

Prozesse und Scheduling

Aufgabe 1 : Scheduling

Gegeben seien die folgenden Prozesse und die Längen des jeweiligen CPU-Bursts:

Prozess	Zeitbedarf	Priorität
P1	10	2
P2	1	4
P3	2	2
P4	1	1
P5	5	3

Nehmen Sie nun an, dass alle Prozesse zum Zeitpunkt $t=0$ in der Reihenfolge P1, P2, P3, P4, P5 ankommen. Vernachlässigen Sie dabei die Scheduler-Aufrufzeit sowie die Prozessumschaltzeit. Priorität: Hohe Zahl bedeutet hohe Priorität. Bei gleicher Priorität werden die Prozesse in der Reihenfolge P1, P2, ... abgearbeitet.

- Untersuchen Sie, wie diese Prozesse mit den vier Algorithmen FCFS, SJF, nicht-unterbrechbare Prioritäten-Scheduling und RR (Zeitscheibe=2) ausgeführt werden. Zeichnen Sie dazu für jeden Algorithmus ein Gantt-Diagramm, das den Ablauf der Prozesse darstellt.
- Was ist die Wartezeit für jeden Prozess für jeden der Algorithmen?

FCFS:

Beginn->	0	10	11	13	14	Wartezeit
P1	10					0
P2		1				10
P3			2			11
P4				1		13
P5					5	14
						$\Sigma=48$, mittlere Wartezeit $48/5=9,6$

SJF:

Beginn->	0	1	2	4	9	Wartezeit
P1					10	9
P2	1					0
P3			2			2
P4		1				1
P5				5		4
						$\Sigma=16$, mittlere Wartezeit $16/5=3,2$

Andere Reihenfolge (P4,P2,...) möglich, jedoch gleiche mittl. Wartezeit

Prioritäten-Scheduling:

Beginn->	0	1	6	16	18	Wartezeit
P1			10			6
P2	1					0
P3				2		16
P4					2	18
P5		5				1
						$\Sigma=41$, mittlere Wartezeit $41/5=8,2$

Andere Reihenfolge (P2,P5,P3,P1,P4) möglich, mit anderer mittl. Wartezeit = 6,6

RR (Zeitscheibe=2)

Beginn->	0	2	3	5	6	8	10	12	14	15	17	Wartezeit
P1	2					2		2		2	2	6+2+1=9
P2		1										2
P3			2									3
P4				1								5
P5					2		2		1			6+2+2=10
												$\Sigma=29$, mittlere Wartezeit $29/5=5,8$

c) Welche Strategie liefert das beste Ergebnis, wenn jeweils die mittlere Wartezeit minimiert werden soll?

Die Wartezeit wird mit SJF optimiert.

Aufgabe 2 : Echtzeit-Scheduling

Ein Echtzeitsystem hat vier periodische Tasks mit den Perioden 50, 100, 200 und 250 ms. Angenommen, die vier Tasks benötigen 35, 20, 10 und x ms Laufzeit. Ab welchem Wert von x sind die Tasks auf keinen Fall mehr zuteilbar?

Die Summe der CPU-Lasten der Tasks darf 100 % nicht überschreiten:

Task 1: $35/50 = 70\%$

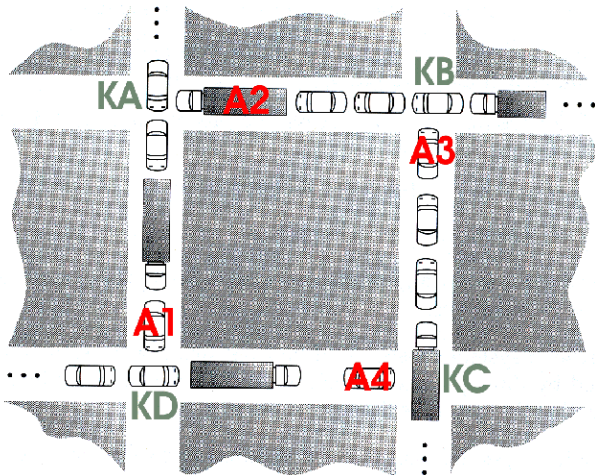
Task 2: $20/100=20\%$

Task 3: $10/200=5\%$

Summe: 95%. Die vierte Task darf also 5 % CPU Last nicht überschreiten -> Laufzeit x muss kleiner gleich 12,5 ms sein ($12,5/250=5\%$).

Betriebsmittelverwaltung

Aufgabe 3 : Deadlock an der Kreuzung



Auf dem Bild ist ein Deadlock an einer Verkehrskreuzung gezeigt.

