

# Übung zur Vorlesung ”Grundlagen Betriebssysteme und Systemsoftware”

(Prof. Dr. J. Schlichter, WS 2008 / 2009)

Übungsleitung: Dr. Georg Groh (grohg@in.tum.de)

Tutoren: Dipl. Inform. Vivian Prinz (prinzv@in.tum.de), Dr. Nils Kammenhuber (kammenhuber@net.in.tum.de), Dipl. Inform. Robert Schmohl (schmohl@in.tum.de), Dipl. Inform. Dipl. Geogr. Jan Herrmann (hermanj@in.tum.de), Dipl. Inform. Robert Eigner, David Brodski (brodski@in.tum.de), Yang Guo (yang.guo@gmx.de), Jan Finis (finis@in.tum.de), Martin Levihn (levihn@in.tum.de)

<http://www11.in.tum.de/Veranstaltungen/GrundlagenBetriebssystemeundSystemsoftware0809>

<http://www11.in.tum.de/Veranstaltungen/GrundlagenBetriebssystemeundSystemsoftware0809/uebung>

## Blatt 7

- Abgabe: bis 08.12.2008 12:00 Uhr per E-Mail an den Tutor der eigenen Gruppe. Die Mail soll einen Zip-Ordner als attachment haben, der für jede Hausaufgabe einen Unterordner enthält, in dem die Lösung als .txt-File(s), als .c-File(s) o.ä. enthalten ist.
- Musterlösungen Hausaufgaben: ab 08.12.2008 12:00 Uhr auf der Übungswebseite zum Download.
- Musterlösungen Tutoraufgaben: ab 15.12.2008 12:00 Uhr auf der Übungswebseite zum Download.

## Stoff

Es sei empfohlen folgende Literatur durchzuarbeiten:

- Skript Kapitel 3.6, 4.1, 4.2, 4.3
- Tanenbaum, Modern Operating Systems, 2001: Kapitel 2.5 (Scheduling); Kapitel 3 (Deadlocks)

Auf diesem Blatt werden im wesentlichen drei Themen behandelt: Deadlock Erkennung und Vermeidung, Deadlock Verhinderung und Process Scheduling.

## 1 Hausaufgabe (Process-Scheduling)

### Lernziele

Vertiefen und Verdeutlichen der verschiedenen Process-Scheduling Algorithmen.

## Aufgabe

In dieser Aufgabe sollen verschiedene Prozessorverwaltungs-Strategien (Scheduling-Strategien) verglichen werden. Als Modell sei dazu ein einfaches Ein-Bediener-System gegeben. Das heißt wir betrachten Aufträge mit Bedienzeiten (Rechenzeiten) sowie Ankunftszeiten (der Zeitpunkt, ab dem der Auftrag im System vorhanden ist und berechnet werden kann) und einen Prozessor, auf dem jeweils immer nur einer der Aufträge bearbeitet werden kann. Wir nehmen vereinfachend an, dass ein Prozesswechsel keine Zeit kostet.

In unserem Beispiel existieren sechs Aufträge  $A_1, \dots, A_6$  mit den Bedienzeiten  $b = (5, 6, 2, 3, 2, 1)$  (bedeutet  $A_1$  benötigt 5 Zeiteinheiten,  $A_2$  6 usw.). Der Vektor der Ankunftszeiten sei durch  $a = (0, 2, 3, 4, 6, 7)$  (bedeutet  $A_1$  ist ab Zeitpunkt 0,  $A_2$  ab 2 usw. in der Warteschlange) gegeben. Vor dem Zeitpunkt  $t = 0$  sei das Bedienungssystem leer.

Die Ausführung eines solchen Systems kann man sehr gut anhand des (beispielhaften) Diagramms in Abb 1 visualisieren, bei dem auf der x-Achse die Zeiteinheiten und auf der y-Achse die verschiedenen Aufträge angetragen sind. Dabei ist jeder Auftrag durch eine Linie von seinem Ankunftszeitpunkt bis zu seiner abgeschlossenen Berechnung eingetragen, wobei die Linie gestrichelt ist solange er wartet und durchgezogen, solange ihm der Prozess zugeteilt ist. Zur besseren Übersicht sind an den Linien zusätzlich die Bedienzeiten angetragen. (Hinweis: In dem Diagramm ist eine unbekannte Scheduling-Strategie realisiert.)

