

8 Betriebssystemunterstützung für Multimedia

8.1 Charakteristika von Multimedia-Daten

8.2 Aufbau von Betriebssystemen

8.3 Prozeßbegriff

8.4 Auftragsplanung

8.4.1 Durchsuchen des Gestaltungsraums

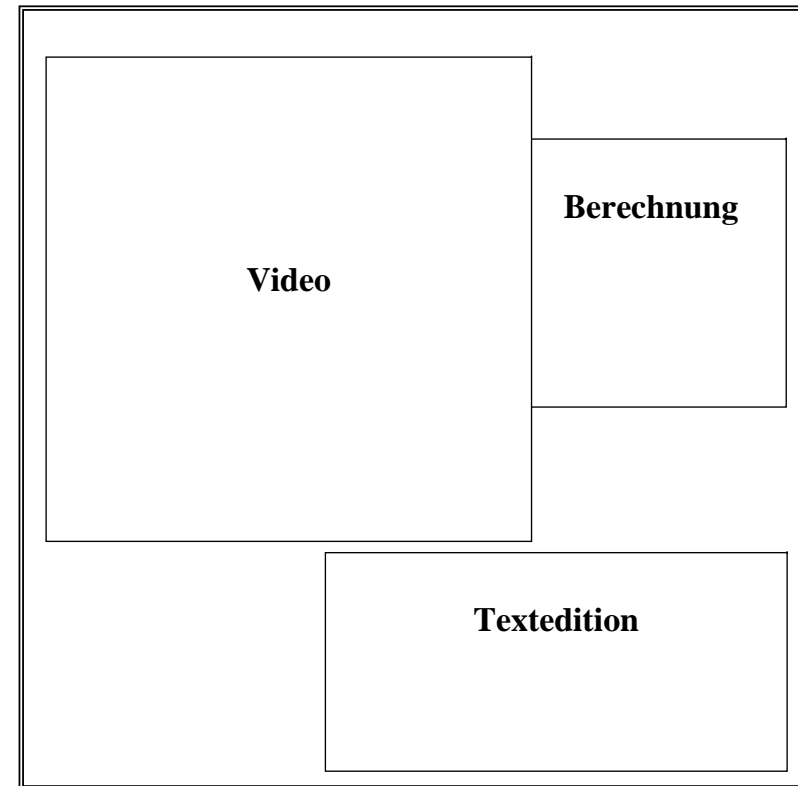
8.4.2 Planen nach Fristen

8.4.3 Planen nach Spielräumen

8.3.4 Ratenplanung

Situation am Arbeitsplatzrechner:

Mehrere parallele Aktivitäten unter einem Fenstersystem.



Wunsch des Nutzers: Gleichmäßiger Fortgang aller Aktivitäten.

Charakteristika von Multimedia-Daten:

Einfache Datenformate: 8-Bit-Ganzzahlen,
16-Bit-Ganzzahlen.

Große Datenmengen: Eine einlagige DVD umfaßt
etwa 2 Stunden Video.

