

Hochschule Ravensburg-Weingarten

Schriftliche Prüfung Betriebssysteme

Prof. Dr. M. Zeller

Datum, Zeit 22. Juli 2009, 08:00 – 09:30 Uhr (90 min)
Aufgabenblätter 13 Seiten (einschl. Deckblatt)
erreichbare Punktzahl 67
zugelassene Hilfsmittel A (s. Prüfungsplan)

Studiengang Prof. Nr. Raum
AI 3618 H142

Name: _____

Matrikelnummer: _____

Vorbemerkung Die Klausur ist ziemlich umfangreich. Lassen Sie sich nicht verunsichern, Sie benötigen nicht alle Punkte für die Note 1,0; Sie benötigen weniger als die Hälfte der Punkte für die Note 4,0.

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	Summe
Max. Punkte	19	20	12	6	10	67
Punkte						

Name:

Mat. Nr:

Aufgabe 1 Virtueller Speicher

Ein Betriebssystem verwendet Paging, um für die verschiedenen Prozesse jeweils einen virtuellen Hauptspeicher zu realisieren. Der virtuelle Speicher wird auf 56 MB Hauptspeicher und 8 MB der Festplatte abgebildet (Swap-Space). Die Gesamtlänge einer Adresse beträgt 28 Bit.

Das Betriebssystem verwendet eine dreistufige Seitentabelle.

Die Länge der ersten Seitenadresse (PT1) beträgt 6 Bit; die Länge der zweiten Seitenadresse (PT2) beträgt 5 Bit; die Länge der dritten Seitenadresse (PT3) beträgt 8 Bit; die Länge des Offsets beträgt 9 Bit. Ein Prozess belegt für das Text- und das Heap-Segment stets Adressen von 0 beginnend aufwärts, für das Stack-Segment belegt er Adressen von $2^{27} - 1 = 134\,217\,727$ beginnend abwärts.

6 Bit	5 Bit	8 Bit	9 Bit
PT1	PT2	PT3	Offset

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

1.1 (11 Punkte)

Ein Prozess belegt folgende Adressbereiche:

Prog. Teil	Adressbereich	Größe in Byte
Text-Segment	0 - 417000	417001
Heap-Segment	417001 - 30650070	30233070
Stack-Segment	133951109 - 134217727	266619

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

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

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

(1 Punkt) Wie viele Seiten belegt das Stack-Segment?

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

(2 Punkte) Benötigt das System eine (oder mehrere) weitere Seitentabelle(n) zweiter Stufe für das Stack-Segment oder kann der Stack über die vorhandenen Seitentabellen zweiter

Name:

Mat. Nr:

Stufe adressiert werden (Begründung)?

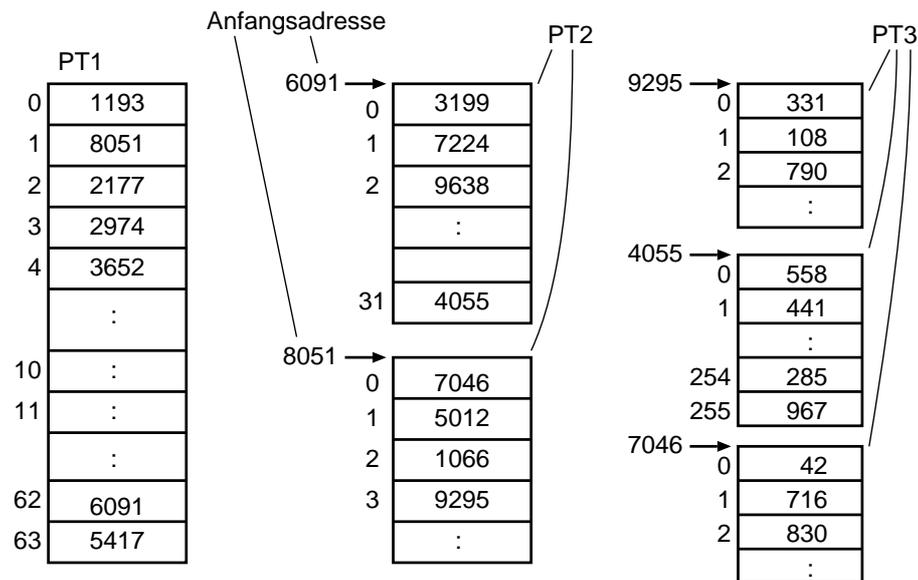
(1 Punkt) Wie viele Kacheln verwaltet das Betriebssystem?

