

Hochschule Ravensburg-Weingarten
Schriftliche Prüfung Betriebssysteme

Prof. Dr. M. Zeller

Datum, Zeit 4. Juli 2008, 08:00 – 09:30 Uhr (90 min)
Aufgabenblätter 12 Seiten (einschl. Deckblatt)
erreichbare Punktzahl 75
zugelassene Hilfsmittel A (s. Prüfungsplan)

Studiengang Prof. Nr. Raum
AI 3618 C109

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	6	Summe
Max. Punkte	19	25	8	10	3	10	75
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 32 MB Hauptspeicher und 16 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 5 Bit; die Länge der zweiten Seitenadresse (PT2) beträgt 7 Bit; die Länge der dritten Seitenadresse (PT3) beträgt 6 Bit; die Länge des Offsets beträgt 10 Bit.

5 Bit	7 Bit	6 Bit	10 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 (10 Punkte)

Ein Prozess belegt folgende Adressbereiche:

Prog. Teil	Adressbereich	Größe in Byte
TextSegment	0 - 420 000	420 001
HeapSegment	420 001 - 19 738 070	19 318 070
Stack Segment	268 365 945 - 268 435 456	69 512

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

Index PT1 5 Bit: $2^5 = 32$

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

Index PT2 7 Bit: $2^7 = 128$

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

Offset 11 Bit: $2^{10} = 1024$

(1 Punkt) Wie viele Seiten belegt das StackSegment?

Größe Stack-Segment / Größe einer Seite $69\,512/1024 = 67,88... \Rightarrow 68$ Seiten

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

Anzahl Seiten / Anzahl Einträge in einer Tabelle dritter Stufe $68/64 = 1,06... \Rightarrow$ Es werden *zwei* Tabellen dritter Stufe benötigt.

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

Anzahl Seiten: $19\,738\,071/1024 = 19\,275,45... \Rightarrow 19\,276$ Seiten. Anzahl Seiten / Anzahl Einträge in einer Tabelle dritter Stufe $19\,276/64 = 301,17... \Rightarrow$ Es werden 302 Tabellen dritter Stufe benötigt.

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

Anzahl der Tabellen dritter Stufe / Anzahl der Einträge zweiter Stufe: $302/128 = 2,35... \Rightarrow$ Es werden 3 Seitentabellen zweiter Stufe benötigt.

(1 Punkt) Wie viele Kacheln verwaltet das Betriebssystem?

Größe Hauptspeicher / Größe Kachel: $32 * 2^{20} / 2^{10} = 32 * 2^{10}$ Kacheln 32 K.

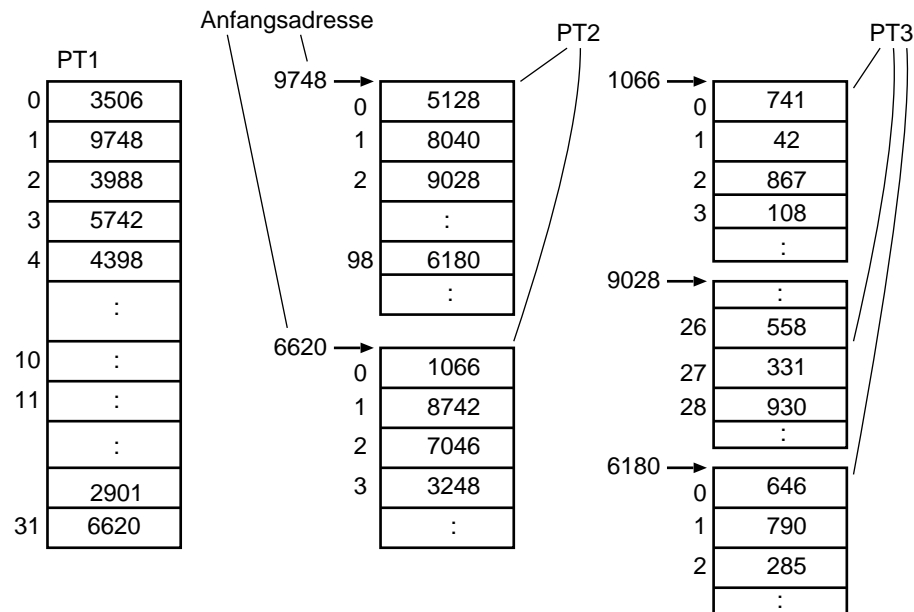
Name:

Mat. Nr:

1.2 (9 Punkte)

Im Weiteren soll eine virtuelle Adresse durch vier Dezimalzahlen für PT1, PT2, PT3 und Offset dargestellt werden. Beispiel: Die dezimalen Werte (11, 27, 33, 213) stehen für die virtuelle Adresse 01011 0011011 100001 00011010101.

