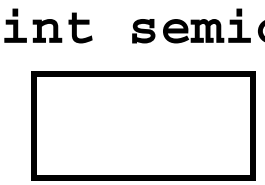
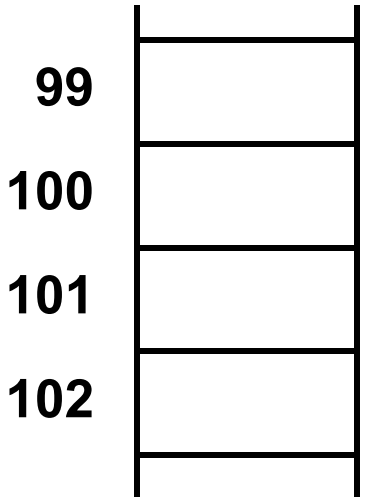
1.) Erzeugung einer neuen Semaphorgruppe

```
semid = semget(IPC_PRIVATE,3,IPC_CREAT|0777);
```

Programm



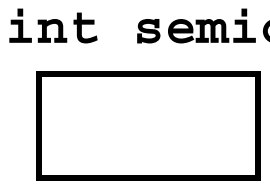
UNIX-Kern



1.) Erzeugung einer neuen Semaphorgruppe

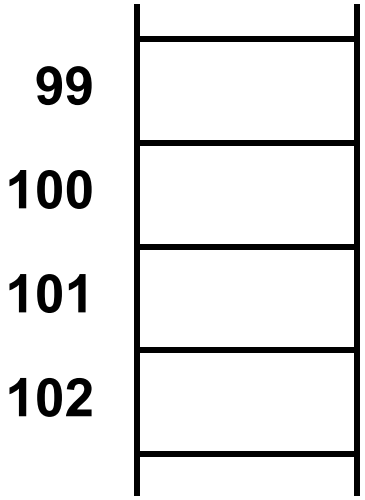
```
semid = semget(IPC_PRIVATE,3,IPC_CREAT|0777);
```

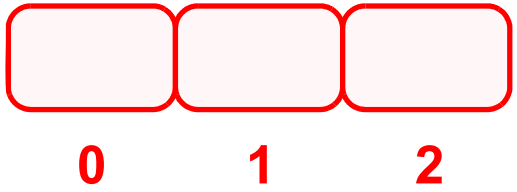
Programm



semget() erzeugt eine Gruppe mit 3 Semaphoren, ...

UNIX-Kern



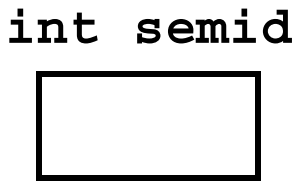


Sem.gruppe

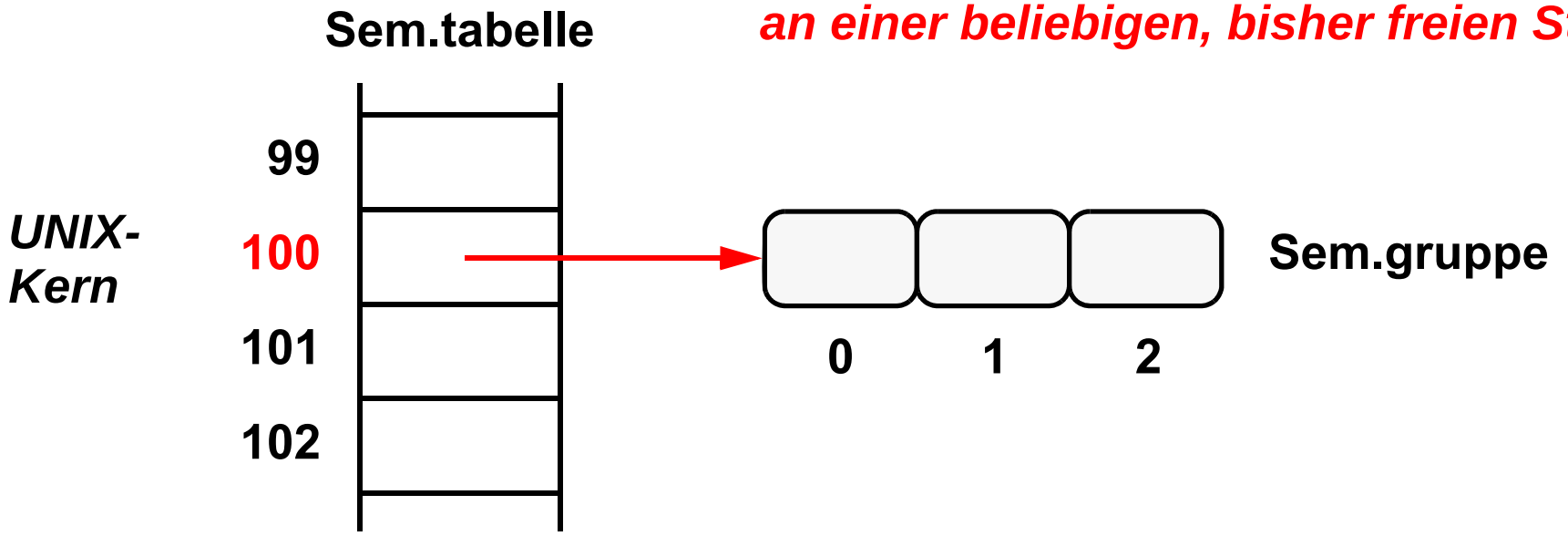
1.) Erzeugung einer neuen Semaphorgruppe

```
semid = semget(IPC_PRIVATE,3,IPC_CREAT|0777);
```

Programm



... verankert sie in der Semaphortabelle an einer beliebigen, bisher freien Stelle ...



1.) Erzeugung einer neuen Semaphorgruppe

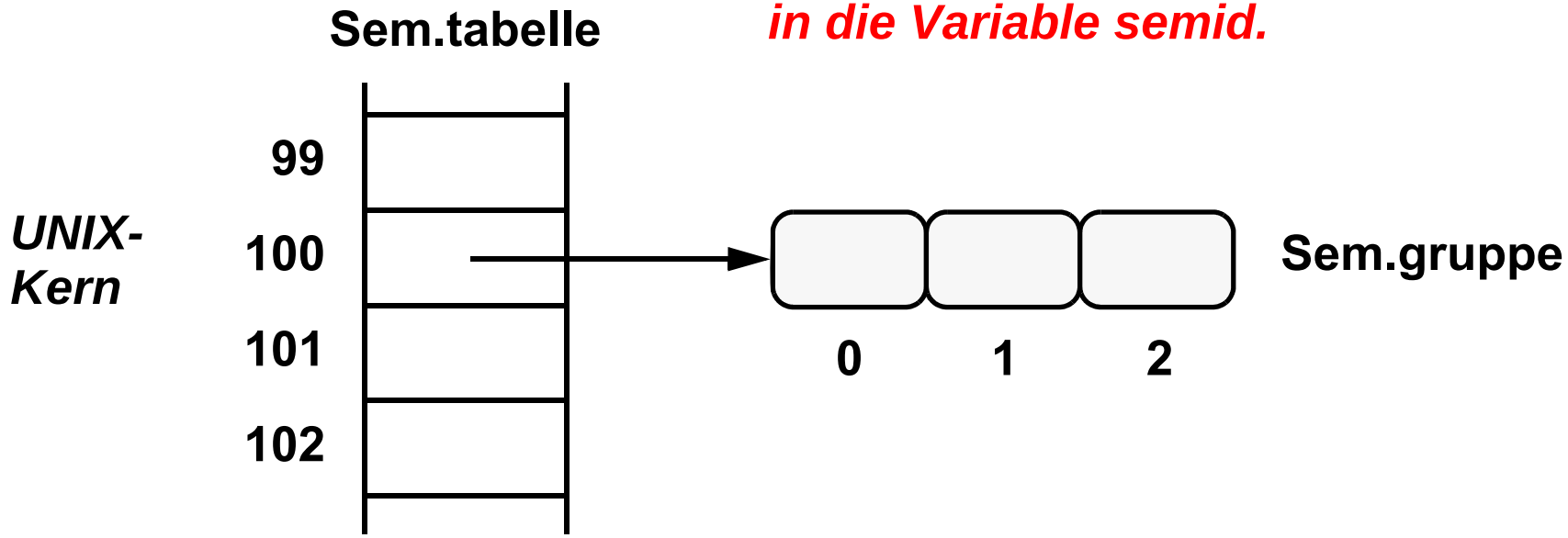
```
semid = semget(IPC_PRIVATE,3,IPC_CREAT|0777);
```

