Hochschule Ravensburg-Weingarten
Schriftliche Prüfung Betriebssysteme

Prof. Dr. M. Zeller

Datum, Zeit	5. Februar 2008, 14:00 – 15:30 Uhr (90 min)
Aufgabenblätter	12 Seiten (einschl. Deckblatt)
erreichbare Punktzahl	62
zugelassene Hilfsmittel	A (s. Prüfungsplan)

Studiengang	Prf. Nr.	Raum
AI	3618	H061

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!

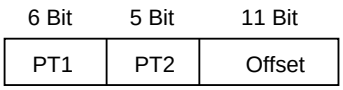
Vom Prüfer auszufüllen:

Aufgabe	1	2	3	4	5	6	Summe
Max. Punkte	14	5	3	13	7	16	62
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 1 MB Hauptspeicher und 2 MB der Festplatte abgebildet (Swap-Space). Die Gesamtlänge einer Adresse beträgt 22 Bit.

Das Betriebssystem verwendet eine zweistufige Seitentabelle. Die Länge der ersten Seitenadresse (PT1) beträgt 6 Bit; die Länge der zweiten Seitenadresse (PT2) beträgt 5 Bit; die Länge des Offsets beträgt 11 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 - 245 000	245 001
HeapSegment	245 001 - 362 070	117 070
Stack Segment	4 051 305 – 4 194 304	143 000

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

Index PT1 6 Bit: $2^6 = 64$

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

Index PT2 5 Bit: $2^5 = 32$

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

Offset 11 Bit: $2^{11} = 2048$

(1 Punkt) Wie viele Seiten belegt das StackSegment?

Größe Stack-Segment / Größe einer Seite $143\,000/2048 = 69,82\ldots \Rightarrow 70$ Seiten

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

Anzahl Seiten / Anzahl Einträge in einer Tabelle zweiter Stufe $70/32 = 2,18\ldots \Rightarrow$ Es werden *drei* Tabellen zweiter Stufe benötigt.

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

Anzahl Seiten: $362\,071/2048 = 176,75\ldots \Rightarrow 177$ Seiten. Anzahl Seiten / Anzahl Einträge in einer Tabelle zweiter Stufe $177/32 = 5,53\ldots \Rightarrow$ Es werden 6 Tabellen zweiter Stufe benötigt.

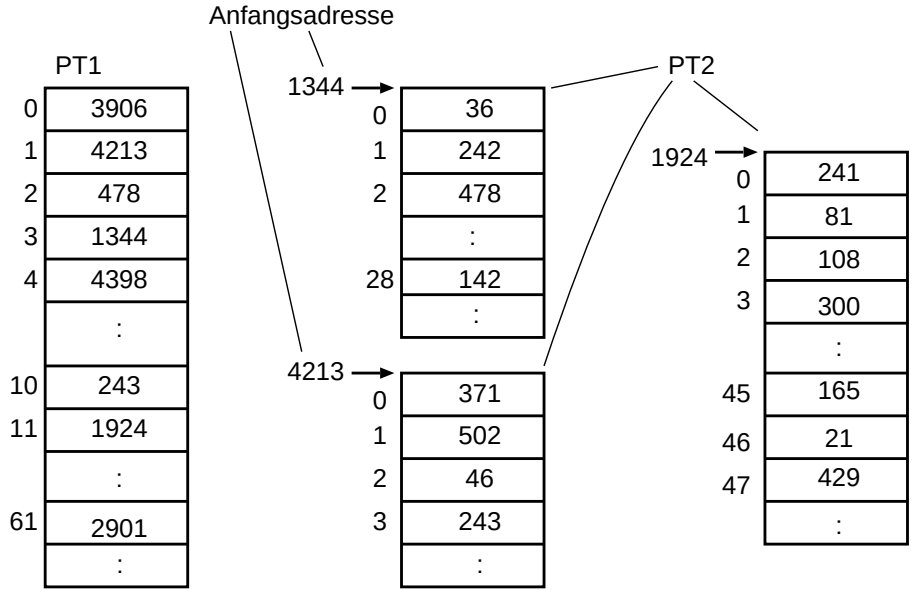
(1 Punkt) Wie viele Kacheln verwaltet das Betriebssystem?

Größe Hauptspeicher / Größe Kachel: $2^{20}/2^{11} = 2^9$ Kacheln (512).