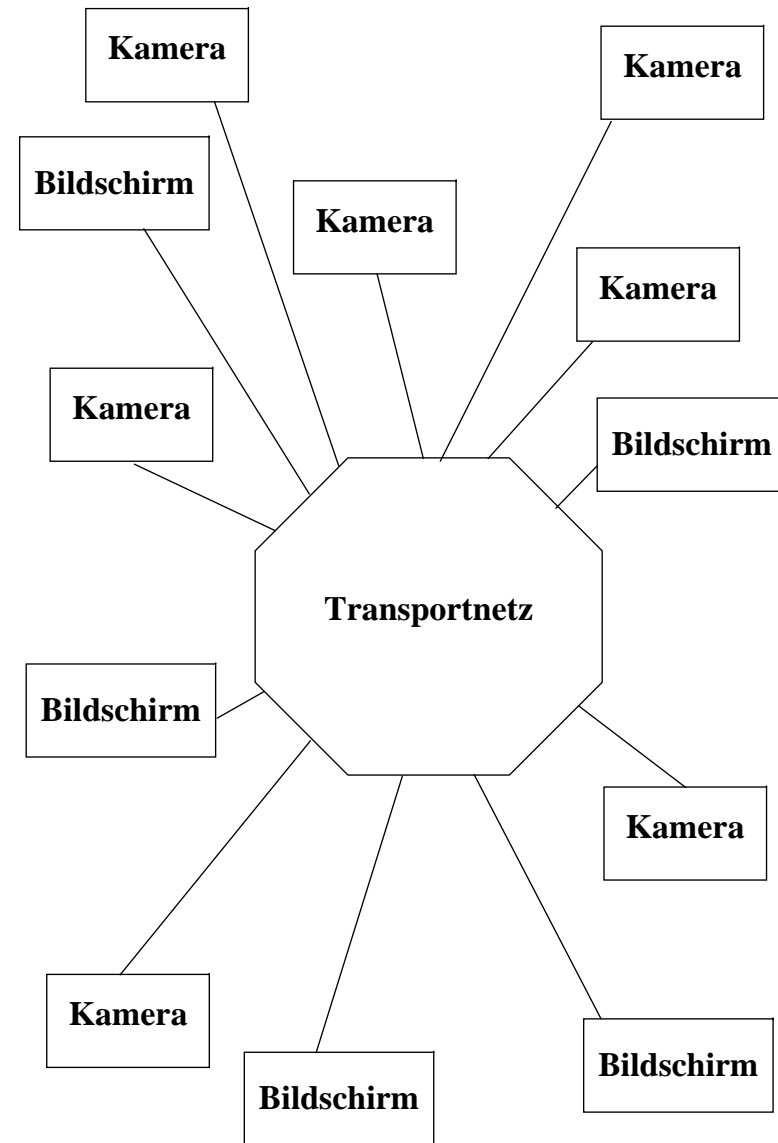
Hohe Bandbreiten: Bei einfacher Geschwindigkeit
liefert eine DVD 11,08 Mbit/s.

Hohe Rechenleistung: Eine MPEG-2 Decodierung
erfordert etwa 2 GOPS.

Fast exakte Synchronisation verschiedener Datenströme.

**Einfache repetitive Operationen, z. B. Transformation
von YUV-Format in RGB-Format:**

$$\begin{pmatrix} R \\ G \\ B \end{pmatrix} = \begin{pmatrix} 1,1644 & 0 & 1,5696 \\ 1,1644 & -0,39 & -0,81 \\ 1,1644 & 2,01 & 0 \end{pmatrix} \times \begin{pmatrix} Y-16 \\ U-128 \\ V-128 \end{pmatrix}$$



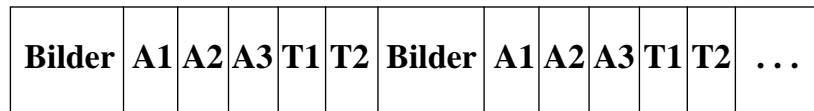
**Schema einer Videokonferenz,
Kompression und Dekompression der Daten**

Video on Demand:

Ein Film besteht aus

einer Bildfolge,
mehreren Tonfolgen (Musik, Sprachen),
mehreren Textfolgen (Untertitel).

Verschachtelte Speicherung:



A_i = Audiospuren, T_i = Textspuren

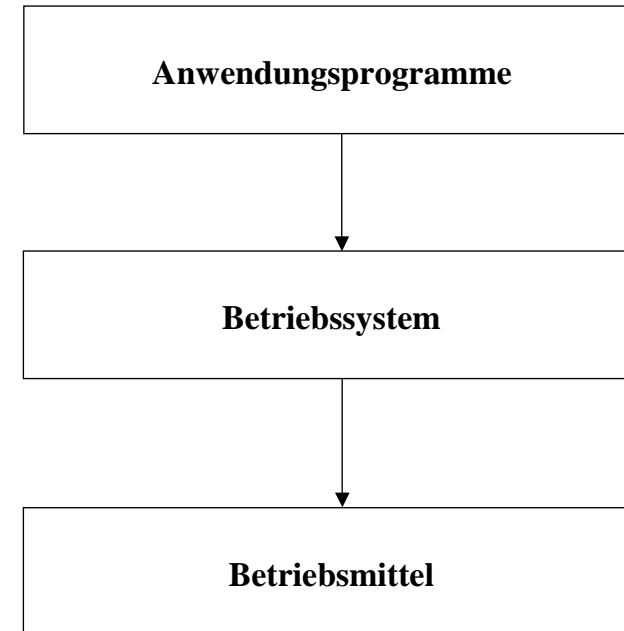
Probleme für den Anbieter:

Viele gleichzeitige Videoströme,
Nachbildung der Funktionen eines Abspielgerätes:
Anhalten eines Films,
(schnelles) Rückspulen und Neubeginn,
(schnelles) Vorspulen.

