FB Informatik und Mathematik
Prof. R. Brause

Klausur + Musterlösung

Betriebssysteme SS 2011

14.7.2011

Vorname:	
Nachname:	
Matrikelnummer:	
Geburtsdatum:	
Studiengang:	

Bitte tragen Sie auf jeder Seite Ihre Matrikelnummer ein und überprüfen Sie diese Klausur auf Vollständigkeit (18 Seiten!). Tragen Sie auf dem Deckblatt Ihre Daten in **Druckbuchstaben** ein.

Verwenden Sie ausschließlich die beigefügten Blätter. Die Rückseite können Sie als Schmierpapier für Ihre Notizen verwenden. Sollten diese nicht ausreichen, so wenden Sie sich bitte an die Aufsicht. Hilfsmittel (z. B. Handys, Bücher, eigenes Papier, etc...) sind verboten. Die Benutzung gilt als Täuschungsversuch und führt zum Ausschluss von der Klausur.

Die Klausur enthält 10 Aufgaben mit insgesamt maximal 144 Punkten. Um 100 % der Klausur zu bestehen, müssen davon Aufgaben mit 120 Punkten erreicht werden. Aufgabe 11 mit 21 Punkten ist eine optionale Bonusaufgabe.

Die Anzahl der Punkte entspricht ungefähr der Bearbeitungszeit in Minuten. Die maximale Bearbeitungszeit für die gesamte Klausur beträgt 180 Minuten.

Viel Erfolg !!!

1 5	**2** 5	**3** 10	**4** 20	**5** 20	**6** 15	**7** 18	**8** 10	**9** 16	**10** 25	***11*** *21*	**Σ** 165	**Übungs-punkte**

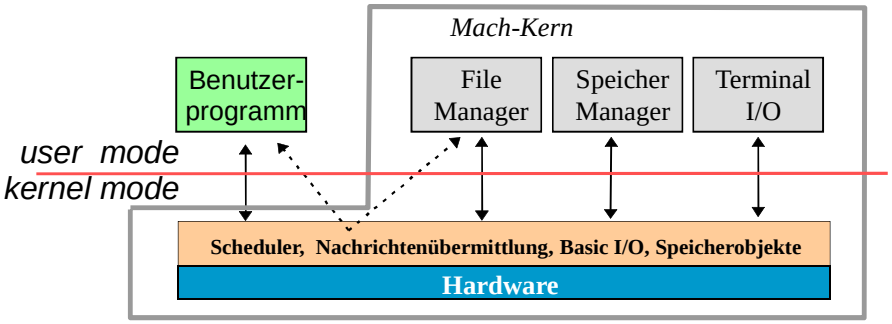
Aufgabe 1: Schichtenmodell 5 Punkte

(a) Zeichen Sie das Schichtenmodell des Mach-Kerns und kennzeichnen Sie die Schnittstellen. (2 Punkte)

(b) Benennen Sie die Vor- und Nachteile des Aufbaus des Mach-Betriebssystems. (2 Punkte)

(c) Welchen Vorteil bietet es, möglichst viele Funktionalitäten in den Kernel-Mode zu verlagern? (1 Punkt)

(a)



(b) ***Vorteile**: minimaler Speicherbedarf für Kern, alle Funktionen modularisiert austauschbar im Betrieb.*
***Nachteile**: Kommunikationsdauer zwischen Managern lang, da user/kernel mode –Schranke überwunden werden muss. Das Nachladen von benötigten Kernfunktionen braucht Zeit.*

(c) Die Verlagerung nutzt die schnellere Kommunikation und leichteren Funktionsaufruf innerhalb des Kerns. Außerdem müssen keine Funktionen nachgeladen werden.

Aufgabe 2: Software-Hardware-Migration 5 Punkte

(a) In Großrechnern gibt es oft keine I/O-Driver, sondern stattdessen Hardwareeinheiten mit gleichen Schnittstellen. Welche Vor – und Nachteile hat eine solche Software-Hardware-Migration? (2 Punkte)

(b) Geben Sie ein weiteres Beispiel für eine Software-Hardware-Migration an. (2 Punkte)

(c) Wenn Sie die Wahl zwischen einer Software- und einer Hardwarelösung hätten, welcher Lösung würden Sie den Vorzug geben? Begründen Sie ihre Antwort. (1 Punkt)

(a) Vorteile: Geräte funktionieren schneller durch HW-Treiber.

