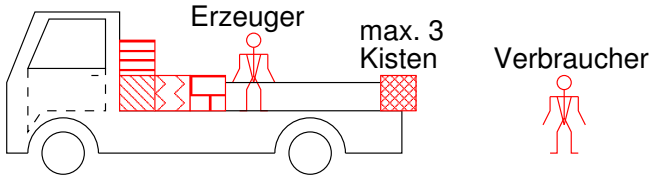
Einseitige Synchronisation mit Semaphoren: Beispiel

Erzeuger-/Verbraucher Problem: Ein Erzeugerprozeß und ein Verbraucher mit Platz für 3 Gegenstände dazwischen sollen synchronsiert werden.



Beispiel: Be- oder entladen des Umzugswagens, einer (Erzeuger) stellt die Kisten auf die Kante der Ladefläche, ein anderer (Verbraucher) nimmt die Kisten von der Ladekante und stellt sie weg, 3 Kisten passen gleichzeitig auf die Ladekante.

Einseitige Synchronisation mit Semaphoren: Beispiel

Lösung:

```
10 frei: semaphore(3); belegt: semaphore(0); Kisten : integer:=5;

100 Erzeuger : process
110   while Kisten>0 do
120     frei.P;
130     Kiste_auf_Kante_stellen
140     Kisten:=Kisten-1;
150     belegt.V;
160 end process
```

```
200 Verbraucher : process
210   repeat
220     belegt.P;
230     Kiste_von_Kante_nehmen
240     frei.V;
250 end process
```


Einseitige Synchronisation: Beispiel Umzugwagen

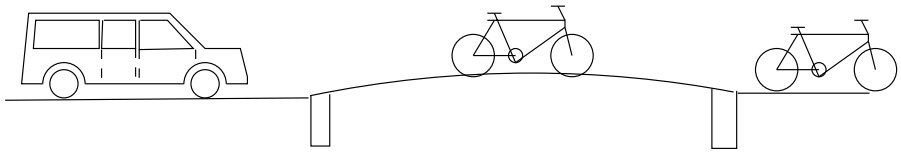
Beispielablauf: E=Erzeuger, V=Verbraucher

Ereignis	frei.z	frei.WS	belegt.z	belegt.WS	Kisten	Bemerkung
Initialisierung	3		0		5	Zeile 10, kein Prozeß läuft
Erzeuger gestartet	2		1		4	1. Iteration
	1		2		3	2. Iteration
	0		3		2	3. Iteration
	-1	E	3		2	4. Iteration blockiert in120
Verbraucher gestartet	0		2		2	1. Iteration, deblockiert E (240)
	1		1		2	2. Iteration
	2		0		2	3. Iteration

Ereignis	frei.z	frei.WS	belegt.z	belegt.WS	Kisten	Bemerkung
	2		-1	V	2	4. Iteration blockiert in 220
Erzeuger (Forts.)	2		0		1	weiter in 4. Iteration, deblockiert V (150)
	1		1		0	5. Iteration, Erzeuger wird beendet
Verbraucher (Forts.)	1		1		0	weiter in 4. Iteration
	2		0		0	5. Iteration
	3		-1	V	0	6. Iteration blockiert in 220

Mehrseitige Synchronisation mit Semaphoren: Beispiel

Zweiseitige Synchronisation: Autos und Fahrradfahrer benutzen eine Brücke. Wegen Baufälligkeit können entweder bis zu 5 Radler oder exclusiv ein Auto über die Brücke fahren. Die Brücke kann nur in eine Richtung benutzt werden.



Mehrseitige Synchronisation: Beispiel Brücke

Ereignis	S.z	S.WS	A.z	A.WS	Bemerkung
initial	1		5		
1. Rad kommt	0		5		Zeile 110
	0		4		Zeile 120
	1		4		Zeile 130, (cont.)
2. Rad kommt	0		4		Zeile 110
	0		3		Zeile 120
	1		3		Zeile 130, (cont.)
1. Auto kommt	0		3		Zeile 210
	0		2		Zeile 220-230 (i=1)
	0		1		Zeile 220-230 (i=2)
	0		0		Zeile 220-230 (i=3)
	0		-1	A1	Zeile 230 (i=4, blockiert)