Programm

```
int semid
```

100

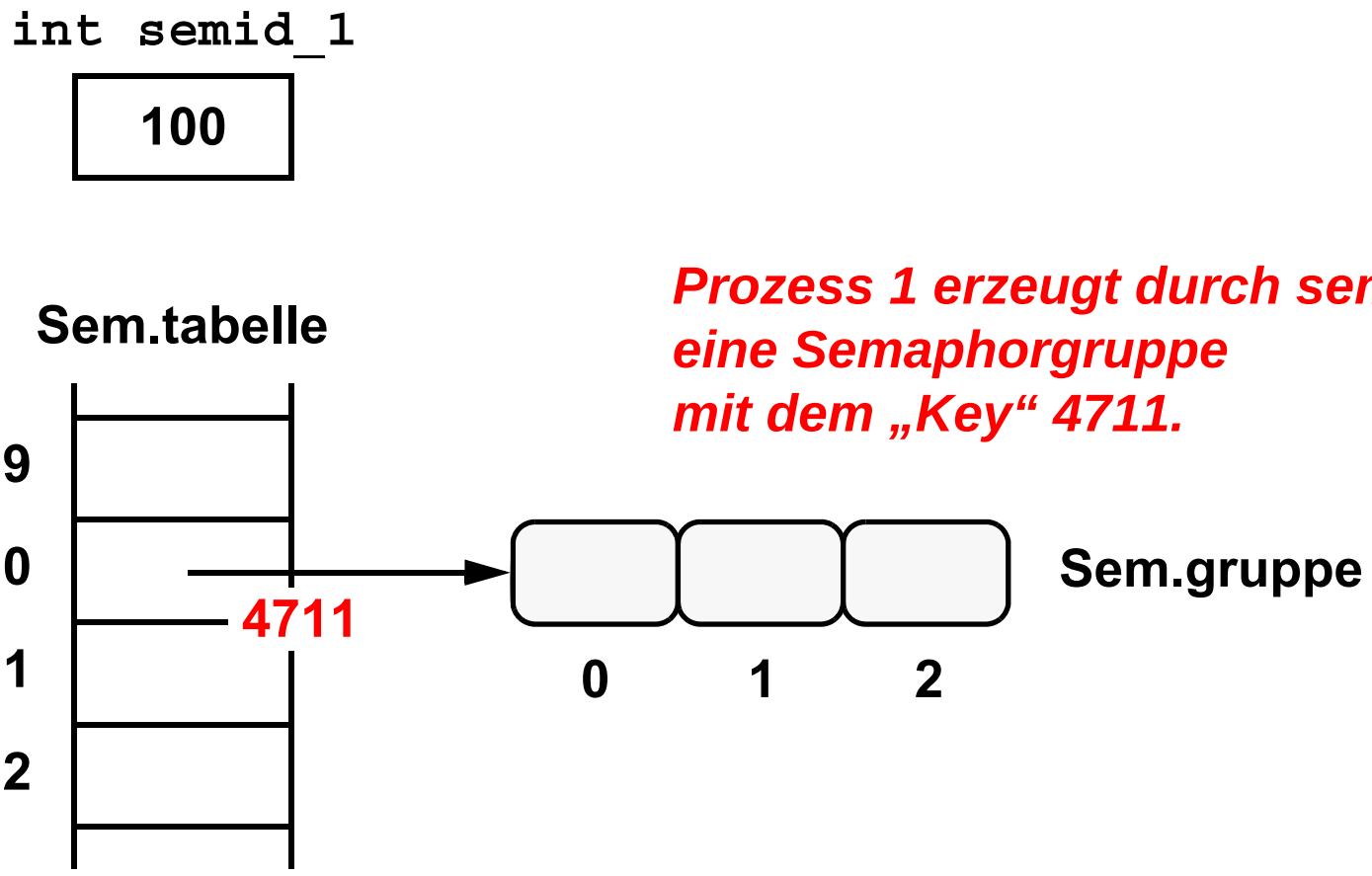
... und schreibt den „Identifier“ dieser Stelle in die Variable semid.