Nachteile: Treiber schwerer änderbar und wartbar.

(b) Beispiel : Zusammenfassung der TCP/IP-Transportschicht in Chips und Boards.

(c) Bei starken Schwankungen der Funktionalität ist eine Softwarelösung besser. Wenn die Software ausgereift und erprobt ist und über längere Zeit stabil bleibt, ist aber eine Hardwarelösung vorzuziehen.

Aufgabe 3: Definitionen — 10 Punkte

(a) Was ist der Unterschied zwischen einer abstrakten und einer virtuellen Maschine? (2 Punkte)

(b) Was sind Prozesse im Unterschied zu *lightweight*, *heavyweight* sowie *User-* und *Kernel*-Threads? (4 Punkte)

(c) Was versteht man unter *Apartments* in Windows NT? Wozu werden sie verwendet? (4 Punkte)

(a) Eine abstrakte Maschine definiert sich über eine Schnittstelle, die eine Funktionalität anbietet wie eine echte Maschine. Eine virtuelle Maschine verfügt ebenso über eine solche Export-Schnittstelle, benötigt aber zusätzlich noch eine Import-Schnittstelle, über die sie die Leistungen in Anspruch nimmt, um ihren Export zu gewährleisten. Sie ist damit nur eine Zwischenschicht zwischen den Anforderungen und dazu benötigten low-level Dienstleistungen.

(b) Jeder ***Prozess*** *habt einen eigenen Kontext, etwa den Hauptspeicherbereich, in dem er arbeitet, die Dateibeschreibungen offener Dateien, die Kommunikationskanäle zu anderen Prozessen usw. Ein Prozess kann viele* ***Threads*** *haben. Sie sind zwar eigenständig, arbeiten aber alle im selben Kontext des einen Prozesses, dem sie angehören und können alle auf diesen Kontext zugreifen. Wenn ein* **lightweight**-*Thread durch einen I/O blockiert wird und warten muss, werden auch alle anderen* lightweight-*Threads des Prozesses blockiert, auch wenn sie selbst keinen I/O benötigen. Die* **heavyweight**-*Threads dagegen werden durch I/O anderer Threads nicht blockiert.*

Ein Prozess, und damit auch seine Threads, kann unterschiedliche Rechte besitzen: im Kern-Modus ist ein privilegierter Zugriff auf alle Kernfunktionen möglich; im User-Modus ist der Zugriff wesentlich restriktiver und es werden vorher alle Rechte geprüft.

(c) Die Menge aller Threads und Objekte eines Prozesses lassen sich in Untermengen unterteilen. Jede Untermenge wird als Apartment bezeichnet und regelt die Zugriffsrechte und –mechanismen von Threads auf Datenobjekte. Innerhalb eine Apartments ist der Zugriff ohne Voranmeldung möglich; möchte ein Thread auf ein Objekt eines anderen Apartments zugreifen, muss der Zugriff mit dem Zugriff anderer Threads synchronisiert werden. Dieser Zwangssynchronisierung wird als data marshalling *(Listenbildung) bezeichnet und verhindert* race conditions.

Aufgabe 4: Multiprozessorscheduling 20 Punkte

Angenommen, Sie haben ein 3-Prozessorsystem und versuchen, eine Menge von Threads eines Prozesses besonders effizient zu synchronisieren. Gegeben seien dazu folgende Präzedenzrelationen sowie in Klammern die Joblängen (in Zeiteinheiten):

B>>D, C>>F, B>>E, E>>F, D>>G, G>>H, F>>H, A>>C, D>>F

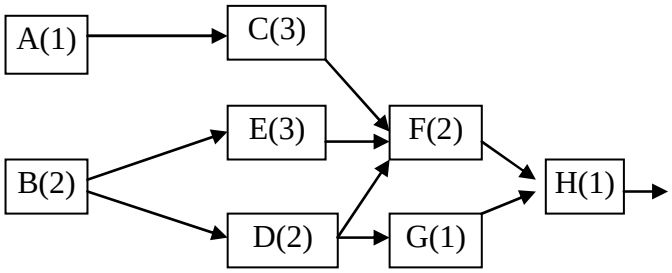
A(1), B(2), C(3), D(2), E(3), F(2), G(1), H(1)

(a) Erstellen Sie den zu der Präzedenzrelation zugehörigen *Präzedenzgraphen* (8 Punkte)

(b) Geben Sie den *kritischen Pfad* in dem Graphen an. (2 Punkte)

