FB Informatik und Mathematik
Prof. R. Brause

Klausur + Musterlösung

Betriebssysteme SS 2009

9.7.2009

Vorname:	
Nachname:	
Matrikelnummer:	
Geburtsdatum:	
Studiengang:	

Bitte tragen Sie auf jeder Seite Ihre Matrikelnummer ein und überprüfen Sie diese Klausur auf Vollständigkeit (14 Seiten!). Tragen Sie auf dem Deckblatt Ihre Daten in **Druckbuchstaben** ein.

Als Hilfsmittel ist ein nicht programmierbarer Taschenrechner zugelassen. Verwenden Sie ausschließlich die beigefügten Blätter. Sollten diese nicht ausreichen, so wenden Sie sich bitte an die Aufsicht. Alle weiteren 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 7 Aufgaben mit insgesamt 90 Punkten; Aufgabe 8 ist optional. Die Punktzahl jeder Aufgabe entspricht der geschätzten Bearbeitungszeit in Minuten. Die maximale Bearbeitungszeit beträgt 180 Minuten.

Viel Erfolg !!!

1	2	3	4	5	6	7	8	Σ	Übungspkte
8	*20*	*8*	*12*	*12*	*15*	*15*	*15*		*10*

Aufgabe 1: Schichtenmodell 8 Punkte

(a) Was ist eine virtuelle Maschine? (2 Pkte)

Eine virtuelle Maschine ist eine Softwareschicht, die durch zwei Schnittstellen gekennzeichnet ist: eine Exportschnittstelle, die Leistungen anderen Programmen zur Verfügung stellt, und einer Importschnittstelle, die besagt, welche Leistungen sie dafür benötigt.

(b) Was ist eine Software-Hardware-Migration? (2 Pkte)

Bei der SW-HW-Migration wird eine SW-Schicht, die meist direkt auf der HW-Schicht liegt, ebenfalls in HW ausgeführt, um die Funktionen schneller zu machen.

(c) Im Unterschied zum Konzept der abstrakten Maschine erlaubt eine virtuelle Maschine eine Software-Hardware-Migration. Warum? (2 Pkte)

Eine abstrakte Maschine hat nur eine Exportschnittstelle und ist deshalb keine Schicht. Für die Migration wird aber eine Schicht mit zwei Schnittstellen benötigt wie sie von der virtuellen Maschine zur Verfügung gestellt wird. Die zweite Schnittstelle definiert die Kommunikation mit der Hardware, die bei einer Ersetzung der Software durch Hardware genau befolgt werden muss. Ist die Kommunikation unbekannt, kann die Software nicht ersetzt werden.

(d) Wann ist eine Software-Lösung zu bevorzugen, wann eine Hardware-Realisierung? (2 Pkte)

Eine SW-Lösung ist bei schnellem Aufbau und Debugging der Funktionalität vorzuziehen; sind die Dienste aber fest und die Ausführungsgeschwindigkeit kritisch, so ist eine HW-Lösung (HW-Migration) vorzuziehen.

Aufgabe 2: Prozesse und Scheduling 20 Punkte

Fünf Stapelaufträge treffen praktisch zeitgleich bei ihrem Computersystem ein. Ihr Computer verfügt über nur einen Prozessor. Die Aufträge werden von einem Scheduler nach verschiedenen Kriterien sortiert. Die Ankunftsreihenfolge ist A-B-C-D-E. Die Zeit für einen Prozesswechsel wird vernachlässigt. Je höher die Prioriätszahl, desto wichtiger der Prozess.

Prozess	***A***	***B***	***C***	***D***	***E***
Zeitdauer	***3***	***2***	***5***	***1***	***2***
Priorität	***1***	***6***	***3***	***2***	***4***

(a) Berechnen Sie die durchschnittliche Verweilzeit für folgende Strategien:

(i) FCFS

(ii) Priority Scheduling

(iii) Shortest-Job-First

(i) FCFS *(2 Pkte)*

Die Reihenfolge der Jobs ist nach der Ankunft A,B,C,D,E mit den jeweiligen Zeiten 3,2,5,1,2. Die Gesamtdauer ist jeweils die Summe aus vorheriger Dauer und eigener Dauer, also 3,5,10,11,13 und der Gesamtsumme 3+5+10+11+13 = 42 mit der mittleren Dauer von 42/5 = **8,4** Zeiteinheiten.