Zwei Server-Modelle:

- Pull-Server:** Der Konsument erhält das Video nur stückweise, die Folgestücke werden explizit angefordert.
- Push-Server:** Der Konsument erhält das vollständige Video in einem Strom.

Aufbau eines Rechensystems:



Bemerkungen:

- (i) Ein Betriebssystem stellt eine einfache Schnittstelle zur Ausführung von Anwendungsprogrammen dar. Insbesondere verbirgt es die Eigenarten der Hardware vor dem Nutzer.
- (ii) Jedem Betriebsmittel ordnet man einen Verwalter zu.

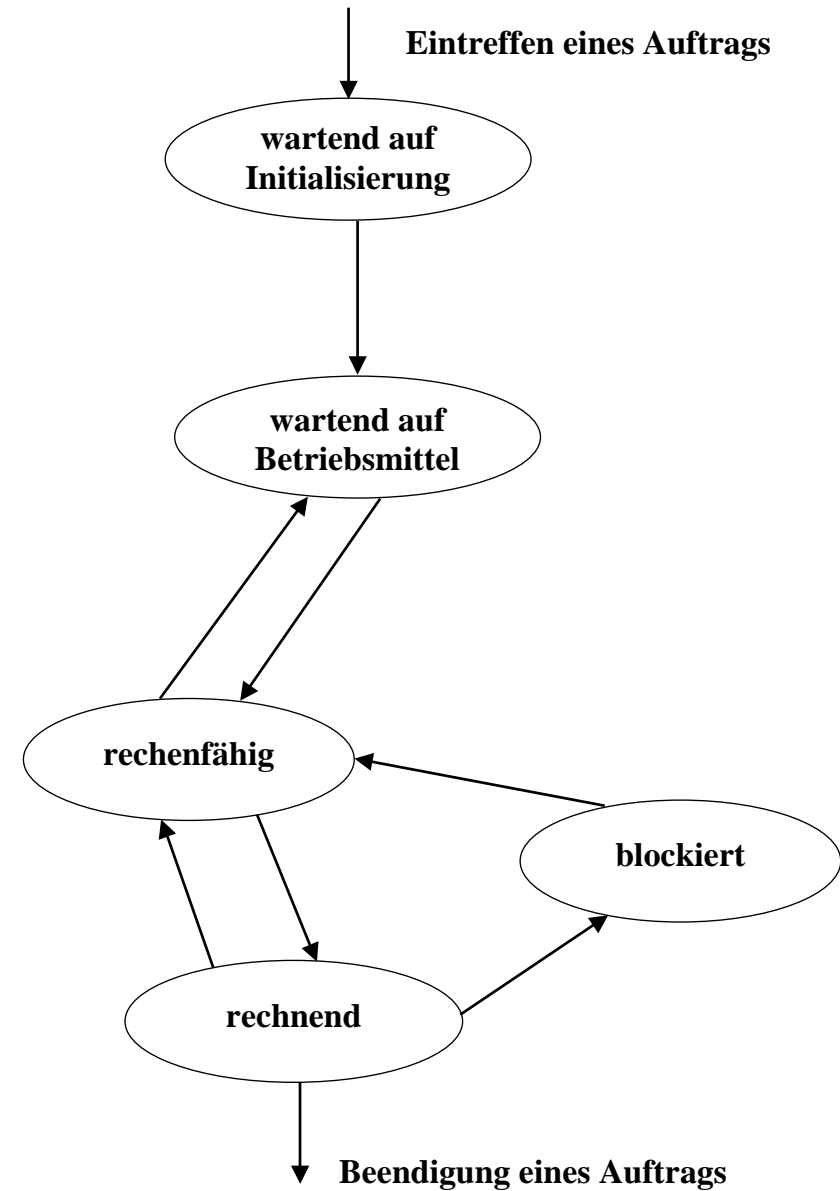
Hauptaufgaben eines Betriebssystems:

Anbieten einer benutzerfreundlichen Schnittstelle,
Betriebsmittelverwaltung,
langfristige Datenhaltung,
Kommunikation,
Ein- und Ausgabe,
Datenschutz,
Schutz gegen Fehlfunktionen,
Verfügbarmachung statistischer Daten.