(c) Erstellen Sie jeweils einen Schedul für *latest scheduling* sowie für *earliest scheduling* mittels eines Gantt-Diagramms. (8 Punkte)

(d) Was ist dabei der Unterschied zwischen *Scheduling* und *Dispatching*? (2 Punkte)



(a)

(b) Der Kritische Pfad ist B-E-F-H der Länge 8

(c)

earliest scheduling:

P1	*A*	*C*				
P2	*B*		*D*	*G*		
P3			*E*		*F*	*H*

latest scheduling:

P1		*A*	*C*			
P2	*B*		*D*		*G*	
P3			*E*		*F*	*H*

(d) Beim Scheduling wird der Ablauf aller Prozesse geplant, beim Dispatching werden sie jeweils zum Ablauf gebracht.

Aufgabe 5: Prozess-Synchronisation 20 Punkte

Sei ein paralleles Programm zur Implementierung eines Produzenten/Konsumenten-Verhältnisses durch den folgenden Pseudocode gegeben. Produzent und Konsument kommunizieren über eine gemeinsame Puffervariable `p`: `append(p,v)` fügt ein Produkt `v` in den Puffer ein, `take(p)` entnimmt ein Produkt aus dem Puffer.

Variable `n` gibt jeweils die Anzahl Produkte im Puffer an. Variable `s` und `d` sind binäre Semaphore. Alle Variablen werden sowohl im Produzenten- als auch im Konsumenten-Thread benutzt, lediglich `lokal` ist eine Variable, die nur lokal im Konsumenten auftritt.

Das `parbegin` Statement in der Hauptfunktion `main()` lässt einen Produzenten und einen Konsumenten-Thread parallel ablaufen.

```
int n = 0 ;
Puffer p ;
binary_semaphore s = 1 ;
binary_semaphore d = 0 ;

void producer(){
    while (true) {
        product v = produce();
        wait(s);
        append(p,v);
        n = n + 1 ;
        if (n == 1)
            signal(d);
        signal(s);
    }
}
```

```
void consumer(){
    int lokal ;
    wait(d) ;
    while (true) {
        product v ;
        wait(s);
        v = take(p);
        n = n - 1 ;
        lokal = n ;
        signal(s) ;
        consume(v) ;
        if (lokal == 0)
            wait(d);
    }
}

void main(){
  n = 0; initialisiere(p) ;
  parbegin(producer,consumer);
}
```

(a) Die Implementierung nutzt binäre Semaphore. Erklären Sie zunächst deren Funktionsweise: Welchen Effekt hat ein Aufruf von `wait(s)` bei s = 0 und s = 1, welchen Effekt hat ein Aufruf von `signal(s)` bei s = 0 und bei s = 1? (4 Punkte)

(b) Welche Variable müssen im wechselseitigen Ausschluss behandelt werden? Begründen Sie ihre Antwort. (2 Punkte)

(c) Markieren Sie die kritischen Abschnitte in dem angegebenen Pseudocode. (2 Punkte)

(d) Sichert die Implementierung den wechselseitigen Ausschluss zu? Begründen Sie Ihre Aussage. (6 Punkte)

(e) Ist eine Flußkontrolle vorhanden? Begründen Sie Ihre Aussage. (6 Punkte)

(a) Ein Aufruf von wait(s) entspricht einem Aufruf von V(s): Wenn s=1 ist wird s auf null dekrementiert und die Methode wieder verlassen. Ist bereits s=0, so wartet der Prozess solange, bis von außen s=1 gesetzt wird, dekrementiert s und verlässt dann erst wait(s). Ein Aufruf von signal(s) entspricht der P(s) – Methode: Wenn s=0 ist so wird s=1 gesetzt und danach die Methode verlassen. Ist s=1, so ändert sich nichts und die Methode wird ebenfalls verlassen.

(b) Es müssen alle globalen Datenbereiche mit wechselseitigem Ausschluss geschützt werden. Dazu gehören sowohl der Puffer p als auch der Zähler n.

(c) siehe Zeichnung: gelb markierter Bereich

(d) Die Implementierung sichert den wechselseitigen Ausschluss zu, da der kritische Bereich, in dem die globalen Daten verändert werden, mit den Semaphor-Operationen wait() und signal() abgesichert sind.

(e) Es ist nur eine einseitige Flusskontrolle vom producer zum consumer über die wait(d)-Anweisung vorhanden für den Fall, dass der Konsument zu schnell ist. Umgekehrt wartet aber der Produzent nicht, wenn der Konsument zu langsam ist.