(ii) Priority Scheduling *(2 Pkte)*

Die Reihenfolge der Jobs ist nach der Priorität B,E,C,D,A mit den jeweiligen Zeiten 2,2,5,1,3. Die Gesamtdauer ist jeweils die Summe aus vorheriger Dauer und eigener Dauer, also 2,4,9,10,13 und der Gesamtsumme 2+4+9+10+13 = 38 mit der mittleren Dauer von 38/5 = **7,6** Zeiteinheiten.

(iii) Shortest-Job-First *(2 Pkte)*

Die Reihenfolge der Jobs ist nach der Länge D,E,B,A,C mit den jeweiligen Zeiten 1,2,2,3,5. Die Gesamtdauer ist jeweils die Summe aus vorheriger Dauer und eigener Dauer, also 1,3,5,8,13 und der Gesamtsumme 1+3+5+8+13 = 30 mit der mittleren Dauer von 30/5 = **6** Zeiteinheiten.

(b) Berechnen Sie für die Round-Robin-Strategie die durchschnittliche Verweilzeit, wenn die Zeitscheiben gegenüber den Ausführungszeiten vernachlässigbar klein sind. (6 Pkte)

Bei quasi-paralleler Ausführung gehen die Jobs nach unterschiedlichen Zeiten zu Ende:

Jobende	restZeiteinh.	* #parallele Jobs =	Gesamtzeit
D	1	5	5
E	1	4	+ 5 = 9
B	1	4	+ 5 = 9
A	1	2	+ 9 = 11
C	2	1	+11= 13
Gesamt			47

Mittlere Ausführungszeit ist also 47/5 = **9,4**

(c) Berechnen Sie für die Round-Robin-Strategie die durchschnittliche Verweilzeit, wenn eine Zeitscheibe genau eine Zeiteinheit beträgt. (6 Pkte)

A	3	■					■				■			
B	2		■					■						
C	5			■					■			■	■	■
D	1				■									
E	2					■				■				
		1	2	3	4	5	6	7	8	9	10	11	12	13
					D			B		E	A			C

Die mittl. Zeit ist mit D+B+E+A+C = 4+7+9+10+13 = 43 genau 43/5 = **8,6** Zeiteinheiten.

(d) Wieso unterscheiden sich die beiden Ergebnisse aus b) und c)? Begründen Sie dies.

(2 Pkte)

Der Unterschied ergibt sich aus dem schnelleren Terminieren der Jobs: Das Laufen eines Jobs innerhalb einer vollen Zeiteinheit bedeutet ein Vorziehen der Jobs vor die anderen, die ebenfalls laufen sollten, aber warten müssen. Bei der letzten Zeitscheibe eines Jobs wird der Job früher beendet, als er eigentlich durch die Gegenwart der anderen Jobs dürfte, was die mittlere Zeit für ihn und damit für alle anderen auch verkürzt.

Aufgabe 3: Prozess-Synchronisierung 8 Punkte

Angenommen, Sie haben folgenden Programmauszug in Pseudocode:

```
global buffer A,B;

main(..)
      P(S1)
   update (A);
      V(S1); P(S2)
   update (B);
      V(S2)
end_main;
```

```
procedure update(buffer buf)
   string text;
      …
   read(buf,text);
      …
   change (text);
      …
   write(buf,text)
      …
end_procedure;
```

(a) Was versteht man unter einer „race condition“? *(2 Pkte)*

Eine „race condition“ tritt auf, wenn ein Prozess einen Code schneller abarbeitet als ein anderer (z.B. bedingt durch den Wechsel der Zeitscheibe) und dadurch Nebeneffekte auftreten können. Typischerweise ist dies der Fall, wenn beide Prozesse lesend und schreibend auf gemeinsame Daten („globale Variable“) zugreifen.

(b) Wo im obigen Codesegment können race conditions *auftreten, wenn es in mehreren Threads gleichzeitig ausgeführt wird? Wieviele kritische Abschnitte gibt es? Begründen Sie Ihre Antwort.* *(2 Pkte)*

Die Nebeneffekte können auftreten, wenn zwei Threads parallel im Code „update()“ Lesen und Schreiben auf denselben Puffer. Es gibt nur einen kritischen Abschnitt (siehe eingezeichnetes Viereck im Code), der aber innerhalb eines Threads für zwei unterschiedliche Puffer aufgerufen wird.

