressraum und ein **Frame** ein Memoryblock im physikalischen Adressraum.

Dabei gilt size(Page) = size(Frame).

Typen von Seitentabellen:

Flache (flat), Hierarchische (n-stufige), Invertierte, und auf Hashtabellen basierende.

Umsetzung von Virtuellem Speicher, der Segmentation unterstützt:

Segmente: Memory-Blöcke mit variabler Länge; virtueller Speicher braucht Adress-Übersetzung! Mögliche Übersetzungsvariante auf Slide 83.

Wegen der Variablen Länge muss in der Segmenttabelle die Startadresse und die Länge bzw. das Ende des Segments stehen.

Aufteilung Adresse/Adressraum

Speichersystem verwaltet physikalischen Adressraum von 64KB. Jedem Prozess wird ein logischer Adressraum von 256 Pages zu je 512 Bytes zur Verfügung gestellt. Gesucht (a) Anzahl Bits der logischen Adresse und (b) Anzahl Bits der physikalischen Adresse.

Problem: Reichen diese Angaben?

Ja, denn: **Grundsätzlich:** Wenn beim Speicher nicht angegeben ist, ob Wort- oder Byteadressierung, sollte man von Byteadressierung ausgehen. Sichergehen in der Prüfung: Hinschreiben, was man verwendet.

Hier ist Wort-Adressierung nicht sinnvoll, weil die Wortbreite nicht bekannt ist. Wenn gegeben wäre, dass es sich z.B. um eine 32Bit-Architektur handelt, wäre die Wortbreite 4 Bytes.

Gegeben sei ein Speicherverwaltungssystem mit zwei Ebenen von je gleich grossen Seitentabellen. Die Grösse des virtuellen Adressraumes sei 64kB. Berechnen Sie für eine Seitenlänge von 1kB die Anzahl Einträge pro Seitentabelle. Gesucht: Anzahl Einträge M pro Seitentabelle.

Wodurch wird M definiert? Alle Pages müssen mit den gegebenen Strukturdaten abgebildet werden können.

Was sind „alle“ Pages? Adressraumgrösse geteilt durch Pagegrösse ergibt 64 Pages.

Formel für Abbildung von N=64 pages mit einer 2-stufigen Pagetable:

$$\prod_{i=1}^{2} \textit{Anzahl Einträge einer PT der Stufe } i = \textit{Anz.E}(PT_1) * \textit{Anz.E}(PT_2) = 64$$

Da die beiden Seitetabellen gleich gross sind, folgt: #E(PT)^2 = 64 und daraus #E = 8. Die Seitentabellen müssen also jeweils 8 Einträge haben.

Prüfung Herbst 2004, Aufgabe 1c

In der Aufgabenstellung steht, dass alle Reference Bits periodisch gelöscht werden. Das bedeutet für die Zugriffssequenz [1 3 6 2 5 8]: Bei Zugriff auf Pages 6, 8, ... werden **alle** Referenzbits gelöscht. Nach dem Zugriff auf beispielsweise 6 ist **nur** das Referenzbit für 6 auf 1, alle anderen auf 0.

Prüfung Herbst 2004, Aufgabe 1b

Sowohl der logische als auch der physikalische Adressraum eines Rechners sei 2^{32} Bytes (4GB) gross. Für das Speichermanagement soll hierarchisches Paging verwendet werden, wobei ein Frame 256 Bytes umfasst. Ein Eintrag in einer Inner Pagetable besteht aus dem Frame-Index und 8 zusätzlichen Bits für diverse Flags. Ein Eintrag in einer Outer Page-Table besteht aus einem 32-Bit-Zeiger auf eine untergeordnete Page-Table. Wieviele Hierarchieebnenen sind notwendig, wenn für jede Seitentabelle genau ein Frame zur Verfügung steht?

Gegeben: Grösse des Adressraumes, der abzubilden ist. Kennzahlen zum Abbildeverfahren: N-Level Paging, Grösse der Einträge für Levels 1 bis N-1: 4 Bytes, Grösse für Einträge des Levels N: Frame Index + 1Byte.

Gesucht: N

Wodurch wird N definiert? Ganzer Adressraum muss abgedeckt werden.

Formel für abgedeckten Speicher Aufschreiben:

Memory <= #Inner-PT Einträge * Framegrösse

2^32 Bits <= #Inner-PT Einträge * 256 Bytes (Werte eingesetzt.)