Aufgabe 6: Verklemmungen 15 Punkte

Bei parallelen und konkurrenten Aktionen kann es zu Verklemmungen kommen.

(a) Nennen Sie die in der Vorlesung genannten Bedingungen, die notwendig sind, damit es zu Verklemmungen kommen kann. (4 Punkte)

1. *Beschränkte Belegung (mutual exclusion)*
2. *Zusätzliche Belegung (hold-and-wait)*
3. *Keine vorzeitige Rückgabe (no preemption)*
4. *Gegenseitiges Warten (circular wait)*

(b) Ein Szenario aus 4 Prozessen, P1, P2, P3 und P4, und 3 Arten von Ressourcen, R1, R2, R3 sei gegeben. **Sei R1 neunfach, R2 dreifach und R3 sechsfach vorhanden.** Kann mit der nachfolgend genannten Belegung eine Verklemmung auftreten? Begründen Sie Ihre Entscheidung. (7 Punkte)

Matrix der maximalen Anforderungen

	R1	**R2**	**R3**
P1	3	1	4
P2	4	2	2
P3	3	2	2
P4	6	1	3

Matrix mit aktueller Ressourcenbelegung

	R1	**R2**	**R3**
P1	2	1	1
P2	0	0	2
P3	1	0	0
P4	6	1	2

*(b) Es sind insgesamt Ressourcen **R** vorhanden, geschrieben als Triple **R**=(9,3,6). Aktuell sind nach der rechten Matrix (9,2,5) Ressourcen belegt, also (0,1,1) frei. Da zuerst P4 mit der Zusatzbelegung (0,0,1) laufen kann, stehen danach (0,1,1) + (6,1,2)=(6,2,3) Ressourcen frei. Somit kann auch beispielsweise P1 mit zusätzlich (1,0,3) laufen, was zu (6,2,3)+(2,1,1)=(8,3,4) führt. Es folgen P2 und P3, so dass am Ende alle Ressourcen frei sind und alle Prozesse zu Ende gehen konnten. Damit ist eine Verklemmung nicht möglich.*

(c) Wie würden Sie nach dem Banker-Algorithmus entscheiden, ob in obiger Situation die Anforderung des Prozesses P3 nach zwei weiteren Ressourcen R2 erfüllt werden darf? Skizzieren Sie knapp das prinzipielle Vorgehen. (4 Punkte)

(c) Wenn wir die Anforderungen von P3 auf (3,4,2) heraufsetzen, so müssen wir testen, ob dann immer noch alle Prozesse in irgend einer Reihenfolge bedient werden können. Leider kann es eine solche Reihenfolge aber nicht geben, da nach Ablauf aller anderen Prozesse nur (8,3,6) frei ist, aber von P3 die Ressourcen (3,4,2) verlangt werden. Also muss das System immer verklemmen, wenn wir den Anforderungen nachgeben würden. Die ´zusätzlichen Anforderungen von P3 sind also abzulehnen.

