Übungskatalog

Übungskatalog zur Vorlesung *Echtzeitsysteme* an der Hochschule Niederrhein.

Jürgen Quade

Übungskatalog Übungskatalog zur Vorlesung *Echtzeitsysteme* an der Hochschule Niederrhein.

von Jürgen Quade

V3.0, 14. September 2011

Inhaltsverzeichnis

Tabellenverzeichnis

Kapitel 1. Systementwurf

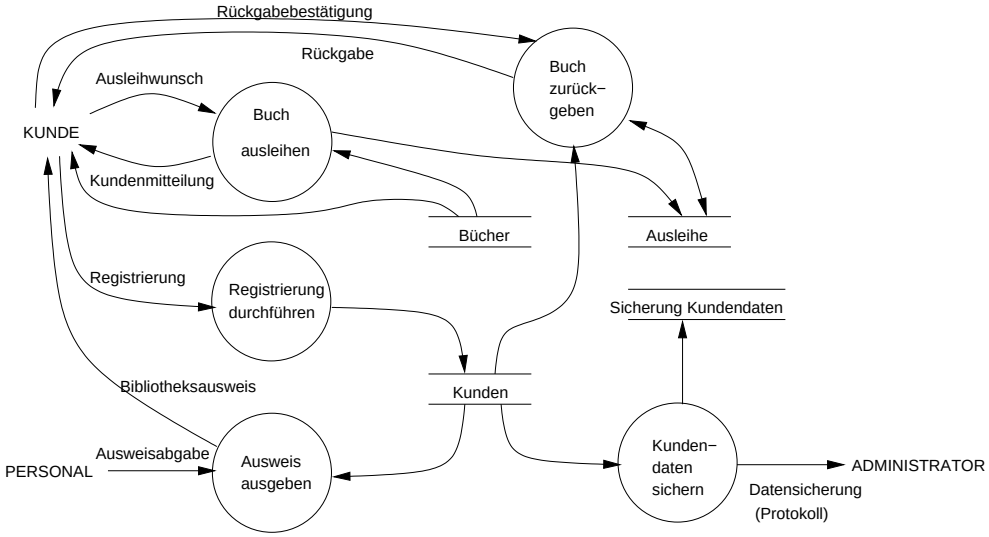
1.1. Geschäftsprozess »Bibliothek«

1.1.1. Beschreibung

Wenn sich ein Kunde als Bibliotheksnutzer registrieren läßt, bekommt er ein Bibliotheksausweis mit Ausweisnummer ausgestellt. Unter Kenntnis der Buchnummer kann ein Bibliotheksnutzer (Kunde mit Ausweis) ausleihen. Ist das von ihm gewünschte Buch vorrätig, bekommt er es ausgeliehen, ansonsten wird ihm eine entsprechende Mitteilung gemacht. Die Ausleihe und die Rückgabe mehrerer Bücher ist möglich.

Von den Kundendaten wird täglich eine Datensicherung erstellt. Dabei wird jeweils ein Protokoll erzeugt.

Von dem oben beschriebenen Geschäftsprozess wird folgendes Datenflussdiagramm erstellt:



1.1.2. Fragen

1. Nennen Sie die Fehler, die das obige Datenflussdiagramm enthält. (Tipp: Überprüfen Sie, ob die in der Vorlesung präsentierten Regeln eingehalten werden und ob die Vorgaben der Beschreibung vollständig erfüllt sind).
2. Erstellen Sie eine korrigierte Version des Datenflussdiagramms.

1.2. Systemstruktur Schleifstation

1.2.1. Beschreibung

Ein Student beschreibt in seiner Diplomarbeit die Steuerungssoftware einer automatisierten Schleifstation (für Wendeschneidplatten), die aus einer Be- und Entladestation (Bunker und Förderband), einem Indus-

trieroboter und einem Vision-System besteht:

> „In dieser Anwendung ruft die System-Task die Peripherie-Task auf, die die Kommunikation des Controllers mit dem Vision-PC und der Steuerung der Schleifmaschine regelt. Sobald die [Peripherie-]Task eine Anfrage [bezüglich eines spezifischen Rohlings] von der Schleifmaschine empfängt, gibt sie einen Impuls an die Band-Task zum Starten des Transportbands und startet gleichzeitig den Bunker über einen digitalen Ausgang. Eine Lichtschranke in der Blackbox des Visionsystems erkennt das Eintreffen eines Bauteils und meldet dies an die Peripherie-Task, die daraufhin über die Band-Task das Transportband stoppt. Anschließend sendet die [Peripherie-]Task einen Bilderkennungsauftrag an den Vision-PC und erhält als Antwort das Vergleichsergebnis, sowie Position und Orientierung des Rohlings [(Koordinaten)]."

> „Falls der Rohling nicht dem von der Schleifmaschine angeforderten Typ entspricht wird das Band wieder gestartet und die Prozedur wiederholt sich. Sollte der Rohling aber zur Anfrage passen, wird das Band eine definierte Strecke bis zum Greifbereich verfahren, wodurch sich die Position rechnerisch ermitteln läßt. Die Peripherie-Task startet dann die Robot-Task, die für alle Roboter Bewegungen zuständig ist. Diese greift den Rohling entsprechend der vom Vision-PC gelieferten Koordinaten und bewegt den Roboter anschließend auf eine sichere Position, wo der Rohling neu orientiert wird."

> „Dort wartet der Roboter bis die Peripherie-Task von der Schleifmaschine die Meldung bekommt, dass der aktuelle Schleifvorgang beendet ist. Dadurch wird das Ent- und Beladen der Schleifstation initiiert. Der Parallelgreifer des Roboters entnimmt der Werkstückaufnahme der Schleifmaschine die bearbeitete Wendeschneidplatte und setzt dafür den angeforderten Rohling ein. Dann fährt der Roboter mit dem Werkstück über die Paletten. Die Peripherie-Task meldet an die Schleifmaschine, dass der Ladevorgang abgeschlossen ist und erwartet die nächste Anfrage. Die Robot-Task palettiert die geschliffene Wendeschneidplatte und fährt dann auf eine sichere Warteposition. Ab hier wiederholt sich der Ablauf."

Fließtextbeschreibungen technischer Vorgänge sind nur bedingt zum Verständnis geeignet. Der Versuch, die Vorgänge in Form eines Ablaufdiagramms darzustellen wird vom Studenten folgendermassen kommentiert:

> „Die Darstellung der verschiedenen Prozesse in einem Flußdiagramm wird durch die z.T. parallele Verarbeitung mehrerer Tasks erschwert."

Mit „Flußdiagramm" meinte der Student nicht wirklich das Datenflussdiagramm, sondern vielmehr ein Programm-Ablaufplan.

