Johann Wolfgang Goethe-Universität
Frankfurt am Main

FB 15 Institut für Informatik *Praktische Informatik* PD Dr. R. Brause

KLAUSUR

zur Vorlesung Betriebssysteme SS 2002

Musterlösung

Vorname Name Matrikelnummer

Bemerkung: Die Punktzahl entspricht ungefähr der Bearbeitungsdauer in Minuten

I) **Multiple Choice-Aufgaben** (Mehrfachantworten sind möglich !) **10 Pkte**

1. Betriebsmittel sind
   - ○ ausschließlich Hardware
   - ○ ausschließlich Software
   - ⊗ können aus Hardware oder Software bestehen.
2. Eine virtuelle Maschine hat
   - ○ nur eine Schnittstelle
   - ⊗ zwei Schnittstellen
   - ○ drei Schnittstellen
3. Der Banker Algorithmus
   - ○ testet, ob in einem System eine Verklemmung vorliegt.
   - ⊗ stellt sicher, dass es zu keinen Verklemmungen kommen kann.
4. Wenn das Request-Bit einer Speicherseite ‚0' ist, wurde sie bisher noch nie referenziert
   ○ JA ⊗ NEIN
5. Eine Speicherverwaltung, die Segmentierung unterstützt
   - ○ hat genau einen linearen Adressraum
   - ⊗ ermöglicht, dass der Adressraum größer ist als der physikalische Hauptspeicher
   - ○ bietet keine automatische Relozierung des Codes
   - ○ teilt den Speicher in Segmente fester Größe.
   - ⊗ bietet explizite Zugriffskontrolle.
6. B-Bäume der Ordnung *m*
   - ⊗ haben Blätter auf nur einer Ebene
   - ○ haben maximal *m* Schlüssel pro Knoten
   - ○ haben Knoten, die im Durchschnitt nur zur Hälfte gefüllt sind.
   - ⊗ verschwenden mehr Platz in den Knoten als B* Bäume.
7. Interleaving wurde für Platten eingeführt, da die Geschwindigkeit der Platten schneller war als der Durchsatz des Festplattenadapters ⊗ JA ○ NEIN
8. Eine Subnetzmaske dient zur
   - ○ Identifikation des Routers
   - ⊗ Test, ob eine IP-Adresse im selben Subnetz liegt
   - ○ Filterung von Fehlerbits im TCP/IP-Protokoll

II) Textaufgaben

1. Virtuelle Maschinen, Interfaces 5 Pkte

Skizzieren sie die Unterschiede zwischen einem Interface und einer virtuellen Maschine.

Ein Interface ist reine Schnittstelle und

- gibt das Format von Funktionen, Daten und Protokolle vor
- enthält keine Implementierung (Code)

VM enthält zwei Schnittstellen

- wird über eine Schnittstelle angesprochen
- gibt Aufträge über zweite Schnittstelle an einen Unterauftragsnehmer weiter, d.h. eine weitere VM oder ein Gerät
- enthält eine Implementierung

2. Prozesse, Threads 10 Pkte

a) Wodurch unterscheiden sich *lightweigt*-Threads von *heavyweight*-Threads? Nennen sie je ein Beispiel in welchem BS (bzw. API oder Technologie) die jeweilige Implementierung verwendet wird.

lightweight Threads

- Implementation durch Bibliothek
- eine blockierender Thread (I/O) blockiert alle anderen des selben Prozesses
- BS weiß nichts von LW Threads: es gibt kein BS-Scheduling, keine echte Parallelität
- Bsp.: UNIX- C-Bibliotheks Threads

heavyweight Threads

- werden bei BS angemeldet und verwaltet (Scheduling, I/O-Blockierung)
- echte Parallelität möglich auf Multiprozessorsystemen
- Bsp: Windows NT, Java (NT,SOLARIS)

b) Nennen sie vier Eigenschaften wodurch sich Prozesse und Threads von einander unterscheiden.

