Hochschule Ravensburg-Weingarten
Schriftliche Prüfung Betriebssysteme

Prof. Dr. M. Zeller

Datum, Zeit	20. Juli 2007, 08:00 – 09:30 Uhr (90 min)
Aufgabenblätter	13 Seiten (einschl. Deckblatt)
erreichbare Punktzahl	67
zugelassene Hilfsmittel	A (s. Prüfungsplan)

Studiengang	Prf. Nr.	Raum
AI	1825	H039/1
AI	3618	H039/1

Name: _____ Matrikelnummer: _____

Hinweise:

- Schreiben Sie bitte Name und Matrikelnummer auf jedes Aufgabenblatt.
- Schreiben Sie Ihre Lösung zu den Aufgaben auf den freien Platz, direkt anschließend an die Fragestellungen. Wenn Sie zusätzliche Blätter verwenden, so schreiben Sie bitte Name und Matrikelnummer auf jedes Blatt.
- Schreiben Sie lesbar!

Falls Sie es wünschen, dass Ihr Prüfungsergebnis auf einer Liste mit Matrikelnummern und Zensuren ausgehängt bzw. per Internet veröffentlicht wird, unterschreiben Sie bitte folgende Erklärung.

Ich bin damit einverstanden, dass mein Klausurergebnis auf diese Weise veröffentlicht wird.

Unterschrift: _____

Bitte haben Sie dafür Verständnis, dass aus Gründen das Datenschutzes keine telefonischen Auskünfte gegeben werden können.

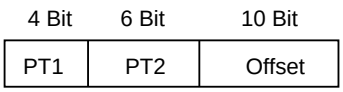
Vom Prüfer auszufüllen:

Aufgabe	1	2	3	4	5	6	7	Summe
Max. Punkte	17	7	9	4	6	16	8	67
Punkte								

Aufgabe 1 Virtueller Speicher

Ein Betriebssystem verwendet Paging, um für die verschiedenen Prozesse jeweils einen virtuellen Hauptspeicher zu realisieren. Der virtuelle Speicher wird auf 512 KB Hauptspeicher und 250 KB der Festplatte abgebildet (Swap-Space). Die Gesamtlänge einer Adresse beträgt 20 Bit.

Das Betriebssystem verwendet eine zweistufige Seitentabelle. Die Länge der ersten Seitenadresse (PT1) beträgt 4 Bit; die Länge der zweiten Seitenadresse (PT2) beträgt 6 Bit; die Länge des Offsets beträgt 10 Bit.



Allg. Hinweis: Schreiben Sie bei den folgenden Aufgaben immer den Rechenweg auf, z. B. "Größe des Speicherbereich XY dividiert durch Anzahl Z".

1.1 (8 Punkte)

Ein Prozess belegt folgende Adressbereiche:

Prog. Teil	Adressbereich	Größe in Byte
TextSegment	0 - 135 000	135 001
HeapSegment	135 001 - 152 070	17 070
Stack Segment	998 576 – 1 048 576	50 000

(1 Punkt) Wie viele Einträge hat die Seitentabelle erster Stufe?

Index PT1 4 Bit: $2^4 = 16$

(1 Punkt) Wie viele Einträge hat eine Seitentabelle zweiter Stufe?

Index PT2 6 Bit: $2^6 = 64$

(1 Punkt) Wie groß (in Kilobyte, KB) ist ist eine Seite, wie groß ist eine Kachel?

Offset 10 Bit: $2^{10} = 1024$

(1 Punkt) Wie viele Seiten belegt das StackSegment?

Größe Stack-Segment / Größe einer Seite $50\,000/1024 = 48,828125 \Rightarrow 49$ Seiten

(1 Punkt) Wie viele Seitentabellen zweiter Stufe werden für das StackSegment benötigt?

Anzahl Seiten / Anzahl Einträge in einer Tabelle 2. Stufe $49/64 = 0,762\ldots \Rightarrow$ Es wird *eine* Tabelle 2. Stufe benötigt.

(2 Punkte) Wie viele Seitentabellen zweiter Stufe werden für das Text und das Heap Segment benötigt?