(c) Ergänzen Sie oben den Code geeignet, so dass race conditions *vermieden werden. (2 Pkte)*

Da nur ein kritischer Abschnitt existiert, würde zum Schutz eigentlich ein Semaphor reichen. Da aber dann der Zugriff auf Puffer A gesperrt sein könnte, obwohl nur auf B zugegriffen wird, muss aus Fairnessgründen jeder globale Speicherbereich seinen eigenen Semaphor bekommen. Deshalb ist in roter Farbe `P()` und `V()` für zwei Semaphore eingezeichnet; als kritischer Abschnitt ist der jeweilige Aufruf von „update“ geschützt.

Alternativ kann man auch `update(buf)` um ein Semaphor-Argument erweitern und den kritischen Abschnitt in `update(buf, S)` mit `P(S)` und `V(S)` schützen.

(d) Angenommen, Sie haben vier Threads, die den obigen Code ausführen. Wieviele Semaphore benötigen Sie? *(2 Pkte)*

Die Anzahl der Threads ist nicht entscheidend, sondern die Anzahl der globalen Speicherbereiche. Also benötige ich zwei Semaphore.

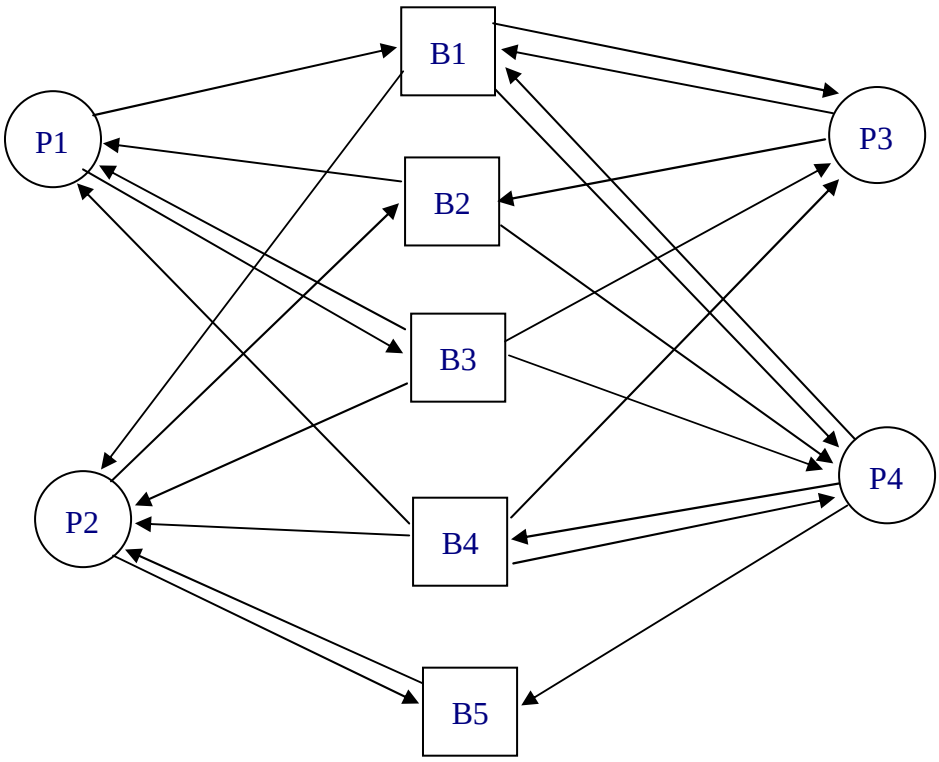
Aufgabe 4: Verklemmungen 12 Punkte

Gegeben seien für einen Systemzustand die folgenden vier Prozesse P1, P2, P3, P4 mit ihren Betriebsmitteltabellen.

	vorhanden						*zusätzlich gewünscht*				
	B1	*B2*	*B3*	*B4*	*B5*		*B1*	*B2*	*B3*	*B4*	*B5*
P1	*0*	*1*	*1*	*1*	*0*		*3*	*0*	*1*	*0*	*0*
P2	*2*	*0*	*1*	*1*	*2*		*0*	*1*	*0*	*0*	*1*
P3	*1*	*0*	*2*	*1*	*0*		*1*	*2*	*0*	*0*	*0*
P4	*1*	*1*	*1*	*1*	*0*		*1*	*0*	*0*	*1*	*1*