Die folgende Abbildung zeigt einen Ausschnitt aus der Seitentabelle erster Stufe und einige Ausschnitte aus 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:

virt. Adresse				phys. Adresse
1	2	28	123	952 443
1	2	26	381	
31	0	3	459	
				661 846
				216 170
1	98	2	503	
2	3	0	967	
				759 574

virt. Adresse				phys. Adresse
1	2	28	123	952 443
1	2	26	381	571 773
31	0	3	459	111 051
1	98	0	342	661 846
X	-	-	106	216 170
1	98	2	503	292 343
Y	2	3	0	- - - -
31	0	0	790	759 574

X: die Kachelnummer 211 taucht in keiner der angegebenen PT3 auf.

Y: Es ist keine PT2 an Adresse 3988 angegeben.

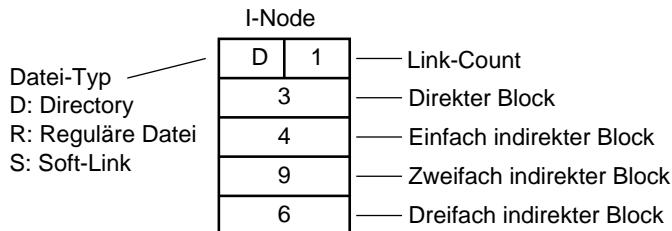
Name:

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 (die Zahlen sind lediglich Beispiele!):



Die Daten sind also über einen direkten Block, einen zweifach indirekten Block und einen dreifach indirekten Block erreichbar. Ein Block enthält 2048 Byte, ein Zeiger auf einen Block enthält 8 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.

$2^1 \text{Byte pro Block} / 8 \text{ Byte pro Zeiger} = 256 \text{ Zeiger pro Block.}$

Anzahl Blöcke: $1 + 256 + 256^2 + 256^3 = 16\,843\,009 \text{ Blöcke} = 34\,494\,482\,432 \text{ Byte (ca. 32 GB)}$

2.2 (2 Punkte)

Wie groß kann das Dateisystem maximal sein (Begründung)?

8 Byte pro Zeiger → Es können max. 2^{64} Blöcke adressiert werden.

$2^{64} \text{ Blöcke} * 2^{11} \text{ Byte/Block} = 2^{75} \text{ Byte.}$

2.3 (3 Punkte)

Wie viele Blöcke belegt eine Datei, die 100 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: $100 * 2^{20} / 2^{11} = 100 * 2^9$ (51200) Blöcke für Daten. Ein direkter Block, 256 Blöcke über den einfach indirekten Block, 50943 über den zweifach indirekten Block. Hier kommen nochmals 199 einfach indirekte Blöcke hinzu. D. h. es werden zusätzlich zu den Datenblöcken 200 einfach indirekte Blöcke und ein zweifach indirekter Block benötigt.

Insgesamt: 51200 Blöcke für Daten, 200 einfach indirekte Blöcke, ein zweifach indirekter Block = 51401 Blöcke.

Name:

Mat. Nr:

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.