2.) Zugriff zweier Prozesse auf dieselbe Semaphorgruppe

Programm A / Prozess 1:

```
semid_1 = semget(4711,3,IPC_CREAT|0777);
```



2.) Zugriff zweier Prozesse auf dieselbe Semaphorgruppe

Programm A / Prozess 1:

```
int semid_1
```

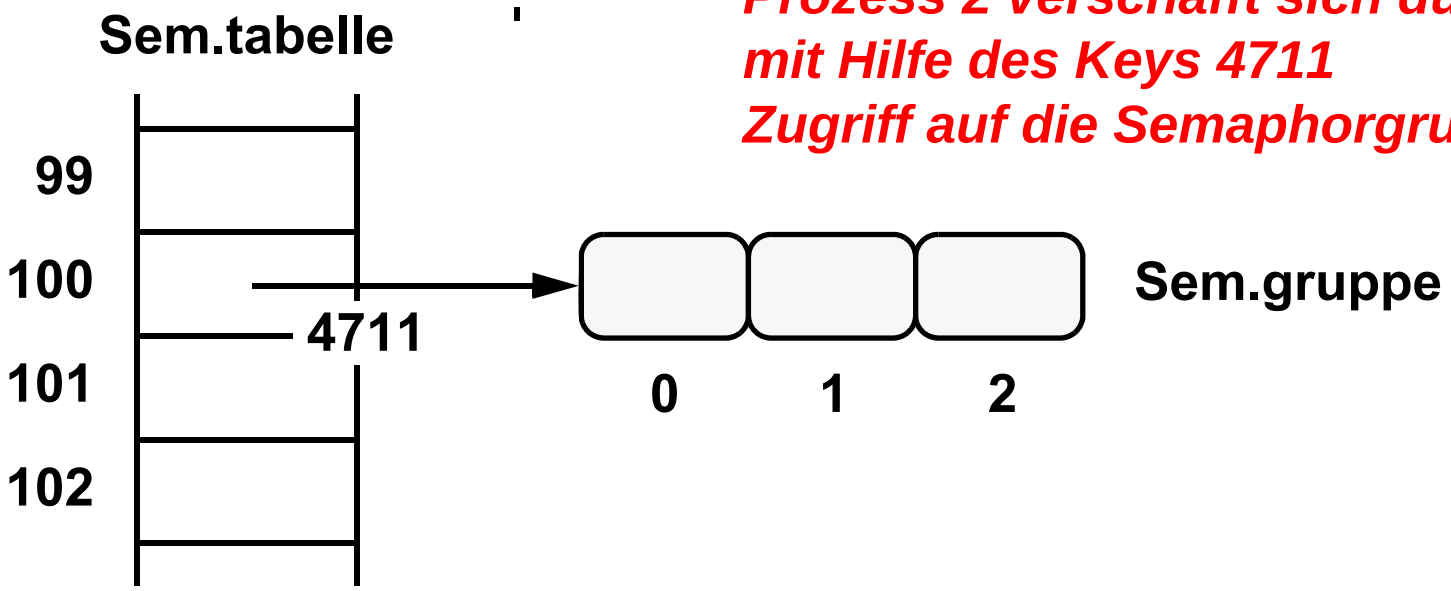
100

Programm B / Prozess 2:

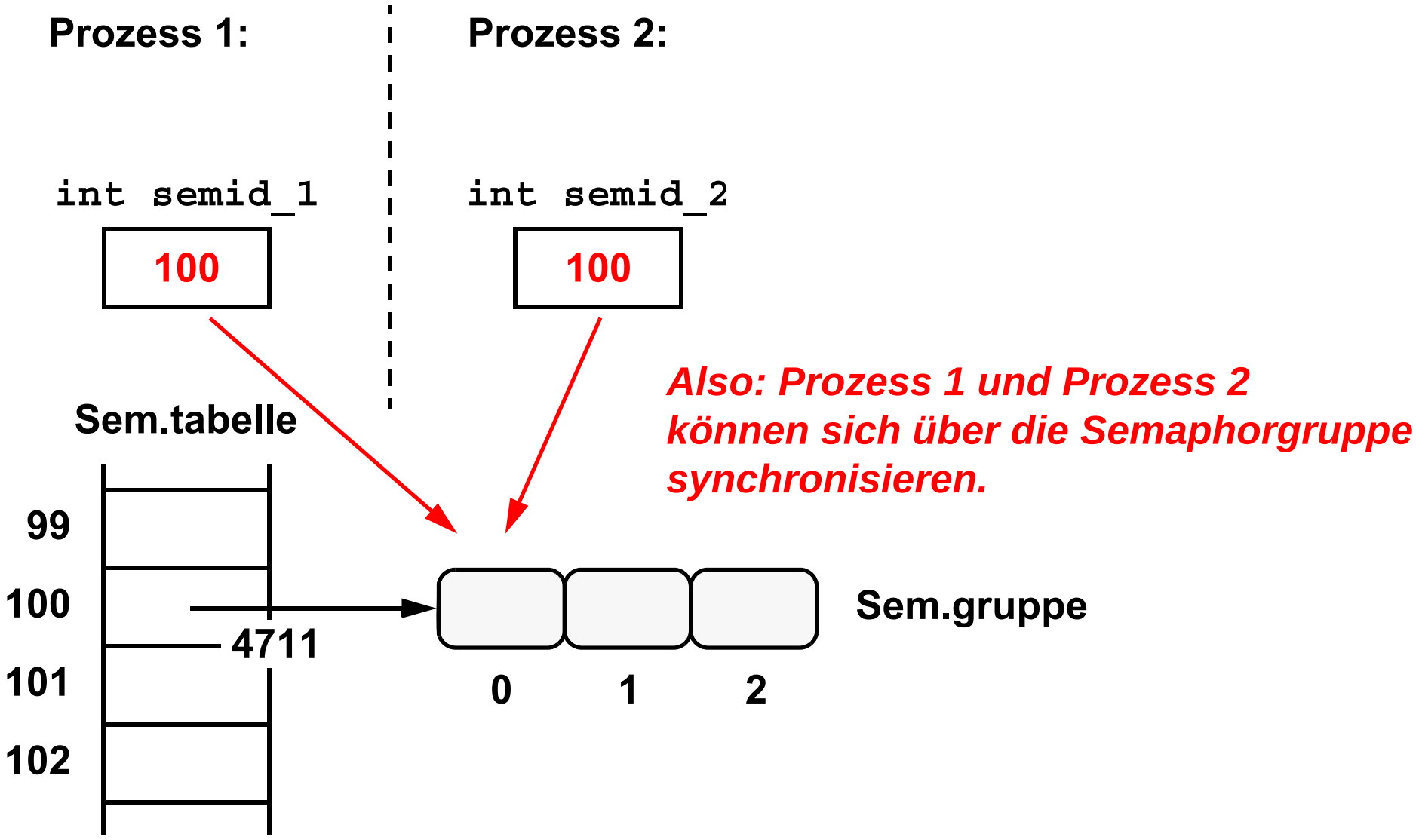
```
semid_2 = semget(4711,3,0777);

int semid_2
```

100

Prozess 2 verschafft sich durch semget() mit Hilfe des Keys 4711 Zugriff auf die Semaphorgruppe.