1.2 (6 Punkte)

Im Weiteren soll eine virtuelle Adresse durch drei Dezimalzahlen für PT1, PT2 und Offset dargestellt werden. Beispiel: Die dezimalen Werte (11, 27, 213) stehen für die virtuelle Adresse 001011 11011 00011010101.

Die folgende Abbildung zeigt einen Ausschnitt aus der Seitentabelle erster Stufe und einige Ausschnitte aus Seitentabellen zweiter Stufe. Achtung: In den Seitentabellen zweiter 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 soweit möglich. Wenn Sie einen Wert nicht eintragen können, so begründen Sie dies bitte stichwortartig:

virt. Adresse	phys. Adresse
3 2 123	979 067
1 2 381	
2 3 459	
11 46 342	
	614 484
	73 827
	223 322

	virt. Adresse			phys. Adresse
	3	2	123	979 067
	1	2	381	94 589
X	2	3	459	- - - - -
	11	46	342	43 350
	11	3	84	614 484
	3	0	99	73 827
Y	- -	- -	- -	223 322

X: Es ist keine PT2 an Adresse 478 angegeben.
Y: die Kachelnummer 109 taucht in keiner der angegebenen PT2 auf.

Aufgabe 2 Funktionsaufruf (5 Punkte)

Ein Compiler verwendet nur den Stack, um Daten zwischen verschiedenen Funktionen eines Programms auszutauschen. Folgendes Programm ist gegeben:

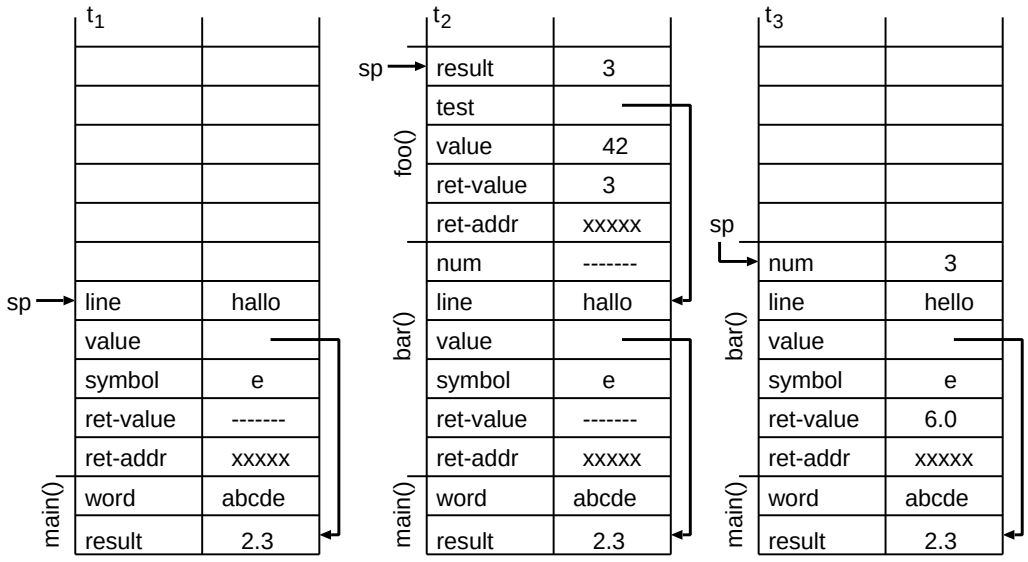
```
1  int foo (char test[ ], int value) {
2     int result = 3;
3     if (test[0] == 'X'){
4        result = value - 1;
5     }
6     return result;
7  }
8
9  float bar (float *value, char symbol){
10    char line[] = "Hallo";
11    int num = foo (line, 42);
12    line[1] = symbol;
13    return num * 2.0;
14 }
15
16 int main (void){
17    char word[ ] = "abcde";
18    float result = 2.3;
19    result = bar(&result, word[4]) ;
20    return result;
21 }
```

Ergänzen Sie die Skizze des Stacks auf der nächsten Seite zu folgenden Zeitpunkten:

- t_1 Zeile 10 unmittelbar nach der Zuweisung zu `line`
- t_2 Zeile 6 unmittelbar vor der Anweisung `return`
- t_3 Zeile 13 unmittelbar vor der Anweisung `return`

Verwenden Sie dabei folgende Symbole:

- $\longrightarrow$ Pointer
- – – – Variable angelegt aber nicht initialisiert
- xxx Variable besitzt einen unbekannten Wert
- sp $\rightarrow$ Stelle, auf die der Stackpointer zeigt


t1
sp
line
hallo
value
symbol
e
ret-value
----
ret-addr
xxxxx
main()
word
abcde
result
2.3
t2
sp
result
3
test
foo()
value
42
ret-value
3
ret-addr
xxxxx
num
----
line
hallo
bar()
value
symbol
e
ret-value
----
ret-addr
xxxxx
main()
word
abcde
result
2.3
t3
sp
num
3
line
hello
bar()
value
symbol
e
ret-value
6.0
ret-addr
xxxxx
main()
word
abcde
result
2.3

Aufgabe 3 Ersetzungsstrategien (7 Punkte)

Das Betriebssystem eines Rechners verwaltet einen Hauptspeicher mit 5 Kacheln. Das Betriebssystem verwendet den Aging-Algorithmus mit einer Zählerbreite von 4 Bit. Auf dem System laufen Prozesse mit insgesamt 10 Seiten. Die Seiten der Prozesse werden gemäß der ersten Zeile der folgenden Tabellen referenziert. Ein Eintrag "AG" bedeutet, dass zu diesem Zeitpunkt die Seiten altern. Wird eine Seite referenziert, so soll die Wirkung in der zugehörigen Spalte dargestellt werden. Bsp.: In der Spalte, in der die Seite 1 referenziert wird, ist dargestellt, dass die Seite 1 in den Hauptspeicher eingelagert wurde. Die Zahl in Klammern gibt den Wert des Zählers an. Die Seite 0 wurde schon vorab in Kachel 0 eingelagert.

Vervollständigen Sie die Werte in der angegebenen Tabelle. Falls Sie größere Korrekturen anbringen müssen, können Sie die zweite Tabelle verwenden.

	Seitenreferenz										
	0	5	AG	6	1	AG	3	0	AG	7	2
K0	0(8)	0(8)	0(4)	0(4)	0(4)	0(2)	0(2)	0(10)	0(5)	0(5)	0(5)
K1	1(8)	1(8)	1(4)	1(4)	1(12)	1(6)	1(6)	1(6)	1(3)	1(3)	1(3)
K2	–	5(8)	5(4)	5(4)	5(4)	5(2)	5(2)	5(2)	5(1)	7(8)	7(8)
K3	–	–	–	6(8)	6(8)	6(4)	6(4)	6(4)	6(2)	6(2)	2(8)
K4	–	–	–	–	–	–	3(8)	3(8)	3(4)	3(4)	3(4)

Aufgabe 4 Synchronisation

Das folgende Petri-Netz zeigt die Synchronisation von 3 Prozessen P_AF, P_BE, P_CD. Es handelt sich um ein Bedingungs-Ereignis-Netz. Die Transitionen A und F gehören zu Prozess P_AF, die Transitionen B und E gehören zu Prozess P_BE, die Transitionen C und D gehören zu Prozess P_CD.

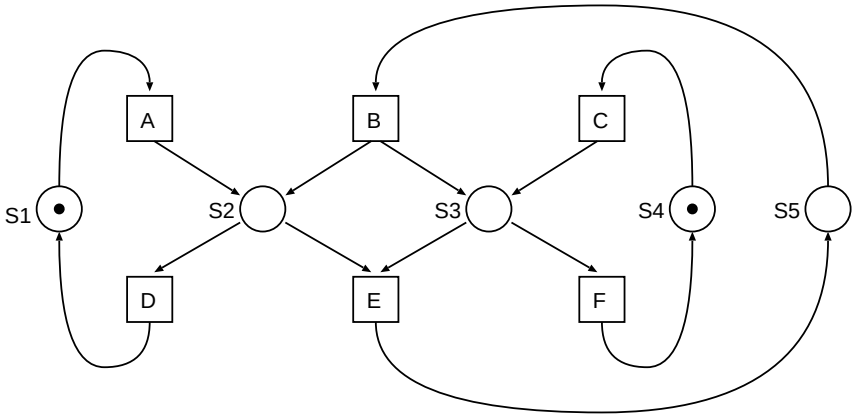


Abbildung 1: Ein paralleles System in Form eines Petri-Netzes

4.1 (2 Punkte)

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

S1, S2, S3, S4

4.2 (4 Punkte)

Geben Sie den Quell-Code für die Prozesse P_AF, P_BE und P_CD an. Sie können dazu PseudoPascal verwenden (s. Skript von Frau Keller) oder (Pseudo)Java.