Anzahl Seiten: $152\,071/1024 = 148,5\ldots \Rightarrow 149$ Seiten. Anzahl Seiten / Anzahl Einträge in einer Tabelle 2. Stufe $149/64 = 2,3\ldots \Rightarrow$ Es werden 3 Tabellen 2. Stufe benötigt.

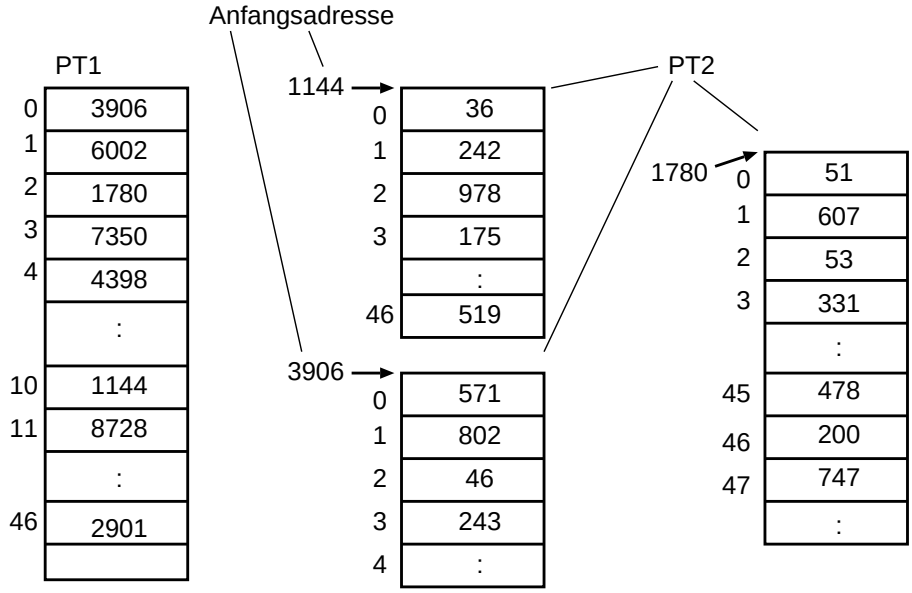
(1 Punkt) Wie viele Kacheln verwaltet das Betriebssystem?

Größe Hauptspeicher / Größe Kachel: $2^{19}/2^{10} = 512$ Kacheln

1.2 (4 Punkte)

Im Weiteren soll eine virtuelle Adresse durch drei Dezimalzahlen für PT1, PT2 und Offset dargestellt werden. Beispiel: Die dezimalen Werte (11, 34, 213) stehen für die virtuelle Adresse 1011 100010 0011010101.

Die folgende Abbildung zeigt einen Ausschnitt aus der Seitentabelle erster Stufe und einige Ausschnitte aus Seitentabellen 2. Stufe. Achtung: In den Seitentabellen 2. Stufe stehen nur die signifikanten Bits, so dass der Offset lediglich angehängt werden muss!



Die physische Adresse soll als *eine* Dezimalzahl dargestellt werden.

Ergänzen Sie die fehlenden Werte in der Tabelle:

virt. Adresse			phys. Adresse
0	2	273	47 377
10	3	881	
2	46	342	
			54 280
			765 051

virt. Adresse			phys. Adresse
0	2	273	47 377
10	3	881	180 081
2	46	342	205 142
2	2	8	54 280
2	47	123	765 051

1.3 (2 Punkte)

Zwei Prozesse teilen sich eine Seite Hauptspeicher (Shared Memory). Jeder Prozess gibt die Anfangsadresse des gemeinsamen Bereichs auf dem Bildschirm aus.

Müssen die beiden Adressen identisch sein, wenn ja, warum?

nein

Können die beiden Adressen identisch sein, wenn nein, warum nicht?

ja

1.4 (3 Punkte)

Prozessoren für Computersysteme besitzen i. Allg. eine Memory Management Unit (MMU). Diese enthält i. Allg. einen Translation Lookaside Buffer (TLB).

Welche Aufgabe hat der TLB?

Er soll zu einer virtuellen Adresse die entsprechende physische Adresse liefern.

Manche TLBs speichern zu jeder Adresse die Process Identification (PID) in ihrem internen Speicher ab, andere nicht.

Welche Vorteil bringt es, wenn der TLB die PID jeweils mit abspeichert?