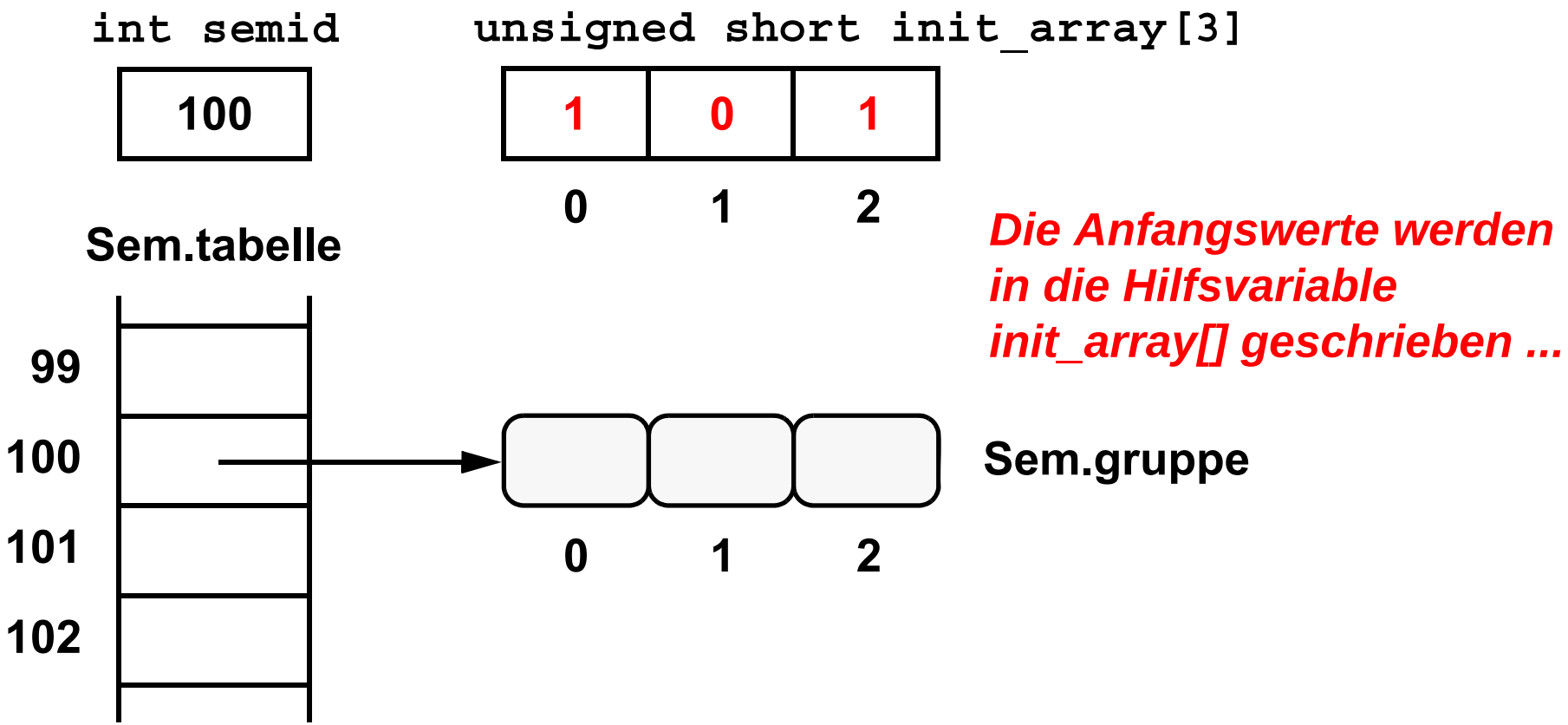


2.) Zugriff zweier Prozesse auf dieselbe Semaphorgruppe



3.) Initialisierung einer Semaphorgruppe

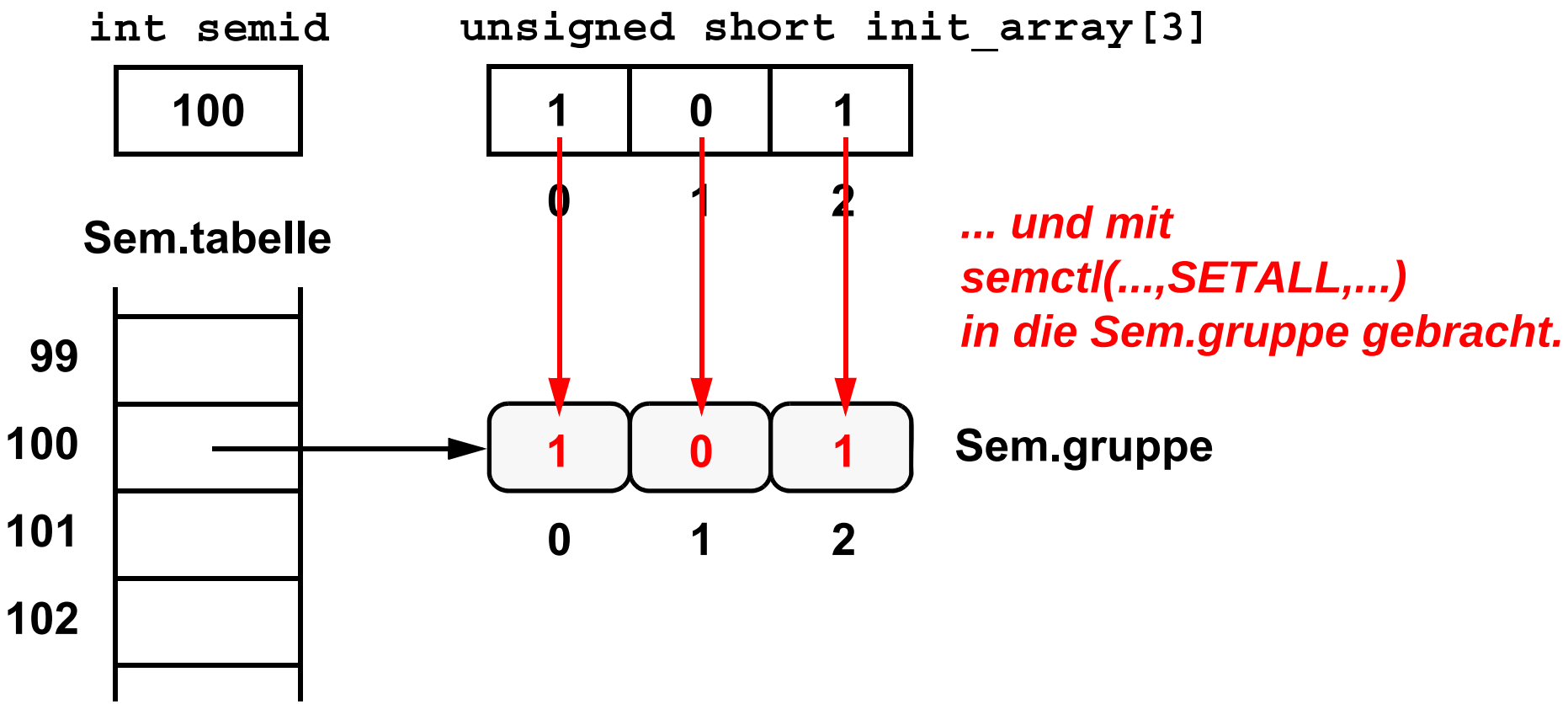
```
init_array[0] = init_array[2] = 1;
init_array[1] = 0;
semctl(semid,0,SETALL,init_array);
```



Die Anfangswerte werden in die Hilfsvariable init_array[] geschrieben ...