- a) Zeigen Sie, dass alle 4 nötigen Bedingungen für ein Deadlock erfüllt sind.
- b) Überlegen Sie sich eine einfache Regel, wie Deadlocks in dieser Situation vermieden werden können.

Die Ressourcen in dem Bild sind die 4 Kreuzungen (KA,KB, KC, KD). Die Kreuzungen werden von Autos A1, A2, A3 und A4 (genauer: Auto-Karawanen) belegt.

Es gilt:

- 1. Mutual Exklusion: nur ein Auto kann zu einer Zeit die Kreuzung passieren.*
- 2. Hold and wait: die Autokarawanene belegt zunächst eine Kreuzung und braucht dann die zweite Kreuzung, um passieren zu können*
- 3. No Preemption: Eine Autokarawane, die eine Kreuzung belegt, gibt diese nicht wieder frei*

4. Circular Wait:

- Die Autokarawane A2 wartet auf Kreuzung KA, die von der Autokarawane A1 belegt ist;
- Die Autokarawane A1 wartet auf KD, die von A4 belegt ist
- A4 wartet auf KC, das von A3 belegt ist
- A3 wartet auf KB, das von A2 belegt ist

Einfache Regeln, das Deadlock zu vermeiden:

Zu 1: Brücke bauen anstelle von Kreuzung

Zu 2: Vorfahrtsregelung oder Ampelschaltung: wenn ein Auto in die erste Kreuzung einfährt, hat es an der zweiten Kreuzung automatisch Vorfahrt

Zu 3: Blockiert die Kreuzung, so müssen die hinteren Autos die (jeweils erste) Kreuzung wieder freigeben. Eine Variante davon wird z.B. in New York praktiziert: der Bereich im inneren einer Kreuzung (rotes Rechteck links oben) ist tabu. Hat ein Auto grün, so darf es nur in die Kreuzung einfahren, wenn es die Kreuzung wieder verlassen kann. Schaltet die Ampel auf Rot, und das Auto bleibt auf dem roten Viereck stehen, wird eine kräftige Strafe fällig. Vor dieser Regelung war ganz New York zur Rush Hour ein Deadlock!

Zu 4: (fällt mir keine einfache Regel ein)

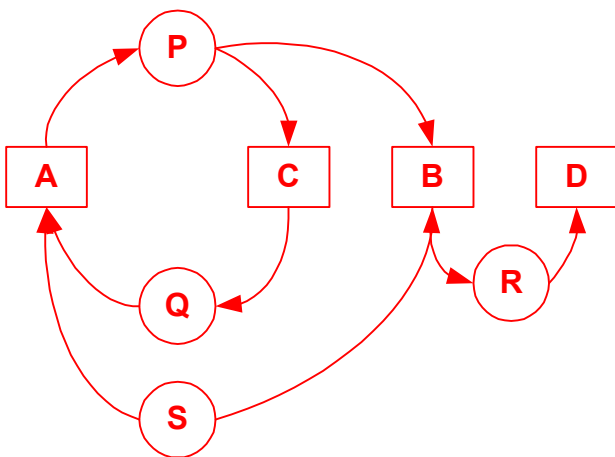
Aufgabe 4 : Betriebsmittelbelegungen

Die Prozesse P, Q, R und S benötigen jeweils Ressourcen A, B, C und D.

Gegeben ist die folgende Situation:

- Prozess P hat A belegt und möchte B und C belegen.
- Prozess Q hat C belegt und möchte A belegen.
- Prozess R hat B belegt und möchte D belegen.
- Prozess S möchte A und B belegen.

a) Zeichnen Sie einen Betriebsmittelbelegungsgraphen der obigen Situation. Befindet sich das System in einem Deadlock?



Das System befindet sich in einem Deadlock: $P \rightarrow C \rightarrow Q \rightarrow A \rightarrow P$

b) Terminieren Sie Prozess Q und geben Sie alle von Q gehaltenen Ressourcen frei. Wie entwickelt sich das System weiter? Entsteht dabei ein Deadlock oder nicht?

Zwischen P und S kann, muss aber nicht, ein neues Deadlock entstehen: $P \rightarrow B \rightarrow S \rightarrow A \rightarrow P$

c) Befindet sich das System unter b) in einem sicherem oder unsicherem Zustand?

Unsicher, da ein Deadlock möglich ist.