(1 Punkt) Auf welche Größe kann der physische Speicher maximal ausgebaut werden (Hauptspeicher plus Swap-Space)?

1.2 (8 Punkte)

Im Weiteren soll eine virtuelle Adresse durch vier Dezimalzahlen für PT1, PT2, PT3 und Offset dargestellt werden. Beispiel: Die dezimalen Werte (33, 13, 56, 116) stehen für die virtuelle Adresse 100001 01101 00111000 001110100.

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



Die physische Adresse soll in Form einer 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:

Name:

Mat. Nr:

virt. Adresse	phys. Adresse
1 0 2 491	425 451
0 18 244 418	
62 31 254 81	
1 3 0 375	
	366 717
	114 000
	495 367

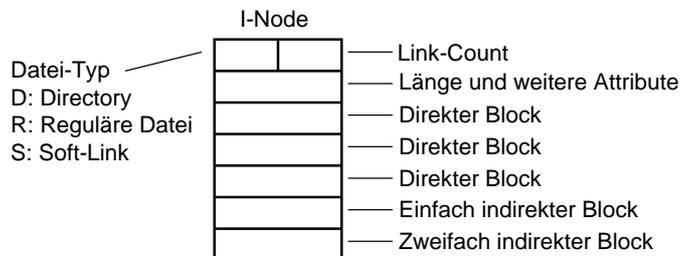
Name:

Mat. Nr:

Aufgabe 2 Datei System mit I-Nodes

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

Ein I-Node des Systems besitzt folgendes Format:



Außer den angegebenen Attributen Die Daten sind also über drei direkte Blöcke, einen einfach indirekten Block und einen zweifach indirekten Block erreichbar. Ein Block enthält 4096 Byte, ein Zeiger auf einen Block enthält 16 Byte. I-Nodes enthalten nie selbst Daten einer Datei.

2.1 (3 Punkte)

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

2.2 (2 Punkte)

Wie groß kann das Dateisystem maximal sein (Begründung)?

2.3 (3 Punkte)

Wie viele Blöcke belegt eine Datei, die 140 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.

Name:

Mat. Nr:

2.4 (8 Punkte)

Verzeichnisse sind 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 nebenstehende Skizze zeigt ein Dateisystem; es gibt in diesem Dateisystem keine anderen Dateien. Die Dateien / (Wurzel-Verzeichnis), etc, users und didi sind Verzeichnisse. Die Datei btx.cip ist ein Hard-Link, der auf hrw.pic verweist. Die Datei notes.txt ist ein Soft-Link, der auf gen.cfg verweist. Die Datei gen.cfg ist 5 KB groß, sie beginnt mit "Do not change ..." und endet mit "... to refresh."

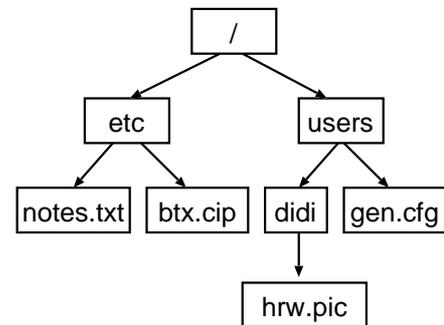


Abb. 1 zeigt alle vom Dateisystem verwendeten I-Nodes und Blöcke sowie einen Ausschnitt der Freispeicherverwaltung. Ergänzen Sie die Skizze an den mit ○ gekennzeichneten Stellen. An Stellen, die keinen definierten Wert besitzen tragen Sie bitte ein Kreuz ein.