Helfen Sie Ihrem Kommilitonen, in dem Sie eine adäquate technische Beschreibung der Abläufe auf Basis von Datenflussdiagrammen zu erstellen.

1.2.2. Fragen

1. Stellen Sie jeweils eine Liste

   a. der beteiligten Tasks,

   b. der Peripheriekomponenten und

   c. der ausgetauschten Daten

   zusammen.
2. Beschreiben Sie daher die Abläufe innerhalb des Systems mit Hilfe eines Datenflußdiagramms. Subsysteme und Peripheriekomponenten sind Datenquellen bzw. Datensenken. Zeichnen Sie auch Kontrollflüße ein.
3. Geben Sie für die Tasks System-Task, Robot-Task, Band-Task und Peripherie-Task jeweils ein Struktogramm an.

1.3. Petrinetz Fertigungsstrasse

1.3.1. Beschreibung

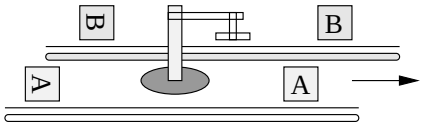


Abbildung 1-1. Handhabungssystem

Ein Handhabungsgerät bearbeitet Werkstücke (A und B), die auf zwei Förderbändern (F1 und F2) zum Handhabungsgerät transportiert werden. Das Handhabungsgerät kann immer nur ein Werkstück bearbeiten.

1.3.2. Aufgaben

1. Skizzieren Sie das zugehörige Petrinetz (Bedingunges/Ereignis Netz). Beschriften Sie die Stellen und Transitionen und geben Sie sinnvolle Anfangsbedingungen an.
2. Weisen Sie anhand des Erreichbarkeitsgraphen die Deadlockfreiheit des Netzes dar.
3. Da das bisher erstellte Petrinetz keine Fairneß garantiert, soll im folgenden ein Modell erstellt werden, welches garantiert, dass abwechselnd Werkstück A und danach Werkstück B bearbeitet wird.
4. Geben Sie ein Stellen/Transitionsnetz an, bei dem jeweils zwei Werkstücke A vom Handhabungsgerät bearbeitet werden, bevor ein Werkstück B bearbeitet wird.

1.4. Produktionsbetrieb I

1.4.1. Beschreibung

Ein Produktionssystem besteht aus 3 Maschinen M1, M2, M3 und 2 Bedienern B1, B2. Jedes auf einem Band eintreffende Werkstück muß zuerst von M1 (»Fräsen«) und dann entweder von M2 oder M3 (»Bohren«) bearbeitet werden. B1 arbeitet an M1 oder M2, B2 arbeitet an M1 oder M3. Zur Bearbeitung eines Werkstücks ist eine Maschine und ein Bediener nötig. Zu einer bestimmten Zeit kann immer nur ein Auftrag von einer Maschine bearbeitet werden und ein Bediener kann nicht gleichzeitig an zwei Maschinen arbeiten.

1.4.2. Aufgaben

1. Entwerfen Sie das Petrinetz (Bedingunges/Ereignis Netz). Beschriften Sie die Stellen und Transitionen und geben Sie sinnvolle Anfangsbedingungen an.

1.5. Produktionsbetrieb II

1.5.1. Beschreibung

Ein Produktionssystem besteht aus 3 Maschinen M1, M2, M3 und 2 Bedienern B1, B2. Jedes auf einem Band eintreffende Werkstück muß zuerst von M1 oder M2 und dann von M3 bearbeitet werden.

B1 arbeitet an M1 oder M3, B2 arbeitet an M2 oder M3. Zur Bearbeitung eines Werkstücks ist eine Maschine und ein Bediener nötig, letzterer kann nicht gleichzeitig an 2 Maschinen arbeiten. Die von M1 oder M2 bearbeiteten Werkstücke werden in einem Lager L3 zwischengelagert. M3 verschweißt jeweils 3 Werkstücke aus L3 miteinander. Je nachdem ob B1 oder B2 an M3 arbeitet, werden die verschweißten Werkstücke im Endlager L1 oder L2 abgelegt.

1.5.2. Aufgaben

1. Worin liegt der Unterschied zwischen einem Bedingungs/Ereignis Netz und einem Stellen/Transitions Netz?
2. Entwerfen Sie das Petrinetz (Stellen/Transitions Netz). Beschriften Sie die Stellen und Transitionen und geben Sie sinnvolle Anfangsbedingungen an.
3. Wie müssen Sie das Netz modifizieren, wenn das Lager eine Kapazität von 9 Werkstücken besitzt?
4. Wie ändert sich das Modell, wenn zwischen den beiden Bedienern nicht unterschieden wird?

1.6. Petrinetz: Schreib-Leselocks

1.6.1. Beschreibung

Kritische Abschnitte sind dann mehrfach betretbar, wenn während des Zugriffes innerhalb des kritischen Abschnittes keine Veränderungen an den Daten vorgenommen werden. Solange also kein Rechenprozess schreibend auf Daten innerhalb eines solchen Abschnittes zugreift, können mehrere Rechenprozesse gefahrlos lesend zugreifen.

Dieses Verhalten eines Lese-/Schreiblocks soll mit Hilfe normaler Semaphore nachgebildet werden, so dass zwei Rechenprozessen der lesende Zugriff oder einem Rechenprozess der schreibende Zugriff erlaubt ist. Dazu wird ein Semaphor mit 2 vorinitialisiert. Ein Prozess, der nur lesend auf den kritischen Abschnitt zugreift, alloziert das Semaphor wie gewohnt einmal, ein Rechenprozess, der schreibend zugreifen möchte, alloziert dagegen das Semaphor zweimal. Die folgende Codesequenzen verdeutlichen den Vorschlag:

Sequenz zum Lesen	Sequenz zum Schreiben
`P(S1)`	`P(S1)`
	`P(S1)`
`... // kritischer Abschnitt`	`... // kritischer Abschnitt`
`V(S1)`	`V(S1)`
	`V(S1)`

Es soll geklärt werden, ob dieser Vorschlag praktikabel ist.

1.6.2. Aufgaben

1. Erläutern Sie den Unterschied zwischen einem Semaphor und einem Mutex.
2. Beschreiben Sie den Unterschied zwischen einem Bedingungs-/Ereignisnetz und einem Stellen-/Transitionsnetz.
3. Modellieren Sie die Zugriffe (Sequenzen) der Tasks auf das Semaphor in einem Bedingungs-/Ereignisnetz. Beschriften Sie Stellen und Transitionen. Geben Sie eine sinnvolle Anfangsmarkierung an. (HINWEIS: Das Betriebsmittel „Prozesse“ wird als *eine* Stelle modelliert, maximal 3 Prozesse seien aktiv).
4. Stellen Sie den zugehörigen Erreichbarkeitsgraphen auf.
5. Erläutern Sie anhand der Sequenz zum Schreiben, wie es zu einer Verklemmung kommt.
6. Zeichnen Sie ein *Stellen-/Transitionsnetz*, bei dem es zu keiner Verklemmung kommt. Beschriften Sie wiederum Stellen und Transitionen und geben Sie eine sinnvolle Anfangsmarkierung an.