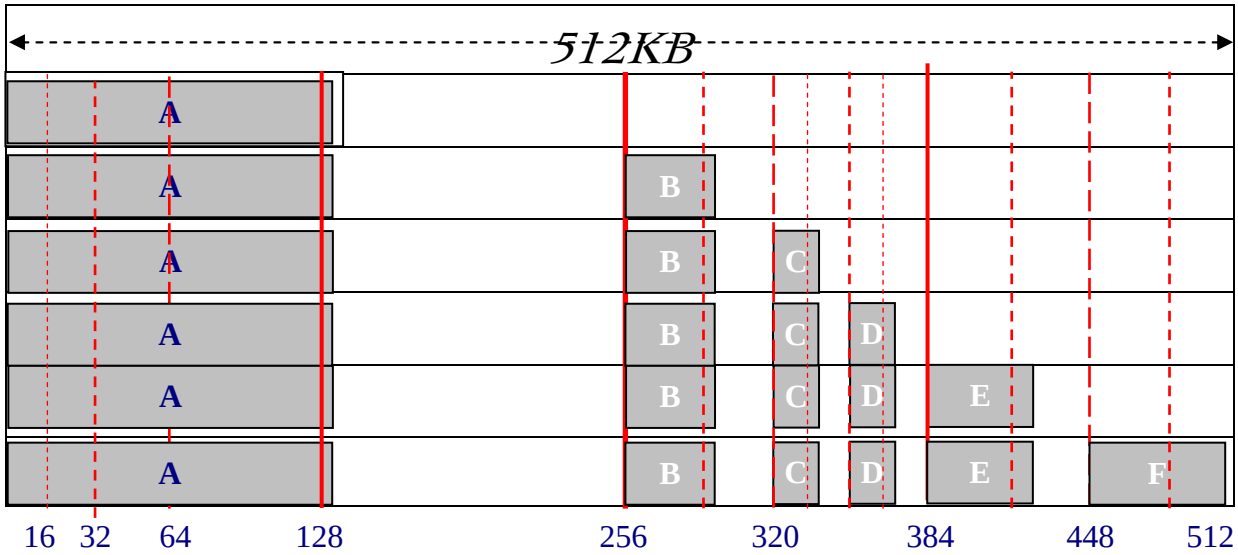
Aufgabe 7: Speicherverwaltung 18 Punkte

Die Verwaltung von Heap-Speicher oder anderem, selbst verwalteten Speicher kann mit einfachen Strategien bewältigt werden.

(a) Folgende freie Speicherstücke seien in der aufgeführten Reihenfolge und genannten Größe gegeben: 8KB, 16KB, 17KB, 9KB, 12KB, 19KB, 18KB und 15KB. Welcher freie Bereich wird jeweils gewählt für eine Anforderung von zunächst (A) 9KB, dann (B) 12KB, anschließend (C) 9KB und zum Schluss (D) 8KB? Beantworten Sie die Frage jeweils für die Strategien *FirstFit*, *NextFit*, *BestFit* und *WorstFit* durch Eintragen der Anforderung in die Tabelle. (6 Punkte)

	8KB	*16KB*	*17KB*	*9KB*	*12KB*	*19KB*	*18KB*	*15KB*
FirstFit	*D*	*A*	*B*	*C*				
NextFit		*A*	*B*	*C*	*D*			
BestFit	*D*			*A*	*B*			*C*
WorstFit		*D*	*C*			*A*	*B*	

(b) Angenommen, Sie verwenden das Buddy-System zur Speicherverwaltung Ihres Programm-Heaps. Zu Beginn verfügen Sie über einen freien Speicherblock von 512KB an der Adresse 0. Geben Sie ein Abbild der Speicherbelegung, wenn die folgenden Speicheranforderungen in der angeführten Reihenfolge bedient werden: (A) 129KB, (B) 37KB, (C) 26KB, (D) 31KB, (E) 41KB und (F) 63KB. (10 Punkte)



(c) Wie groß kann der Verschnitt einer Speicheranforderung für ein Buddy-System im schlimmsten Fall werden? (2 Punkte)

*(c) Ist die Anforderung gerade 2^n+1 Bytes, so wird die Anforderung auf 2*2^n Bytes aufgerundet. Es bleiben also von dem zweiten Teil 2^n gerade 2^n-1 Bytes übrig als Verschnitt.*

Aufgabe 8: Virtueller Speicher 10 Punkte

(a) Begründen Sie, warum statt Multi-Level-Tabellen invertierte Seitentabellen genutzt werden. (3 Punkte)

(b) Was verbirgt sich hinter dem Begriff *Translation Lookaside Buffer*, kurz TLB? (2 Punkte)

(c) Beschreiben Sie die Funktionsweise eines Assoziativspeichers. (5 Punkte)

(a) Bei invertierten Seitentabellen müssen nur die wenigen physikalischen Seiten benannt werden, welche tatsächlich von den virtuellen Adressen referiert werden und nicht der gesamte mögliche virtuelle Adressraum. Damit ist die notwendige Tabellengröße wesentlich geringer als bei den Multi-Level-Tabellen, in denen alle erdenklichen virtuellen Adressen des Programms gespeichert sind, auch wenn sie aktuell nicht benutzt werden.

(b) Der TLB kombiniert die invertierten Seitentabellen mit einer Assoziativspeicherlogik, so dass die invertertierte Seitentabelle nicht lienear durchsucht werden muss, sondern in einem einzigen Zeittakt nach der Eingabe einer virtuellen Adresse direkt die korrespondierende physikalische Adresse ausgelesen werden kann.