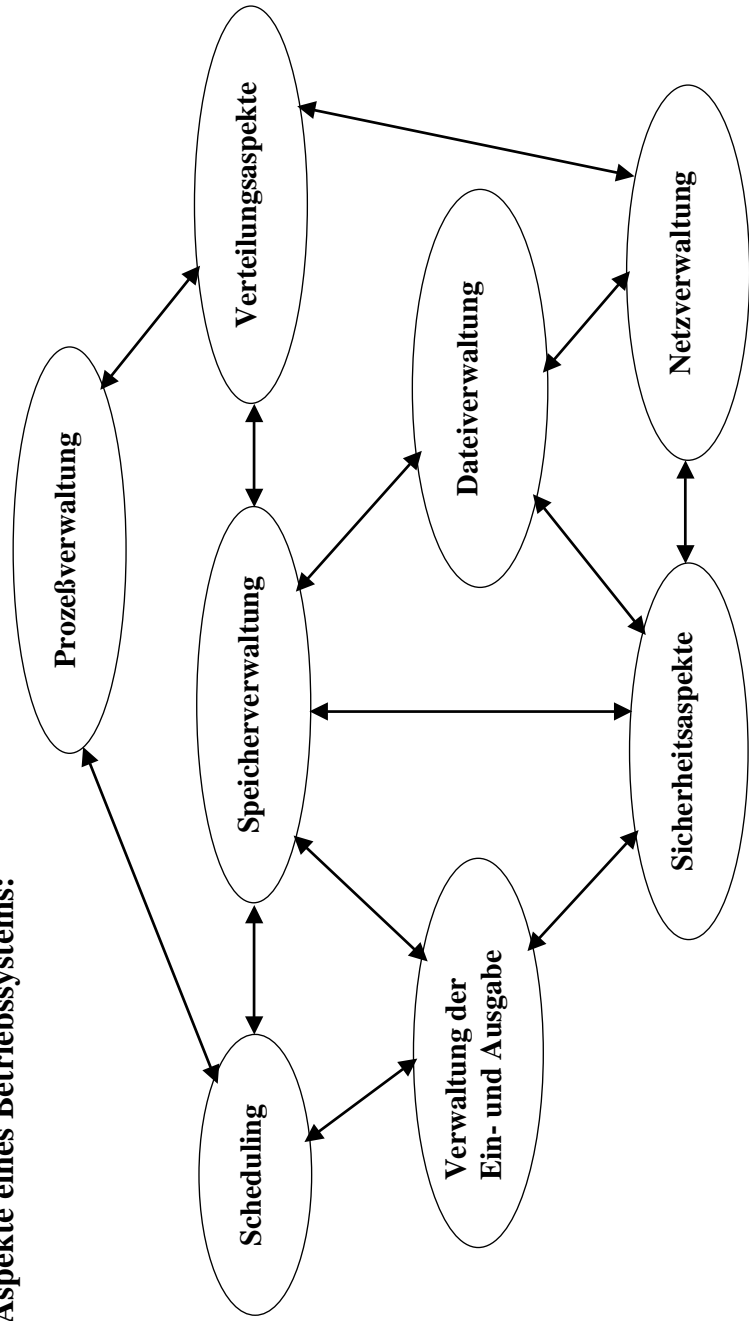
Bewertungskriterien für Betriebssysteme:

Effizienz,
Funktionsumfang,
Einfachheit,
Erweiterbarkeit,
Schutzfähigkeit,
Echtzeitfähigkeit.

Verwaltung von Aufträgen in einem Rechesystem:

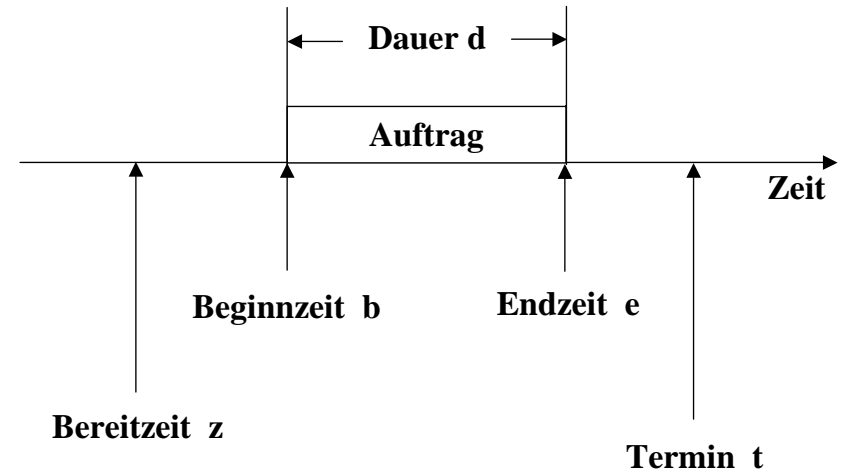


Aspekte eines Betriebssystems:



Arbeitsauftrag:

Die Zeiten eines Arbeitsauftrags:



Es gelten folgende Beziehungen:

- (i) $z \leq b$
- (ii) $b + d = e$
- (iii) $e \leq t$

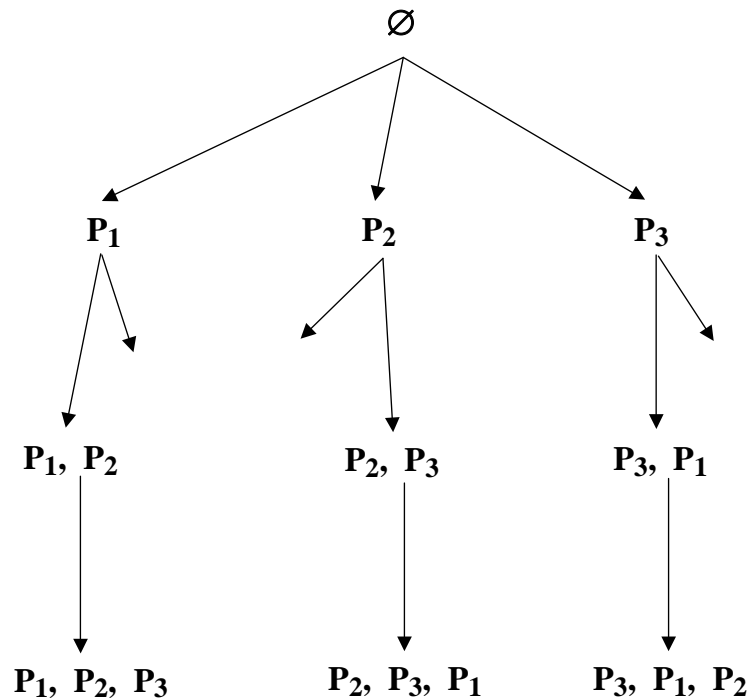
Bezüglich des Fertigstellungstermins unterscheidet man drei Regime:

- (i) Der Endtermin ist unbedingt einzuhalten.
- (ii) Der Endtermin kann geringfügig überschritten werden.
- (iii) Der Endtermin ist weitgehend belanglos.

**Kennzeichnung eines Prozesses durch drei Zeiten:
(Bereitzeit, Dauer, Endtermin).**

Gegeben seien drei Prozesse P_1 , P_2 , P_3 :

Ausschnitt aus Planungsbaum:



Bemerkung: Die Zahl der zu untersuchenden Pläne ist $n!$.

Existieren strikte Randbedingungen, dann ist die Zahl der zu untersuchenden Pläne bedeutend geringer. Gegeben seien die folgenden drei Prozesse:

$$P_1 = (1, 4, 7)$$

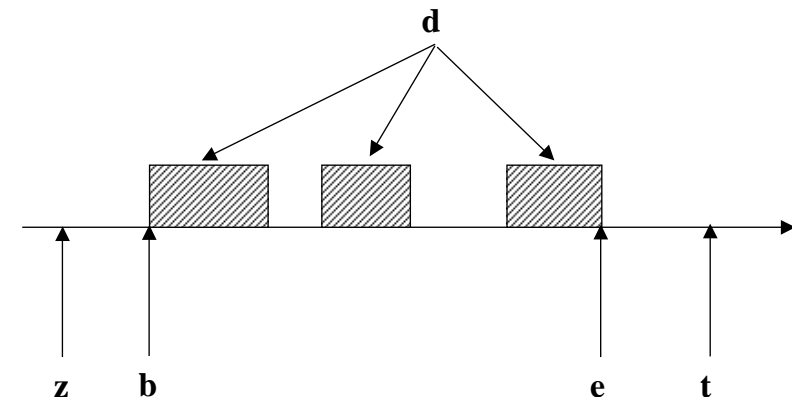
$$P_2 = (4, 5, 12)$$

$$P_3 = (0, 3, 14)$$

Eine Planung, die beim Zeitpunkt 0 beginnt, liefert nur die Folge P_3, P_1, P_2 . Eine Planung, die beim Zeitpunkt 1 beginnt, liefert die Folge P_1, P_2, P_3 .