3.) Initialisierung einer Semaphorgruppe

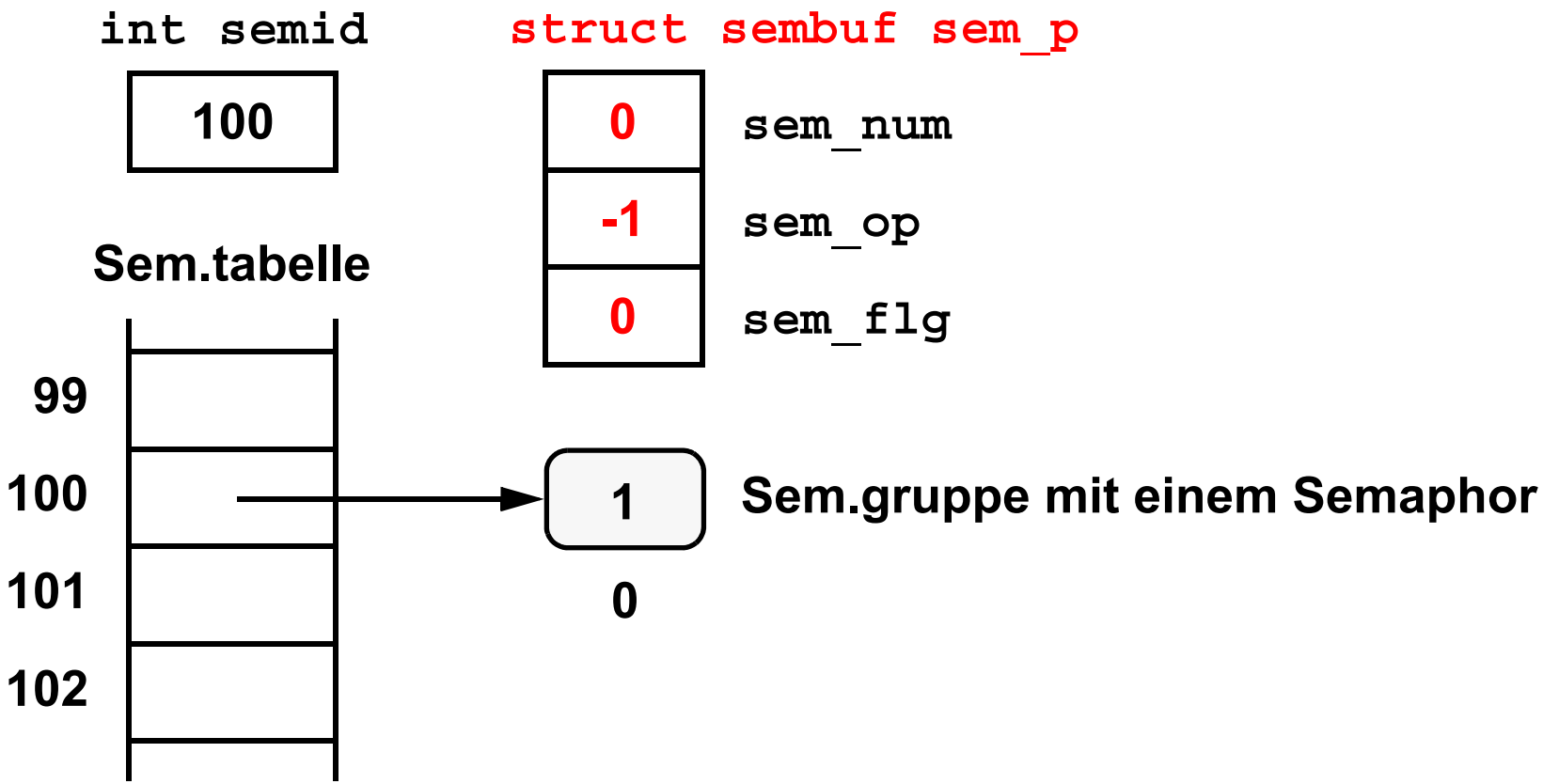
```
init_array[0] = init_array[2] = 1;
init_array[1] = 0;
semctl(semid,0,SETALL,init_array);
```



4.) Operation auf einem Semaphor

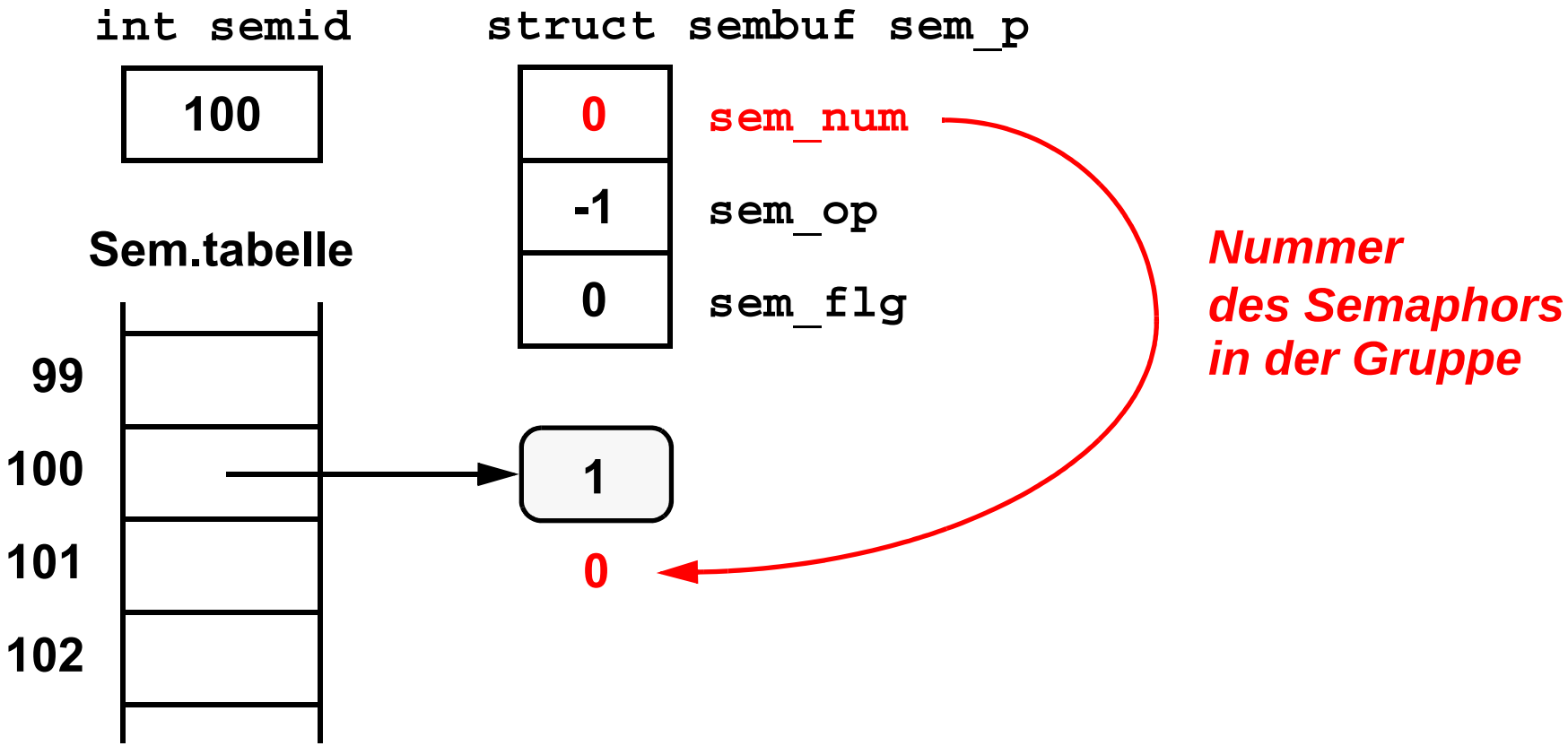
```
sem_p.sem_num = 0;
sem_p.sem_op  = -1;
sem_p.sem_flg = 0;
semop(semid,&sem_p,1);
```

Die Operation wird durch Zuweisungen an sem_p vorbereitet ...