Ereignis	S.z	S.WS	A.z	A.WS	Bemerkung
1. Rad fährt	0		0		Zeile 150 (term., deblockiert 1.Auto)
1. Auto (Forts.)	0		-1	A1	Zeile 230 (i=5, blockiert)
3. Rad kommt	-1	R3	-1	A1	Zeile 110 (blockiert)
4. Rad kommt	-2	R3, R4	-1	A1	Zeile 110 (blockiert)
2. Rad fährt	-2	R3, R4	0		Zeile 150 (term., deblockiert 1.Auto)
1. Auto (Forts.)	-1	R4	0		Zeile 240, (cont., deblockiert 3.Radler)
3. Rad (Forts.)	-1	R4	-1	R3	Zeile 120 (blockiert)
1. Auto fährt	-1	R4	4		5x Zeile 270, (term., debl. 3.Radler)
3. Rad (Forts.)	0		4		Zeile 130 (cont., deblockiert 4. Radler)
4. Rad (Forts.)	1		3		Zeile 120-130 (cont.)
3. Rad fährt	1		4		Zeile 150 (term.)
4. Rad fährt	1		5		Zeile 150 (term.)

Mehrseitige Synchronisation
Allgemeiner Fall: 2 Prozeßtypen, Keine Priorisierung

- Zwei unterschiedliche Prozeßtypen (Typ A und Typ B) greifen auf eine gemeinsame Ressource zu.
- Die beiden Prozeßtypen dürfen die Ressource nie gleichzeitig benutzen.
- Es dürfen maximal n Prozesse vom Typ A die Ressource gleichzeitig benutzen.
- Es dürfen maximal m Prozesse vom Typ B die Ressource gleichzeitig benutzen.

Bemerkung: Ist schon ein Prozeß vom Typ A im kritischen Abschnitt, so kann ein weiterer Prozeß vom Typ A ebenfalls in den kritischen Abschnitt eintreten und dabei einen warten Prozeß vom Typ B überholen. Das gleiche gilt umgekehrt. (Anders als das Beispiel mit der Brücke!)

Allgemeiner Fall: 2 Prozeßtypen, Keine Priorisierung

```
10 AZaehler : integer :=0; BZaehler : integer :=0;
20 AZSchutz, BZSchutz, Ausschluss : semaphore(1);
30 A_Ausschluss : semaphore(n); // weggelassen, wenn n unendlich
40 B_Ausschluss : semaphore(m); // weggelassen, wenn m unendlich
```



```
100 Typ_A : process
        ...
110     A_Ausschluss.P;
120        AZSchutz.P;
130           AZaehler:=AZaehler+1;
140           if AZaehler=1 then
150              Ausschluss.P;
160           endif;
170        AZSchutz.V;
180           _Ressource_benutzen_
190        AZSchutz.P;
200           AZaehler:=AZaehler-1;
210           if AZaehler=0 then
220              Ausschluss.V;
230           endif;
240        AZSchutz.V;
250     A_Ausschluss.V;
        ...
260 end process
```

```
300 Typ_B : process
        ...
310     B_Ausschluss.P;
320        BZSchutz.P;
330           BZaehler:=BZaehler+1;
340           if BZaehler=1 then
350              Ausschluss.P;
360           endif;
370        BZSchutz.V;
380           _Ressource_benutzen_
390        BZSchutz.P;
400           BZaehler:=BZaehler-1;
410           if BZaehler=0 then
420              Ausschluss.V;
430           endif;
440        BZSchutz.V;
450     B_Ausschluss.V;
        ...
460 end process
```


Allgemeiner Fall: 2 Prozeßtypen, Keine Priorisierung

Erstes Leser-/Schreiber-Problem als Spezialfall

1. Es wird n auf ∞ und m auf 1 gesetzt.
2. Damit wird `A_Ausschluss` überflüssig.
3. Es kann immer nur ein Prozeß in den Block, der durch `B_Ausschluss` geschützt wird. Damit wird `BZSchutz` sowie der Zaehler mit zugehöriger Abfrage überflüssig.
4. Im Prozessen vom Typ B kann nunmehr `B_Ausschluss` und `Ausschluss` zusammengefaßt werden.