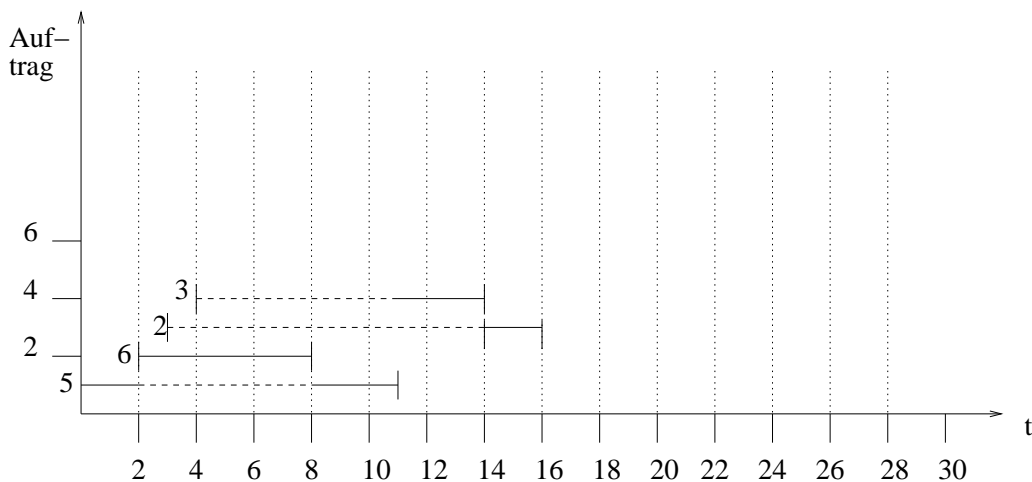


Figure 1: Diagramm des Bediener-Systems

### 1.1 Teilaufgabe

Zeichnen Sie entsprechende Diagramme für das angegebene Beispiel-Bediener-System für folgende Scheduling-Strategien:

1. FCFS (First Come First Served):

FCFS bedeutet, dass die Aufträge in der Reihenfolge ihrer Ankunft ausgeführt werden. Auftragsunterbrechung ist nicht möglich.

## 2. SRPT (Shortest Remaining Processing Time):

SRPT bedeutet, dass jeweils der Auftrag mit kürzester Restbedienzeit ausgeführt wird. Auftragswechsel sind nur bei Ankunftszeitpunkten neuer Aufträge möglich.

### 1.2 Teilaufgabe

Berechnen Sie die **mittlere Verweilzeit**  $\bar{V}$  und die **mittlere Wartezeit**  $\bar{W}$  für die beiden Strategien der vorangehenden Teilaufgabe. Nutzen Sie zur Berechnung folgende Formeln:

- $c_i$  bezeichnet die **Abgangszeit** (der Zeitpunkt seiner fertigen Berechnung) des Auftrags  $A_i$ ,  $a_i$  seine Ankunftszeit und  $b_i$  seine Bedienzeit;
- $v_i = c_i - a_i$  bezeichnet die **Verweilzeit** des Auftrags  $A_i$ ;
- $w_i = v_i - b_i = c_i - a_i - b_i$  bezeichnet die **Wartezeit** des Auftrags  $A_i$ ;
- Gibt es  $n \in \mathbb{N}$  Aufträge, so gilt für die **mittlere Verweilzeit**:

$$\bar{V} = \frac{1}{n} \sum_{i=1}^n v_i$$

- Für die **mittlere Wartezeit** gilt:

$$\bar{W} = \frac{1}{n} \sum_{i=1}^n w_i$$

- Bezeichnet  $\bar{b} = \frac{1}{n} \sum_{i=1}^n b_i$  die mittlere Bedienzeit, so gilt folgender Zusammenhang:

$$\bar{V} = \bar{W} + \bar{b}$$

### 1.3 Teilaufgabe

Der Vektor der Ankunftszeiten werde modifiziert zu  $a' = (0, 0, 0, 0, 0, 0)$ , d.h. die Aufträge liegen alle zu Beginn vor. Die Aufträge werden nun nach einer Zeitscheibenstrategie mit dem Quantum  $q$  bedient (Round-Robin Strategie), ein Auftrag wird also nach dem Zeitquantum  $q$  unterbrochen und der nächste in der angegebenen Reihenfolge setzt seine Ausführung fort. Die **zyklische** Reihenfolge, in der die Aufträge bedient werden, ist durch die Permutation  $\pi = (1, 2, 3, 4, 5, 6)$  festgelegt.