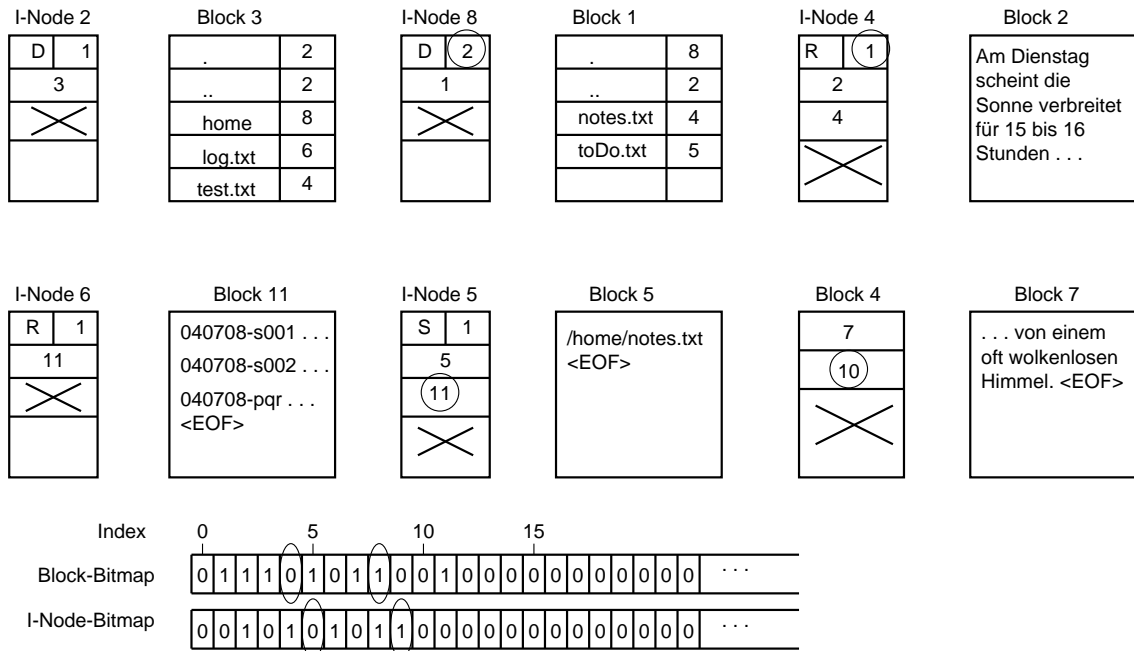


Abbildung 1: I-Nodes und Blöcke des Dateisystems

Abb. 1 zeigt alle vom Dateisystem verwendeten I-Nodes und Blöcke sowie einen Ausschnitt der Freispeicherverwaltung. Der Eintrag <EOF> bedeutet: Gemäß Längeneintrag im I-Node endet die Datei an dieser Stelle. Der erste Eintrag der Freispeicher-Bitmmaps 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.

2.4 (9 Punkte)

Das System enthält einige Inkonsistenzen. Listen Sie die Einträge auf, die fehlerhaft sind. Z. B. Block X, Eintrag Y müsste den Wert Z haben, I-Node-Bitmap, n-te Stelle müsste den Wert k haben.

2.5 (5 Punkte)

Die Datei `log.txt` wird um 2 KB vergrößert. Welche Änderungen ergeben sich in dem gegebenen Dateisystem? Das System verwendet dafür nur Blöcke ab Block 15 einschließlich. Was wird angelegt, welche Werte werden wo eingetragen bzw. verändert?

Neu: Ein Block (z. B. 15) für die zusätzlichen Daten und ein indirekter Block (z. B. 16), der einen Verweis auf den Datenblock (15) enthält.

Änderungen: Im I-Node 6 Verweis auf indirekten Block eintragen (16), Länge der Datei anpassen (+ 2048). In der Block-Bitmap die neu belegten Blöcke als belegt markieren (z. B. Stelle 15 und 16 auf 1 setzen).

Name:

Mat. Nr:

2.6 (3 Punkte)

Eine Datei (XY.Z) besteht aus einem I-Node und drei Datenblöcken. Ein Programm öffnet die Datei mit dem Befehl `fopen("XY.Z", "r+")`. Das Programm liest den ersten Block und überschreibt den Rest der Datei. Nehmen Sie an, dass die Stromversorgung des Rechners ausfällt, während die Datei überschrieben wird. Mit welcher Strategie lässt sich verhindern, dass die Datei in einen inkonsistenten Zustand kommt. Erläutern Sie das Vorgehen ggf. mit einer Skizze.

Stichwort: Copy and Write. Es kann nicht zu einem inkonsistenten Zustand kommen, wenn die neuen Inhalte und ggf. Änderungen an indirekten Blöcken in neue Blöcke geschrieben werden. Zuletzt muss die Wurzel der Datei in einer atomaren Aktion ersetzt werden. Z. B. indem eine neue Version erstellt wird, die zuletzt durch Schreiben von einem Bit zur gültigen Version erklärt wird.