Frei	*X*	*0*	*0*	*1*	*1*

(a) Zeichnen Sie den Betriebsmittelgraphen auf. *(4 Pkte)*



(b) Wenden Sie den Banker-Algorithmus zur Verklemmungserkennung an. Was ist das kleinste x für Betriebsmittel 1, für den der Systemzustand sicher ist? (8 Pkte)

P4: X=1 frei: 2 1 1 2 1

P2: frei: 4 1 2 3 3

P1: frei: 4 2 3 4 3

P3: frei: 5 2 5 5 3

Das kleinste X ist gleich eins, da bei X=0 P4 (und kein weiterer Prozess) zu Ende gehen kann.

Aufgabe 5: Virtuelle Adressen 12 Punkte

Gegeben seien die folgenden drei virtuellen 16-Bit Speicheradressen in Hexadezimaldarstellung:

(a) 75B4 (b) 8AC6 (c) 5B3E

Die Zuordnung der Bits zu den drei Seitentabellen und zum offset ist durch folgende Zeichnung gegeben:

Basis			*indx1*		*indx2*		*offset*							
15	*14*	*13*	*12*	*11*	*10*	*9*	*8*							*0*

Notizen:

(a)	0	1	1	1	0	1	0	1	B	4
(b)	1	0	0	0	1	0	1	0	C	6
(c)	0	1	0	1	1	0	1	1	3	E

Neben der Basis-Seitentabelle sind in der folgenden Abbildung für jede Stufe die benötigten Tabellen angegeben. Bestimmen Sie damit die zu (a), (b), (c) gehörenden physikalischen Adressen. Geben Sie diese in Hexadezimaldarstellung an. Man beachte, dass von der in der Tabelle angegebenen Hexzahl nur die letzten 7 Bits verwendet werden können.

7	*4*
6	*2*
5	*1*
4	*3*
3	*1*
2	*4*
1	*5*
0	*7*

Basistabelle

3	*–*
2	*5*
1	*7*
0	*0*

indx1-Tabelle1

3	*–*
2	*4D*
1	*0A*
0	*04*

indx2-Tabelle0

3	*7*
2	*2*
1	*0*
0	*–*

indx1-Tabelle3

3	*71*
2	*20*
1	*–*
0	*62*

indx2-Tabelle5

3	*5*
2	*–*
1	*3*
0	*7*

indx1-Tabelle4

3	*–*
2	*34*
1	*5F*
0	*15*

indx2-Tabelle7

Ergebnisrechnung:

	Basis	**idx1**	**idx2**	**Phys.**	**offset**
(a)	3	→1:2	→5:2	20	1B4
(b)	4	→3:1	→0:1	0A	0C6
(c)	2	→4:3	→5:1	**?**	03E

Lösung

(a) 41B4 *(4 Pkte)*

(b) 14C6 *(4 Pkte)*

(c) page fault *(4 Pkte)*

Die Hexdarstellungen der Physikalischen Adresse und des offsets werden zusammengeführt, nicht addiert.

Hilfstabelle: *Dezimal, hexadezimal und Binärdarstellung für 4 Bits.*

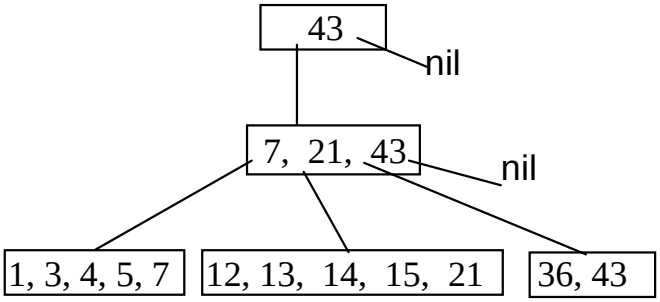
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*10*	*11*	*12*	*13*	*14*	*15*
0	*1*	*2*	*3*	*4*	*5*	*6*	*7*	*8*	*9*	*A*	*B*	*C*	*D*	*E*	*F*
0000	*0001*	*0010*	*0011*	*0100*	*0101*	*0110*	*0111*	*1000*	*1001*	*1010*	*1011*	*1100*	*1101*	*1110*	*1111*

Aufgabe 6: Dateistrukturen 15 Punkte

Ihr System erreicht die Zugriffsfolge auf die Schlüssel 7, 4, 15, 12, 21, 1, 36, 43, 14, 13, 5, 3 eines Dateisystems.