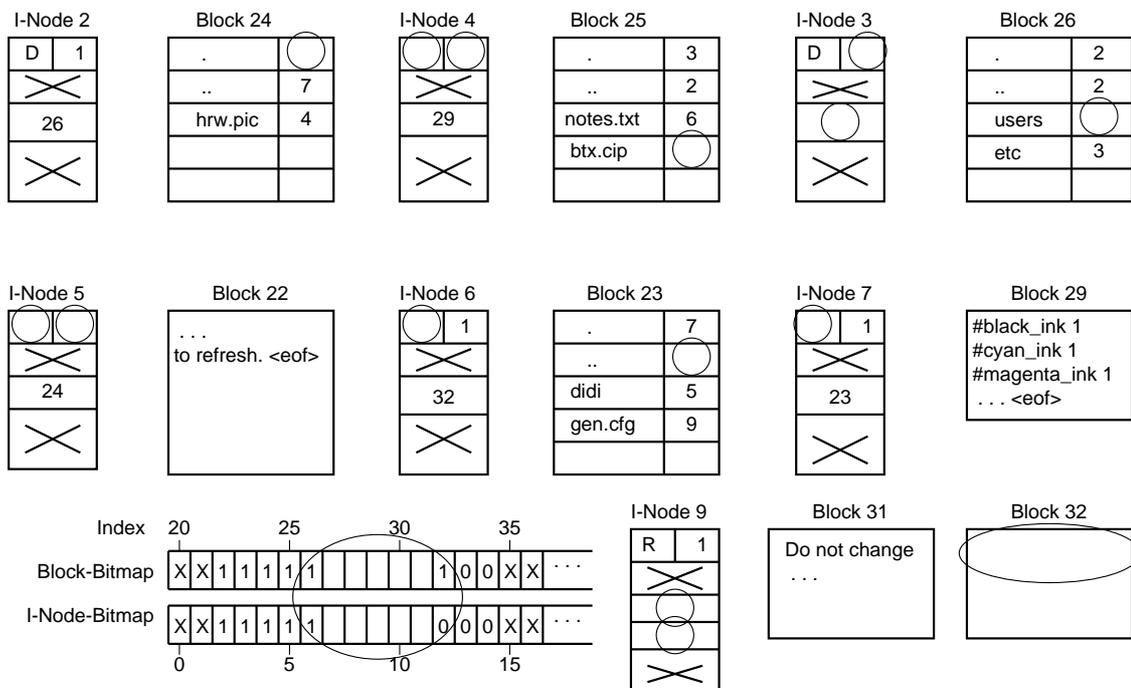


Abbildung 1: Skizze des Dateisystems

Der Eintrag <EOF> bedeutet: Gemäß Längen-Eintrag im I-Node endet die Datei an dieser Stelle. Der erste Eintrag der Freispeicher-Bitmaps bezeichnet den Block 0 bzw. den I-Node 0. Der Wert 1 bedeutet, dass der entsprechende Block bzw. I-Node belegt ist, der Wert 0 bedeutet, dass der entsprechende Block bzw. I-Node frei ist.

2.5 (2 Punkte)

Die Datei gen.cfg wird um 4 KB vergrößert. Welche Änderungen ergeben sich in dem gegebenen Dateisystem? Wenn das System zusätzliche Blöcke verwendet, so sind dies die

Name:

Mat. Nr:

Blöcke 33, 34, 35 Wenn das System zusätzliche I-Nodes verwendet, so sind dies die I-Nodes 10, 11, 12

Was wird angelegt, welche Werte werden wo eingetragen bzw. verändert?

2.6 FAT Datei-System (2 Punkte)

Eine Partition der Größe 1 GB soll durch ein FAT-Datei-System verwaltet werden. Die Größe eines Blocks beträgt 2 KB, ein Zeiger auf einen Block besteht aus 3 Byte

Wie groß ist die FAT für diese Dateisystem?

Name: _____

Mat. Nr: _____

3.2 Optimale Strategie (5 Punkte)

Wie würde die Ersetzung gemäß der optimale Strategie aussehen? Füllen Sie nur die Felder ohne 'X' aus.