Name:

Mat. Nr:

Aufgabe 3 Ersetzungsstrategien (8 Punkte)

Das Betriebssystem eines Rechners verwaltet einen Hauptspeicher mit 4 Kacheln. Das Betriebssystem verwendet den NRU-Algorithmus. Auf dem System läuft ein Prozess mit insgesamt 8 Seiten. Die Seiten der Prozesse werden gemäß der ersten Zeile der folgenden Tabellen referenziert. Dabei steht 'r' für lesenden Zugriff und 'w' für schreibenden Zugriff. Der Eintrag ZS bedeutet, dass die R-Bits durch NRU-Algorithmus zurückgesetzt werden. In der Spalte unter einem Zugriff sollen die Folgen dieses Zugriffs dargestellt werden. So wird z. B. in der ersten Spalte durch den Zugriff auf Seite 1 diese Seite in Kachel 3 eingelagert. Die Seiten 0, 4, und 3 wurden schon vorher eingelagert.

Ergänzen Sie die Tabelle.

	Seitenreferenz/Art des Zugriffs									
	1/r	4/r	2/w	ZS	4/r	3r	2/r	1/w	ZS	3/r
K0	0(1,1)	0(1,1)	0(1,1)	0(0,1)	0(0,1)	0(0,1)	0(0,1)	1(1,1)	1(0,1)	1(0,1)
K1	4(1,0)	4(1,0)	4(1,0)	4(0,0)	4(1,0)	4(1,0)	4(1,0)	4(1,0)	4(0,0)	4(0,0)
K2	3(0,1)	3(0,1)	2(1,1)	2(0,1)	2(0,1)	2(0,1)	2(1,1)	2(1,1)	2(0,1)	2(0,1)
K3	1(1,0)	1(1,0)	1(1,0)	1(0,0)	1(0,0)	3(1,0)	3(1,0)	3(1,0)	3(0,0)	3(1,0)

Name:

Mat. Nr:

Aufgabe 4 Synchronisation

Das folgende Petri-Netz zeigt die Synchronisation von 2 Prozessen P_ABC , und P_DEF . Es handelt sich um ein Bedingungs-Ereignis-Netz. Die Transitionen A, B und C gehören zu Prozess P_ABC , die Transitionen D, E und F gehören zu Prozess P_DEF .

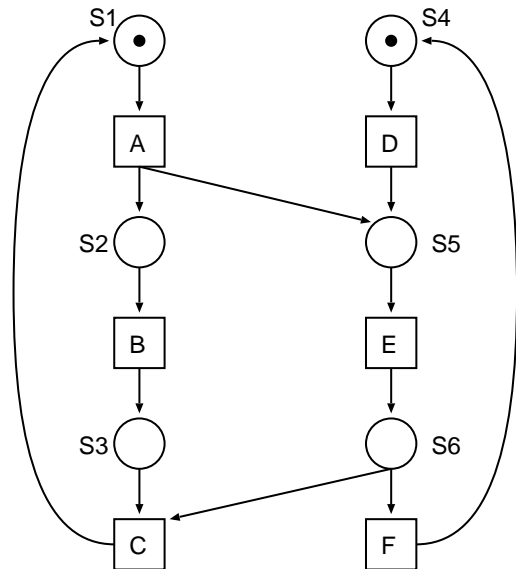


Abbildung 2: Ein paralleles System in Form eines Petri-Netzes

4.1 (2 Punkte)

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

S5 und S6

4.2 (3 Punkte)

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

```

Prozess P_ABC{
  while (true) {
    A();
    S5.up();
    B();
    S6.down();
    C();
  }
}
    
```

```

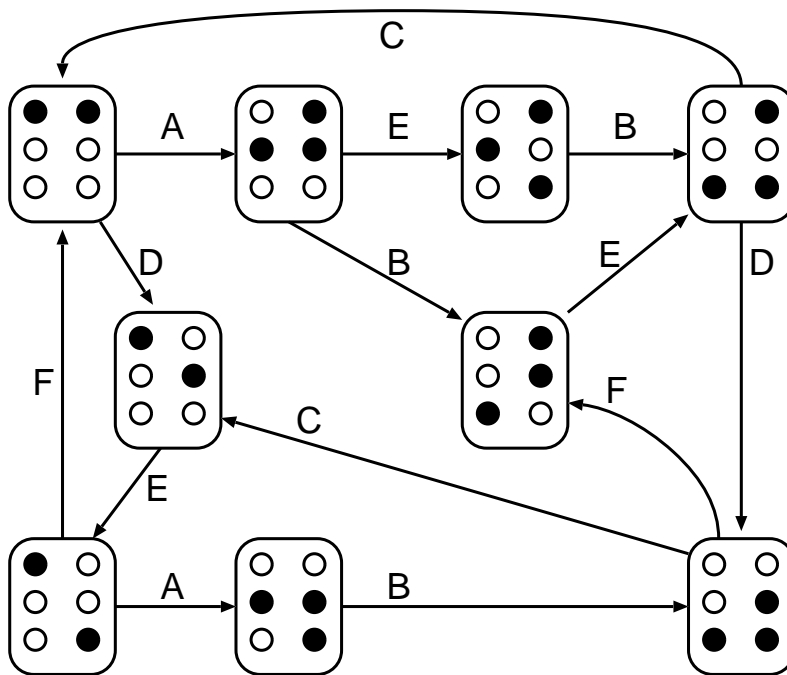
Prozess P_DEF{
  while (true) {
    D();
    S5.up();
    S5.down();
    E();
    S6.up();
    S6.down();
    F();
  }
}
    
```