(a) Zeichnen Sie für die gegebene Zugriffsfolge eine Baumstruktur entsprechend eines zweistufigen, index-sequentiellen Dateizugriffs. Gehen Sie davon aus, dass ein Behälter maximal 5 Schlüssel Speichern kann. (5 Pkte)

Lösung:

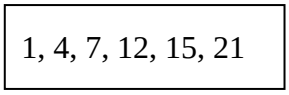


(b) Gegeben sei ein leerer B-Baum mit m = 6. Fügen sie die oben beschriebene Schlüsselfolge in den Baum ein. Benutzen Sie hierfür das in der Vorlesung vorgestellte Verfahren. Zeichnen Sie nach jeder Einfüge-Operation, die die Struktur verändert, den vollständigen Baum. Beachten Sie, dass die Wurzel max. 2 ⌊(2m–2)/3⌋ Schlüssel enthalten kann.*

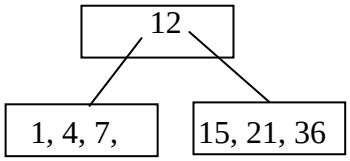
(10 Pkte)

Max. 5 Schlüssel pro Container, aber max. 6 Schlüssel in der Wurzel. Also ergibt sich folgender Baumaufbau:

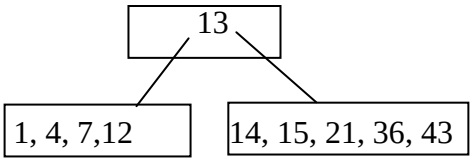
Wurzel nach Aufnahme der ersten 6 Schlüssel:



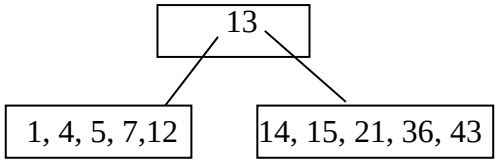
Der nächste Schlüssel ist 36. Also **teilt** sich die Wurzel zu (3 Pkte)



Der folgende Schlüssel 43 und 14 füllen nur den rechten Container. Erst bei Schlüssel 13 wird der rechte Container zu voll, so dass er nach links **überfließt** zu (2 Pkte)



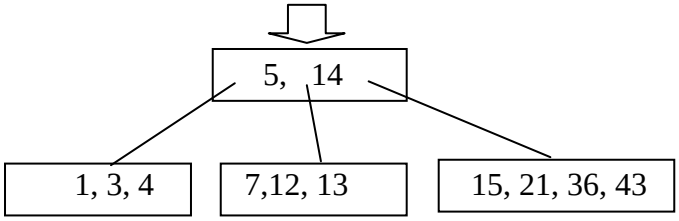
Schlüssel 5 ergänzt links den Container.



Nun erscheint Schlüssel 3. Dieser findet weder rechts noch links Platz: Die Container werden neu aufgeteilt.

Zusammensetzen aller beteiligten Schlüssel: (5 Pkte)

1, 3, 4, **5**, 7,12, 13,**14**, 15, 21, 36, 43



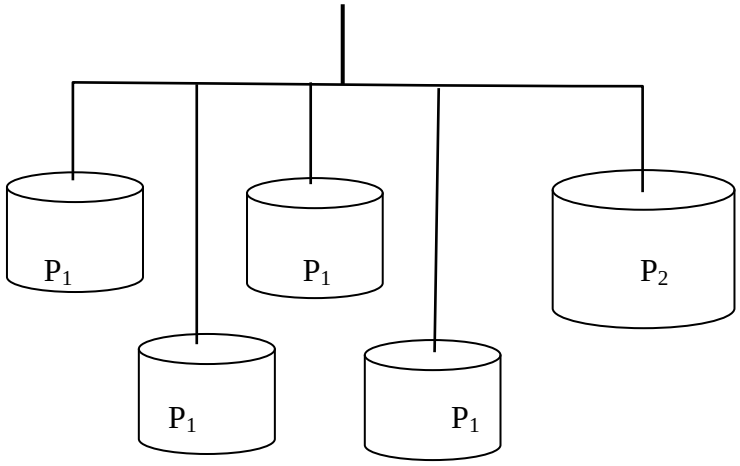
und Neuaufteilung der Schlüssel nach Schlüsselzahlen $\lfloor(2m-2)/3\rfloor$, $\lfloor(2m-1)/3\rfloor$ und $\lfloor 2m/3\rfloor$ erbringt den endgültigen B*-Baum.