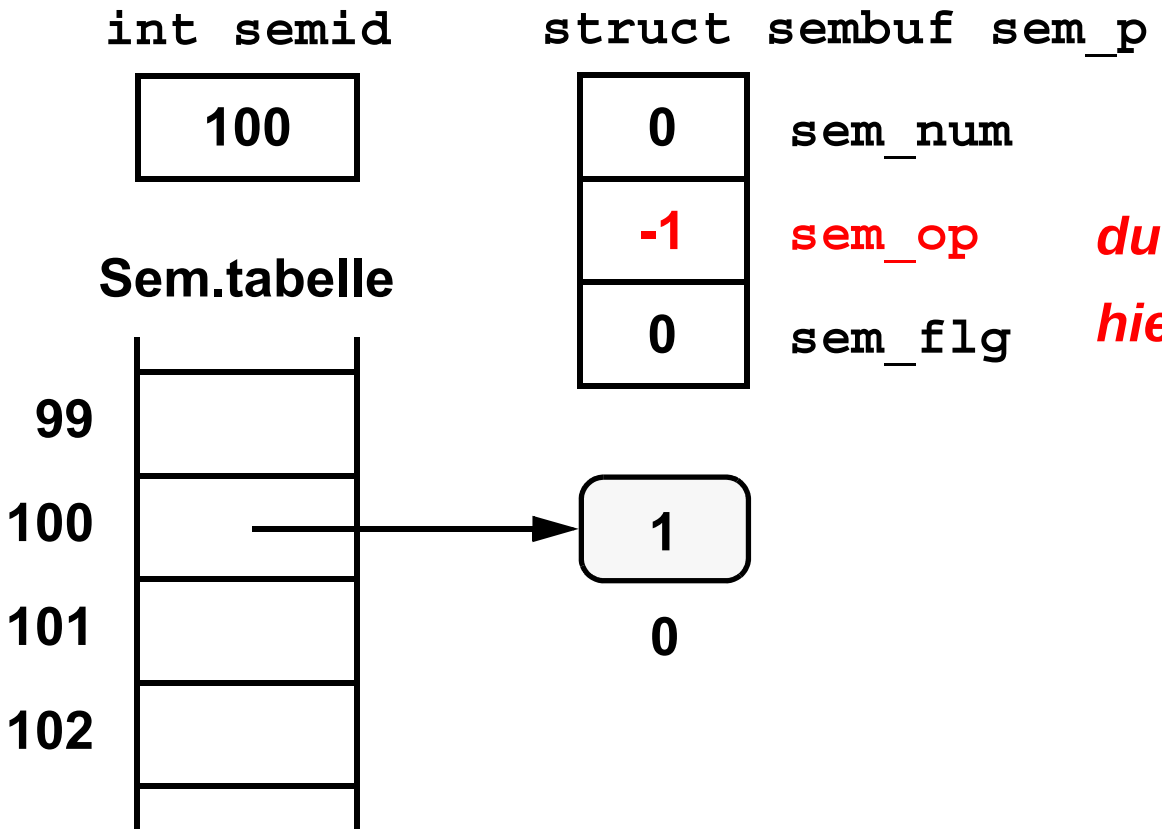
4.) Operation auf einem Semaphor

```
sem_p.sem_num = 0;
sem_p.sem_op  = -1;
sem_p.sem_flg = 0;
semop(semid,&sem_p,1);
```



4.) Operation auf einem Semaphor

```
sem_p.sem_num = 0;
sem_p.sem_op  = -1;
sem_p.sem_flg = 0;
semop(semid,&sem_p,1);
```