Thema	Prozess	Thread
Kontext	eigene Dateideskriptoren je Prozess	teilen sich Deskriptor einer Datei
Adressraum	eigener Adressraum	mehrere Threads teilen sich Adressraum
Scheduling	BS übernimmt Scheduling	Programmierer hat Einfluss auf Schedul
Kooperation	Prozesse konkurrieren um Betriebsmittel	Threads müssen kooperieren
Kommunikation	Kommunikation über Shared Mem., Pipes etc	Kommunikation über lokale Variable

3. Scheduling 10 Pkte

Fünf Stapelaufträge treffen in einem Computer fast zur gleichen Zeit ein. Sie besitzen geschätzte Ausführungszeiten von 7, 16, 4, 9 und 2 Minuten und die Prioritäten 5, 2, 4, 1 und 3, wobei 5 die höchste Priorität ist. Geben sie für jeden der folgenden Scheduling-Algorithmen die durchschnittliche Verweilzeit an. Vernachlässigen sie dabei die Kosten für einen Prozesswechsel. Alle Aufträge benutzen als einziges Betriebsmittel die CPU.

i) Round Robin

ii) Priority Scheduling
iii) First-Come-First-Serve (Reihenfolge: 7, 16, 4, 9, 2)
iv) Shortest-Job-First

Annahmen: Nehmen Sie für i) an, dass das System präemptiven Mehrprogrammbetrieb verwendet und jeder Auftrag einen fairen Anteil an Prozessorzeit erhält. Des weiteren sei die Zeitscheibe sehr viel kürzer als die Zeiteinheiten, die für die geschätzten Ausführungszeiten angeführt sind, so dass die Aufträge quasi parallel ausgeführt werden.
Nehmen sie für ii) - iii) an, dass die Aufträge nacheinander (non-präemptiv) ausgeführt werden.

7	16	4	9	2	Joblängen
5	2	4	1	3	Prioritäten

7	16	4	9	2	**26,2**	**FCFS**
7	23	27	36	38	131	$= \Sigma$

7	4	2	16	9	**19,6**	**Prio**
9	11	13	29	38	98	$= \Sigma$

2	4	7	9	16	**16,2**	**SJF**
2	6	13	22	38	81	$= \Sigma$

5	2	2	10		
4	4	2	18		
3	7	3	27		
2	9	2	31		
1	16	7	38		
		$\Sigma =$	124	**24,8**	**RR**

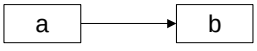
4. Prozesssynchronisation **20 Pkte**

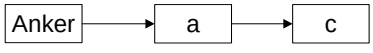
a) Geben sie ein Beispiel für eine *Race Condition*

Bei einer RC greife zwei Prozesse konkurrierend und nicht synchronisiert auf eine globale Ressource zu.
Eine Race Condition entsteht z.B. dann, wenn ein Prozess Informationen aus der Ressource abruft, lokal speichert und unterbrochen wird. Der zweite Prozess wird erweckt bzw. gestartet und ändert die Daten, die Prozess 1 lokal speichert: Prozess 2 überholt Prozess 1. Prozess 1 kommt wieder zum Zug und führt Änderungen basierend auf den mittlerweile ungültigen Informationen aus.

Beispiel Einfügen und Aushängen in Lineare Liste, die Einträge b und c enthält:
Prozess A will Eintrag a Liste einhängen, B den ersten Eintrag b entfernen.

1. B kommt vom Anker zu b
2. B speichert Zeiger auf nächstes Element c
3. B wird unterbrochen, A kommt zur Ausführung
4. A speichert Zeiger auf b
5. A setzt Anker auf a
6. A setzt next-Zeiger von a auf b
7. A wird unterbrochen, B kommt zur Ausführung
8. B setzt next-Zeiger des Ankers auf c





Die oberen beiden Graphen zeigen das Resultat, unten ist die korrekte Struktur aufgeführt.

b) Die Prozesse P_1 und P_2 benötigen exklusiven Zugriff auf die Ressourcen R_1, R_2, R_3. Eine Ressource wird mit dem Befehl open(R_x) geöffnet und mit close(R_x) freigegeben. Wenn ein Prozess versucht eine Ressource zu öffnen, die bereits ein anderer Prozess benutzt, blockiert dieser mindestens so lange, bis die Ressource mittels close() freigegeben wird. Der Programmcode für die Prozesse P_1 und P_2 ist:

P_1:

```
open(R1);
if(b) {
   open(R2);
   close(R1);
   open(R3);
} else {
   open(R3);
   close(R1);
   open(R2);
}
close(R2);
close(R3);
```

P_2:

```
open(R3);
if(b) {
   open(R2);
   close(R3);
   close(R2);
   open(R1);
} else {
   close(R3);
   open(R1);
   open(R2);
   close(R2);
}
close(R1);
```

Untersuchen Sie unter der Bedingung, dass die boolesche Variable b beliebig initialisiert wird und nicht geändert wird, ob es zu Verklemmungen kommen kann.