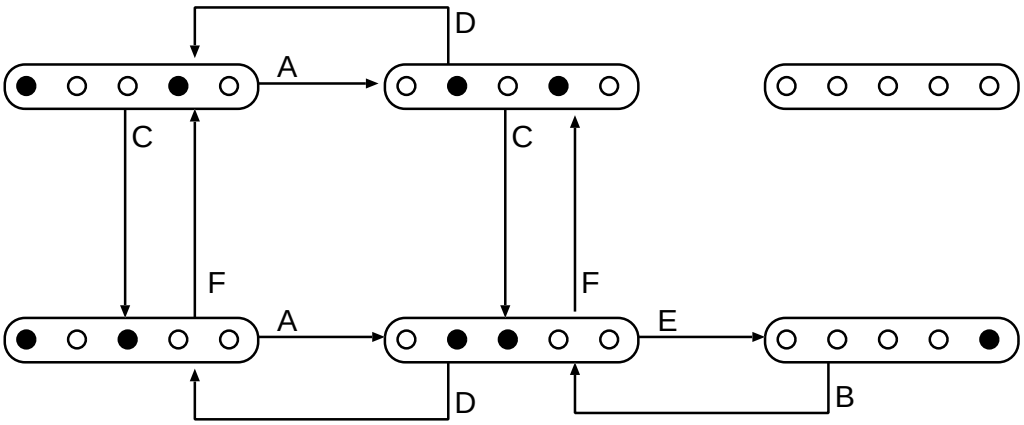
```
Prozess P_AF{
 while(true){
    S1.down();
    A();
    S2.up();
    S3.down();
    F();
    S4.up();
  }
}
```

```
Prozess P_BE{
 while(true){
    B():
    S2.up();
    S3.up();
    S2.down();
    S3.down();
    E();
  }
}
```

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

4.3 (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.



4.4 (2 Punkte)

Kann das System, das in dem oben angegebenen Petri-Netz (s. Abb. 1) dargestellt ist, in einen Deadlock geraten; wenn ja, wie; wenn nein, warum nicht?

Es kann nicht in einen Deadlock geraten; aus jedem Zustand gibt es einen Übergang in einen anderen Zustand.

4.5 Variante

Betrachten Sie nun das System gemäß Abb. 2. Es handelt sich ebenfalls um ein Bedingungs-Ereignis-Netz.

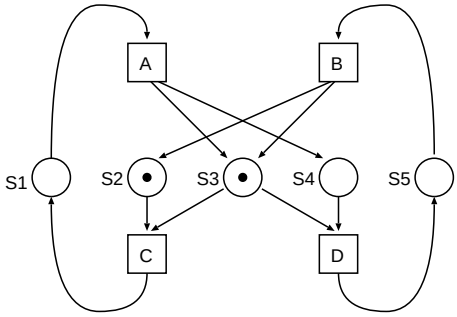


Abbildung 2: Ein paralleles System in Form eines Petri-Netzes

Es ist durch die zwei Prozesse P_AC, P_BD implementiert. Die Semaphore dürfen nur die Werte 0 und 1 annehmen (Bedingungs-Ereignis-Netz).

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

```
Prozess P_BD{
 while(true){
    B():
    S2.up();
    S3.up();
    S3.down();
    S4.down();
    D();
  }
}
```

4.6 (2 Punkte)

Kann die gegebene Implementierung des Systems in Pseudo-Java in einen Deadlock geraten; wenn ja, wie; wenn nein, warum nicht?

Ja, wenn der Prozess P_BD die Anweisungen `S2.up(); S3.up(); S3.down();` ohne Unterbrechung durchläuft.

4.7 (2 Punkte)

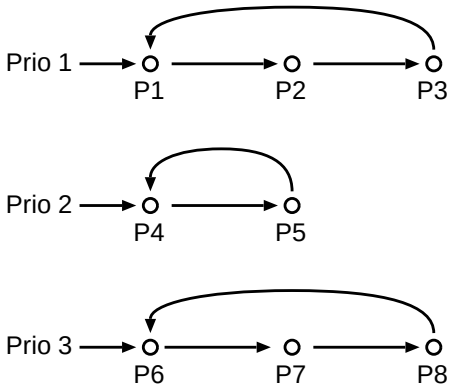
Kann das Systems in C mit Hilfe der Semaphore aus der C-Standard-Bibliothek so implementiert werden, dass es nicht in einen Deadlock geraten kann; wenn ja, wie; wenn nein, warum nicht? Achtung: Es genügt eine stichwortartige Antwort als Begründung.