1. Berechnen Sie die mittlere Verweilzeit und die mittlere Wartezeit für die Quanten  $q_1 = \frac{3}{2}$  und  $q_2 = \frac{1}{2}$ . Visualisieren Sie dazu zunächst wieder die Abläufe als Diagramme!
2. Welchen Einfluß hat die (zyklische) Reihenfolge  $\pi$ , in der die Aufträge bedient werden, auf die mittlere Verweilzeit? (Abgabe: 1 Satz)
3. Welche Vorteile hat eine Zeitscheibenstrategie im Gegensatz zur Verweilzeit-optimalen Strategie? (Abgabe: 1-2 Sätze)
4. Überlegen Sie sich, wie sich die Größe des Zeitquantums in realen Systemen auswirkt! Argumentieren Sie in zwei Sätzen, welche Vor- und Nachteile eine Verkleinerung bzw. eine Vergrößerung des Quantums bei einer Zeitscheiben-Strategie ergibt.

## Abgabe

Entweder

- die ganze Aufgabe als Word2003-Datei (.doc) oder PDF (.pdf)

ODER

- die grafischen Teile als .ppt (Office 2003), .pdf, .jpg, .gif oder .png Dateien und die Textteile als .txt Dateien.

## 2 Hausaufgabe (Deadlocks und Deadlock-Verhinderung)

### Lernziele

Vertiefung des Wissens um Verklemmungen (Deadlocks), speziell Konzept des Belegungs-Anforderungs-Graphs (Resource-allocation-Graph), Möglichkeiten zur Verklemmungs-Verhinderung (Deadlock-Prevention).

### Aufgabe

Es soll die Situation modelliert werden, dass an einer Kreuzung in Deutschland ohne Verkehrszeichen, gleichzeitig vier Autos ankommen. Nehmen Sie an, dass es sich bei den vier Fahrern jeweils um Spinner handelt, die auf JEDEN schießen, der Ihnen ihr Vorfahrtsrecht wegnimmt.

#### 2.1 Teilaufgabe

Geben Sie einen Belegungs-Anforderungs-Graph an, der ein Deadlock modelliert! Benennen Sie Ihre Ressourcen entsprechend aussagekräftig! Überprüfen Sie die 4 Deadlock Bedingungen (Abgabe: 4 Stichpunkte)! Hat der Graph Zyklen?

#### 2.2 Teilaufgabe

Was kann man tun um in dieser Situation Deadlocks zu vermeiden? Nennen Sie dazu für jede der vier Deadlock Bedingungen eine mögliche vermeidende Maßnahme!

## Abgabe

Entweder

- die ganze Aufgabe als Word2003-Datei (.doc) oder PDF (.pdf)

ODER

- die grafischen Teile als .ppt (Office 2003), .pdf, .jpg, .gif oder .png Dateien und die Textteile als .txt Dateien.

### 3 Hausaufgabe (Deadlock-Vermeidung)

#### Lernziele

Vertiefen des Banker's-Algorithmus.

#### Aufgabe

Führen Sie den Banker's-Algorithmus an zwei Beispielen für einzelne und einem Beispiel für mehrfache Ressourcen durch! Programmieren Sie entweder den Algorithmus in C oder beantworten Sie die Teilaufgaben 3.1, 3.2 und 3.3!

Wir gehen dazu aus von  $n$  Prozessen und  $m$  Ressourcen. Weiterhin seien die existierenden Zahlen  $e_i \in \mathbb{N}_0$  der  $m$  Ressourcen durch das Tupel  $(e_1, e_2, \dots, e_m)$  gegeben und die  $m$  Zahlen der im Moment noch freien Ressourcen durch das Tupel  $(a_1, a_2, \dots, a_m)$ . Die aktuelle Belegung von Ressourcen durch Prozesse sei durch die  $n \times m$  Matrix  $c_{ij} \in \mathbb{N}_0$  und die aktuelle maximale Anforderung von Ressourcen durch Prozesse durch die  $n \times m$  Matrix  $r_{ij} \in \mathbb{N}_0$ .

#### 3.1 Teilaufgabe

Klarerweise sollte gelten:

$$\forall j : \sum_{i=1}^n c_{ij} + a_j = e_j \quad (1)$$

und

$$\forall i, j : c_{ij} + r_{ij} \leq e_j \quad (2)$$

Warum ist dies sinnvoll? (jeweils 1 Satz)

#### 3.2 Teilaufgabe

Sind folgende Systemzustände sicher (safe) (safe heisst: das System ist nicht verklemmt (deadlocked) und es gibt eine Ausführungsreihenfolge der Prozesse in der kein Deadlock auftritt auch wenn alle Prozesse alle noch anzufordernden Ressourcen anfordern)? Geben Sie jeweils in knappen Strichpunkten die Ausführung des Banker's Algorithmus an! Geben Sie ggf. die Ausführungsreihenfolge der Prozesse an!