Bemerkung: Für nicht unterbrechbare Prozesse ist das Planungsproblem NP-vollständig.

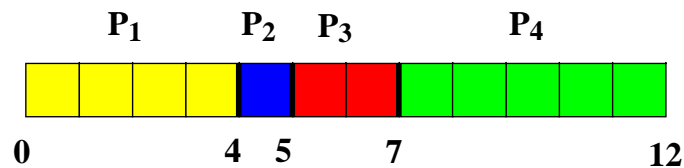
Illustration einer unterbrechbaren Ausführung:



Planen nach Endtermin:

Beispiel: Gegeben seien $P_1 = (0, 4, 5)$, $P_2 = (0, 1, 7)$,
 $P_3 = (0, 2, 7)$, $P_4 = (0, 5, 13)$.

Ein zulässiger Plan:



Algorithmus: Bei der Einplanung des nächsten Prozesses berücksichtigt man einen mit dem wichtigsten Endtermin.

Ein weiteres Beispiel: Gegeben seien $P_1 = (0, 2, 4)$,
 $P_2 = (3, 3, 14)$, $P_3 = (6, 3, 12)$,
 $P_4 = (5, 4, 10)$.

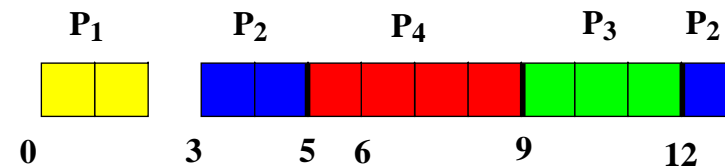
Zeitpunkt 0: Auswahl von P_1

Zeitpunkt 3: Auswahl von P_2

Zeitpunkt 6: Auswahl von P_4

Zeitpunkt 10: Auswahl von P_3 , Terminverletzung!

Betrachtet man die Prozesse als nicht unterbrechbar, dann existiert kein zulässiger Plan. Lässt man Unterbrechungen zu, dann ist der folgende Plan zulässig. Er wurde auch über das Prinzip "earliest deadline first" gefunden.



Bemerkung: Das Verfahren "earliest deadline first" findet immer einen zulässigen Plan, falls ein solcher existiert.

Planen nach Spielräumen:

Beispiel:

Gegeben seien die drei Prozesse, $P_1 = (0, 8, 10)$, $P_2 = (0, 5, 9)$ und $P_3 = (0, 4, 9)$; es stehen zwei Prozessoren zur Verfügung.

Eine Planerstellung nach dem frühesten Endtermin führt zu folgender Prozessorbelegung.

Prozessor 1: P_2

Prozessor 2: P_3, P_1

Es tritt eine Fristverletzung für Prozeß P_1 ein. Eine Planerstellung nach dem kleinsten Spielraum führt in diesem Fall zu einem zulässigen Schedule.

Prozessor 1: P_2, P_3

Prozessor 2: P_1

Das folgende Beispiel für zwei Prozessoren demonstriert, daß nach dem Prinzip des kleinsten Spielraums für nichtunterbrechbare Prozesse nicht immer ein zulässiger Schedule auffindbar ist, obgleich ein solcher existiert.

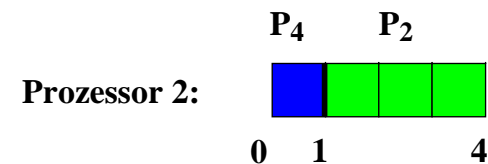
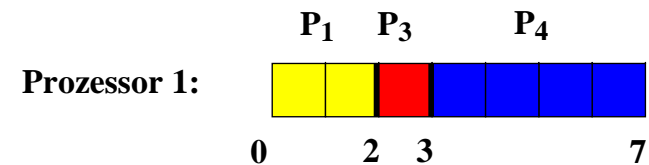
Beispiel: Gegeben seien $P_1 = (0, 1, 1)$, $P_2 = (0, 5, 6)$, $P_3 = (0, 3, 5)$, $P_4 = (0, 5, 8)$.

Bemerkung: Das Problem, nichtunterbrechbare Prozesse auf einem Mehrprozessorsystem einzuplanen, ist NP-vollständig.

Sind Prozesse unterbrechbar, dann liefert die Berücksichtigung der aktuellen Spielräume oft einen zulässigen Schedule.