Ja, die Semaphore der C-Standard-Bibliothek können zu Gruppen zusammengefasst werden, so dass alle Semaphore einer Gruppen in einer atomaren Aktion manipuliert werden. Wenn die Semaphore S2, S3 und S4 zu einer Gruppe zusammengefasst werden, kann der Deadlock nicht auftreten. Anm.: Es wäre sicherlich sinnvoll, alle benötigten Semaphore (S2, S3, S4 und S5) zu einer Gruppe zusammenzufassen; notwendig ist allerdings nur die Gruppierung von S2, S3 und S4.

Aufgabe 5 Scheduling

Ein Betriebssystem verwendet preemptive Multitasking mit einer Kombination aus Round-Robin und prioritätsbasiertem Scheduling. Es gibt drei Prioritätsstufen: 1, 2 und 3, wobei die Stufe 1 die höchste Priorität darstellt und die Stufe 3 die niedrigste. Der Scheduler wird aktiv, wenn eine Zeitscheibe abläuft oder wenn ein Prozess blockiert wird. Das System verwaltet 8 Prozesse (P1 ... P8).

Ein Prozess ist rechnend (RE), einige Prozesse sind bereit (BR) einige sind blockiert (BL).



5.1 (7 Punkte)

Die folgende Tabelle soll die Zustände der Prozesse zu verschiedenen Zeitpunkten darstellen. Der Zustand zum Zeitpunkt t_0 ist gegeben. Es treten nun der Reihe nach Ereignisse auf. Mit dem Ausdruck "Prozess P7 wird deblockiert" ist gemeint, dass die Ursache für die Blockade aufgehoben ist. Z. B. weil eine Aus- oder Eingabe des Prozess beendet wurde.

Ergänzen Sie die Tabelle gemäß den auftretenden Ereignissen.

Zeitpunkt: Ereignis

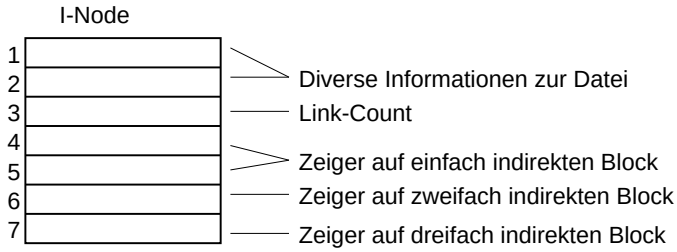
- t_0: Ein Zeitscheibe für das preemptive Multitasking läuft ab
- t_1: Prozess P7 wird deblockiert
- t_2: Prozess P1 wird deblockiert
- t_3: Prozess P2 wird blockiert
- t_4: Prozess P1 wird blockiert
- t_5: Prozess P5 wird blockiert
- t_6: Prozess P4 wird blockiert
- t_7: Ein Zeitscheibe für das preemptive Multitasking läuft ab

			t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7	
P1	BL	BL	BL	BL	BL	BR	RE	BL	BL	BL	BL
P2	BL	BL	BR	RE	RE	RE	BL	BL	BL	BL	BL
P3	BL	BL	BL	BL	BL	BL	BL	BL	BL	BL	BL
P4	BR	RE	RE	BR	BR	BR	BR	BR	RE	BL	BL
P5	BR	BR	BR	BR	BR	BR	BR	RE	BL	BL	BL
P6	RE	BR	BR	BR	BR	BR	BR	BR	BR	BR	BR
P7	BL	BL	BL	BL	BR	BR	BR	BR	BR	RE	BR
P8	BR	BR	BR	BR	BR	BR	BR	BR	BR	BR	RE

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 zwei indirekte Blöcke, einen zweifach indirekten Block und einen dreifach indirekten Block erreichbar. Ein Block enthält 512 Byte, ein Zeiger auf einen Block enthält 4 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.

2^9Byte pro Block/ 4 Byte pro Zeiger = 128 Zeiger pro Block.
Anzahl Blöcke: $2 * 128 + 128^2 + 128^3 = 2113792$ Blöcke = 1082261504 Byte (ca. 1 GB)

6.2 (2 Punkte)

Wie groß kann das Dateisystem maximal sein (Begründung)?