Der TLB kann gleichzeitig Einträge von unterschiedlichen Prozessen enthalten. Er muss daher bei einem Prozesswechsel nicht vollständig geleert werden.

Ist es sinnvoll, bei Programmen mit mehreren Threads auch die Thread-ID im TLB abzulegen (Begründung!)?

Dies ist nicht sinnvoll, da die verschiedenen Threads einen gemeinsamen Adressraum verwenden. D. h. die Umrechnung läuft für alle Threads des Prozess gleich.

Aufgabe 2 Ersetzungsstrategien (7 Punkte)

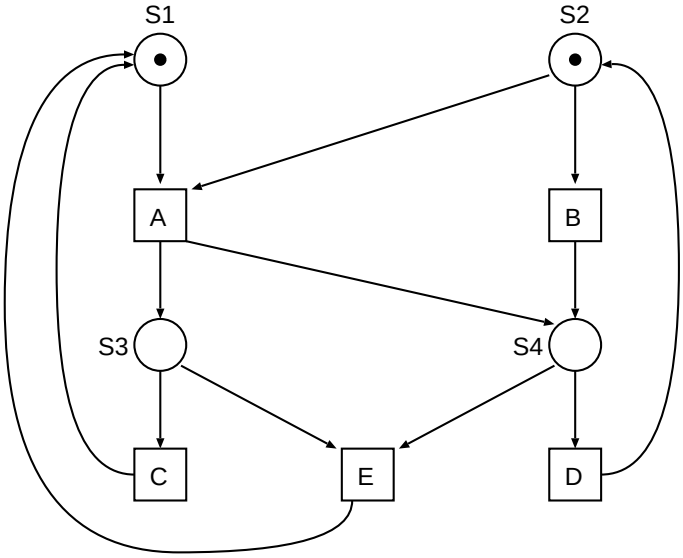
Das Betriebssystem eines Rechners verwaltet einen Hauptspeicher mit 5 Kacheln. Das Betriebssystem verwendet den Clock-Algorithmus. Auf dem System laufen Prozesse mit insgesamt 10 Seiten. Die Seiten der Prozesse werden gemäß der ersten Zeile der folgenden Tabellen referenziert. Wird eine Seite referenziert, so soll die Wirkung in der zugehörigen Spalte dargestellt werden. Bsp.: In der Spalte, in der die Seite 0 referenziert wird, ist dargestellt, dass die Seite 0 in den Hauptspeicher eingelagert wurde. Zahl in Klammern gibt den Wert des R-Bits an, der * Bezeichnet den Zeiger des Clock-Algorithmus.

SeitenNr.	0	1	3	2	5	1	6	3	1	4	7
K1	0(1)	0(1)	0(1)	0(1)	0(1)*	0(1)*	6(1)	6(1)	6(1)	6(1)	6(1)*
K2	– *	1(1)	1(1)	1(1)	1(1)	1(1)	1(0)*	1(0)*	1(1)*	1(0)	1(0)
K3	–	– *	3(1)	3(1)	3(1)	3(1)	3(0)	3(1)	3(1)	3(0)	3(0)
K4	–	–	– *	2(1)	2(1)	2(1)	2(0)	2(0)	2(0)	4(1)	4(1)
K5	–	–	–	– *	5(1)	5(1)	5(0)	5(0)	5(0)	5(0)*	7(1)

Spalte 3, 2, 5 insges. 1 Punkt, jede weitere Spalte ein Punkt.

Aufgabe 3 Synchronisation

Das folgende Petri-Netze zeigt die Synchronisation von 3 Prozessen *P_AC*, *P_BD*, *P_E*. Es handelt sich um ein Bedingungs-Ereignis-Netz. Die Transitionen *A* und *C* gehören zu Prozess *P_AC*, die Transitionen *B* und *D* gehören zu Prozess *P_BD*, die Transition *E* gehört zu Prozess *P_E*.



3.1 (2 Punkte)

Welche Stellen müssen Sie als Semaphor realisieren, um die drei Prozesse gemäß dem obigen Petri-Netz zu synchronisieren?