Kapitel 2. Schritthaltende Verarbeitung

2.1. Parameterbestimmung

2.1.1. Beschreibung

Bei einem Realzeitsystem liegt folgende Prozessorbelegung durch die drei Tasks Task A, Task B und Task C vor:

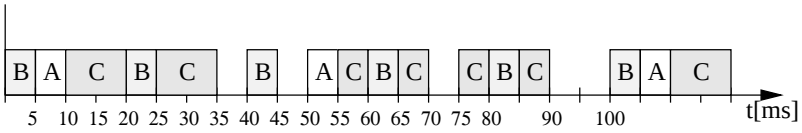


Abbildung 2-1. Prozessorbelegung

2.1.2. Aufgaben

1. Leiten Sie aus dem Diagramm die Prozesszeiten $t_{p,i}$ und die Verarbeitungszeiten $t_{e,i}$ der einzelnen Tasks Task A, Task B und Task C her. Gehen Sie davon aus, dass die Prozesszeiten und die Verarbeitungszeiten der einzelnen Tasks konstant sind.
2. Nach welcher *Faustregel* erfolgt im allgemeinen die Prioritätenvergabe?
3. Geben Sie die Prioritäten (von 1 bis 3) der Tasks an, wobei 1 der höchsten und 3 der niedrigsten Priorität entspricht.
4. Skizzieren Sie die möglichen Zustände und Zustandsübergänge einer Task in einem Echtzeitbetriebssystem. Welche Taskzustände lassen sich im obigen Diagramm nicht eindeutig unterscheiden?
5. Erstellen Sie für jede Task ein Diagramm, in dem Sie den Taskzustand über der Zeit eintragen, wenn von der oben angegebenen Prozessorbelegung der einzelnen Tasks ausgegangen wird.
6. Geben Sie die relative Belastung durch die einzelnen Tasks, sowie die Gesamtbelastung für den Realzeitrechner an.
7. Wie groß sind im Beispiel die maximalen Reaktionszeiten der Tasks bei Einsatz eines prioritätengesteuerten Schedulings (Rechnung!)?

2.2. Auslastung

2.2.1. Beschreibung

Ein Steuerungssystem ist durch die folgenden Anforderungen gekennzeichnet:

Anforderung	$t_{e,i}$	$t_{p,i}$	$t_{a,i}$
A	1 ms	5 ms	0 ms
B	4 ms	10 ms	0 ms
C	3 ms $\leq t_{e,c} \leq$ 5 ms	20 ms	0 ms
D	5 ms	50 ms $\leq t_{p,d} \leq$ 100ms	0 ms

2.2.2. Aufgaben

1. Berechnen Sie die Auslastung des Rechners durch die Anforderungen A und B.
2. Durch welche Werte ist der Worst Case für den Rechner gekennzeichnet? Geben Sie für diesen Fall die Gesamtauslastung durch alle vier Anforderungen an.
3. Wie hoch ist die durchschnittliche Auslastung, wenn Sie von einer Gleichverteilung der Prozess- und Verarbeitungszeiten ausgehen?
4. Geben Sie den vier Jobs Prioritäten, falls jede Anforderung durch einen eigenen Job bearbeitet wird.
5. Rechnerkernbelegung
    a. Zeichnen Sie die Anforderungsfunktion (worst case) im Bereich $0 \leq t \leq 60$ms (Auflösung 1cm=5ms).
    b. Zeichnen Sie die Rechnerkernbelegung durch die 4 Jobs (worst case) in das gleiche Diagramm, in das Sie die Anforderungesfunktion eingetragen haben.
6. Geben Sie die maximal zulässigen Reaktionszeiten $t_{Dmax,i}$ der 4 Jobs unter der Bedingung an, dass bis zum Neustart des Jobs der vorhergehende Job abgearbeitet sein muss.
7. Ermitteln Sie auf grafischem Wege die maximale Reaktionszeit t_{Rmax} der niedrigpriorsten Task im Fall von prioritätengesteuertem Scheduling.
8. Geben Sie die maximalen Wartezeiten der einzelnen Rechenprozesse an.
9. Mathematische Betrachtung:
    a. Stellen Sie die Zeitparameter für die vier Anforderungen zusammen ($t_{Emin,i}$, $t_{Emax,i}$, $t_{Pmin,i}$, $t_{Dmin,i}$, $t_{Dmax,i}$, $t_{Rmin,i}$, $t_{S,i}$ und die Priorität).
    b. Berechnen Sie für alle vier Rechenprozesse $t_{Rmax,i}$.
    c. Überprüfen Sie die beiden Echtzeitbedingungen. Ist schritthaltende Verarbeitung möglich?

2.3. Prioritätenverteilung

2.3.1. Beschreibung

Ein Rechnersystem soll die vier Jobs J1, J2, J3 und J4 bearbeiten. Der Prozess J_i besitze die periodische Prozesszeit $t_{p,i}$ und die konstante Verarbeitungszeit $t_{e,i}$:

Tabelle 2-1. Kennzahlen der Prozesse

Anforderung	$t_{e,i}$	$t_{p,i}$	$t_{s,i}$
JA	2.5 msec	18 msec	0ms
JB	1 msec	3 msec	0ms
JC	0.5 msec	2 msec	0ms
JD	1.5 msec	6 msec	0ms

2.3.2. Aufgaben

1. Ist es von der Zeitanforderung her gesehen prinzipiell möglich, alle vier Prozesse von einem Echtzeitrechner bearbeiten zu lassen?
2. Wie müssen die Prioritäten verteilt werden, damit jeder Prozess schritthaltend verarbeitet werden kann (prioritätengesteuertes Scheduling)? Jedes Programm muss sicher beendet sein, bevor ein neuer Alarm mit erneutem Start des gleichen Programms eintreffen kann (Rechtzeitigkeitsbedingung, die maximal zulässige Reaktionszeit $t_{Dmax,i}$ ist hier also durch die Prozesszeit $t_{p,i}$ gegeben). Die Programme dürfen ansonsten beliebig unterbrochen werden. Zeichnen Sie den Zeitverlauf der einzelnen Alarme (Anforderungsfunktion) und Programmzeitanforderungen und die Rechnerkernbelegung.