(c) Ein Assoziativspeicher besteht aus Speicherzellen, die mit einer Entscheidungslogik ausgestattet sind. Wird eine Anfrage in Form eines Bitmusters angelegt, so werden alle Zellen der n-ten Tabellenspalte auf den Wert des n-ten Bits abgefragt. Parallel dazu werden alle anderen Spalten für die anderen Bits abgefragt. Erst wenn alle Zellen einer Tabellenspalte, also einer Adresse, Übereinstimmung mit dem Bitmuster melden, wird ein Erfolg („Hit“) angezeigt und die dazu gespeicherte Adresse kann ausgelesen werden.

Aufgabe 9: B-Baum / B*-Baum — 16 Punkte

Eine entscheidende Rolle bei der Schnelligkeit von Dateisystemen spielt die Organisation der Datenstrukturen.

a) Zeichnen Sie für folgende Eingabe von Schlüsseln eines Dateisystems

7, 13, 5, 9, 2, 1, 12, 14, 15, 16, 4, 6, 8

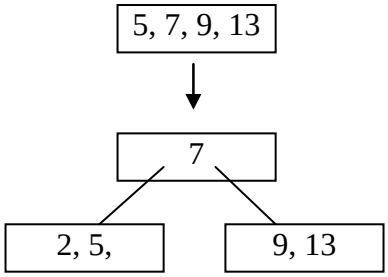
einen B-Baum und einen B*-Baum mit m = 5. Zeichnen Sie auch Zwischenschritte zu den einzelnen Bäumen! (14 Punkte)

b) Vergleichen Sie beide Bäume. Was fällt Ihnen auf und warum ist dies so? (2 Punkte)

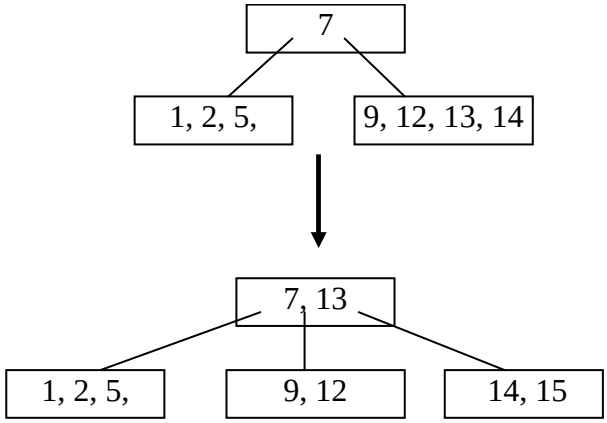
(a) Mit m=5 gibt es max. m-1=4 Schlüssel pro Container.

B-Baum:

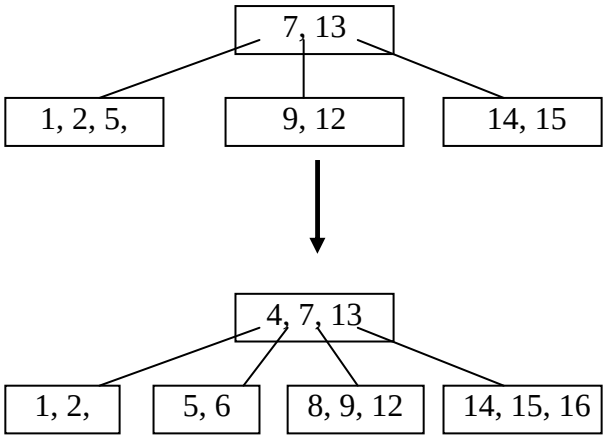
Nach max. 4 Schlüsseln 7, 13, 5, 9 *wird mit Schlüssel* 2 *die Wurzel geteilt und der mittlere Schlüssel wandert nach oben.*



Beim weiteren Auffüllen mit 1, 12, 14, 15 *wird der rechte Container zu voll und ebenfalls geteilt. Der mittlere Schlüssel* 13 *wandert nach oben in den bestehenden Container.*

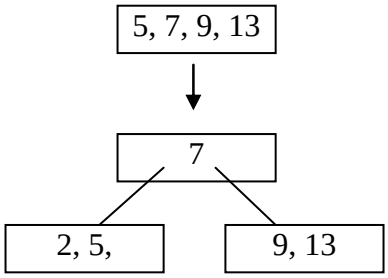


Dies geschieht auch bei den Containern für kleine Schlüssel.