Fall 1: b = true

P_1:

```
open(R1);
open(R2);
close(R1);
open(R3);
close(R2);
close(R3);
```

P_2:

```
open(R3);
open(R2);
close(R3);
close(R2);
open(R1);
close(R1);
```

Verklemmung möglich:

```
P2:   open(R3);
P1:   open(R1);
P1:   open(R2);
P1:   close(R1);
P1:   open(R3);     // P1 blockiert, da R3 geöffnet ist
P2:   open(R2);     // P2 blockiert, da R2 geöffnet ist
```

Typische Situation: P_1 hat R_2 und will R_3, P_2 hat R_3 und will R_2.

Fall 2: b = false

P_1:

```
open(R1);
open(R3);
close(R1);
open(R2);
close(R2);
close(R3);
```

P_2:

```
open(R3);
close(R3);
open(R1);
open(R2);
close(R2);
close(R1);
```

Keine Verklemmung möglich: Für alle Paare (i,j), i, j∈{1,2,3}, i≠j gibt es keine Kombination, in der ein Prozess zuerst R_i und dann R_j während der andere zuerst R_j und dann R_i anfordert, ohne dass die jeweils zuerst angeforderte Ressource wieder frei gegeben worden ist.

5. Speicherverwaltung **30 Pkte**

a) Wir betrachten die folgende Referenzfolge von Seitenzugriffen bei einer Speichergröße von 4 Seiten: 1, 2, 3, 1, 5, 3, 4, 1, 2, 3, 5, 4 ausgehend von einem vollständig freien Speicher.
Skizzieren Sie die Auswirkungen bekannter Ersetzungsstrategien (15 Pkte)

1	2	3	1	5	3	4	1	2	3	5	4	**FIFO**
						x	**x**	**x**	**x**	**x**	**x**	
1	1	1	1	1	1	**4**	4	4	4	**5**	5	
	2	2	2	2	2	2	**1**	1	1	1	**4**	
		3	3	3	3	3	3	**2**	2	2	2	
				5	5	5	5	5	**3**	3	3	

1	2	3	1	5	3	4	1	2	3	5	4	**LRU**
						x		**x**		**x**	**x**	
1	1	1	1	1	1	1	1	1	1	1	**4**	
	2	2	2	2	2	**4**	4	4	4	**5**	5	
		3	3	3	3	3	3	3	3	3	3	
				5	5	5	5	**2**	2	2	2	

1	2	3	1	5	3	4	1	2	3	5	4	**OPT**
						x				**x**		
1	1	1	1	1	1	1	1	1	1	**5**	5	
	2	2	2	2	2	2	2	2	2	2	2	
		3	3	3	3	3	3	3	3	3	3	
				5	5	**4**	4	4	4	4	4	

b) Ein Computer benutzt das Buddy-System zur Speicherverwaltung. Zu Beginn verfügt er über einen freien Speicherblock von 1 Megabyte, beginnend bei der Adresse 0. Geben Sie jeweils ein Abbild der Speicherbelegung nach jedem Speicherzugriff, wenn die folgenden Speicheranforderungen (request) und Freigaben (release) in der angeführten Reihenfolge bedient werden:
A: request 64K, B: request 230K, C: request 100K, D: request 129K, release A, release B, E: request 75K, release C, release E, release D
Beschriften Sie dabei die zusammenhängenden Speicherblöcke jeweils mit ihrer Größe. (8 Pkte)

A: request 64K	B: request 230K	C: request 100K	D: request 129K	release A	release B	E: request 75K	release C	release E	release D
512	512	512	256	256	256	256	256	256	1024
			D 129	D 129	D 129	D 129	D 129	D 129	
256	B 230	B 230	B 230	B 230	256	256	256	512	
128	128	C 100	C 100	C 100	C 100	C 100	128		
64	64	64	64	128	128	E	E		
A	A	A	A						

c) Die Erweiterung des Hauptspeicher (eine größere Anzahl von Seitenrahmen) garantiert nicht für alle Seitenersetzungsstrategien, dass auch die Anzahl der Seitenfehler sinkt. Wann kommt es zu diesem Effekt ? (7 Pkte)