1.  $m = 1; n = 4; c_1 = (2, 2, 4, 8)^T; r_1 = (10, 8, 4, 6)^T; e = 20; a = 4$

2.  $m = 1; n = 4; c_1 = (2, 4, 4, 8)^T; r_1 = (10, 6, 6, 12)^T; e = 20; a = 2$

3.  $m = 4; n = 5; c = \begin{pmatrix} 6 & 0 & 2 & 2 \\ 0 & 2 & 0 & 0 \\ 2 & 2 & 2 & 0 \\ 2 & 2 & 0 & 2 \\ 0 & 0 & 0 & 0 \end{pmatrix}; r = \begin{pmatrix} 2 & 2 & 0 & 0 \\ 0 & 2 & 2 & 4 \\ 6 & 2 & 0 & 0 \\ 0 & 0 & 2 & 0 \\ 4 & 2 & 2 & 0 \end{pmatrix}; e = (12, 6, 8, 4); a = (2, 0, 4, 0)$

### **3.3 Teilaufgabe**

Warum wird der Banker's Algorithmus in aktuellen Betriebssystemen nicht verwendet? (1 Satz)

#### **Abgabe**

Die ganze Aufgabe als Word2003-Datei (.doc) oder PDF (.pdf)

## 4 Tutoraufgabe (Scheduling in 4.3 BSD UNIX)

### Abgabe

Die Aufgabe wird in den Tutorübungen gemeinsam erarbeitet. Die Aufgabe soll NICHT abgegeben werden.

### Lernziele

Es soll die Scheduling Strategie eines realen Betriebssystems behandelt werden. Ein BSD Unix Derivat ist z.B. Grundlage für das beliebte Mac OS X.

### Aufgabe

Eine Strategie für die Prozessorverwaltung besteht darin, an die Prozesse **Prioritäten** zu vergeben. Der Prozessor wird dann jeweils dem Prozess mit der höchsten Priorität zugeteilt. Die Prioritätenvergabe kann dabei statisch oder dynamisch erfolgen. Im ersten Fall hat jeder Prozess für die Dauer seiner Existenz eine feste Priorität; im zweiten Fall können sich die Prioritäten der Prozesse dynamisch verändern, d.h. sie werden in gewissen Zeitabständen neu berechnet.

Bei einer **Zeitscheibenstrategie** (Round-Robin-Strategie) werden die Prozesse an den Prozessor jeweils für ein festgelegtes Zeitquantum (in 4.3 BSD Unix beträgt z.B. die Zeitscheibe 100 ms) gebunden und spätestens nach dem Ablauf dieser Zeitspanne wird den Prozessen der Prozessor wieder entzogen.

Zeitscheibenstrategien und Prioritätenvergabe können zu effizienten Verwaltungsstrategien kombiniert werden.

In dieser Aufgabe wird das Scheduling in dem Betriebssystem **4.3 BSD Unix** genauer betrachtet. Bei dem Unix-Scheduling handelt sich um eine Zeitscheibenstrategie mit dynamischer Prioritätenvergabe. Unix vergibt für seine Prozesse Prioritäten von 0 - 127 (0 ist die höchste Priorität), die in 32 Warteschlangen verwaltet werden. Ein Prozess mit Priorität *PRIO* wird in die Schlange *PRIO/4* eingeordnet. Alle Prozesse einer Prioritätsklasse befinden sich in einer Warteschlange, die nach einer Round-Robin Strategie abgearbeitet wird. Zunächst wird allen Prozessen der Warteschlange mit höchsten Prioritäten die CPU zugeteilt bis die Warteschlange leer ist. Dann kommen die Prozesse aus der Warteschlange mit den nächstniedrigeren Prioritäten zum Zuge.