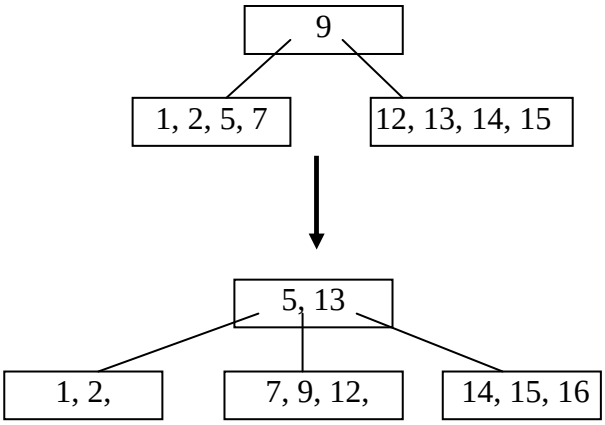


***B*-Baum**:*

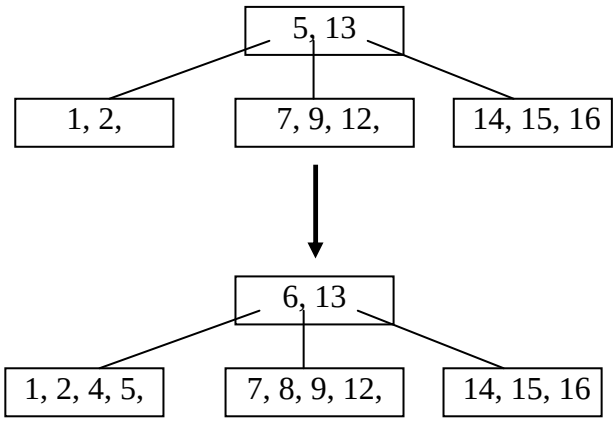
Nach max. $2\lfloor(2m-2)/3\rfloor = 4$ *Schlüsseln* 7, 13, 5, 9 *wird die Wurzel ebenfalls in zwei Hälften geteilt.*



Allerdings wird beim weiteren Abspeichern der Schlüssel 2, 1, 12, 14, 15 *zuerst der Überlauf aktiv, so dass das Teilen erst später bei Schlüssel* 16 *einsetzt. Die Trennelemente haben hierbei die Indizes* $A=\lfloor 2(m-1)/3\rfloor+1 = 3$ und $B = A + \lfloor(2m-1)/3\rfloor +1 = 7$



Die weitere Füllung mit 4,6,8 *nutzt nur noch Überfließen, keine weitere Teilung.*



(b) der B-Baum hat weniger Container, da sie besser gefüllt sind durch die Operation „überfließen“ und „Schlüsselzahl teilen durch drei“.*

Aufgabe 10: RAID-Systeme 25 Punkte

Gegeben sei ein System bestehend aus 8 kleinen und einer großen Festplatte, die unabhängig voneinander ausfallen können. Die Ausfallwahrscheinlichkeit der kleinen Platte betrage p_1, die der großen p_2.

(a) Berechnen Sie die Ausfallwahrscheinlichkeit des Systems für Raid-0. (5 Punkte)

(b) Berechnen Sie die Ausfallwahrscheinlichkeit des Systems für Raid-1. Nehmen Sie an, dass die große Festplatte einzeln arbeitet und die kleinen Festplatten je paarweise eine Gruppe bilden. (10 Punkte)

(c) Berechnen Sie die Ausfallwahrscheinlichkeit des Systems für Raid-2. Nehmen Sie an, dass das System einen Fehler korrigieren kann. Der Einfachheit halber darf angenommen werden, dass die Größe der Festplatte keine Rolle spielt bei der Verteilung der Paritätsinformation. (10 Punkte)

(a) Seien P1=1- p_1 und P2 = 1- p_2 die Überlebenswahrscheinlichkeiten. Dann ist die Überlebenswahrscheinlichkeit aller unabhängigen Einheiten gemeinsam $P1^8P2$ und die Ausfallwahrscheinlichkeit somit 1- $P1^8P2$, da bei nur einem Ausfall beim nicht-fehlertoleranten RAID0 das Gesamtsystem ausfällt.

(b) Bei RAID-1 gibt es 4 Paare von Spiegelplatten. Jedes Spiegelsystem (a,b) hat folgendes Ausfallverhalten

a	*b*	*Ausfall*
0	*0*	*1*
0	*1*	*0*
1	*0*	*0*
1	*1*	*0*

Es gibt nur Ausfall, wenn beide Platten defekt sind und es gilt Ps= p_1 p_1 für den Ausfall einer Spiegeleinheit. Da alle Spiegeleinheiten unabhängig voneinander sind, gilt also die Logik der Aufgabe vorher, so dass die Ausfallwahrscheinlichkeit 1- Ps^4P2 beträgt.