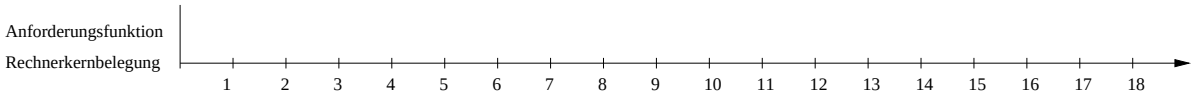


Abbildung 2-2. Rechenzeitanforderungen und Rechnerkernbelegung

3. Bestimmen Sie die maximalen Reaktionszeiten $t_{Rmax,i}$ der Rechenprozesse zunächst grafisch.
4. Bestimmen Sie die maximalen Reaktionszeiten der Rechenprozesse mathematisch.

2.4. Echtzeitnachweis »Steuerung«

2.4.1. Beschreibung

Die Anwendungssoftware einer Steuerung besteht aus drei Jobs, deren Zeitparameter in der folgenden

Tabelle gelistet sind:

Job	$t_{Emin,i}$	$t_{Emax,i}$	$t_{Pmin,i}$	$t_{Dmin,i}$	$t_{Dmax,i}$	$t_{A,i}$	$\rho_{max,i}$
A	10 ms	30 ms	80 ms	0 ms	60 ms	0 ms	
B	20 ms	30 ms		0 ms	50 ms	0 ms	0,375
C	28 ms	32 ms	160 ms	0 ms	155 ms	0 ms	

1. Berechnen Sie die Auslastungen
   a. $\rho_{max,A}$ (ein Job)
   b. $t_{Pmin,B}$
   c. ρ_{ges} (Gesamtauslastung für alle drei Jobs)
   d. Ist eine schritthaltende Verarbeitung *prinzipiell* möglich?

2. Prioritätenvergabe
   a. Wie lautet die Faustformel zur Vergabe von Prioritäten?
   b. Bei welchem der drei Rechenprozesse ist eine eindeutige Vergabe der Prioritäten möglich? Welche Priorität bekommt dieser Job?

3. Echtzeitnachweis bei Einsatz eines prioritätengesteuerten Schedulings für das bestehende System. Gehen Sie hierzu von den folgenden Prioritäten aus: A=1 (hoch), B=2 (mittel) und C=3 (niedrig).
   a. Geben Sie die Rechenzeitanforderungsfunktion $t_{C,1}(t)$ für Jobs der höchsten Priorität an.
   b. Berechnen Sie die maximale Reaktionszeit $t_{Rmax,1}$.
   c. Geben Sie die Rechenzeitanforderungsfunktion $t_{C,2}(t)$ für Jobs mittlerer Priorität an.
   d. Berechnen Sie die maximale Reaktionszeit $t_{Rmax,2}$.
   e. Geben Sie die Rechenzeitanforderungsfunktion $t_{C,3}(t)$ für Jobs der niedrigsten Priorität an.
   f. Berechnen Sie die maximale Reaktionszeit $t_{Rmax,3}$.
   g. Zeigen Sie anhand der 2. Echtzeitbedingung (Rechtzeitigkeitsbedingung), dass schritthaltende Verarbeitung nicht möglich ist.

4. Was können Sie unter Beibehaltung des prioritätengesteuerten Schedulings ändern, damit dennoch eine schritthaltende Verarbeitung gewährleistet wird? Geben Sie für diesen Fall die maximale Reaktionszeit $t_{Rmax,A}$ für Job A an.

5. Echtzeitnachweis bei Einsatz eines Deadline-Schedulings
   a. Geben Sie die Gesamt-Rechenzeitanforderungsfunktion $t_{C,ges}(I)$ an.
   b. Für welche I müssen Sie $t_{C,ges}(I)$ konkret untersuchen? Geben Sie den Bereich an, in dem I zu untersuchen ist und dazu die zu untersuchenden Punkte in diesem Bereich.
   c. Führen Sie den Echtzeitnachweis durch. Ist schritthaltende Verarbeitung möglich?

2.5. Realzeitnachweis bei abhängigen Ereignissen

2.5.1. Beschreibung

Gegeben sind zwei Rechenprozess RP_1 und RP_2 mit den folgenden Kenndaten:

Anf.	$t_{Emax,i}$	$t_{Pmin,i}$	$t_{Dmin,i}$	$t_{Dmax,i}$	$t_{A,i}$
A	2 msec	6 msec	0 msec	3 msec	
B	3 msec	6 msec	0 msec	4 msec	

Beide Rechenprozesse sollen per Deadline-Scheduling verarbeitet werden.

2.5.2. Aufgaben

1. Führen Sie für den gegebenen Fall den Echtzeitnachweis durch. Gehen Sie zunächst davon aus, dass die beiden Ereignisse unabhängig voneinander sind. Welcher Wert ergibt sich in diesem Fall für t_A?

   Bestimmen Sie dafür die Rechenzeitanforderungsfunktion $t_c(I)$. Sind die Rechenprozesse fristgerecht zu bearbeiten?

2. Zwischen den beiden Ereignissen existiert der Zusammenhang, dass genau 1 ms nach Auftreten des Ereignisses A das Ereignis B auftritt. Geben Sie für diesen Fall t_S an.

3. Führen Sie jetzt unter Berücksichtigung der Abhängigkeit den Echtzeitnachweis durch. Sind die Rechenprozesse fristgerecht zu bearbeiten?

Kapitel 3. Realzeit-Betriebssystem

3.1. Auf dem Bauernhof

3.1.1. Beschreibung

Auf einem Bauernhof werden Kühe, Schweine und Pferde gehalten. Die Tiere müssen regelmässig gefüttert werden, die Kühe alle 8 Stunden, die Pferde alle 6 Stunden und die Schweine alle 4 Stunden - genau einmal in dem jeweiligen Zeitraum (Fütterungszeit). So müssen die Kühe beispielsweise einmal zwischen 0 und 8 Uhr, das zweite Mal zwischen 8 und 16 Uhr und ein drittes Mal zwischen 16 und 24 Uhr gefüttert werden. Dabei spielt es keine Rolle, ob die Fütterung um 0 Uhr oder (im Fall eines jungen, sportlichen Bauern, erst um 6 Uhr beginnt). Alle Tiere befinden sich in einem Stall, daher kann bei der Fütterung problemlos zwischen den Tierarten gewechselt werden. Tiere, die nicht rechtzeitig gefüttert wurden, bekommen schlechte Laune und brechen aus dem Stall aus (harte Realzeitbedingung).