SeitenNr.	0	1	2	3	4	5	6	3	7	1	3	6	4	5	2	0
K1	0	0	0	0	0						X	X	X	X	X	X
K2	-	1	1	1	1						X	X	X	X	X	X
K3	-	-	2	2	2						X	X	X	X	X	X
K4	-	-	-	3	3						X	X	X	X	X	X
K5	-	-	-	-	4						X	X	X	X	X	X

Zweite Tabelle für evtl. Korrekturen:

SeitenNr.	0	1	2	3	4	5	6	3	7	1	3	6	4	5	2	0
K1	0	0	0	0	0						X	X	X	X	X	X
K2	-	1	1	1	1						X	X	X	X	X	X
K3	-	-	2	2	2						X	X	X	X	X	X
K4	-	-	-	3	3						X	X	X	X	X	X
K5	-	-	-	-	4						X	X	X	X	X	X

Aufgabe 4 Funktionsaufruf (6 Punkte)

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

```
1 int foxi(int *, char []);
2
3 int fix (char word [], int value) {
4     int result = 1;
5     result = foxi(&value, word);
6     return result * 2;
7 }
8
9 int foxi(int *result, char line []){
10    *result = line [0];
11    line [0] = 68;
12    return 42;
13 }
14
15 int main (void){
16     short result = 0;
17     char line [] = "Lupo";
18     result = fix (line, 10);
19     printf("\n line: %s\n\n", line);
20     return result;
21 }
```

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

- t_1 nach Zeile 17 unmittelbar vor dem Funktionsaufruf `fix(...)`
- t_2 nach Zeile 10 unmittelbar vor der Anweisung `line[0] = 68;`
- t_3 in Zeile 6 unmittelbar vor dem Rücksprung zur aufrufenden Funktion

Verwenden Sie dabei folgende Symbole:

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

Ausschnitt aus der ASCII-Code-Tabelle:

D	L	o	p	u
68	76	111	112	117

Hinweis: Es sind verschiedene Varianten möglich. Achten Sie aber bitte darauf, dass Ihre Lösung in sich konsistent ist.

Name:

Mat. Nr:

Aufgabe 5 Synchronisation

Das folgende Petri-Netz zeigt die Synchronisation von drei Prozessen P_{AC} , P_{BD} und P_E . Es handelt sich um ein Bedingungs-Ereignis-Netz. Die Transitionen A, C gehören zu Prozess P_{AC} , die Transitionen B, D gehören zum Prozess P_{BD} , die Transition E gehört zu Prozess P_E .

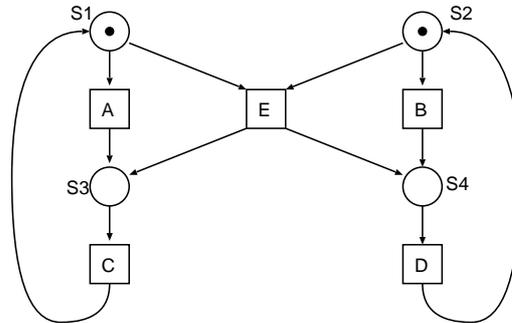


Abbildung 2: Ein paralleles System in Form eines Petri-Netzes

5.1 (2 Punkte)

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

5.2 (3 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} {

Prozess P_{BD} {

}

}

Name:

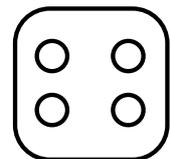
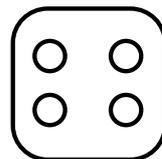
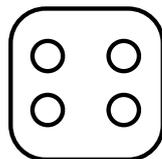
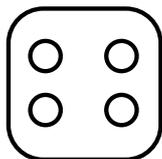
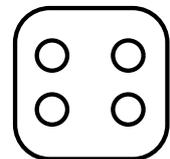
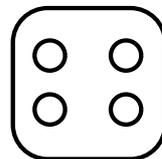
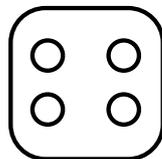
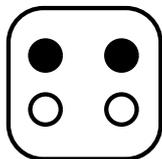
Mat. Nr:

Prozess P_E{

}

5.3 (3 Punkte)

Zeichnen Sie den Ereignisgraphen 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.



5.4 (2 Punkte)

Kann das System, das in dem oben angegebenen Petri-Netz (s. Abb. 2) dargestellt ist, in einen Deadlock geraten; wenn ja, wie; wenn nein, warum nicht?
