Semaphore Implementation

- Typically, semaphore operations are implemented within the operating system itself.
- Instead of a busy wait (`while S <= 0;`), the operating system can block the process
  - Saves the CPU for the use of other processes
- A process executing a V may cause another previously blocked process to be put on the ready queue.

Semaphore Implementation

```
typedef struct sem {
    int value;
    list_of_processes plist; (processes blocked on this semaphore)
 } semaphore;
```

```
 P(S):
     S.value--;
     if (S.value < 0) {
         add this process to S.plist;
         block;
     }
```

```
V(S):
   S.value++;
   if (S.value <= 0) {
       get a process P from S.plist;
       wakeup(P);
   }
```

- How to ensure P and V are atomic?
  - uni-processor: disable interrupts (P & V are fairly quick)
  - multi-processor: software or hardware solutions
    - busy wait is negligible for P & V

Readers-Writers Problem

- A data set is shared among a number of concurrent processes
  - Readers – only read the data set; they do **not** perform any updates
  - Writers – can both read and write
- Problem – allow multiple readers to read at the same time. Only one single writer can access the shared data at the same time

Readers-Writers Problem (Cont.)

- Shared Data
    - Data set
    - Semaphore *mutex* – to control access to critical section
    - Semaphore *wrt* – to ensure mutual exclusivity when writing
    - Integer *readcount* – to count the readers

Readers-Writers Problem (Cont.)

- The structure of a writer process:

```
writer: while (1) {
                P (wrt) ;
                    //    writing is performed
                V (wrt) ;
        }
```

Readers-Writers Problem (Cont.)

- The structure of a reader process:

```
while (1) {
                P (mutex) ;
                readcount++ ;
                if (readcount == 1)
                         P (wrt) ;
                V (mutex)

                       // reading is performed

                 P (mutex) ;
                 readcount-- ;
                 if (readcount  == 0)
                        V (wrt) ;
                 V (mutex) ;
  }
```

```
SEM mutex = 1;
SEM wrt = 1;
int readcount = 0;
```