Ein junger, sportlicher Bauer benötigt zum Füttern der Kühe 2 Stunden, für die Pferde 1,5 Stunden und für die Schweine 1 Stunde. Er geht nach folgender Strategie I vor: Die Tierarten sind unterschiedlich anspruchsvoll, am schwierigsten die Schweine und am unkompliziertesten die Kühe. Die anspruchsvollen Tiere werden in ihrer Fütterungszeit immer sofort bedient, die weniger komplizierten erst danach.

3.1.2. Aufgaben

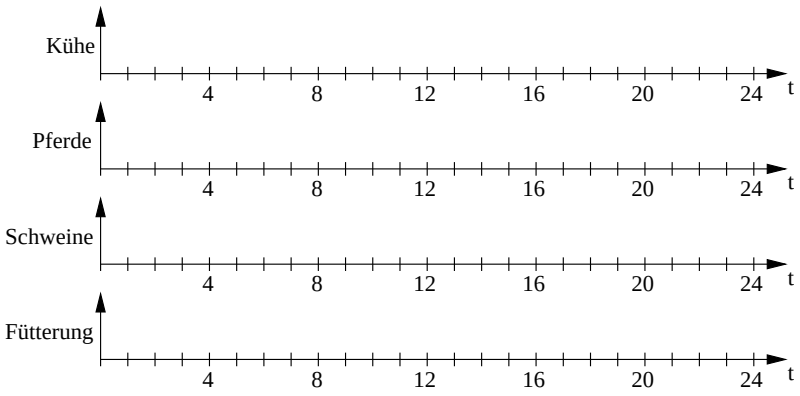


Abbildung 3-1. Lösungsdiagramm

1. Tragen Sie in ein Diagramm die Fütterungszeitpunkte und -dauern der einzelnen Tierarten ein, sowie die Fütterungsreihenfolge nach der oben genannten Strategie über 24 Stunden.
2. Ein älterer Bauer ist nicht mehr ganz so schnell unterwegs und benötigt zum Füttern der Kühe 2.5 Stunden, der Pferde 2 Stunden und der Schweine 1 Stunde. Zunächst geht er nach dem oben beschriebenen Strategie I vor.

   Tragen Sie auch für diesen Fall die Fütterungsreihenfolge in das Diagramm ein. Was passiert?

3. Er wendet nun ein anderes Fütterungsschema (Strategie II) an: Er füttert zu jedem Zeitpunkt die hungrigsten Tiere (bei denen das Ende des Fütterungszeit/Zeitraums am nächsten ist). Wie sieht jetzt die Fütterungsreihenfolge aus?
4. Ordnen Sie den in dieser Aufgabe verwendeten Begriffen die Fachbegriffe aus der Echtzeitwelt zu:

Tabelle 3-1. Begriffe

Bauernhof	Echtzeitwelt
Bauer	
Füttern einer Tierart	
Fütterungsdauer	
Fütterungszeitraum	
Strategie I	
Strategie II	

3.2. Scheduling

3.2.1. Beschreibung

Ein Echtzeitrechner bearbeite vier Tasks, deren Verarbeitungszeiten für diesen Rechner und deren Prozesszeiten gegeben sind:

Tabelle 3-2. Taskspezifikation

Taskname	t_e	t_p
A	50 ms	200 ms
B	75 ms	300 ms
C	100 ms	500 ms
D	125 ms	800 ms

3.2.2. Fragen

1. Ist der Rechner von den Zeitanforderungen (Auslastungsbedingung) prinzipiell in der Lage, die Aufgaben zu bearbeiten?
2. Zunächst werde ein *prioritätengesteuertes* Scheduling verwendet. Welche Task bekommt dabei welche Priorität? Geben Sie die Rechnerkernbelegung durch die vier Tasks an.

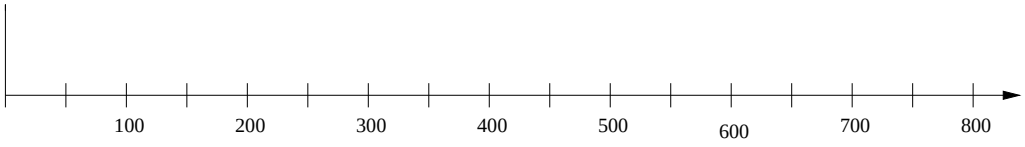


Abbildung 3-2. Prioritätengesteuertes Scheduling

3. Als zweites soll ein Round-Robin Scheduling untersucht werden. Ein Quantum sei 50 ms. Geben Sie die Rechnerkernbelegung durch die vier Tasks beim Round-Robin Scheduling an. Hinweis: Erstellen und verfolgen Sie ständig die Liste (Queue) der rechenbereiten Rechenprozesse!

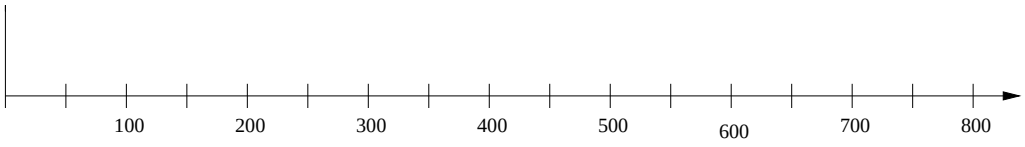


Abbildung 3-3. Round Robin Scheduling

4. Jetzt werde ein Deadline-Scheduling verwendet. Die maximal zulässigen Reaktionszeiten seien dabei durch die Prozeßzeiten gegeben ($t_{Dmax}=t_p$). Geben Sie auch für diesen Fall die Rechnerkernbelegung durch die vier Tasks an. Hinweis: Nummerieren Sie in der Anforderungsfunktion die einzelnen Anforderungen durch und zeichnen Sie zusätzlich die Deadlines ein.

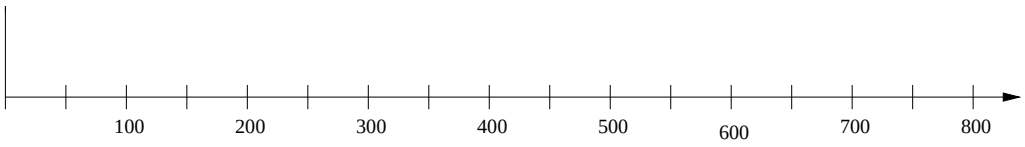


Abbildung 3-4. Deadline Scheduling

3.3. Gerätetreiber

3.3.1. Aufgaben

1. Schreiben Sie eine Applikation, die von dem Gerät »digital_io« einen 32-Bit-Wert einliest und mit `printf` auf dem Bildschirm ausgibt.
2. Welche drei Funktionen müssen in einem zugehörigen Linux-Treiber mindestens impelementiert sein?
3. Welche Parameter werden der Funktion `driver_read` neben *`struct file *instanz`* und *`loffs_t *offset`* übergeben?
4. Gegeben ist der folgende Code einer Funktion `driver_read`:

```
static ssize_t driver_read( struct file *instanz, char *userbuf,
    size_t count, loffs_t *offset )
{
    int to_copy, not_copied;
    __u32 control;

    control = inl( basisaddress+0x30 );
    to_copy = min( count, sizeof(control) );
    not_copied = copy_to_user( userbuf, &control, to_copy );
    return to_copy-not_copied;
}
```

   a. Ändern Sie den Code, so dass dieser mit invertierter Logik arbeitet (die eingelesenen Bits müssen invertiert werden).

   b. Welche Aufgabe hat die Funktion `min` im gegebenen Code? Warum ist sie wichtig?