Finden der Anzahl Inner-PT Einträge:

$$E = \prod_{i=1}^{N} \textit{Anzahl Einträge einer PT der Stufe } i$$

$$E = \left(\prod_{i=1}^{N-1} \frac{256}{4} \right) \cdot \frac{256}{2} = 64^{N-1} \cdot 128 \quad \text{(Werte eingesetzt.)}$$

Gleichung „lösen“ (ausprobieren): 2^32/256/128 <= 64^(N-1), das führt auf N=2 zu klein, also **N=3.**

Input/Output

Disk-Scheduling

(Übung 3, Aufgabe 5) Verschiedene Disk Scheduling Algorithmen: FCFS, SSTF, SCAN, SCAN-C, C-LOOK. Angenommen, die mittlere Länge der Auftragswarteschlange sei klein, d.h. sie enthält 0 oder 1 Element. Wie verhalten sich die Verfahren FCFS, SCAN und SSTF in diesem Fall?

FCFS ungefähr gleich schnell wie SSTF.

Bei welchem Scheduling kann „Starvation“ auftreten?

Nur FCFS schützt vor Verhungern. Beispiel für die Entstehung von Starvation bei den anderen Algorithmen: Ständig neue Anfragen für die Spur X. Oft haben die Disks einen Schutz gegen diese Probleme, Random Seed).

RAID: Abschätzen der Geschwindigkeit bei RAID? Auch irgend eine Aufgabe in den Übungen.

Read: Parallel Read ist immer schneller, da jede Platte nur ein n-tel der Daten lesen muss.

Write: Parallel Write ist immer schneller, siehe oben.

ABER: Falls N+X Daten geschrieben werden (Redundanz!), blockiert dies Resourcen und daher sinkt die Gesamtleistung des Systems.

Dateisysteme

Verzeichnisse

Was ist ein Verzeichnis?

Verzeichnisse sind auch Dateien, welche bestimmte Struktur haben. Es gelten daher die selben Beschränkgungen (Anzahl Blocks, etc.) wie für Files. Siehe VL.

Übung 3

Vorgehen um eine Datei zu löschen?

Eintrag im Directory löschen, Datenblöcke in die Liste der freien Blöcke aufnehmen, Inode löschen.

Was passiert bei einem Absturz?

Journaling Dateisystem oder Suche nach Blöcken/Inodes, welche nicht mehr referenziert werden (Garbage Collector). D.h. zuerst die höheren Datenstrukturen (Inodes) löschen, um dies

zu ermöglichen!

Hard-/Softlinks

Softlinks: Verweist auf den **Pfad** der Datei oder des Verzeichnisses und ist daher partitionsübergreifend möglich.

Hardlinks: Direkter Verweis auf die Datei, d.h. auf eine eindeutige Referenz (Inode!), daher nur innerhalb der gleichen Partition möglich. Hard Links auf Verzeichnisse sind in der Regel verboten (man kann, wenn es das OS nicht verhindert, aber man sollte nicht).

Netzwerke

Leitungskodierungen

NRZ-L: Nullniveau ist in der Mitte, und wird nie angenommen.

NRZI: Immer wenn eine eins auftritt, wird ein Übergang gemacht.

Bipolar – AMI: Bei 0 auf Niveau 0 und bei 1 abwechselnd zwischen positiv und negativ.

Manchester: Taktrückgewinnung möglich.

Warum diese komplexen Leitungskodierungen?

Man möchte (i) den Gleichspannungsanteil unterdrücken und/oder (ii) Taktrückgewinnung ermöglichen.

CRQ-SR: Folgenummern

	Sendepuffer	*Empfangspuffer*	*Folgenummern*	*Timer*
IRQ	1	1	2	Ja
Selective CRQ	N	n-1	2*n	Ja
Go-Back-N CRQ	N	1	N+1	Ja

(Skript aus Internt: Dann sind alle Tabellen etc. Ausgefüllt!)

Warum 2n Folgenummern bei CRQ mit Selective Retransmission?

Prüfungsstoff

Kapitel 6.8, 6.9 und 7.0 sind **KEIN** Prüfungsstoff; alle anderen, speziell auch „Bit- bzw. Charstuffing“, gehören zum Prüfungsstoff.

Diese Notizen erheben weder Anspruch auf Vollständigkeit noch auf Korrektheit!