Implementation of General Semaphores Using Binary Semaphores

Anthony Howe

June 18, 2001

The Problem

Can we implement a general semaphore from a binary semaphore?

Semaphores Review

- One of the first mechanisms proposed to handle inter-process synchronization
- A semaphore is an integer value that is only accessed through two atomic operations: *wait* and *signal*
- Busy-wait version

```
Wait(S):        while S <= 0 do no-op;
                S := S - 1;

Signal(S):      S := S + 1;
```

Semaphores Continued

- Blocking version

```
Wait(S):        if S > 0 then
                        S:=S-1
                else
                        block execution of
                        calling process


Signal(S):      if processes blocked on S then
                        awaken one of them
                else
                        S:=S+1
```

- Binary semaphores only allow the integer to hold the values 0 and 1
- Binary semaphores are easier to implement than general semaphores

Solution #1

```
var
        mutex=1: binary-semaphore;
        delay=0: binary-semaphore;
        C={initvalue}: integer;

Procedure Wait()
        begin
        wait(mutex);
        C:=C-1;
        if C < 0 then begin
                signal(mutex);
                wait(delay);
                end
        else
                signal(mutex);
        end

Procedure Signal()
        begin
        wait(mutex);
        C:=C+1;
        if C <= 0 then
                signal(delay)
        signal(mutex)
        end
```

Solution #2

```
var
        mutex=1: binary-semaphore;
        delay=0: binary-semaphore;
        C={initvalue}: integer;

Procedure Wait()
        begin
        wait(mutex);
        C:=C-1;
        if C < 0 then begin
                signal(mutex);
                wait(delay);
                end
        signal(mutex);
        end

Procedure Signal()
        begin
        wait(mutex);
        C:=C+1;
        if C <= 0 then
                signal(delay)
        else
                signal(mutex)
        end
```

Solution #3

```
var
        mutex=1: binary-semaphore;
        delay=0: binary-semaphore;
        barrier=1: binary-semaphore;
        C={initvalue}: integer;

Procedure Wait()
        begin
        wait(barrier);
        wait(mutex);
        C:=C-1;
        if C < 0 then begin
                signal(mutex);
                wait(delay);
                end
        else
                signal(mutex);
        signal(barrier);
        end

Procedure Signal()
        begin
        wait(mutex);
        C:=C+1;
        if C = 1 then
                signal(delay)
        signal(mutex)
        end
```

Solution #4

```
var
        mutex=1: binary-semaphore;
        delay={min(1,initvalue)}: binary-semaphore;
        C={initvalue}: integer;

Procedure Wait()
        begin
        wait(delay);
        wait(mutex);
        C:=C-1;
        if C > 0 then
                signal(delay);
        signal(mutex);
        end

Procedure Signal()
        begin
        wait(mutex);
        C:=C+1;
        if C = 1 then
                signal(delay)
        signal(mutex)
        end
```

Performance

- Semaphore Operations

	c<=0		c=1		c>1	
	Wait()	Signal()	Wait()	Signal()	Wait()	Signal()
Semaphore	*1*	*1*	*1*	*1*	*1*	*1*
Solution #1	*Incorrect*	*Incorrect*	*Incorrect*	*Incorrect*	*Incorrect*	*Incorrect*
Solution #2	4	2	2	2	2	2
Solution #3	5	3	4	2	4	2
Solution #4	3	3	3	2	4	2

- Restrictive/Unrestrictive

Conclusions

- The implementation of general semaphores using binary semaphores must be implemented carefully so no concurrency errors are introduced

- Various solutions exist, when choosing a solution examine the performance characteristics of each that best suits your needs

- Implementation of general semaphores using binary semaphores is not recommended when efficiency is a concern

References

E. W. Dijkstra, *Cooperating Sequential Processes*, In F. Genuys (ed.) Programming Languages, 43-112. New York: Academic Press. 1968.

D. Hemmendinger, “A correct implementation of general semaphores”, *Operating Systems Review*, vol. 22, no. 3 (July, 1988), pp. 42-44.

D. Hemmendinger, “Comments on “A correct and unrestrictive implementation of general semaphores””, *Operating Systems Review*, vol. 23, no. 1 (January, 1989), pp. 7-8.

C. Samuel Hsieh, “Further comments on implementation of general semaphores”, *Operating Systems Review*, vol. 23, no. 1 (January, 1989), pp. 9-10.

P. Kearns, “A correct and unrestrictive implementation of general semaphores”, *Operating Systems Review*, vol. 22, no. 4 (October, 1988), pp. 46-48.

A. Silberschatz and P.B. Galvin. Operating System Concepts, fifth edition, Addison Wesley, Reading, Massachusetts, 1998.

A. Silberschatz and P.B. Galvin. Operating System Concepts, fifth edition, Addison Wesley, Reading, Massachusetts, August 1998.