Prozesserzeugung und -synchronisierung

- ➔ Prozesserzeugung mit fork() und CreateProcess() / Mini-Shellübung
- ➔ Win32-Übung zur Synchronisierung
- ➔ Java-Threads-Übung

Aufgabe 5: Semaphoren mit Java

Java bietet zur Synchronisierung von Threads das `synchronized`-Schlüsselwort und die Funktionen `wait()` und `notify()` an (neben anderen Funktionen). Eine Firma möchte nun alten C-Code nach Java portieren, der zur Synchronisierung Semaphore verwendet. Dazu hat man folgende Java-Klasse entworfen, die Semaphore nachbilden sollen:

```
class Semaphore {
    private int counter;

    Semaphore(int start=0) {
        counter = start;
    }

    synchronized public void up() {
        // hier Code für up-Operation einfügen
        counter++;
        if( counter<=0 )
            notify();
    }

    synchronized public void down() {
        // hier Code für down-Operation einfügen
        counter--;
        if( counter<0 )
            wait();
    }
}
```

Ergänzen Sie den notwendigen Code.

Virtuelles Speichermanagement

Aufgabe 6: Seitenwechsel

Gegeben ist die folgende Seitenreferenzfolge:

1,2,3,4,2,1,5,6,2,1,2,3,7,6,3,2,1,2,3,6

Wie viele Seitenfehler treten jeweils für die unten genannten Seitenaustauschgorithmen auf, wenn dem Programm 1, 2, 3, 4, 5, 6 oder 7 Seitenrahmen zur Verfügung stehen? Beachten Sie, dass alle Seitenrahmen ursprünglich leer sind, und somit die ersten Seiten ebenfalls einen Seitenfehler erzeugen.

- a) LRU
- b) FIFO
- c) Optimal

1 Seitenrahmen:

a) LRU:

1	2	3	4	2	1	5	6	2	1	2	3	7	6	3	2	1	2	3	6
---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---	---

20 Seitenwechsel; FIFO und Optimal sind hier gleich.

2 Seitenrahmen:

a) LRU:

1	1	3	3	2	2	5	5	2	2	2	7	7	3	3	1	3	3		
	2	2	4	4	1	1	6	6	1	3	3	6	6	2	2	2	6		

18 Seitenwechsel

b) FIFO:

1	1	3	3	2	2	5	5	2	2	3	3	6	6	2	2	3	3		
	2	2	4	4	1	1	6	6	1	1	7	7	3	3	1	1	6		

18 Seitenwechsel

c) Optimal

1	1	3	4	1	5	6	1	3	3	3	3	1	3	6					
	2	2	2	2	2	2	2	2	7	6	2	2	2	2					

15 Seitenwechsel (die letzten 2 sind beliebig)

3 Seitenrahmen:

a) LRU:

1	1	1	4	4	5	5	5	1	1	7	7	2	2	2					
	2	2	2	2	2	6	6	6	3	3	3	3	3	3					
		3	3	1	1	1	2	2	2	2	6	6	1	6					

15 Seitenwechsel

b) FIFO:

1	1	1	4	4	4	6	6	6	3	3	3	2	2	2	6				
	2	2	2	1	1	1	2	2	2	7	7	7	1	1	1				
		3	3	3	5	5	5	1	1	1	6	6	6	3	3				

16 Seitenwechsel

c) Optimal

1	1	1	1	1	1	3	3	3	3	3									
	2	2	2	2	2	2	7	2	2	2									
		3	4	5	6	6	6	6	1	6									

11 Seitenwechsel (der letzte ist beliebig)

4 Seitenrahmen:

a) LRU:

1	1	1	1	1	1	1	1	6	6										
	2	2	2	2	2	2	2	2	2										
		3	3	5	5	3	3	3	3										
			4	4	6	6	7	7	1										

10 Seitenwechsel

b) FIFO:

1	1	1	1	5	5	5	5	3	3	3	3	1	1						
	2	2	2	2	6	6	6	6	7	7	7	7	3						
		3	3	3	3	2	2	2	2	6	6	6	6						
			4	4	4	4	1	1	1	1	2	2	2						

14 Seitenwechsel

c) Optimal

1	1	1	1	1	1	7	1												
	2	2	2	2	2	2	2												
		3	3	3	3	3	3												
			4	5	6	6	6												

8 Seitenwechsel

5 Seitenrahmen:

a) LRU:

1	1	1	1	1	1	1	1												
	2	2	2	2	2	2	2												
		3	3	3	6	6	6												
			4	4	4	3	3												
				5	5	5	7												

8 Seitenwechsel

b) FIFO:

1	1	1	1	1	6	6	6	6	6										
	2	2	2	2	2	1	1	1	1										
		3	3	3	3	3	2	2	2										
			4	4	4	4	4	3	3										
				5	5	5	5	5	7										