5. Gegeben ist der folgende Code einer Treiberfunktion `driver_write`:

```
static ssize_t driver_write( struct file *instanz, const char *userbuf,
    size_t count, loffs_t *offset )
{
    int to_copy, not_copied;
    __u32 control;

    to_copy = max( count, sizeof(control) );
    not_copied = copy_from_user( &control, userbuf, count );
    outl( control, basisaddress+0x34 );
    return 0;
}
```

   Nennen und beschreiben Sie die drei Fehler in diesem Code.

6. Schreiben Sie eine Applikation, die von dem Gerät »digital_io« einen 32-Bit-Wert einliest und an die gleiche Schnittstelle wieder zurückschreibt.
7. Ein portmapped Ausgaberegister an der Speicherstelle `basisadresse+0x38` darf erst dann beschrieben werden, wenn der Wert an der Speicherstelle `basisadresse+0x3c 0x11223344` ist.

   a. Zeichnen Sie ein Struktogramm für eine Funktion `driver_write` die so lange die Speicherstelle `basisadresse+0x3c` überprüft, bis diese den geforderten Wert hat. Erst danach dürfen Sie den übergebenen Wert schreiben.

   b. Skizzieren Sie den zugehörigen C-Code.

Kapitel 4. Programmierung

4.1. Zeitverwaltung

Die Funktion `gettimeofday()` liefert die Zeit in Sekunden und Mikrosekunden über die folgende (vereinfacht dargestellte) Datenstruktur (siehe Vorlesungsskript):

```
struct timeval {
    unsigned int tv_sec;
    unsigned int tv_usec;
};
```

Mit Hilfe der Funktion soll eine Differenzzeitmessung durchgeführt werden, die auch für größere Zeitabstände ausgelegt ist. Das Ergebnis soll wieder in Form der Datenstruktur `struct timeval` vorliegen (also in Sekunden und in Mikrosekunden). Hinweis: Der Datentyp `unsigned int` ist 32 Bit breit und kann damit einen Wert von 0 bis 4294967295 annehmen.

1. Welcher Zeitbereich lässt sich in einer derartigen Datenstruktur unterbringen?
2. Im Praktikum wurde die per `gettimeofday()` zurückgelieferte Zeit in eine einzelne Variable (Typ Integer) umgerechnet. Welchen Vorteil bringt das mit sich? Was sind die Nachteile?
3. Erläutern Sie das Prinzip der Differenzzeitmessung.
4. Geben Sie für die folgenden drei Paare von Zeitstempeln die jeweilige Differenz in Sekunden und Mikrosekunden an.

Vorher	tv_sec=12345	tv_usec=309111
Nachher	tv_sec=12347	tv_usec=309156

Vorher	tv_sec=12345	tv_usec=1300
Nachher	tv_sec=12347	tv_usec=1156

Vorher	tv_sec=4294967295	tv_usec=200
Nachher	tv_sec=1	tv_usec=100

5. Skizzieren Sie in Form eines **Struktogramms** einen Algorithmus, der die Differenzzeit berechnet und in die Variablen `diff_seconds` und `diff_useconds` ablegt. Der Algorithmus soll für alle angegebenen Zeitwerte das jeweils richtige Ergebnis liefern.

4.2. Critical Section - Messwerterfassung

4.2.1. Beschreibung

Ein Echtzeitrechner führt eine Meßwerterfassung durch. Für drei unterschiedliche Eingangskanäle (Temperatur, Druck, Durchfluss) gibt es drei Tasks, die jeweils für die Steuerung eines Kanals zuständig sind. Da mit dem Meßwert auch die Dauer der Meßwertaufnahme zu bestimmen ist, gibt es genau einen Zäh-

ler, der zu Beginn der Task gestartet wird und nach dem Ende der Meßwerterfassung mit dem Stoppen des Timers ausgelesen wird. Die erfaßten Meßwerte werden von den drei Tasks zusammen mit dem Zählerstand in einen gemeinsamen Speicher abgelegt. Der Zugriff auf diesen Speicher ist für alle Tasks ohne Synchronisation möglich.

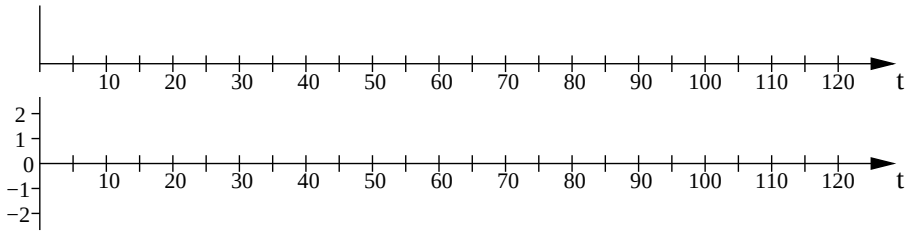
Die drei Tasks haben die folgenden Verarbeitungs- und maximal zulässigen Reaktionszeiten:

Task	t_E	t_{Dmax}
Task A	10 ms	40 ms
Task B	20 ms	60 ms
Task C	30 ms	80 ms

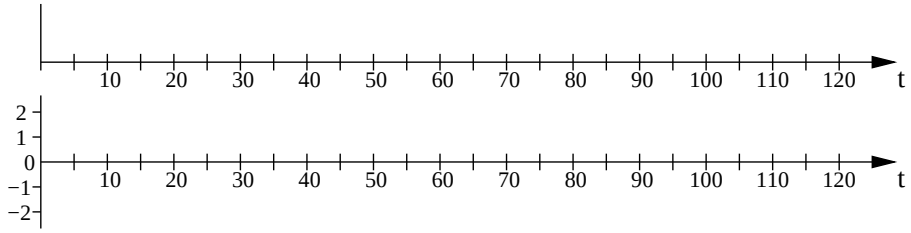
4.2.2. Aufgaben

1. Ist eine Lösung der Aufgabe in Realzeit *prinzipiell* möglich (Überprüfen der Auslastungsbedingung)? Sie können zur Lösung der Aufgabe davon ausgehen, dass die Prozeßzeit mit der maximal zulässigen Reaktionszeit der jeweiligen Task identisch ist. Bestimmen Sie dazu die Gesamtauslastung des Echtzeitrechners durch die drei Tasks.
2. Beschreiben Sie kurz die „critical section“ bei der Meßwerterfassung. Wie kann softwaretechnisch sichergestellt werden, dass es bei dieser „critical section“ zu keiner race condition kommt?
3. Erstellen Sie ein DFD der Konstellation. Ergänzen Sie das DFD durch die Kontrollflüsse.
4. Skizzieren Sie den Code einer der drei Tasks in Form eines Struktogramms.
5. Geben Sie den drei Tasks Prioritäten.
6. Skizzieren Sie die Rechnerkernbelegung und den Wert des Semaphor im Zeitraum von 0 bis 120 ms. Gehen Sie davon aus, dass die einzelnen Tasks zu den folgenden Zeitpunkten “lauffähig” werden:

Task	**Startzeitpunkt**
Task A	20 ms, 80 ms
Task B	10 ms, 70 ms
Task C	0 ms, 90 ms



7. Geben Sie die maximale Reaktionszeit $t_{Rmax,A}$ der Task A für den Fall an, dass kein Semaphor notwendig wäre (ohne Synchronisation) und für den hier beschriebenen Fall mit dem Semaphor (Worst Case).
8. Um die maximale Reaktionszeit (Antwortzeit) der Task A zu verringern, wird ein zweiter Zähler zur Verfügung gestellt. Wie wird jetzt das Semaphor verwendet?
9. Zeichnen Sie für den Fall, dass zwei Zähler zur Verfügung stehen, die Rechnerkernbelegung und den Wert des Semaphor.



10. In welchem Fall wird durch die beschriebene Maßnahme die maximale Reaktionszeit der Task A tatsächlich verbessert?

4.3. Semaphor und Spinlock

4.3.1. Beschreibung

Gegeben ist folgendes, aus einem Gerätetreiber stammendes Codefragment:

```
extern struct global_access_list *gal;
extern struct global_content *cvar;

static int driver_open( struct inode *geraetedatei, struct file *instanz )
{
    int i;

    read_and_modify( gal );

    read_content( cvar );
    if( cvar->count > CONTENT_MAX_COUNT ) {
        write_content( cvar );
        return -1;
    }
    ...
    return 0;
}

static irqreturn_t driver_isr( int irq_nr, void *dev_id, struct pt_regs *regs )
{
     read_content( cvar );
     cvar->count++;
     write_content( cvar );
     ...
     return IRQ_HANDLED;
}
```

Der Betriebssystemkern stellt zum Schutz kritischer Abschnitte sowohl Semaphore als auch Spinlocks mit den folgenden Funktionen zur Verfügung:

Betreten	**Verlassen**
`P(S)`	`V(S)`
`spin_lock(L)`	`spin_unlock(L)`
`localir_disable()`	`localir_enable()`
`spin_lock_localir_disable(L)`	`spin_unlock_localir_enable(L)`

4.3.2. Aufgaben

1. Spinlocks

   a. Erläutern Sie den Unterschied zwischen einem Semaphor und einem Spinlock anhand des Zustandsübergangsdiagramms für Rechenprozesse.

   b. Auf welchen Rechnertypen (Ein-/Mehrprozessormaschinen) können Spinlocks eingesetzt werden?

   c. Bei welchen Gelegenheiten setzen Sie eine Interruptsperre zum Schutz eines kritischen Abschnitts ein?

2. Unterbrechungsmodell

   a. Skizzieren Sie das Unterbrechungsmodell des Linux-Betriebssystems.

   b. Sie haben eine Codesequenz A, die auf der Kernel-Ebene abläuft. Codesequenzen auf welcher Ebene können die Codesequenz A unterbrechen?

   c. Ordnen Sie die Funktionen `driver_open()` und `driver_isr()` jeweils einer Ebene des Unterbrechungsmodells zu.

   d. Kann die Funktion `driver_isr()` auf einer Einprozessormaschine unterbrochen werden?

3. Schutz kritischer Abschnitte

   a. Beschreiben Sie die beiden kritischen Abschnitte in den Codefragmenten.

   b. Mit welcher Methode schützen Sie den ersten kritischen Abschnitt, wenn die Funktion `read_and_modify()` eine Verarbeitungszeit von $t_{E,rm}$=2ms hat? Begründung!

   c. Warum spielt die Verarbeitungszeit der Funktion `read_and_modify()` eine Rolle bei der Wahl des Elementes, mit dem der kritischen Abschnitt geschützt werden soll?

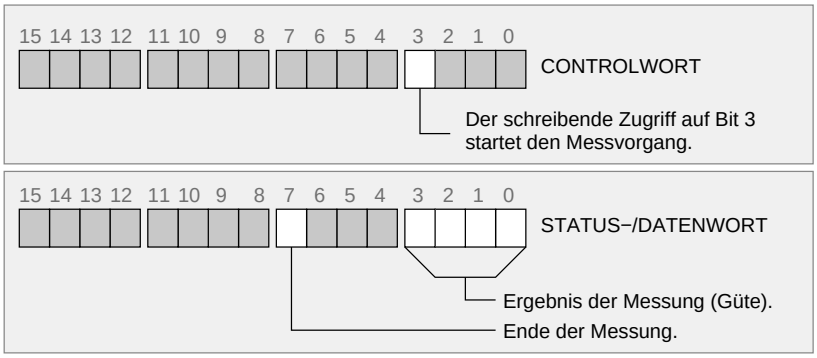
   d. Mit welcher Methode schützen Sie den zweiten kritischen Abschnitt? Begründung!

   e. Ergänzen Sie das gegebene Codefragment um die sinnvollen Funktionen zum Schutz kritischer Abschnitte.

4.4. Ankopplung einer Messstation

4.4.1. Beschreibung

Eine Messstation zur Erfassung eines Gütewertes soll über ein Control- und ein kombiniertes Status-/Datenwort angesteuert werden:



Das Controlwort besteht aus nur einem relevanten Bit: Der schreibende Zugriff auf Bit 3 startet den Messvorgang. Die übrigen Bits haben keine Bedeutung („don't care").

Im Status-/Datenwort sind die Bits von 0 bis 3 und das Bit 7 von Bedeutung. Wenn Bit 7 gesetzt ist, ist der Messvorgang abgeschlossen; in diesem Fall enthalten die Bits 0 bis 3 das Messergebnis (Maßzahl für die Güte).

Das Controlwort und das Status-/Datenwort sollen beide an der einen Adresse `0xFFA600` in den Adressraum des Rechners eingeblendet werden (Controlwort: schreibender Zugriff; Status-/Datenwort: lesender Zugriff).