Bei den so genannten Stack-Algorithmen führt eine Vergrößerung der Seitenzahl immer zu einer Verringerung der Seitenfehler. Um festzustellen, welche Seite aus dem Speicher verdrängt ist, führt man bei allen Strategien eine Ordnung ein. Bei den Stack-Algorithmen wird dadurch eine Reihenfolge der Seiten realisiert, die unabhängig ist von der Anzahl der zur Verfügung stehenden Seitenrahmen. Bei FIFO ist dies beispielsweise nicht der Fall. Hier ist der Zeitpunkt, wann eine Seite verdrängt wird, davon abhängig, wann sie einen Seitenrahmen erhält, wie viele Seitenfehler seit diesem Zeitpunkt auftreten und wie viele Seitenrahmen insgesamt zur Verfügung stehen.

6. Virtueller Arbeitsspeicher **20 Pkte**

Gegeben ist die folgende Seitentabelle für einen Prozeß. Alle Zahlen sind dezimal und starten mit 0. Die Seitengröße beträgt 2048 Bytes.

Seite	Valid	R	M	Seitenrahmen
0	0	1	0	--
1	1	1	1	5
2	1	0	0	17
3	1	0	0	1
4	0	0	0	--
5	1	0	1	0

a) Beschreiben Sie, wie aus den virtuellen Adressen die physikalischen Hauptspeicheradressen erzeugt werden. Nehmen Sie dazu (unnötigerweise) an, dass eine virtuelle Adresse 16 Bit lang ist.

Eine virtuelle Adresse teilt sich in zwei Teile: Den Index der Seitentabelle und den Offset. Bei einer Seitengröße von 2048 Bytes werden für den Offset 11 Bit benötigt. Also bleiben

im Beispiel einer 16 Bit Adresse 5 Bit für den Index. Mit dem Index erhält man die Zeile der Seitentabelle, in der Informationen zur gewünschten Seite abgelegt sind. Ist die Seite im Hauptspeicher, so findet sich dort die Nummer des Seitenrahmens. Die physikalische Adresse erhält man, indem die höchstwertigen 5 Bit der virtuellen Adresse ersetzt werden durch die Bits für die Nummer des Seitenrahmens.

b) An welchen physikalischen Adressen, so sie eine haben, finden sich die folgenden virtuellen Adressen? Falls Seitenfehler auftreten geben Sie nur dies an.

i) 2052 = 1·2048 + 4, also Index = 1 mit *offset* = 4:
Zeile 1, Seitenrahmen 5, physikalische Adresse oberste Bits = 5 mit dem Wert 5·2048 = 10240. Die *offset* Bits mit dem Wert 4 werden an die obersten Bits konkateniert, was bei der Dezimalzahl eine Addition bedeutet: 10240 + 4 = 10244.

ii) 6177 = 3×2048 + 33, also Index = 3 mit *offset* = 33:
Zeile 3, Seitenrahmen 1, physikalische Adresse = 1·2048 + 33 = 2081

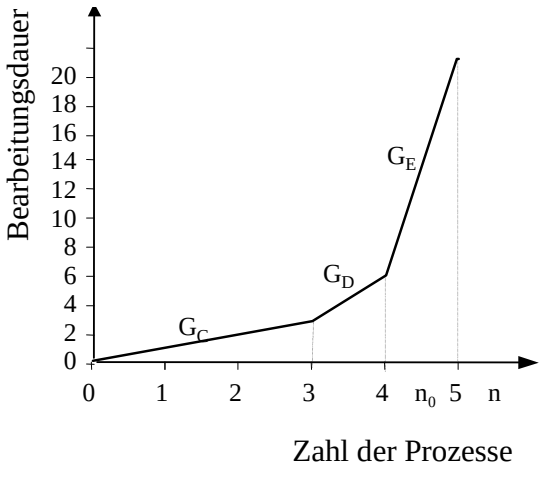
iii) 8999: Zeile 4, Seitenfehler

c) Ein Computer mit 32-bit Adressen benutzt eine zweistufige Adresskonversion. Virtuelle Adressen bestehen aus 9 Bit für die Top-Level-Seite, 11 Bit für die zweite Seitentabelle und dem Offset. Wie groß ist eine Seite und wie viele Seiten können im Adressraum verwaltet werden?

