Übungen zu
Grundlagen der Informatik für Studierende des Maschinenbaus
im Sommersemester 2001
(Dipl.-Inf. Jörg Barner)

Aufgabe 24: Prozeßumschalter
Erläutern Sie die Aufgaben und die Arbeitsweise des Prozeßumschalters eines Betriebssystems.

Lösung:
Der Prozeßumschalter (Scheduler, Dispatcher) ist die Instanz im Betriebssystem, die zwischen Prozessen umschaltet und damit einen parallelen Ablauf der Prozesse vortäuscht. Neben der eigentlichen Umschaltung muß der Scheduler auch entscheiden, welcher der lauffähigen Prozesse als nächster in den Zustand „laufend“ übergehen soll. Für diese Entscheidung sind mehrere Strategien möglich, z.B. nach Prioritäten oder mit Warteschlangen. Eine verbreitete Strategie ist „Round-Robin“: Wenn ein Prozeß in den Zustand „bereit“ übergeht, wird er an das Ende einer Warteschlange einsortiert. Der Erste Prozeß in dieser FIFO-Warteschlange wird jeweils aktiviert. Wenn ein Prozeßeine gewisse Zeit im Besitz der CPU war, wird er unterbrochen und wieder an das Ende der Warteschlange gesetzt.

Aufgabe 25: Prozeßsynchronisation

a) Was versteht man unter Prozeßsynchronisation?

b) Erläutern Sie *kurz* die Ihnen bekannten Synchronisationskonzepte.

Lösung:

a) Die Prozeßsynchronisation stellt sicher, daß gewisse Operationen in verschiedenen Prozessen in einer bestimmten Reihenfolge ausgeführt werden.

   Dies kann zum einen die sogenannten „kritischen Abschnitte“ betreffen. Solche Abschnitte dürfen nur von einem (oder einer begrenzten Anzahl) von Prozessen einer Prozeßgruppe gleichzeitig durchlaufen werden, weil z.B. der Code in dem kritischen Abschnitt Ressourcen benötigt, die nur begrenzt vorhanden sind. In welcher Reihenfolge aber die einzelnen Prozesse in die kritischen Abschnitte eintreten, ist gleichgültig.

   Zum anderen kann Synchronisation nötig sein, wenn Prozesse miteinander kooperieren wollen und manche Operationen in verschiedenen Prozessen hintereinander ausgeführt werden müssen. Z.B.: Prozeß `A` schreibt in einen Puffer, Prozeß `B` liest daraus. Wenn der Puffer leer ist, muß `B` warten, bis `A` den Puffer gefüllt hat.

b) **Semaphore**: (einfachster Fall: binäre Semaphore) Variable mit dem Wertebereich $\{0, 1\}$ und Operationen P und V. P belegt den Semaphor, wenn er frei ist, und wartet falls nicht. V gibt ihn wieder frei. Etliche Variationen sind möglich: zählende Semaphore (größerer Wertebereich), Vektoren von Semaphoren,....

Monitor: Synchronisation über ein Modul, das zu einem Zeitpunkt nur von einem Prozeß betreten werden kann. Kritische Abschnitte sind also alle Teile dieses Moduls. Sobald einer der Prozesse den Monitor betreten hat, müssen die anderen warten, bis er ihn wieder verläßt.

Rendezvous: Operationen zum Senden und Empfangen von „Nachrichten". Sender blockiert, bis ein anderer Prozeß die Empfangsoperation aufruft. Umgekehrt blockiert ein Empfänger, bis eine Nachricht gesendet wird.

Aufgabe 26: Betriebsmittelvergabe

Im Arbeitsspeicher stehen zum Zeitpunkt $t = 0$ die Prozesse $P_1, \ldots, P_4$ in der Anordnung, wie die Abbildung unten zeigt. Die Betriebsmittelvergabe erfolgt nach der Best-Fit-Methode. Als Reihenfolgestrategie wird FIFO verwendet.

reserviert f. Betr.system	P1 64K	frei 64K	P2 128K	frei 32 K	P3 32 K	frei 64K	P4 128K	frei 256K

0K 128K 256K 384K 512K 640K 768K 896K

a) Zum Zeitpunkt $t = 1$ kommt folgender Bedarf hinzu:

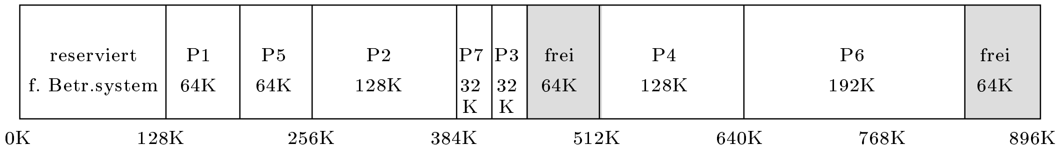
Prozeß	Speicherbedarf
P_5	64K
P_6	192K
P_7	32K

Kann dieser Bedarf gedeckt werden? Begründen Sie Ihre Antwort!

b) Zum Zeitpunkt $t = 2$ sind die Prozesse P_1, P_2 und P_3 beendet. Hinzu kommen die Prozesse P_8 und P_9, die jeweils 128k Speicherplatz anfordern. Können diese Anforderungen erfüllt werden? Begründen Sie Ihre Antwort!

Lösung:

a) Der Bedarf kann gedeckt werden, die Anordnung sieht nach der Vergabe folgendermaßen aus:



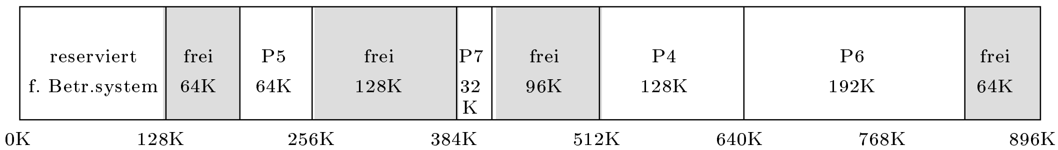
Die Freiliste – die wegen Best-Fit nach der Blockgröße geordnet ist – sieht also so aus

Beginn	Größe
448K	64K
832K	64K

b) Nach der Beendigung von P_1, P_2 und P_3 werden ihre Speicherblöcke freigegeben, der Block von P_3 kann mit dem benachbarten 64K-Block zu einem größeren 96K-Freiblock vereinigt werden. Anschließend erhält man folgende Freiliste:

Beginn	Größe
256K	128K
416K	96K
128K	64K
832K	64K

also:



Man sieht, daß zwar noch die Anforderung von P_8 erfüllt werden kann, nicht jedoh die 128K für P_9, obwohl noch insgesamt 224K frei sind. Eine Lösung wäre eine Speicherkompaktierung, also das „Zusammenschieben" der belegten Blöcke, so daß nur ein großer freier Block übrig bleibt.

Aufgabe 27: Verdrängungstrategien

Gegeben sind die Seiten 0,1,2,3 und 4. Die Seiten sollen in einem Speicher mit 3 Kacheln verwaltet werden. Zu den ersten drei Zeitpunkten werden die Kacheln mit den Seiten 0, 1 und 4 gefüllt.

Führen die die folgende Tabelle

Zeitpunkt	1	2	3	4	5	6	7	8	9	10	11	12
Referenz	0	1	4	3	2	3	0	1	0	3	0	2
Kachel 1	0	0	0									
Kachel 2	-	1	1									
Kachel 3	-	-	4									

entsprechend der Referenzliste (= Nummer der jeweils als nächstes benötigten Seite) mit

a) der LRU-Methode

b) der FIFO-Methode

fort.

Lösung:

a) LRU:

Zeitpunkt	1	2	3	4	5	6	7	8	9	10	11	12
Referenz	0	1	4	3	2	3	0	1	0	3	0	2
Kachel 1	0	0	0	3	3	3	3	3	3	3	3	3
Kachel 2	-	1	1	1	2	2	2	1	1	1	1	2
Kachel 3	-	-	4	4	4	4	0	0	0	0	0	0
Einlagerung	x	x	x	x	x	-	x	x	-	-	-	x

b) FIFO:

Zeitpunkt	1	2	3	4	5	6	7	8	9	10	11	12
Referenz	0	1	4	3	2	3	0	1	0	3	0	2
Kachel 1	0	0	0	3	3	3	3	1	1	1	1	1
Kachel 2	-	1	1	1	2	2	2	2	2	3	3	3
Kachel 3	-	-	4	4	4	4	0	0	0	0	0	2
Einlagerung	x	x	x	x	x	-	x	x	-	x	-	x

Bei FIFO muß also eine Einlagerung mehr vorgenommen werden. Bei dieser Strategie wird Seite 3 zum Zeitpunkt 8 verdrängt, obwohl erst bei $t = 6$ auf sie zugegriffen wurde. Dies geschieht, weil Seite 3 relativ früh eingelagert wurde, und FIFO nimmt im Gegensatz zu LRU keine Rücksicht auf den letzten Zugriffszeitpunkt.