Bemerkung: Wie in VL erwähnt, wird das Verhalten zum ursprünglichen 1. L-S-Problem leicht verändert: Bei Schreiber im kritischen Abschnitt und wartenden Schreiber reiht sich neuer Lesen HINTER Schreibern ein!


```
10 AZaehler : integer:=0;
20 AZSchutz : semaphore(1)
30 Ausschluss : semaphore(1)

100 Typ_A : process
        ...
120        AZSchutz.P;
130           AZaehler:=AZaehler+1;
140           if AZaehler=1 then
150              Ausschluss.P;
160           endif;
170        AZSchutz.V;
180           _Ressource_benutzen_
190        AZSchutz.P;
200           AZaehler:=AZaehler-1;
210           if AZaehler=0 then
220              Ausschluss.V;
230           endif;
240        AZSchutz.V;
        ...
260 end process
```

```
300 Typ_B : process
        ...
350              Ausschluss.P;
380           _Ressource_benutzen_
420              Ausschluss.V;
        ...
460 end process
```

Mehrseitige Synchronisation
Allgemeiner Fall: 2 Prozeßtypen, Priorisierung

- Zwei unterschiedliche Prozeßtypen (Typ A und Typ B) greifen auf eine gemeinsame Ressource zu.
- Die beiden Prozeßtypen dürfen die Ressource nie gleichzeitig benutzen.
- Es dürfen maximal n Prozesse vom Typ A die Ressource gleichzeitig benutzen.
- Es dürfen maximal m Prozesse vom Typ B die Ressource gleichzeitig benutzen.
- Prozesse vom Typ B sind gegenüber Prozessen vom Typ A priorisiert.

Bemerkung: Ein Prozeß vom Typ B überholt alle wartenden Prozesse vom Typ A und verhindert deren Eintritt in den kritischen Abschnitt.

Allgemeiner Fall: 2 Prozeßtypen, Priorisierung

```
10 AZaehler : integer :=0; BZaehler : integer:=0;
20 AZSchutz, BZSchutz, Ausschluss : semaphore(1);
30 A_Ausschluss : semaphore(n);
40 B_Ausschluss : semaphore(m);
50 Halt, Vorhalt : semaphore(1); /* Neu! */
```



```
100 Typ_A : process
        ...
110     A_Ausschluss.P;
112       Vorhalt.P;
114         Halt.P;
120           AZSchutz.P;
130             AZaehler:=AZaehler+1;
140             if AZaehler=1 then
150               Ausschluss.P;
160             endif;
170           AZSchutz.V;
172         Halt.V;
174       Vorhalt.V;
180         _Ressource_benutzen_
190       AZSchutz.P;
200         AZaehler:=AZaehler-1;
210         if AZaehler=0 then
220           Ausschluss.V;
230         endif;
240       AZSchutz.V;
250     A_Ausschluss.V;
260 end process
```

```
300 Typ_B : process
        ...
320     BZSchutz.P;
330       BZaehler:=BZaehler+1;
340       if BZaehler=1 then
345         Halt.P;
350         Ausschluss.P;
360       endif;
370     BZSchutz.V;
375     B_Ausschluss.P; /* 310 */
380       _Ressource_benutzen_
385     B_Ausschluss.V; /* 450 */
390     BZSchutz.P;
400       BZaehler:=BZaehler-1;
410       if BZaehler=0 then
420         Ausschluss.V;
425         Halt.V;
430       endif;
440     BZSchutz.V;
        ...
460 end process
```


Allgemeiner Fall: 2 Prozeßtypen, Priorisierung

Zweites Leser-/Schreiber-Problem als Spezialfall

1. Es wird n auf ∞ und m auf 1 gesetzt.
2. Damit wird `A_Ausschluss` überflüssig.
3. Im Prozessen vom Typ B kann `Ausschluss` die Funktion von `B_Ausschluss` übernehmen, muß dazu allerdings anders plaziert werden.

```
10 AZaehler : integer:=0; BZaehler : integer:=0;
20 AZSchutz, BZSchutz, Ausschluss : semaphore(1);
50 Halt, Vorhalt : semaphore(1);
```

```
100 Typ_A : process
112    Vorhalt.P;
114      Halt.P;
120        AZSchutz.P;
130          AZaehler:=AZaehler+1;
140          if AZaehler=1 then
150            Ausschluss.P;
160          endif;
170        AZSchutz.V;
172      Halt.V;
174    Vorhalt.V;
180      _Ressource_benutzen_
190    AZSchutz.P;
200      AZaehler:=AZaehler-1;
210      if AZaehler=0 then
220        Ausschluss.V;
230      endif;
240    AZSchutz.V;
260 end process
```

```
300 Typ_B : process
        ...
320    BZSchutz.P;
330       BZaehler:=BZaehler+1;
340       if BZaehler=1 then
345          Halt.P;
360       endif;
370    BZSchutz.V;
375    Ausschluss.P;
380       _Ressource_benutzen_
385    Ausschluss.V;
390    BZSchutz.P;
400       BZaehler:=BZaehler-1;
410       if BZaehler=0 then
425          Halt.V;
430       endif;
440    BZSchutz.V;
        ...
460 end process
```