1 IPS - Semaphore

In einem Parkhaus mit insgesamt P Parkplätzen werden Ein- und Ausfahrt mit Lichtschranken überwacht. An der Einfahrt steht eine Ampel, die auf "rot" geschaltet wird, wenn kein Parkplatz mehr frei ist.

Skizzieren Sie in Pseudocode ein System, das aus drei Prozessen sowie einem Initialisierungsteil besteht und die Ampel steuert. Es verwendet dazu eine allen drei Prozessen gemeinsame Zählvariable, die die Anzahl der freien Parkplätze angibt, und zur Wahrung der Daten-Konsistenz ein binäres Semaphor.

Der Beobachter-Prozess E an der Einfahrt zählt die Autos, die in das Parkhaus einfahren, indem er auf ein Lichtschranken-Ereignis wartet und nach dessen Eintritt die Zählvariable um 1 erniedrigt.

Der Beobachter-Prozess A an der Ausfahrt zählt die Autos, die es wieder verlassen, indem er die Zählvariable um 1 erhöht.

Der Berichterstatter-Prozess B prüft einmal pro Sekunde die Zählvariable. Wenn sie den Wert 0 erreicht hat, stellt er die Ampel auf "rot". Wenn sie wieder einen positiven Wert hat, stellt er sie wieder auf "grün". Die Schaltung der Einfahrtschranke gewährleistet, dass zwei aufeinander folgende Autos einen Mindestabstand von 5 Sekunden beim Passieren der Schranke nicht unterschreiten.

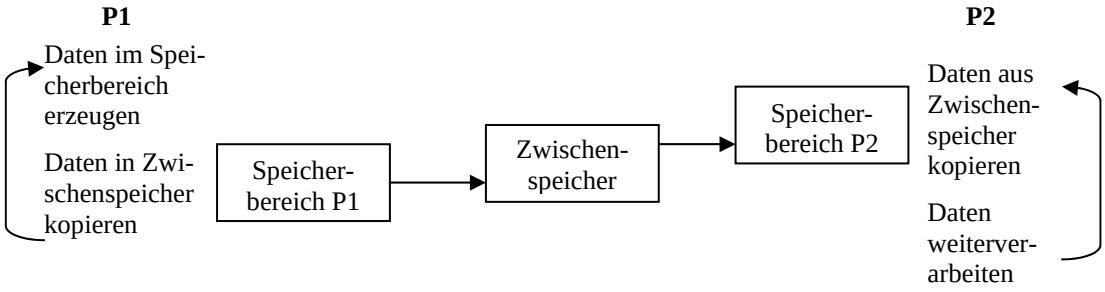
Achten Sie darauf, dass bei Ihrer Lösung keine Dateninkonsistenzen eintreten können! Überlegen Sie sich hierzu die Inkonsistenzmöglichkeiten, die entstehen, wenn man die kritischen Bereiche nicht mit Semaphoren schützt!

2 Datenaustausch zwischen zwei Prozessen (Klausur WS 1997/98)

Gegeben sei ein Prozesssystem mit den zwei Prozessen P1 und P2. P1 produziert in einer Schleife immer wieder Datenpakete, welche P2 weiterverarbeiten soll. Es muß jedoch sichergestellt werden, dass P1 immer kontinuierlich arbeiten kann – d.h. dass P1 nie länger als für einen kurzen Zeitraum blockiert wird.

P2 läuft i.a. langsamer ab als P1. Daher werden nicht alle Datenpakete weiterverarbeitet, sondern nur solche, die gerade zu einem Zeitpunkt fertig werden, wenn P2 wieder aufnahmebereit ist (jeweils ein Schnappschuß)! Wichtig ist dabei, dass Datenpakete jeweils entweder vollständig und konsistent von P1 nach P2 übergeben werden oder gar nicht!

Damit der langsamere Prozess P2 nicht den schnellen Prozess P1 „ausbremst“, werden die Daten in jedem Zyklus von P1 nach dem Erzeugen in einen Zwischenpuffer kopiert, aus dem P2 – sofern aufnahmebereit – dann lesen kann! Das folgende Bild zeigt diese Anordnung:



Der Aufwand für das Kopieren der Daten ist bei beiden Prozessen gegenüber dem Erzeugen/Verarbeiten der Daten niedrig. Speicherschutzaspekte beim Kopieren können hier unberücksichtigt bleiben.

Skizzieren Sie in (PASCAL oder C-ähnlichem Pseudocode) diese Anordnung und achten Sie unter Verwendung von binären Semaphoren auf Einhaltung der folgenden Bedingungen:

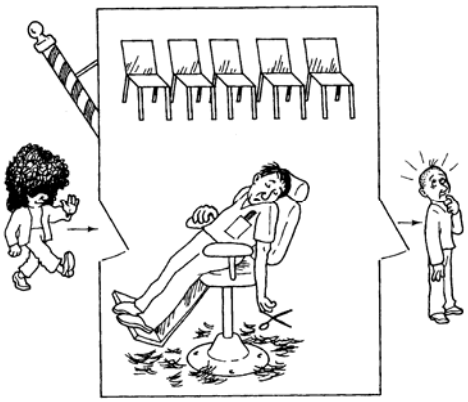
- konsistentes Kopieren der Daten in den / aus dem Zwischenspeicher
- zusätzlich: P2 soll mit seinem Kopieren/Verarbeiten nicht eher beginnen als sicher neue, von P1 erzeugte Daten konsistent im Zwischenspeicher vorliegen. Aktives Warten soll dabei vermieden werden.

Hinweis:
Bei binären Semaphoren bewirken auch mehrere V()-Aufrufe nur ein Setzen des Semaphorzählers auf 1!

3 Interprozesskommunikation und Synchronisation (Gewichtung: mittel/hoch)

Ein klassisches IPKS-Problem bei Client/Server-Prozesskonstellationen kann in eine etwas freundlichere Formulierung gekleidet werden. Der ‚Schauplatz' ist nun ein Friseurladen...

Hier arbeitet **ein** Friseur (Server). Falls kein Kunde anwesend ist, setzt sich der Friseur auf seinen Friseurstuhl und schläft.



Ein eintreffender Kunde muss den schlafenden Friseur aufwecken. Falls weitere Kunden eintreffen, während der Friseur Haare schneidet (m.a.W. eine Clientanforderung bearbeitet), warten die Kunden (auf einem der Stühle). Der Friseur kann jeweils nur einen Haarschnitt zu einer Zeit durchführen – dieser wird dafür jedoch vollständig durchgeführt.

Bitte geben Sie für diese Situation jeweils eine Funktion/Prozedur für **den Friseur** und eine für **die Kunden** an, ohne dass zeitkritische Abläufe entstehen können. Bitte listen Sie auch die verwendeten Semaphore sowie deren Initialisierungen auf!

Hinweise:

1. *Sie benötigen keine Puffer zum Merken der auf den Friseur wartenden Kunden! Alle Ausschlüsse und Pufferungen können durch Semaphore realisiert werden!!! Bitte überlegen Sie hierzu zuerst, welches die handelnden Objekte in dieser Situation sind und dann, wie sich diese verständigen müssen.*
2. *Es gibt Parallelen zu dem in der Vorlesung besprochenen Erzeuger- / Verbraucherproblem! Bitte denken Sie jedoch trotzdem zuerst über Hinweis 1 nach!*

4 Ablaufplanung (Gewichtung: hoch)

Auf einem Unix System V – System mit einer Zeitscheibendauer von 1sec und 50 Ticks / Zeitscheibe laufen drei Prozesse A, B und C.

C ist ein reiner Rechenprozess ohne sonstige Blockierungen und benötigt eine reine Rechenzeit von 60 Ticks. Alle drei Prozesse werden zum Zeitpunkt 0 gestartet.

Die "Programme" der Prozesse A und B sind nachfolgend abgedruckt. Zu den angegebenen Zeiten (in Anzahl Ticks seit Prozeßstart) erfolgen jeweils Semaphorfunktionsaufrufe. Die bei einem V() – Aufruf evtl. wieder freigegebenen Prozesse werden lediglich "bereit"-gesetzt, d.h. sie erhalten nicht sofort wieder die CPU!

Initialisierung der Semaphoren:
`S0 := 0`
`S1 := 1`	

Prozeß A:	**Prozeß B:**

`...`
`10: P(S0)`
`...`
`20: P(S1)`
`...`
`30: V(S1)`
`...`
`40: "`*Prozeßende*`"`
	`...`
	`20: V(S0)`
	`P(S1)`
	`...`
	`50: V(S1)`
	`...`
	`60: "`*Prozeßende*`"`

Die Startprioritäten seinen wie folgt vergeben: A: 60, B: 62, C: 64
Alle drei Prozesse werden zum Zeitpunkt 0 gestartet.

Bitte tragen Sie den Ablauf der drei Prozesse in das untenstehende Diagramm ein, benennen Sie die Gründe für einen Prozeßwechsel und berechnen Sie für jeden Prozeß den Parameter 'P' – Penalty Ratio.

Das Ausfüllen soll gemäß folgendem ***(nicht-korrekten*!) Muster** geschehen:

	10	20	30	40	50	10	20	30	40	50	10	20	30	40	50	10	20	30	40	50	P
Proz. A					Sem	Block															P = 1.0
C	0					30															
Prio	60					67															

Hier bitte Ihre Lösung:

Ticks:	10	20	30	40	50	10	20	30	40	50	10	20	30	40	50	10	20	30	40	50	P
Proz. A																					P =
C																					
Prio	60																				
Proz. B																					P =
C																					
Prio	62																				
Proz. C																					P =
C																					
Prio	64																				

5 Programmieraufgabe: Beobachter- u. Auswerteprozess

Programmieren Sie das Beispiel mit dem beobachtenden und dem auswertenden Prozess (s.u.) in C und zwar derart, dass die ohne Interprozesssynchronisation entstehenden Fehlermöglichkeiten nicht entstehen {Hinweis: Hierzu reichen die Systemaufrufe aus dem Abschnitt Threads im Kapitel Prozessverwaltung aus!).

```
Beobachter-Prozeß                          Berichterstatter-Prozeß
wiederhole ewig                                  wiederhole ewig
   warte auf Ereignis;                              warte 1 Minute
                                                    falls zaehler <> 0
                                                      berechne durchschnitt;
                                                      write (zaehler, durchschnitt);
   miß gewicht;
   gewichtssumme:=gewichtssumme+gewicht;
   zaehler := zaehler + 1;                          zaehler:=0; gewichtssumme:=0;
```

Bitte versuchen Sie auch die möglichen Fehlerstellen – die natürlich in einem Programm ohne geeignete Synchronisationselemente entstehen können – genauer zu untersuchen, d.h. herauszufinden, wann diese tatsächlich auftreten.