Semaphores

E.W. Dijkstra (1965) abstracted the key notion of mutual exclusion in his concepts of semaphores.

Definition
A semaphore is a protected variable whose value can be accessed and altered only by the operations P and V and initialization operation called 'Semaphoiinitislize'.

Binary Semaphores can assume only the value 0 or the value 1 counting semaphores also called general semaphores can assume only nonnegative values.

The P (or wait or sleep or down) operation on semaphores S, written as P(S) or wait (S), operates as follows:

```
P(S):  IF   S  >  0
          THEN  S :=  S - 1
          ELSE   (wait on S)
```

The V (or signal or wakeup or up) operation on semaphore S, written as V(S) or signal (S), operates as follows:

```
V(S):  IF  (one or more process are waiting on S)
          THEN (let one of these processes proceed)
          ELSE   S := S +1
```

Operations P and V are done as single, indivisible, atomic action. It is guaranteed that once a semaphore operations has stared, no other process can access the semaphore until operation has completed. Mutual exclusion on the semaphore, S, is enforced within **P(S)** and **V(S)**.

If several processes attempt a P(S) simultaneously, only process will be allowed to proceed. The other processes will be kept waiting, but the implementation of P and V guarantees that processes will not suffer indefinite postponement.

Semaphores solve the lost-wakeup problem.

Producer-Consumer Problem Using Semaphores

The Solution to producer-consumer problem uses three semaphores, namely, full, empty and mutex.

The semaphore 'full' is used for counting the number of slots in the buffer that are full. The 'empty' for counting the number of slots that are empty and semaphore 'mutex' to make sure that the producer and consumer do not access modifiable shared section of the buffer simultaneously.

Initialization

- Set full buffer slots to 0.
  i.e., semaphore Full = 0.
- Set empty buffer slots to N.
  i.e., semaphore empty = N.
- For control access to critical section set mutex to 1.
  i.e., semaphore mutex = 1.

```
Producer ( )
WHILE (true)
        produce-Item ( );
    P (empty);
    P (mutex);
    enter-Item ( )
    V (mutex)
    V (full);

Consumer ( )
WHILE (true)
    P (full)
    P (mutex);
    remove-Item ( );
    V (mutex);
    V (empty);
```

consume-Item (Item)

Source:

http://www.personal.kent.edu/~rmuhamma/OpSystems/Myos/semaphore.htm