4.4.2. Fragen

1. Welche Werte (Wertebereich) ergeben sich für die Güte eines Messergebnisses aufgrund der Angaben bezüglich des Status-/Datenwortes?
2. Software
   a. Erstellen Sie ein Struktogramm für die Durchführung einer Messung.
   b. Skizzieren Sie den (hardwarenahen) C-Code für die Durchführung einer Messung (Starten der Messung, Warten auf das Ende und Einlesen des Messergebnisses).
   c. In welcher Systemkomponente befindet sich dieser Code typischerweise?
   d. Wie sieht der Zugriff auf das Interface für die Applikation aus?

Kapitel 5. Verfügbarkeit und Zuverlässigkeit

5.1. Echtzeitrechner

5.1.1. Beschreibung

Ein Echtzeitrechner besteht aus den in der folgenden Tabelle aufgelisteten Einzelkomponenten mit der jeweiligen Dauerunverfügbarkeit q_i:

Tabelle 5-1. Verfügbarkeit der Einzelkomponenten eines Echtzeitrechners

Komponente	Dauerunverfügbarkeit
Prozessor	q=1%
DMA-Interface	q=2%
Terminal	q=2%
Speicher	q=1%
Analog-I/O-Interface	q=3%
Bildschirm	q=0.5%

5.1.2. Aufgaben

1. Der Echtzeitrechner sei verfügbar, wenn der Prozessor, der Speicher, das DMA-Interface, das analoge Ein-/Ausgabe-Interface und das Terminal oder der Bildschirm funktionsfähig sind. Wie groß ist die Ausfallwahrscheinlichkeit des Echtzeitrechners?
2. Um eine höhere Verfügbarkeit zu erreichen, werden einige Komponenten des Echtzeitrechners als Zweifachsystem ausgeführt. Für die dazu notwendige Koppel- und Vergleichselektronik wird eine Verfügbarkeit von 1 angenommen. Welche Komponenten erbringen dabei den größten Beitrag zur Erhöhung der Verfügbarkeit? Welche Verfügbarkeit läßt sich für das Gesamtsystem maximal erreichen, wenn Sie zwei Komponenten verdoppeln?
3. Um die Anlage bei einer Störung in einen sicheren Zustand zu bringen, seien der Prozessor und das DMA-Interface nötig. Welche Strategie der Komponentenverdoppelung ist unter diesem Aspekt sinnvoll?

   Unter welchen Umständen muß die Anlage in den sicheren Zustand überführt werden, wenn ein Gewinn an Sicherheit gegenüber 2. erreicht werden soll?

   Wie groß ist nun die Verfügbarkeit?

5.2. Doppelrechnersystem

5.2.1. Beschreibung

Die Verfügbarkeit einer Prozeßsteuerung soll durch den Einsatz von zwei Rechnern mit gleicher Verfügbarkeit p_r vergrößert werden. Die dafür benötigte Rechnerkopplung hat die Verfügbarkeit p_k. Das System ist ausgefallen, wenn beide Rechner oder die Rechnerkopplung nicht mehr verfügbar sind.

5.2.2. Aufgaben

1. Wie groß muß p_k mindestens sein, damit die Systemverfügbarkeit gleich der Verfügbarkeit eines Einzelrechners ist?
2. Wie groß ist die Systemverfügbarkeit, wenn $p_r = 99\%$ und $p_k = 99.9\%$ sind?

5.3. Forschungssatellit

5.3.1. Beschreibung

Für einen Forschungssatelliten soll ein geeignetes Rechnersystem so konfiguriert werden, dass auch für einen längeren Raumflug eine hohe Verfügbarkeit gewährleistet ist. Als einzelne Rechner des Rechnersystems werden Standardsysteme eingesetzt, für die der Hersteller folgende Daten angibt: `MTBF=25000h`, `MTTR=200h` (die unterschiedlichen Umgebungsbedingungen im Weltraum und auf der Erde sollen hier vernachlässigt werden).

5.3.2. Aufgaben

1. Wie groß ist die Dauerverfügbarkeit p_R eines einzelnen Rechners (auf der Erde)?

Ein Rechner besteht dabei aus zwei Komponenten, einer Prozessor-Baugruppe und einer Ein-/Ausgabe-Baugruppe, die beide die gleiche Verfügbarkeit besitzen.

2. Wie groß ist ihre jeweilige $MTBF_K$?
3. Wie groß ist die Dauerverfügbarkeit p_K jeder einzelnen Komponente?
4. Warum ist die Größe „Dauerverfügbarkeit“ für den Einsatz im Weltraum irrelevant?
5. Wie groß ist die Wahrscheinlichkeit dafür, dass ein *einzelner* Rechner beim Eintreffen des Satelliten am Jupiter nach 3 Jahren (1Jahr=9000h) noch funktionsfähig ist?

Es wird nun überlegt, wie durch den Einsatz von mehreren Rechnern diese Wahrscheinlichkeit erhöht werden kann. Erreicht werden kann dies durch ein Mehrrechnersystem, das selbst dann noch verfügbar ist, wenn nur noch einer der n parallel arbeitenden Rechner funktionsfähig ist.

6. Geben Sie *allgemein* für *n* Rechner die Verfügbarkeit $p_n(t)$ des beschriebenen Mehrrechnersystems zum Zeitpunkt t an.
7. Wieviele Rechner müssten eingesetzt werden, d.h. wie groß muß *n* sein, um für den Jupiterflug eine Wahrscheinlichkeit der Verfügbarkeit des Rechnersystems von mindestens 0.995 zu erreichen?

Da dies aus Gewichtsgründen nicht realisierbar ist, versucht man nun durch eine andere Konfiguration des Mehrrechnersystems die gewünschte Verfügbarkeit zu erreichen. Dabei macht man sich die Tatsache zunutze, dass jeder einzelne Rechner aus zwei unabhängigen Komponenten (Prozessor- und Ein/Ausgabe-Baugruppe) besteht. Das Satelliten-Rechnersystem wird nun so umkonstruiert, dass es selbst dann noch funktionsfähig ist, wenn nur noch eine beliebige Prozesssor-Baugruppe und eine beliebige Ein/Ausgabe-Baugruppe funktionsfähig sind. Es sollen nun 7 Rechner auf diese Weise in den Satelliten eingebaut werden.

8. Zeichnen Sie das Zuverlässigkeits-Ersatzschaltbild für die so gegebene Rechnerkonfiguration.
9. Wie groß ist nun die Wahrscheinlichkeit für ein Funktionieren des Satelliten-Rechnersystems beim Eintreffen des Satelliten am Jupiter?