S1, S2, S3, S4

3.2 (4 Punkte)

Geben Sie den Quell-Code für die Prozesse *P_AC*, *P_BD* und *P_E* an. Sie können dazu PseudoPascal verwenden (s. Skript von Frau Keller) oder (Pseudo)Java.

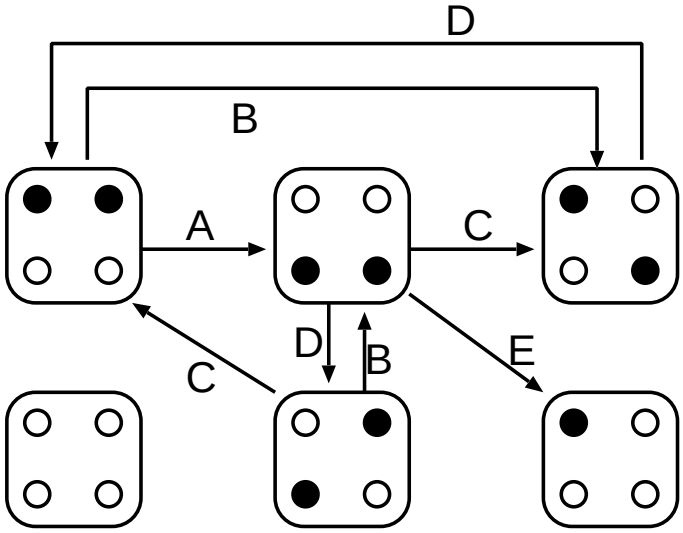
```
Prozess P_AC{
 while(true){
    S1.down();
    S2.down();
    A();
    S3.up();
    S3.down();
    S4.up();
    C();
    S1.up();
  }
}
```

```
Prozess P_BD{
 while(true){
    S2.down();
    B():
    S4.up();
    S4.down();
    D();
    S2.up();
  }
}
```

```
Prozess P_D{
 while(true){
    S3.down();
    S4.down();
    E();
    S1.up();
  }
}
```

3.3 (3 Punkte)

Zeichnen Sie den Ereignisgrafen des Petri-Netzes. Sie können die Vorlage unten verwenden oder eine eigene Skizze anfertigen. Geben Sie zu jedem Übergang die Transition, die ihn auslöste, an.

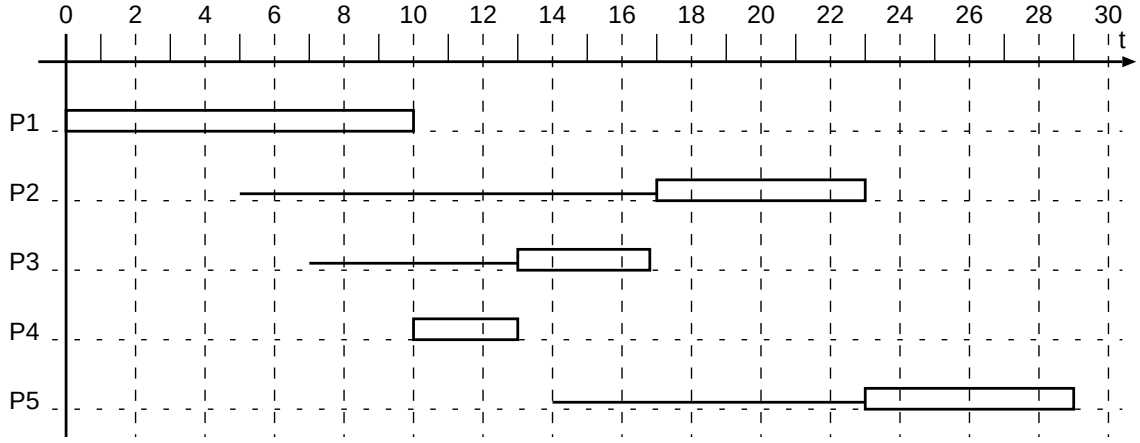


Aufgabe 4 Scheduling (4 Punkte)

Ein Betriebssystem bearbeitet fünf Prozesse. Das System verwendet Scheduling gemäß "shortest remaining time" (auch "least remaining time" genannt). Wenn zwei Prozesse die gleicher Rangstufe besizen, darf der Prozess rechnen, der zuerst eingeplant wurde. Der Scheduler wird immer dann aktiv, wenn ein Prozess gestartet oder beendet wird.

ProzessNr.	Startzeit	Dauer
1	0	10
2	5	6
3	7	4
4	10	3
5	14	6

Tragen Sie die Wartezeiten und die Laufzeiten der Prozesse in das Diagramm ein. Verwenden Sie für Wartezeiten einen Strich: — und für Laufzeiten einen Balken: ▭ oder verwenden Sie unterschiedliche Farben.



Aufgabe 5 FAT Datei System

Ein Dateisystem wird per File Allocation Table (FAT) verwaltet. Ein Block enthält 2 KB, ein Zeiger auf einen Block enthält 4 Byte. Verzeichnisse sind normale Dateien, die zu jeder verwalteten Datei einen Eintrag enthalten (s. Abb. 1). Ein Eintrag besteht aus dem Namen, den Attributen und einem Verweis auf den ersten Block der Datei. Die Länge eines Eintrags beträgt 128 Byte. Ein neuer Eintrag wird stets an das Ende der Verzeichniss-Datei geschrieben.

Ein Programm möchte die Datei `D:\Tmp\abc.log` einlesen. Die Blockadresse des Verzeichnis `D:\Tmp` ist 3. In diesem Verzeichnis waren bereits 18 Dateien enthalten, als die Datei `abc.log` angelegt wurde. Nehmen Sie an, die unten abgebildete FAT und das Verzeichnis `D:` ist bereits in den Hauptspeicher geladen.

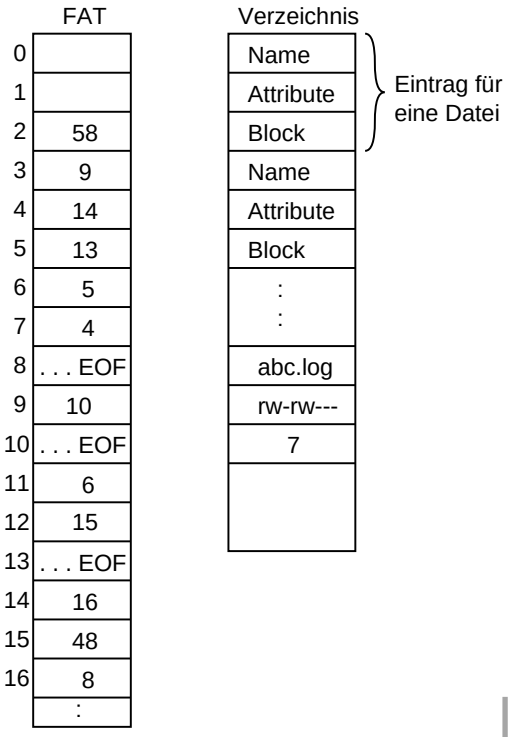


Abbildung 1: FAT und Verzeichniss

5.1 (3 Punkte)

Welche Plattenblöcke muss das Programm einlesen, um die Datei zu lesen?

Blöcke 3, 9, (10) für das Verzeichnis und 7, 4, 14, 16, 8 für die Datei.

5.2 (1 Punkte)

Wie groß kann die Platte bzw. Partition für `D:` maximal sein?

Blockadress: 4 Byte = 32 Bit. 2^{32} Blöcke * 2 KB pro Block = 2^{43} = 8 TB

5.3 (2 Punkte)

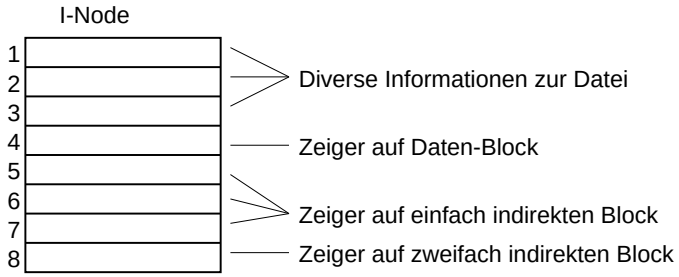
Wie groß ist die FAT, wenn eine Partition mit 2 GB verwaltet werden soll.

2GB / 2 KB pro Block = 2^{20} Blöcke. 2^{20} * 4 Byte pro Block = 4 MB

Aufgabe 6 Datei System mit I-Nodes

Ein Dateisystem verwendet I-Nodes für die Verwaltung von Dateien. Für die Freispeicherverwaltung von I-Nodes und Blöcken verwendet das System je eine Bitmap.

Ein I-Node des Systems besitzt folgendes Format:



Die Daten sind also über einen direkten Block, drei einfach indirekte Blöcke und einen zweifach indirekten Block erreichbar. Ein Block enthält 2 KB, ein Zeiger auf einen Block enthält 3 Byte. I-Nodes enthalten nie selbst Daten einer Datei.

6.1 (3 Punkte)

Wie groß kann eine Datei in diesem Dateisystem maximal sein? Bitte geben Sie alle Rechenschritte an.

2 KB pro Block/ 3 Byte pro Zeiger = 682 Zeiger pro Block.
Anzahl Blöcke: $1 + 3 * 682 + 682^2 = 467171$ Blöcke = 956766208 Byte (ca. 912,44 MB)

6.2 (2 Punkte)

Wie groß kann das Dateisystem maximal sein (Begründung)?

3 Byte pro Zeiger →Es können max. 2^{24} Blöcke adressiert werden.
2^{24} Blöcke*2 KB/Block = 2^{35} Byte (32 GB)

6.3 (3 Punkte)

Wieviel Blöcke belegt eine Datei, die 6 MB Daten enthält. Berücksichtigen Sie *nicht* den Platz, der im Datei-Verzeichnis (Directory) belegt wird und ebenfalls *nicht* den Platz, der durch den I-Node belegt wird.

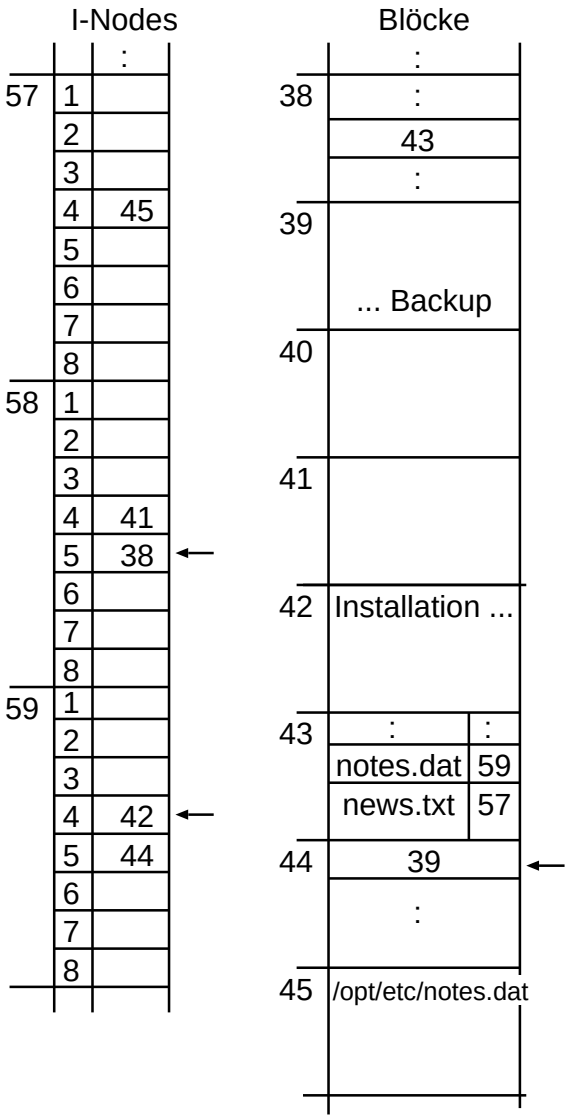
Dateigröße/Blockgröße = 3072 Blöcke. 1 direkter Block; Rest 3071. Drei einfach indirekt adressierte Blöcke = 2046 Datenblöcke; Rest 1025 Blöcke. Ein zweifach indirekter Block und zwei einfach indirekte Blöcke: 682 Blöcke und 343 Blöcke.

3078: Es sind zusätzlich zu den 3072 Blöcke noch 6 Blöcke belegt (5 einfach und 1 zweifach indirekter Block).