10 Seitenwechsel

c) Optimal

1	1	1	1	1	1	1													
	2	2	2	2	2	2													
		3	3	3	3	3													
			4	4	4	7													
				5	6	6													

7 Seitenwechsel (vorletzte Seite verschiedene Möglichkeiten)

6 Seitenrahmen:

a) LRU:

1	1	1	1	1	1	1													
	2	2	2	2	2	2													
		3	3	3	3	3													
			4	4	4	7													
				5	5	5													
					6	6													

7 Seitenwechsel

b) FIFO:

1	1	1	1	1	1	7	7	7	7										
	2	2	2	2	2	2	1	1	1										
		3	3	3	3	3	3	2	2										
			4	4	4	4	4	4	3										
				5	5	5	5	5	5										
					6	6	6	6	6										

10 Seitenwechsel

c) Optimal

1	1	1	1	1	1	1													
	2	2	2	2	2	2													
		3	3	3	3	3													
			4	4	4	7													
				5	5	5													
					6	6													

7 Seitenwechsel (letzter Seitenwechsel: 2 Möglichkeiten)

7 Seitenrahmen:

Alle Algorithmen benötigen 7 Seitenwechsel

Aufgabe 7: Seitenwechsel (2)

Ein Computer hat 5 Seitenrahmen. Für jede Seite sind in der folgenden Tabelle der Zeitpunkt des Ladens, des letzten Zugriffs und die R- und M-Bits angegeben:

Seite	Geladen zum Zeitpunkt	Letzter Zugriff	R- Bit	M- Bit
0	126	280	1	0
1	230	265	0	1
2	140	270	0	0
3	130	267	0	1
4	110	285	1	1

- a) Welche Seite wird FIFO ersetzen?
Seite 4 (älteste Seite)
- b) Welche Seite wird LRU ersetzen?
Seite 1 (letzter Zugriff am längsten her)
- c) Welche Seite wird Second Chance ersetzen?
Seite 3 (älteste Seite mit R=0)
- d) Welche Seite wird Enhanced Second Chance ersetzen?
Seite 2 (R=0, M=0: niedrigste Klasse)

Aufgabe 8: Arbeitsmenge (Working Set)

Ein Prozess bekommt vom Betriebssystem 4 Seitenrahmen Hauptspeicher zugeteilt. Das Betriebssystem untersucht alle 200 ms den Prozess und stellt fest, dass er jeweils verschiedene 4 Seiten pro Intervall benötigt.

- a) Können Sie ausschließen, dass sich der Prozess im Zustand des Seitenflatterns befindet?
Ja, da für alle benötigten Seiten Seitenrahmen zur Verfügung stehen.
- b) Später stellt das Betriebssystem fest, dass der Prozess 5 Seiten pro Intervall benötigt. Können Sie Seitenflattern immer noch ausschließen?
Nein. Seitenflattern kann, muss aber nicht auftreten.

Aufgabe 9: Virtueller Adressraum

- a) Bei Memory-Mapped Files wird die Datei mit Hilfe des virtuellen Speichermanagements in den virtuellen Adressspeicher des Prozesses abgebildet. Auf einem 32-Bit System beträgt der virtuelle Speicherraum maximal 4 GB. Angenommen, Sie besitzen solch einen Rechner mit 256 MB Hauptspeicher. Kann eine Video-Datei mit 1 GB Größe komplett als Memory-Mapped File in den Adressraum eingeblendet werden? Begründen Sie Ihre Antwort.

Ja, kann ziemlich wahrscheinlich eingeblendet werden, da der freie Adressraum groß genug für 1GB Adressen ist. Die Seiten (Daten) selbst werden je nach Bedarf durch das Virtuelle Speichermanagement in den Hauptspeicher eingeblendet.

- b) Windows XP unterteilt auf 32-Bit Rechnern den Adressraum in zwei Bereiche. Die unteren 2 GB (von 1MB bis 2GB) stehen für den User-Prozess zur Verfügung, die oberen 2 GB (2 GB – 4 GB) sind für das Betriebssystem reserviert. Kann eine Video-Datei der Größe 2 GB komplett in den Adressraum gemappt werden? Begründen Sie Ihre Antwort.

Nein, da weniger als 2 GB vorhanden sind (2GB-1MB=1,999GB; zudem werden Adressen für Programmcode, Stack, Daten, etc. benötigt, die aus dem User-Space vergeben werden)

- c) Wie können Sie erreichen, dass auch große Dateien (z.B. eine Video-Datei mit 9 GB) komplett in den Adressraum eingeblendet werden kann?

Größerer Adressraum, z.B. durch einen 64-Bit Prozessor.