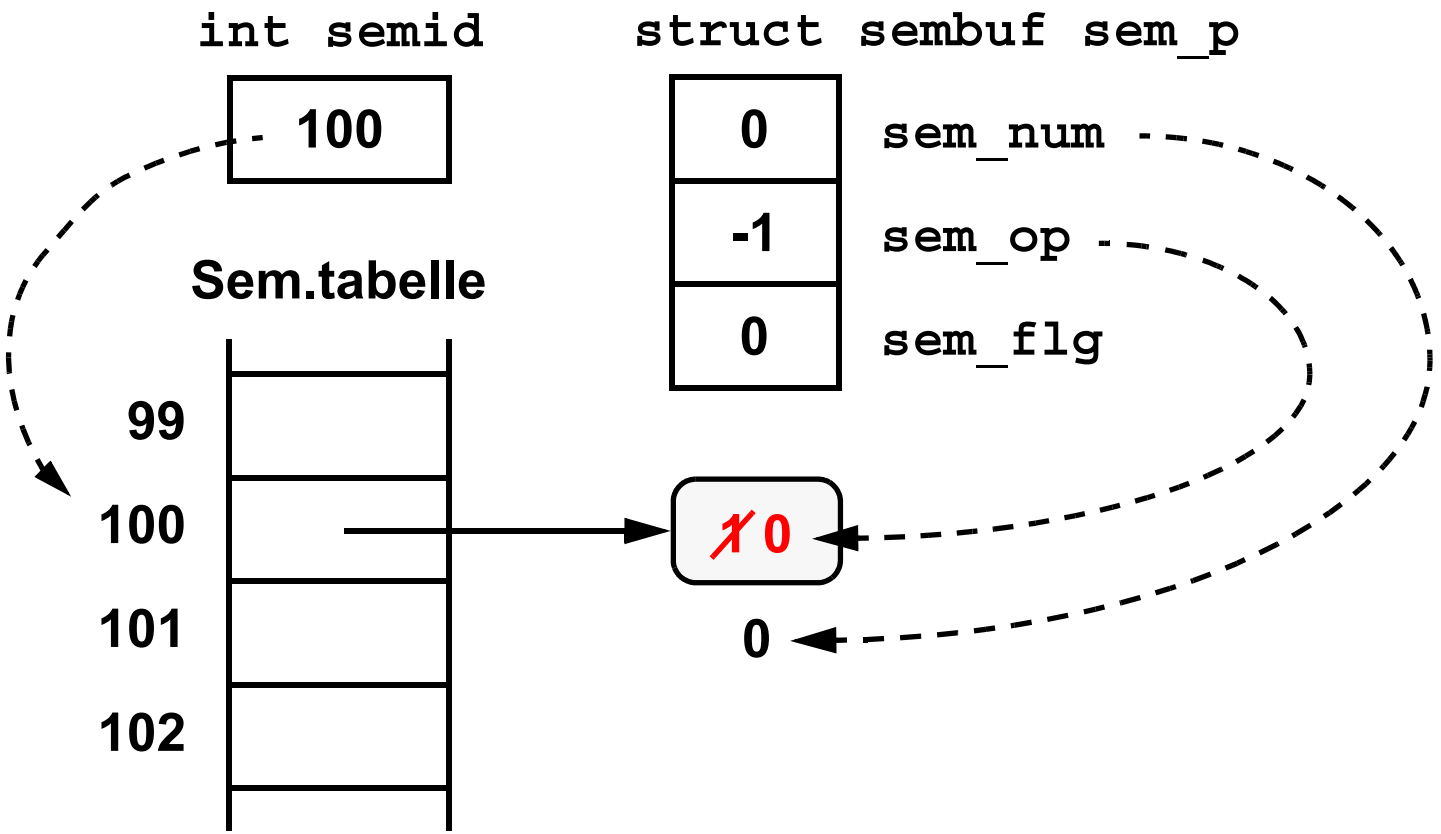
durchzuführende Operation

hier: Senken des Werts um 1, also P-Operation

4.) Operation auf einem Semaphor

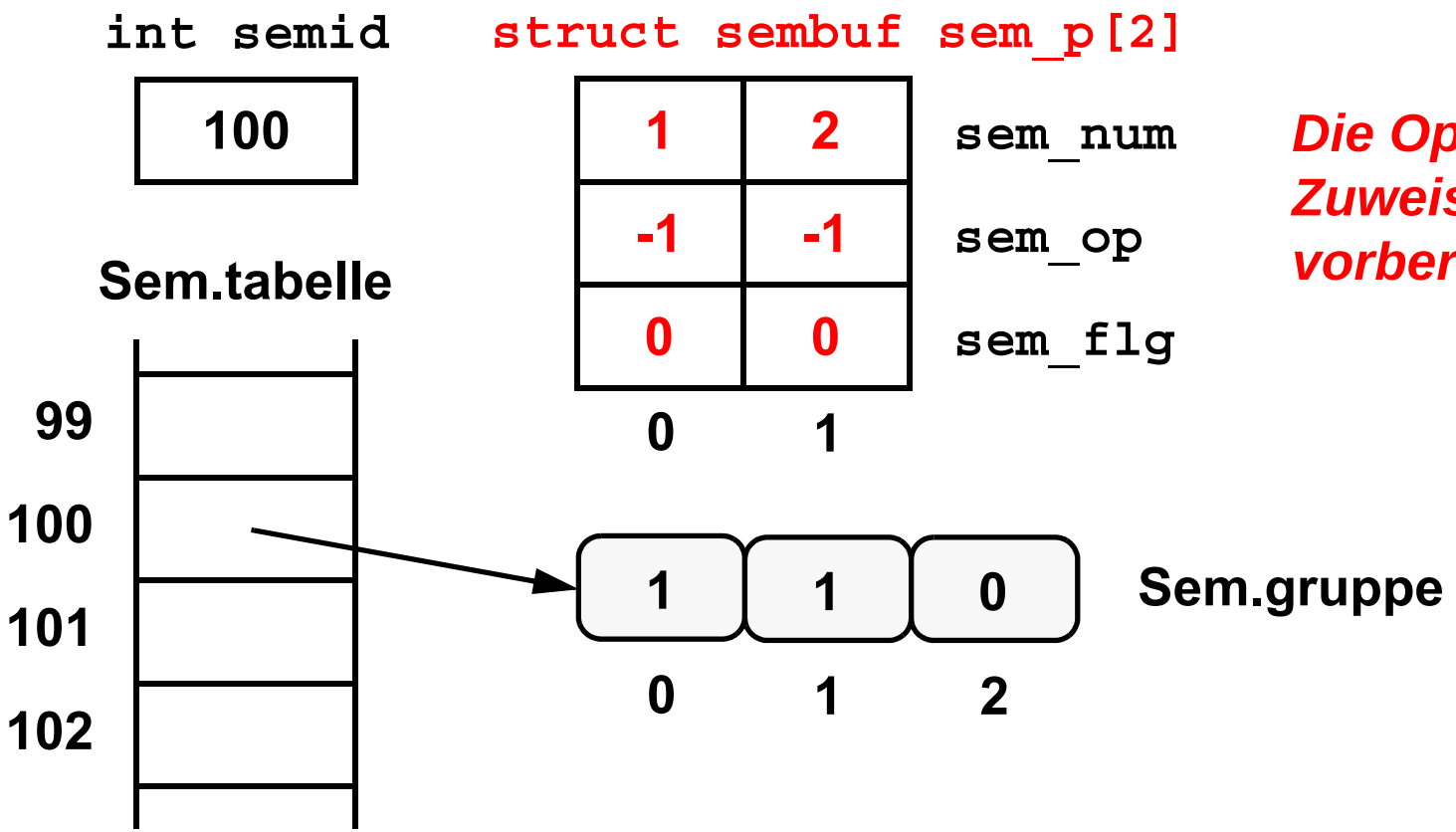
```
sem_p.sem_num = 0;
sem_p.sem_op  = -1;
sem_p.sem_flg = 0;
semop(semid,&sem_p,1);
```

... und mit semop() ausgeführt.



5.) Operation auf mehreren Semaphoren

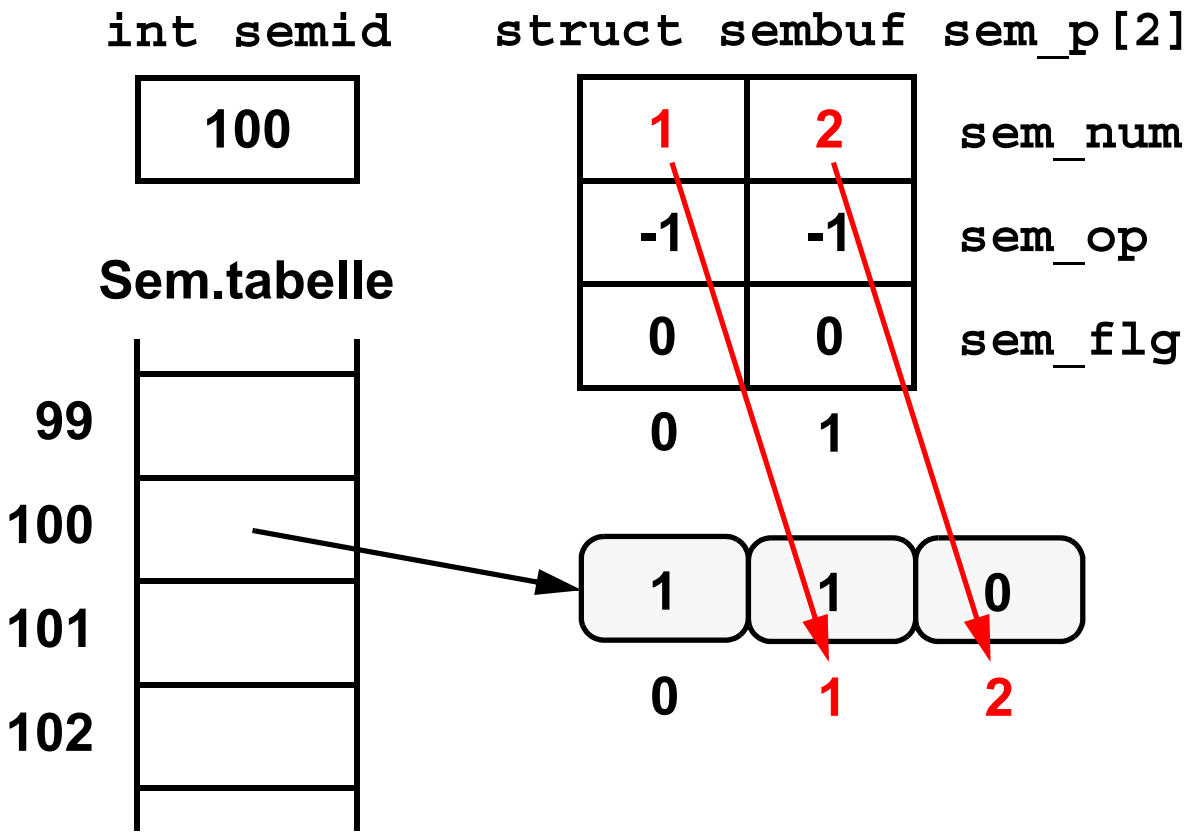
```
sem_p[0].sem_num = 1;   sem_p[1].sem_num = 2;
sem_p[0].sem_op  = -1;  sem_p[1].sem_op  = -1;
sem_p[0].sem_flg = 0;   sem_p[1].sem_flg = 0;
semop(semid,sem_p,2);
```



Die Operation wird durch Zuweisungen an sem_p vorbereitet.