4 Byte pro Zeiger →Es können max. 2^{32} Blöcke adressiert werden.
2^{32} Blöcke * 2^9 Byte/Block = 2^{41} Byte (ca. 2 TB)

6.3 (3 Punkte)

Wie viele Blöcke belegt eine Datei, die 10 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.

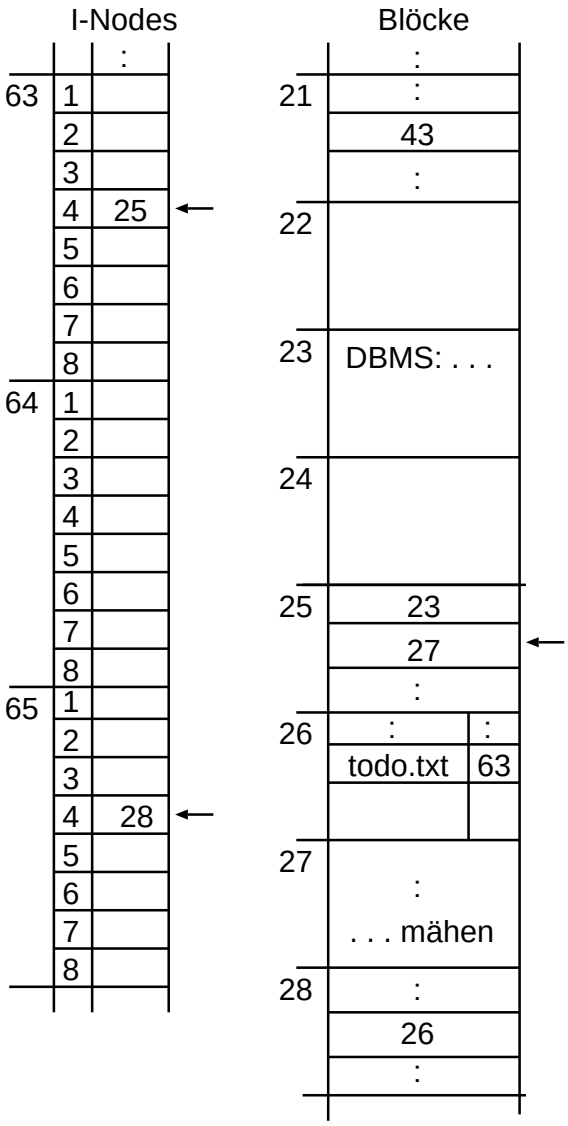
Dateigröße/Blockgröße: $10 * 2^{20}/2^9 = 10 * 2^{11}$ (20480) Blöcke für Daten. Insgesamt 160 einfach indirekte Blöcke. Zwei werden direkt aus dem I-Node adressiert (Rest 158), 128 weitere werden über den zweifach indirekten Block adressiert (Rest 30), die letzten 30 werden über den dreifach indirekten Block – mit einem weiteren zweifach indirekten Block – adressiert.

Es werden also benötigt: 20480 Blöcke für Daten, 160 einfach indirekte Blöcke, zwei zweifach indirekte Blöcke und einen dreifach indirekten Block. Insgesammt: 20480 + 160 + 2 + 1 = 20643

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 64 Byte. Ein neuer Eintrag wird stets an das Ende der Verzeichniss-Datei geschrieben.

Die nebenstehende Abbildung zeigt einen Ausschnitt aus der Reihe der I-Nodes (I-Node 63, 64 und 65) und einen Ausschnitt der Reihe der Blöcke des Dateisystems (Block 21 – 28).

Das Verzeichnis `/home/fritz` wird von I-Node 65 verwaltet. Bevor die Datei `todo.txt` angelegt wurde enthielt das Verzeichnis `/home/fritz` bereits 20 Dateien. Die Datei ist 943 Byte groß, sie beginnt mit "DBMS: Aufgabe 3..." und endet mit " ...Rasen mähen". Das Dateisystem enthält zunächst keine Links.



6.4 (3 Punkte)

Ergänzen Sie die Skizze an den mit ← markierten Stellen.

6.5 (3 Punkte)

Ein symbolischer Link wird innnerhalb des Verzeichnis angelegt angelegt, so dass die Datei `todo.txt` auch über den Namen `morgen.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)

Ein Hard-Link wird innnerhalb des Verzeichnis angelegt angelegt, so dass die Datei `todo.txt` auch über den Namen `nie.txt` zugreifbar ist. Tragen Sie die Änderungen in die Skizze ein.