Für den Offset stehen 32–(11+9) = 12 Bit zur Verfügung. Somit ist eine Seite 2^{12} = 4096 = 4KB groß. Es können 2^{20} = 1.048.576 = 1 Million Seiten adressiert werden.

7. Thrashing **8 Pkte**

In dem folgenden Diagramm ist die Bearbeitungsdauer gegen die Prozesszahl aufgetragen. Beschreiben Sie was an den Knickpunkten geschieht.



Ist die Seitentauschrate für das *working set* ρ_T größer als die Rate beim Übergang von der Rechenbelastung zur Seitentauschbelastung $\rho_T \geq \rho_W$ entstehen drei Bereiche, die durch folgende Relationen geprägt sind:

Bereich C: relatives Speicherangebot $\sigma \geq \sigma_W$, Normalbetrieb

Bereich D: $\sigma_T \leq \sigma \leq \sigma_W$. Ab dem ersten Punkt bei n = 3 setzt *thrashing* ein. Ab hier gilt, dass weniger effektive Rechenzeit je Prozess zur Verfügung steht als es dauert, eine Seite einzulagern.

Bereich E: $0 \leq \sigma \leq \sigma_{T.}$ Der Knickpunkt bei n = 4 bezeichnet die Stelle, ab der das *working set* eines Prozesses nicht mehr in den Hauptspeicher geladen werden kann.

8. Dateisysteme **30 Pkte**

a) i-nodes (10 Pkte)

- Erklären sie wie mittels i-nodes sowohl sehr kleine als auch riesig große Dateien elegant abgebildet werden können.

  Ein i-node enthält die ersten 10 Adressen einer Datei. Dadurch können Dateien, die höchstens 10 Blöcke belegen (davon gibt es i.A. sehr viele, z.B. Verzeichnisse) sehr schnell bearbeitet werden. Des weiteren gibt es drei weitere Adressen von Blöcken für die einfach-, doppelt- und dreifach-indirekte Adressierung. Dabei werden Tabellen mit jeweils 256 Verweisen adressiert. Je nach Grad der Indirektion verweisen die Tabelleneinträge auf Datenblöcke oder weitere Indextabellen. Bei einer (sehr großen) Datei stehen die Adressen der ersten 10 Blöcken in dem i-node, die nächsten 256 Blöcke in der Tabelle, deren Adresse unter einfach indirekt vermerkt ist, dann kommen die zweifach- und schließlich die dreifach-indirekt adressierten Blöcke.

- Angenommen, die Tabellen enthalten je 20 Verweise, wie viele Seiten können durch eine i-node Datenstruktur maximal verwaltet werden ?

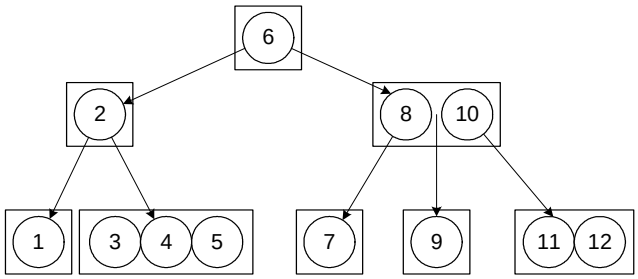
  $10+20+20^2+20^3 = 30+400+8000 = 8430$ Blöcke

b) Zeichnen Sie einen B-Baum (m=4), welchen man erhält, wenn die Folge

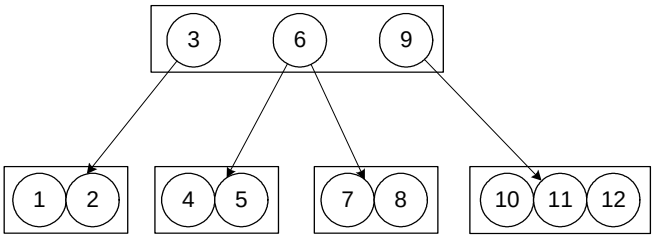
{8,7,6,2,5,9,10,1,4,11,3,12}

in einen anfangs leeren Baum einfügt wird. Wiederholen Sie die Aufgabe mit einem B*-Baum. (20 Pkte)