(c) Bei RAID-2 gibt es eine Fehlerkorrektur, so dass eine von 9 Platten ausfallen darf und das System immer noch funktioniert. Die Wahrscheinlichkeit für null Fehler ist mit Teil (a) $P1^8P2$, mit einem Fehler $8P1^7(1-P1)P2 + P1^8(1-P2)$. Ein Systemausfall geschieht also, wenn diese beiden Fälle nicht auftreten. Damit ist die Ausfallwahrscheinlichkeit 1-$8P1^7(1-P1)P2$ - $P1^8(1-P2)$.

Aufgabe 11: Eingebettete Systeme (Bonusaufgabe) 21 Punkte

Echtzeitsysteme bilden eine wichtige Gruppe der eingebetteten Systeme und verlangen durch ihre spezielle Widmung besondere Algorithmen.

(a) Welche drei Eigenschaften muss ein Echtzeitsystem erfüllen? (6 Punkte)

1. ***Rechtzeitigkeit***

Für jeden Task existieren Zeitschranken, innerhalb derer die gewünschte Leistung erbracht sein muss.

2. ***Gleichzeitigkeit***

Sind mehrere Tasks vorhanden, so müssen die Zeitschranken für alle Tasks erfüllt sein.

3. ***Verfügbarkeit***

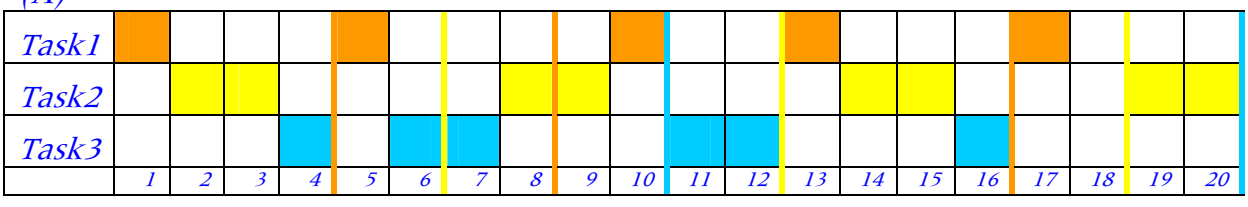
Das System darf nicht wegen Wartung, Reorganisation (garbage collection) oder Hardwaredefekten ausfallen.

(b) In der Vorlesung wurden für Echtzeitsysteme unter anderem die drei folgenden Strategien zur Erstellung eines Echtzeitschedulingstrategien vorgestellt: (A) *Minimal Deadline First,* (B) *Minimal Processing Time First,* (C) *Rate-Monotonic Scheduling* (RMS).
Erstellen Sie mit diesen drei Strategien für folgende Eingabe einen gültigen Schedule für die ersten 20 Sekunden. Alle Jobs werden zum Zeitpunkt 0 gestartet. (15 Punkte)

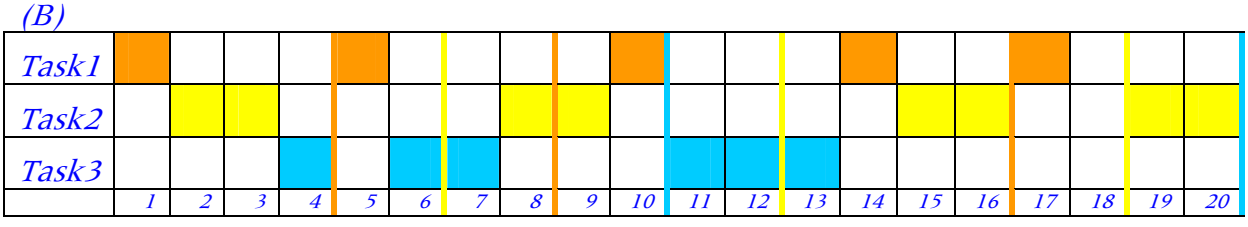
Job	Dauer (Sekunden)	Periode (Sekunden)
Task 1	1	4
Task 2	2	6
Task 3	3	10

Es ist nicht angegeben, ob die Jobs präemptiv odr nicht-präemptiv ausgeführt werden. Es wird deshalb „präemptiv" angenommen.

(A)



(B)



Matrikelnummer:

(C)