5.) Operation auf mehreren Semaphoren

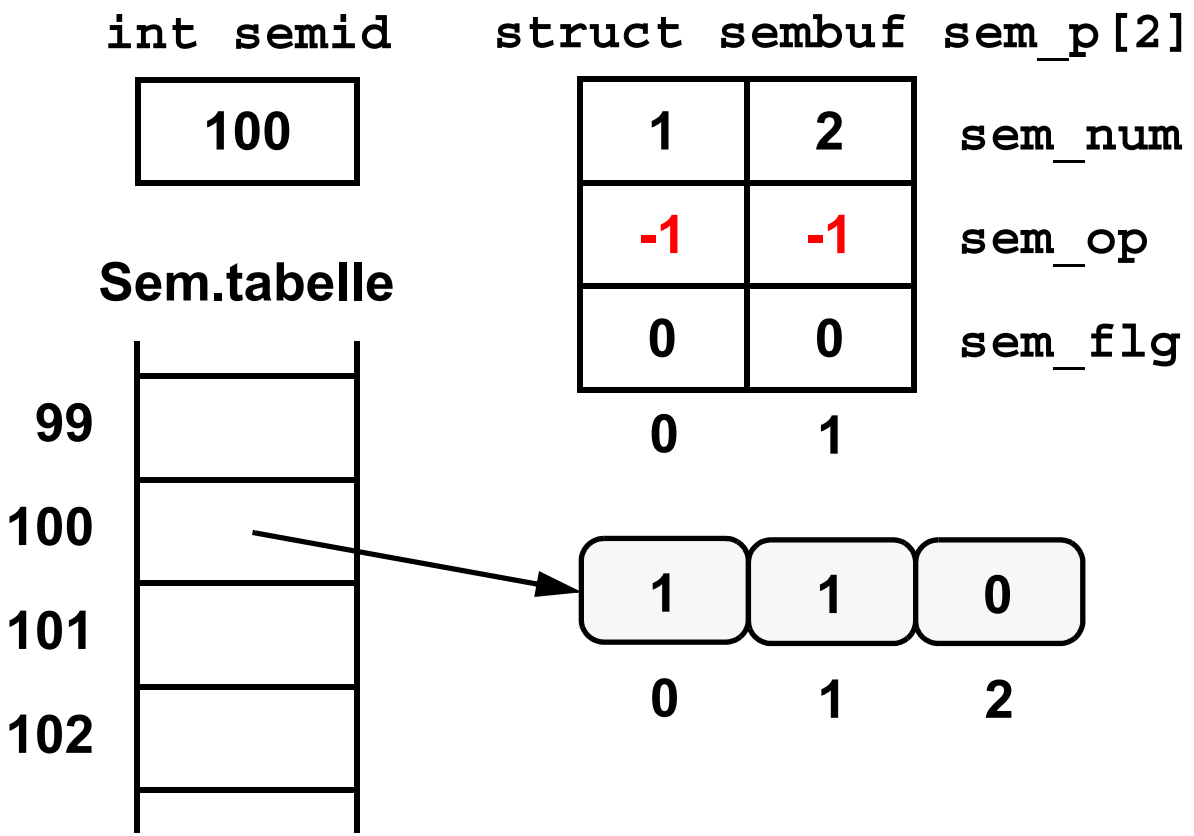
```
sem_p[0].sem_num = 1;  sem_p[1].sem_num = 2;
sem_p[0].sem_op  = -1; sem_p[1].sem_op  = -1;
sem_p[0].sem_flg = 0;  sem_p[1].sem_flg = 0;
semop(semid,sem_p,2);
```



Die Operation betrifft die Semaphoren 1 und 2, aber nicht Semaphor 0.

5.) Operation auf mehreren Semaphoren

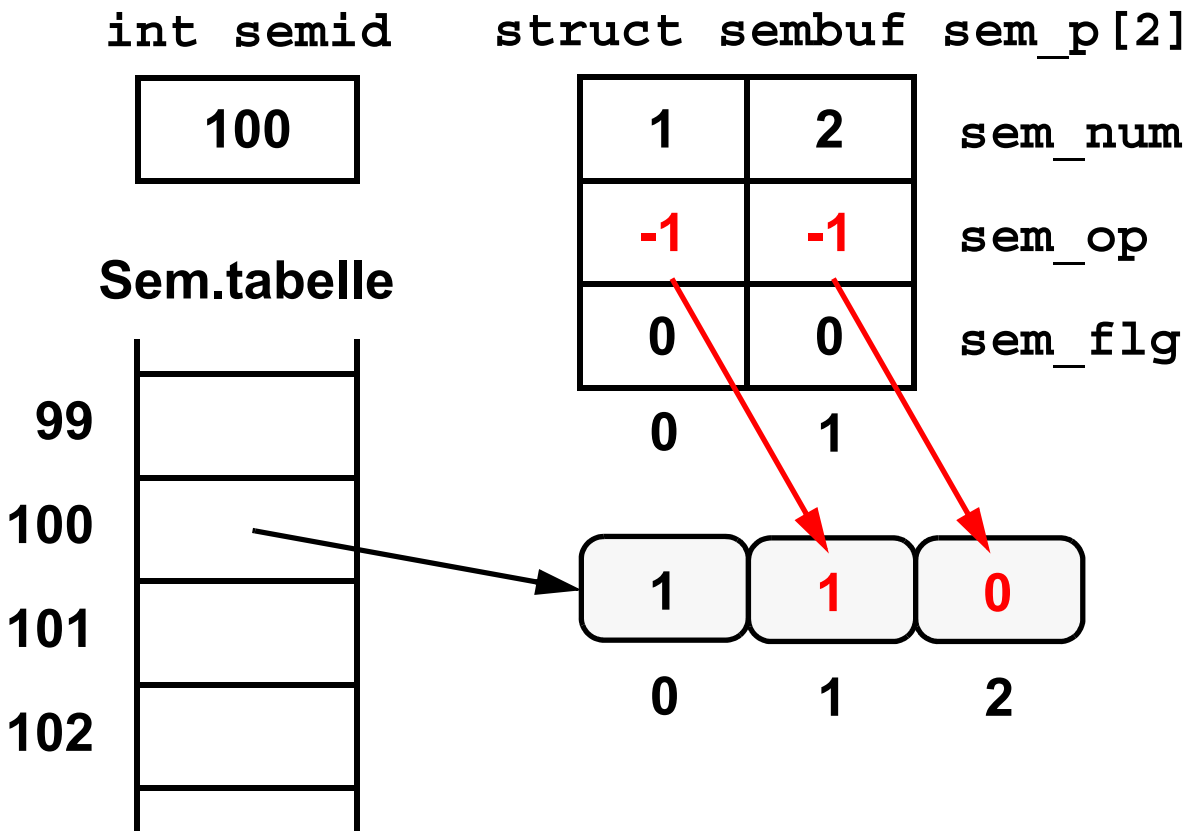
```
sem_p[0].sem_num = 1;   sem_p[1].sem_num = 2;
sem_p[0].sem_op  = -1;  sem_p[1].sem_op  = -1;
sem_p[0].sem_flg = 0;   sem_p[1].sem_flg = 0;
semop(semid,sem_p,2);
```



Auf den Semaphoren soll je eine P-Operation ausgeführt werden.

5.) Operation auf mehreren Semaphoren

```
sem_p[0].sem_num = 1;   sem_p[1].sem_num = 2;
sem_p[0].sem_op  = -1;  sem_p[1].sem_op  = -1;
sem_p[0].sem_flg = 0;   sem_p[1].sem_flg = 0;
semop(semid,sem_p,2);
```



semop() blockiert, da Sem. 2 nicht gesenkt werden kann. Alle Semaphorwerte bleiben unverändert.