Verzeichnisse sind normale Dateien, die zu jeder verwalteten Datei einen Eintrag enthalten. Ein Eintrag besteht aus dem Namen und einem Verweis auf den I-Node der Datei. Die Länge eines Eintrags beträgt 128 Byte. Ein neuer Eintrag wird stets an das Ende der Verzeichniss-Datei geschrieben.

Die folgende Abbildung zeigt einen Ausschnitt aus der Reihe der I-Nodes (I-Node 57 58 und 59) und einen Ausschnitt der Reihe der Blöcke des Dateisystems (Block 38 - 44).

Das Verzeichnis `/opt/etc` wird von I-Node 58 verwaltet. Bevor die Datei `notes.dat` angelegt wurde enthielt das Verzeichnis `/opt/etc` bereits 37 Dateien. Die Datei ist 3 KB groß, sie beginnt mit "Installation . . . " und endet mit " . . . Backup".



6.4 (3 Punkte)

Ergänzen Sie die Skizze an den mit ← markierten Stellen.

6.5 (3 Punkte)

Ein symbolischer Link wird innerhalb des Verzeichnis angelegt angelegt, so dass die Datei `notes.dat` auch über den Namen `news.txt` zugreifbar ist.

Tragen Sie die Änderungen in die Skizze ein. Verwenden Sie nur I-Nodes und Blöcke, die bereits in der Skizze vorhanden sind.

6.6 (2 Punkte)

Wenn ein Prozess eine Datei öffnet, werden die Daten des I-Nodes in eine Tabelle im Hauptspeicher geladen. Ein Kommilitone, der zusammen mit Ihnen ein neues Dateisystem entwirft, macht folgenden Vorschlag: Die Tabelle im Hauptspeicher kann vergößert werden, da Hauptspeicher immer billiger wird. Dann muss beim Öffnen einer Datei nicht in der Tabelle gesucht werden, ob der I-Node schon vorhanden ist; es wird einfach ein Eintrag mit dem I-Node in der Tabelle angelegt, egal ob dieser I-Node schon in der Tabelle enthalten ist oder nicht.

Wie stehen Sie zu diesem Vorschlag?

Duplikate von I-Nodes in der Tabelle können zu Inkonsistenzen führen, wenn die Größe der Datei oder die Datei-Attribute geändert werden. Wenn z. B. ein Prozess die Datei über den einen I-Node so verändert, dass ein Zeiger im I-Node verändert wird. Ein Prozess, der über eine Kopie des I-Nodes zugreift, würde noch den alten Zeiger-Wert verwenden.

Aufgabe 7 Ein-Ausgabe

Sie brennen ein Dateisystem auf CD. Anschließend schauen Sie sich stichprobenartig einige Dateien auf der CD an.

7.1 (4 Punkte)

Ist der CD-Brenner eine blockorientiertes oder ein zeichenorientiertes Gerät (Begründung)?

Beim Brennen arbeitet der CD-Brenner zeichenorientiert. Alle Daten müssen in einem Strom geschrieben werden, der Brenner kann nicht vor- oder zurückspringen.

Beim Lesen arbeitet der CD-Brenner blockorientiert. Das Datei-System 9660 erlaubt es, auf einzelne Blöcke wahlfrei zuzugreifen.

7.2 (2 Punkte)

Sie überarbeiten eine Treiber für ein CD-Laufwerk. Immer, wenn der Treiber einen Datensatz anfordert, fängt er an, regelmäßig abzufragen, ob die Daten schon im Controller sind. Kennen Sie eine geschicktere Methode, um die Eingabe abzuwickeln (stichwortartige Beschreibung)?

Interruptgesteuert: Der Treiber gibt einen Leseauftrag an den Controller des CD-Laufwerks. Der Controller erzeugt einen Interrupt, wenn die Daten zur Verfügung stehen. Die Interruptroutine übergibt die Daten an das aufrufende Programm.

7.3 (2 Punkte)

Was versteht man unter Memory-Mapped I/O? Was ist die Alternative dazu?

Memory-Mapped: Ein-/Ausgabe-Geräte werden über spezielle Adressen angesprochen.

Alternativ dazu können Ein-/Ausgabe-Geräte auch über spezielle Assembler-Befehle angesprochen werden.