Wie viele Schlüssel enthält jeder innere Knoten (ausgenommen die Wurzel und die Blätter) mindestens bzw. höchstens?

Jeder innere Knoten enthält mindestens $\lfloor(2m–2)/3\rfloor = 3$ Schlüssel und höchstens $m–1 = 5$ Schlüssel.

Aufgabe 7: RAID-Systeme 15 Punkte

Gegeben sind 4 kleine Festplatten, die mittels RAID-1 in einem Verbund zusammen mit einer großen Festplatte organisiert sind.



Die Ausfallwahrscheinlichkeit einer kleinen Platte betrage P_1 = 0,05%, die der großen Platte P_2 = 0,0001%.

(a) Was ist zuverlässiger, das RAID-System aus den 4 Platten, organisiert in je zwei Spiegelplattenpärchen, oder die große Platte? (5 Pkte)

Ein Spiegelsystem ist nur dann defekt, wenn beide Platten defekt sind. Also ist die Ausfallwahrscheinlichkeit P_P eines Pärchens bei unabhängigen Platten

$$P_P = P_1 \cdot P_1 = 5 \cdot 10^{-2}\ \% \cdot 5 \cdot 10^{-2}\ \% = 25 \cdot 10^{-4} \cdot 10^{-4} = 2{,}5 \cdot 10^{-7}$$

Zwei unabhängige Pärchen funktionieren nicht mehr, wenn eines oder beide davon defekt sind. Die Ausfallwahrscheinlichkeit P_G dafür ist

$$P_G = P_P\,(1 - P_P) + (1 - P_P)\,P_P + P_P P_P = 2\,P_P\,(1 - P_P) + P_P P_P$$
$$= 2\,P_P - P_P^2 < 2\,P_P = 5 \cdot 10^{-7} < 10^{-6} = P_2$$

Also ist das System aus zwei kleinen Spiegelpärchen zuverlässiger als die große Platte.

(b) Wie groß ist die Ausfallwahrscheinlichkeit des Gesamtsystems? Geben Sie eine Formel an. (5 Pkte)

Das Gesamtsystem fällt aus, wenn nicht alle drei intakt sind. Also ist die Ausfallwahrscheinlichkeit

$$P_{sys} = 1 - (1-P_G)(1-P_G)(1-P_2)$$

Dies lässt sich auch über eine Zustandstabelle erschließen der Pärchen G_1, G_2 und der Großen Platte G_R. Eine Eins bedeutet „intakt“, eine Null „Ausfall“.

G_1	G_2	G_R	def
0	0	0	1
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	0

Dies bedeutet eine Ausfallwahrscheinlichkeit von

$$P_{sys} = P_G P_G P_2 + P_G P_G (1-P_2) + P_G (1-P_G) P_2 + P_G (1-P_G)(1-P_2) + (1-P_G) P_G P_2 + (1-P_G) P_G (1-P_2) + (1-P_G)(1-P_G) P_2$$

(c) Angenommen, die vier Platten bilden ein einheitliches 4-faches Spiegelsystem, in dem jede Platte identisch zu den anderen drei gehalten wird. Wie groß ist dann die Ausfallwahrscheinlichkeit des 4-fach-Systems sowie des Gesamtsystems? Geben Sie Formeln an.

(5 Pkte)

Das Untersystem aus 4 Platten ist nur dann ausgefallen, wenn alle vier Platten defekt sind, also

$$P_4 = P_1^4$$ (3 Pkte)

Das Gesamtsystem ist ausgefallen, wenn eines der beiden (oder beide) Systeme, große Platte bzw. 4-Subsystem, ausgefallen sind:

$$P_{sys} = P_4 (1 - P_2) + (1 - P_4) P_2 + P_4 P_2$$

oder wenn nicht alle funktionieren

$$P_{sys} = 1 - (1 - P_2)(1 - P_4)$$ (2 Pkte)

Aufgabe 8: Datenstrukturen im Netzwerk *optional* **15 Punkte**

Zwei Personen A und B arbeiten gleichzeitig an einem Dokument. Beide wollen sowohl lesend als auch schreibend auf dieselbe Datei zugreifen.