4.3 (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.

Name:

Mat. Nr:



4.4 (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?

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

Name:

Mat. Nr:

Aufgabe 5 Interprozesskommunikation

Zwei Prozesse auf einem Rechner sollen Daten austauschen. Die Prozesse sind nicht miteinander verwandt. Ein Datensatz enthält zwei kurze Texte, zwei ganze Zahlen und drei Gleitkommazahlen und drei einzelne Buchstaben. Das Programm muss voraussichtlich regelmäßig angepasst werden, da sich die Zusammensetzung der Datensätze ändern kann.

Für die Interprozesskommunikation müssen Sie sich zwischen Shared Memory und einer Named Pipe entscheiden. Nennen Sie kurz die spezifischen Vor- und Nachteile dieser Verfahren für die gegebene Aufgabe.

5.1 (3 Punkte)

Vorteil Pipe: Synchronisation übernimmt die Pipe, Lese- bzw. Schreibzugriffe blockieren bei leere bzw. voller Pipe. Shared Memory erfordert externe Synchronisation.

Vorteil Shared Memory: (Eine komplexe Datenstruktur ist leichter zu übertragen; die Daten müssen nicht (um-)kodiert werden, wie bei Pipes. Shared Memory ist Schneller.

Aufgabe 6 Scheduling (10 Punkte)

Ein Betriebssystem verwendet prioritätsbasiertes Scheduling. Der Scheduler wird aktiv, wenn eine Zeitscheibe abläuft und wenn ein Prozesse einen SVC (Aufruf in den Betriebssystem-Kernel) absetzt und dadurch blockiert wird.

- Wenn ein Prozess eine höhere Priorität besitzt als alle anderen Prozesse, so darf er laufen, sobald der Scheduler aktiv ist/war.
- Wenn ein Zeitscheibe abläuft passiert folgendes: Die Priorität des Prozesses, der gerade rechnet, wird um den Wert 1 vermindert. Die Prioritäten der Prozesse, die bereit sind, werden um den Wert 1 erhöht. Danach ermittelt der Scheduler den Prozess, der als nächstes laufen darf.
- Wenn einen Prozess einen SVC abgesetzt, wird er blockiert. Seine Priorität wird um den Wert 1 erhöht.
- Die minimale Priorität eines Prozess ist die Basis-Priorität - 2, die maximale Priorität eines Prozess ist die Basis-Priorität + 2.
- Wenn mehrere Prozesse die maximale Priorität besitzen und einer davon läuft, so darf dieser weiterlaufen.
- Wenn mehrere Prozesse die maximale Priorität besitzen und keiner davon läuft, so darf der Prozess mit der kürzesten Wartezeit als nächster laufen.

Name:

Mat. Nr:

Prozesse

Prozess Nr.	Basis-Priorität
1	5
2	6
3	7
4	6

Ereignisse

- t_1 : Prozess P1, P2 und P3 werden gestartet
- t_2 : Ein Zeitscheibe für Multitasking läuft ab
- t_3 : Prozess P3 macht einen SVC (blockiert)
- t_4 : Prozess P4 wird gestartet
- t_5 : Ein Zeitscheibe für Multitasking läuft ab
- t_6 : Ein Zeitscheibe für Multitasking läuft ab
- t_7 : Prozess P2 macht einen SVC (blockiert)
- t_8 : Prozess P3 wird deblockiert
- t_9 : Ein Zeitscheibe für Multitasking läuft ab
- t_{10} : Ein Zeitscheibe für Multitasking läuft ab
- t_{11} : Prozess P3 macht einen SVC (blockiert)

Zeichnen Sie die Zustände und die Prioritäten der Prozesse in die folgende Tabelle ein:

