Allow multiple readers to execute in the critical section at once.
Cleanse mutual exclusion for writers.
Each read or write of the shared data must happen within a critical section.

Correctness criteria:

How do we control access to the object to permit this protocol?

- Allow only one writer at any point
- Want many readers reading the object at once

Using a single lock on the data object is overly restrictive

- Writers: read data and modify it
- Readers: read data, never modify it

Classes:

An object is shared among many threads, each belonging to one of two

Today: Synchronization for Readers/ Writers Problem

Semaphores are useful for mutual exclusion, progress and bounded waiting

Semaphore

A semaphore supports two atomic operations:

- V (S) → Increment semaphore S
- P (S) → Decrement semaphore S
```java
{ mutex<->S<->T<->G<->U<->V<->W<->X<->Y<->Z

// enable writers
mutex<->S<->T<->G<->U<->V<->W<->X<->Y<->Z
(if (readers == 0)
  readers -= 1;
else
  mutex<->S<->T<->G<->U<->V<->W<->X<->Y<->Z
<wait>

// try to get lock
mutex<->S<->T<->G<->U<->V<->W<->X<->Y<->Z
(if (readers == 1)
  mutex<->S<->T<->G<->U<->V<->W<->X<->Y<->Z
<wait>

// block writers
mutex<->S<->T<->G<->U<->V<->W<->X<->Y<->Z
(if (readers == 1)
  readers -= 1;
else
  mutex<->S<->T<->G<->U<->V<->W<->X<->Y<->Z
<wait>

// enable others
mutex<->S<->T<->G<->U<->V<->W<->X<->Y<->Z
<wait>

// any writers or readers
}

// enable writer
mutex<->S<->T<->G<->U<->V<->W<->X<->Y<->Z
<wait>

// enable reader
mutex<->S<->T<->G<->U<->V<->W<->X<->Y<->Z
<wait>

// semaphore mutex
Semaphore mutex;

// semaphore writer
Semaphore writer;

// semaphore reader
Semaphore reader;

private:
  void write() {
    if (write < value)
      write = value;
  };

public:
  void write() {
    if (write < value)
      write = value;
  }

class ReaderWriter {
);

Readers/Writers Problem
```
Read

Write

Read

Write

Readers/Writers: Scenario 1

Readers/Writers: Scenario 2
- Let a writer enter its critical section as soon as possible.

Alternative desirability semantics:

5. Does this solution guarantee all threads will make progress?

through (at least one is waiting on writ and zero or more can be waiting on writex).

4. If a writer exists and a reader goes next, then all readers that are waiting will fail.

depends on the schedule.

3. When a writer exists, if there is both a reader and writer waiting, which goes next.

2. The last reader to exit signals a waiting writer.

1. The first reader blocks if there is a writer, any other readers who try to enter block.

Implementation notes:

**Readers/Writers Solution: Discussion**

---

**Reader/Writers: Scenario 3**

```c
Read() Write()

Read() Read()

Read() Write()

Read() Write()
```
Readers/ Writers Solution Favoring Writers

Readers/Writers Solution Favoring Writers

{  
read mutex->lock
if (readers == 0) { 
    // enable readers
    readers = 1;
    mutex->lock
    // ensure mutual exclusion
    read mutex->mutex
    // perform read
}
write->lock

read->lock

if (readers == 1) { 
    // another reader
    readers = 1;
    mutex->lock
    // ensure mutual exclusion
    write->mutex
    // perform write
}}
write->lock

read->lock

if (readers == 1) { 
    // another reader
    readers = 1;
    mutex->lock
    // ensure mutual exclusion
    write->mutex
    // perform write
}}
write->mutex

Read ()
Write ()

Read ()
Write ()

Read ()
Write ()

Read ()
Write ()

Readers/Writers: Scenario 5

Readers/Writers: Scenario 4
After eating, put down both chopsticks and go back to thinking

- Block if neighbor has already picked up a chopstick
- Eating: need two chopsticks, try to pick up two closest chopsticks
- Thinking: do nothing
- Share a circular table with five chopsticks
- Five philosophers, each either eats or thinks

Other Synchronization Problems: Dining Philosophers

```
    (Write ()
     Read ()
     Write ()
    )
    
    Read: R1: R2:
    M1: M2:

Reader/Writers: Scenario 6
```
Summary

Starvation is possible in either case:

- Favor writers
- Favor readers

Two possible solutions using semaphores:

- Allow only one writer at a time
- Allow multiple readers to concurrently access a data

Readers/writers problem:

...