B-Baum:



B*-Baum:



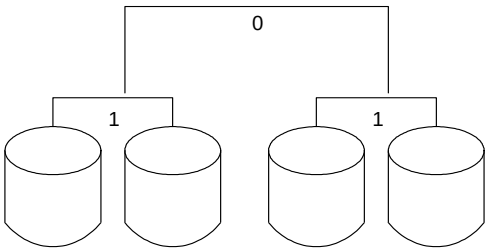
9. I/O, Geräte **15 Pkte**

a) Funktioniert *Cylinder Skew* beim Interleaving für alle Spur-Schedulingstrategien ?

Nein nur bei C-Scan

b) RAID

Die Wahrscheinlichkeit, dass eine Festplatte in einem gegebenen Zeitraum einen Fehler (Head-Crash) aufweist betrage p. Bestimmen Sie die Wahrscheinlichkeit, dass im Beobachtungszeitraum Daten verloren gehen, für eine Konfiguration mit: vier Platten in RAID-1/0, wobei mittels RAID-0-*striping* zwei RAID-1 gespiegelte Platten verbunden werden. Berechnen Sie die Wahrscheinlichkeit, dass es im Beobachtungszeitraum zu Datenverlust kommt.



Das Problem lässt sich auf drei verschiedenen Wegen lösen:

1.) Die Gesamtwahrscheinlichkeit eines Ausfalls ist die Summe aller disjunkten Ereignisse:- Wahrscheinlichkeit; dass 4 Platten ausfallen: p^4

- Wahrscheinlichkeit; dass 3 Platten ausfallen: $4 \cdot p^3 \cdot (1-p)$
- Wahrscheinlichkeit; dass die 2 Platten eines Armes ausfallen: $2 \cdot p^2 \cdot (1-p)^2$

$p(\textit{fault}) = p^4 + 4 \cdot p^3 \cdot (1-p) + 2 \cdot p^2 \cdot (1-p)^2) = 2 \cdot p^2 - p^4$

2.) Die Wahrscheinlichkeit s, dass in RAID 1 beide Platten ausfallen, ist p^2. Das Ereignis, dass entweder das linke Paar, das rechte Paar, oder beide ausfallen, ist (wobei der Fall, dass beide ausfallen, nicht doppelt gezählt wird)

$p(\textit{fault}) = s_1 + s_2 - s_1 s_2 = p^2 + p^2 - p^2 \cdot p^2 = 2 \cdot p^2 - p^4$

3.) Die Wahrscheinlichkeit, dass kein Datenverlust auftritt, ist $(1-p^2)$ pro gespiegelte Platten, also $(1-p^2)^2$ im System. Damit ist die Wahrscheinlichkeit eines Datenverlustes

$p(\textit{fault}) = 1-(1-p^2)^2 = 1 - 1 + 2p^2 - p^4 = 2p^2 - p^4$

10. Treiber, Netzwerke **10 Pkte**

a) Skizzieren sie die wichtigsten Unterschieden zwischen dem TCP und UDP Protokoll.

Typ	TCP	UDP
Acknowledgement	ja	nein
Fehlerkorrektur	ja	nein
Reihenfolge der IP-Pakete gewährleistet	ja	nein
Verbindungsorientiert	ja	nein
Geschwindigkeit (wg.Overhead)	niedrig	hoch

b) Erläutern sie die Probleme, die entstehen können wenn mehrere Personen gleichzeitig in Sitzungssemantik ein Dokument bearbeiten. Wie kann es zu Inkonsistenzen kommen, welche Vorteile bietet die Sitzungssemantik?

Inkonsistenzen: Bei der Sitzungssemantik überschreibt die Person, die zuletzt speichert, alle Änderungen, die andere Benutzer vorgenommen haben, seit sie die Datei gelesen hat. Damit kommt es zu Inkonsistenzen zwischen den lokalen Versionen der Personen.
Vorteile: Die Datei ist immer in sich konsistent in einem gültigen Zustand. Durch die lokale Pufferung werden Performancevorteile erreicht.