Die Prioritäten werden jedoch fortlaufend neu berechnet (multilevel-feedback-queue). Die Prioritäten der Prozesse, die in einem gewissen Zeitabschnitt viel Rechenzeit verbraucht haben, werden erniedrigt; Prozesse, die lange gewartet haben, erhalten eine höhere Priorität ('short-term-scheduling'). Die Prioritäten werden folgendermaßen berechnet:

$$(1) u_{prio} = \text{USER\_PRIO} + p_{cpu} / 4 + 2 * p_{nice} ,$$

wobei  $p_{cpu}$  die Prozessornutzung des rechnenden Prozesses ist und alle 10 ms um 1 inkrementiert wird.  $p_{nice}$  ist ein vom Benutzer bestimmter Gewichtungsfaktor ( $-20 \leq p_{nice} \leq 20$ ) und  $\text{USER\_PRIO}$  ist die Priorität, die dem Prozess beim Start zugeteilt worden ist.  $p_{cpu}$  wird jede Sekunde angepasst durch:

$$(2) p\_cpu = ((2 * load) / (2 * load + 1)) * p\_cpu + p\_nice,$$

wobei *load* eine Abschätzung der CPU-Auslastung ist. Die Anpassung (2) sorgt dafür, dass bisher verbrauchte Rechenzeit nach einer gewissen Zeit nicht mehr ins Gewicht fällt.

#### 4.1 Teilaufgabe

In dieser Teilaufgabe sollen Sie sich das Unix-Scheduling an einem kleinen Beispiel verdeutlichen. Gegeben seien dazu folgende Prozesse mit ihrer Bedienzeit und Anfangspriorität:

Prozess	Bedienzeit	Priorität
1	10	3
2	1	1
3	2	3
4	1	4
5	5	2

Zeichnen Sie die Abarbeitungsfolge der Prozesse, die sich bei Anwendung der oben erläuterten Strategie des 'short-term-schedulings' ergibt, geeignet auf. Zur Vereinfachung nehmen Sie an, dass ein Zeitquantum den Wert 1 hat, *p\_cpu* (des rechnenden Prozesses!) in jedem Zeitquantum um 8 erhöht wird, *p\_nice* den Wert 0 hat und *load* gleich 1 ist. Die Priorität aller Prozesse soll nach jedem *vierten Quantum* neu berechnet werden.

#### 4.2 Teilaufgabe

Welche Beweggründe stehen hinter dem 'short-term-scheduling'?

### 5 Tutoraufgabe (Quizfragen zu Tanenbaum)

#### Abgabe

Die Aufgabe wird in den Tutorübungen gemeinsam erarbeitet. Die Aufgabe soll NICHT abgegeben werden.

#### Lernziele

Das Thema Deadlocks soll weiter vertieft werden.

#### Aufgabe

Beantworten Sie die folgenden Fragen!

### Teilaufgabe

In einem System seien 4 Prozesse, 5 Ressourcen, folgende momentane Belegungen  $c = \begin{pmatrix} 1 & 0 & 2 & 1 & 1 \\ 2 & 0 & 1 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 \\ 1 & 1 & 1 & 1 & 0 \end{pmatrix}$ ;

und folgende maximale Restanforderungen  $r = \begin{pmatrix} 0 & 1 & 0 & 1 & 1 \\ 0 & 2 & 1 & 0 & 0 \\ 1 & 0 & 3 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 \end{pmatrix}$ ; und folgender Vektor von

noch freien Ressourcen  $a = (0, 0, x, 1, 1)$ .

Was ist der kleinste wert von  $x$  für den dieser Systemzustand sicher ist?

### Teilaufgabe

Zwei Prozesse A und B brauchen jeder 3 Datensätze d1, d2, d3 aus einer Datenbank. Wenn A diese in der Reihenfolge d1, d2, d3 anfragt und B sie in der gleichen Reihenfolge anfragt, kann es keinen Deadlock geben. Wenn B in der Reihenfolgen d3, d2, d1 fragt, ist Deadlock möglich. Mit 3 Ressourcen gibt es 3! mögliche Anfrage-Reihenfolgen für jeden Prozess. Welche Kombinationen sind garantiert Deadlock-frei?

### Teilaufgabe

Analysieren sie die Situation der vorangegangenen Teilaufgabe unter Verwendung von Two-Phase-Locking. Eliminiert dies die Möglichkeit von Deadlocks? Hat dies evtl. unerwünschte Nebenwirkungen? Wenn ja, welche?

### Teilaufgabe

Nennen und erläutern Sie kurz die 4 Deadlock-Bedingungen! (Klausur-Teilaufgabe aus dem letzten Jahr)

### Teilaufgabe

Machen Sie sich den Unterschied zwischen Verklemmungs Verhinderung (Deadlock Prevention) und Verklemmungs Vermeidung (Deadlock Avoidance) klar!