(a) Welchen Zustand hat die Datei nach der Arbeit (beide öffnen die Datei, nehmen Änderungen vor und schließen die Datei), wenn das Netzdateisystem als Zugriffssemantik...
- *i) ...die Operationssemantik benutzt.*
- *ii) ...die Sitzungssemantik benutzt*
- *iii) ...die Transaktionssemantik benutzt*

(3 Pkte)

(i) Nach jeder Aktion von A bzw. B wird die Datei sofort verändert. Nach Schließen der Datei ist sie in dem Zustand, der einer bunten Mischung aller Aktionen entspricht, wobei die Reihenfolge stark zufallsabhängig ist.

(ii) je nachdem, ob A oder B als letzter abschließt, sind ausschließlich nur die Änderungen entweder von A oder von B enthalten.

(iii) je nachdem, ob A oder B zuerst Zugriff hatte, sind ausschließlich nur die Änderungen entweder von A oder von B enthalten. Allerdings hatte im Unterschied zu (ii) nur einer von beiden Zugriff, so dass der andere bemerkt, dass seine Aktionen obsolet sind.

(b) Beschreiben Sie bei welcher dieser Zugriffssemantiken es zu ungewollten Resultaten kommen kann und wann ggf. Nutzer den Zugriff verwehrt bekommen. (2 Pkte)

Ungewollte Resultate kann es nur bei (i) und (ii) geben, da die Interferenz der Nutzer unbemerkt bleibt. Bei (iii) wird dem anderen Nutzer der Zugriff verwehrt, so dass dies von ihm bemerkt wird und keine ungewollten Resultate entstehen können.

(c) Angenommen, die Änderungen an den Daten werden über ein Netzwerk ausgeführt. Nennen Sie jeweils ein Beispiel für ein zustandsbehaftetes und zustandsloses Netzprotokoll.

(2 Pkte)

Zustandslos: http, NFS, UDP

Zustandsbehaftet: FTP, Sockets, TCP,

(d) Welche Aufräum- und Wiederherstellungsarbeiten müssen durchgeführt werden, wenn bei einem zustandsbehafteten Netzprotokoll...

i) ... der Client abstürzt. *(3 Pkte)*

Neuer Client muss alte, nicht mehr benutzte Puffer, Lockfiles und Dateideskriptoren, falls noch existent (z.B. auf Platte), löschen bzw. deallozieren.
Danach muss eine neue Verbindung aufgebaut werden (Puffer allozieren etc.) und die Anfrage erneut gestellt werden.
Beim Server muss ein *time-out* bewirken, dass die Verbindung abgebrochen, Puffer dealloziert, lock-files gelöscht usw. werden.

ii) ... der Server abstürzt. *(3 Pkte)*

Wie im Fall (i), nur sind die Rollen symmetrisch vertauscht: der Client bemerkt über *time-out* den Absturz des Servers und muss nach dem Beenden der Verbindung (Deallocation etc.) erneut eine Verbindung aufbauen und sein Anliegen vorbringen.
Der Server muss seinen Dauerspeicher (Festplatte) nach Lockfiles, temporärem Speicher etc. absuchen und dieses entsorgen.

(e) Wie kann man trotz eines zustandslosen Netzprotokolls die Anfragen von verschiedenen Clients auf der Serverseite eindeutig unterscheiden? Die Anfragen kommen alle von der gleichen IP-Adresse, da sie einen gemeinsamen Router benutzen. Gehen Sie davon aus, dass die Anfragen in unterschiedlichen zeitlichen Abständen eintreffen. Nennen Sie hier mindestens zwei verschiedene Konzepte. *(2 Pkte)*

Zur Unterscheidung muss eine Verbindungsinformation beim Client gespeichert werden. Beispielsweise geht dies über

(i) Cookies: Ein Informationsstück wird beim Client gespeichert, das bei Bedarf vom Server abgefragt werden kann.

(ii) Sitzungsnummer: Bei jeder Anfrage und Rückantwort wird ein Sitzungsschlüssel mitgeschickt, der als Argument den Client identifiziert.

Natürlich kann man mit MAC-Adressen des Senders und anderen low-level-Konstanten (BS-Nummer, CPU-ID, etc.) eine Identität herstellen, aber dies verlangt einen Zugriff auf Ressourcen des Rechners, die nicht jedem Clientprogramm möglich sind.