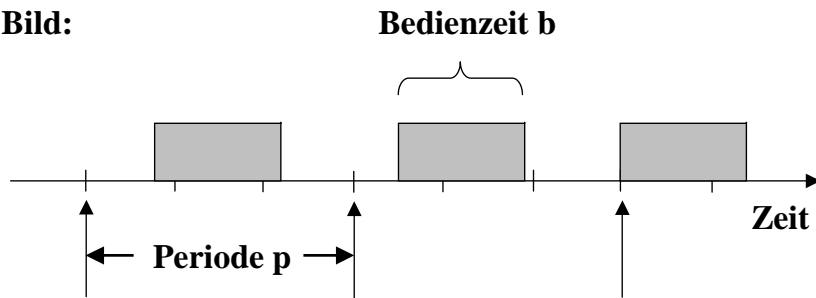
Beispiel: Gegeben seien $P_1 = (0, 2, 3)$, $P_2 = (1, 3, 4)$, $P_3 = (2, 1, 4)$, $P_4 = (0, 5, 7)$.

Ein möglicher Schedule:



Übergang zu zyklischen Systemen:

Bild:

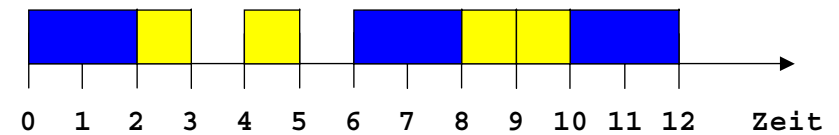


Beispiel: 2 unabhängige, periodische Aufträge

	Periode	Bedienzeit	Symbol
A1	4	2	■
A2	3	1	■

Bemerkung: Alle Aufträge sind zum Zeitpunkt 0 lauffähig, dies ist der ungünstigste Fall. Ein Schedule heißt gültig, falls alle Endtermine eingehalten werden.

Der folgende Schedule ist gültig.



Bemerkungen:

- (i) Da 12 das kleinste gemeinsame Vielfache von 3 und 4 ist, ist der obige Schedule für beliebige Zeiträume gültig.
- (ii) Der Beispielschedule zeigt auch die großen Freiheiten, die ein Schedule-Ersteller hat.

Randbedingungen beim Echtzeitscheduling:

1. Es existiert eine feste Zahl unabhängiger Aufträge.
2. Die Bedienzeiten der einzelnen Aufträge sind konstant.
3. Die einzelnen Aufträge werden periodisch aufgerufen.
4. Die Aufrufintervalle sind fest.
5. Jeder Auftrag muß innerhalb seines Aufrufintervalls abgearbeitet sein.

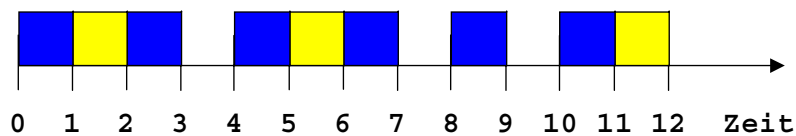
Beispiel zum Scheduling nach festen Prioritäten:

Beispiel: 2 periodische Aufträge

	Periode	Bedienzeit	Symbol
A1	2	1	■
A2	5	1	■

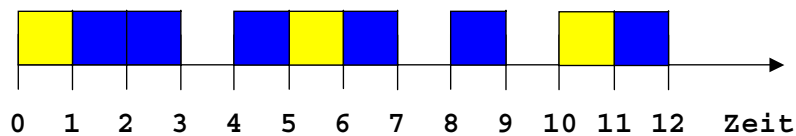
Fall 1: Priorität (A1) > Priorität (A2) ("vor")

Schedule:



Fall 2: Priorität (A2) > Priorität (A1) ("vor")

Schedule:



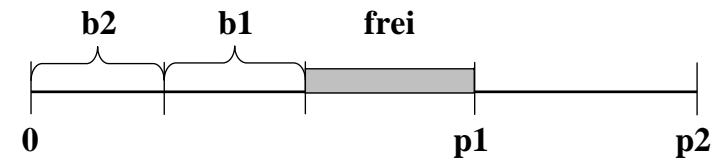
Bemerkungen:

- (i) Im Fall 1 können die Bedienzeiten für A1 und A2 erhöht werden, im Fall 2 nicht.
- (ii) Bei Scheduling nach festen Prioritäten erhält man einen optimalen Schedule, falls man die Prioritäten nach fallenden Perioden vergibt.

Graphische Demonstration:

Gegeben seien 2 zum Zeitpunkt 0 lauffähige Aufträge A1(p1, b1) und A2(p2, b2) mit p1 < p2; A2 habe die größere Priorität. Es muß gelten:

$$b2 + b1 \leq p1$$



Eine andere Anordnung der Prozesse im Intervall [0, p1) ändert nichts wesentliches!

- (iii) In ihrer Arbeit "Scheduling Algorithms for Multiprogramming in a Hard-Real-Time Environment", JACM 20, 1 (1973) bewiesen Liu und Layland die folgende Aussage: Die kleinste obere Schranke der Prozessornutzung U bei Scheduling nach festen Prioritäten ist bei m Aufträgen:

$$U = m * (2^{1/m} - 1)$$

(iv) **Prozessornutzung** $U = \sum_{i=1}^m \frac{b_i}{p_i} \leq 1$

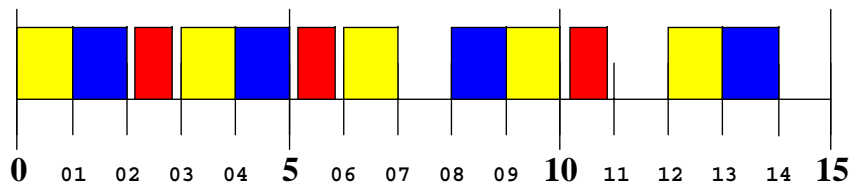
(v) $\lim_{m \rightarrow \infty} m \cdot (2^{1/m} - 1) = \ln 2$

(vi) **Da $\ln 2 \approx 0,69$, muß in Extremfällen bei Scheduling nach festen Prioritäten auf etwa 30% der Rechenleistung verzichtet werden. Andererseits erfordert ein Scheduling nach festen Prioritäten kaum Verwaltungsaufwand.**

Beispiel: 3 Aufträge

	Periode	Bedienzeit	Symbol
A1	3	1	■
A2	4	1	■
A3	5	X	■

Schaubild eines "rate monotonic" Schedules:



Beobachtungen:

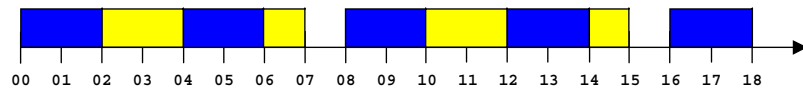
- (i) **Der maximale Wert für die Bedienzeit des Auftrags 3 ist 1. Dies entnimmt man dem Zeitintervall von 2 bis 3. Die Zeitintervalle 7 bis 8, 11 bis 12 und 14 bis 15 sind wegen der Monotonie-Forderung nicht nutzbar.**
- (ii) **Setzt man die Bedienzeit für Auftrag 3 auf 1, dann erhält man eine Nutzung von $1/3 + 1/4 + 1/5 \approx 0,78$, dies entspricht in etwa $3 \cdot (2^{1/3} - 1) \approx 0,78$.**
- (iii) **Wählt man als Schedulingkriterium die Nähe zum Endtermin, dann läßt sich die Bedienzeit des Auftrags 3 auf 2 erhöhen.**

Vergleich zwischen termingesteuertem und ratenmonotonomem Scheduling:

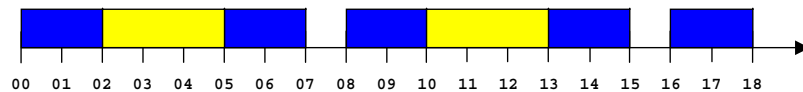
Kriterium: Zahl der Kontextwechsel

Auftrag	Periode	Bedienzeit	Symbol
A	4	2	■
B	8	3	■

RM-Schedule: pro 8 Zeiteinheiten eine Verdrängung.



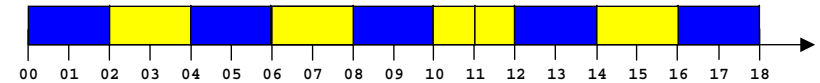
EDF-Schedule: keine Verdrängung notwendig.



Kriterium: Termintreue

Auftrag	Periode	Bedienzeit	Symbol
A	4	2	■
B	10	5	■

RM-Schedule: Terminverletzung bei Zeitpunkt 10.



EDF-Schedule: keine Terminverletzung.

