Starve-Free Readers Writers Problem

Overview

Starve Free Readers-Writers Problem: All ***readers*** and ***writers*** will be granted *access to the resource in their order of arrival*. If a writer arrives while readers are accessing the resource, it will wait until those readers free the resource, and then modify it. The same goes for readers a writer has the access to the resource.

This repo contains the pseudocode of the solution.

Documentation

Global variables:

- Use semaphores for mutex. All `semaphores` are initialized to `1`.
- `orderMutex`: Materialize order of arrival. Taken by the entity that requests the access to the resource and is released after it gains the access.
- `accessMutex`: Requested by a writer before modifying a resource.
- `readers`: Counter for the number of readers accessing the resource.
- `readersMutex`: Protect the counter against conflicting accesses.

```
semaphore orderMutex;         // Initialized to 1
semaphore accessMutex;        // Initialized to 1
semaphore readersMutex;       // Initialized to 1

unsigned int readers = 0;     // Number of readers accessing th
```

Readers Part:

- `Wait()`: Decrements the value of a semaphore by 1.
- `Signal()`: Increments the value of a semaphore by 1.

These are same as P() or V() which are generally used with semaphores.

```
void reader(){
  Wait(orderMutex);             // Remember our order of arrival

  Wait(readersMutex);           // We will manipulate the readers c
  if (readers == 0)             // If there are currently no reader
    Wait(accessMutex);          // requests exclusive access to the
  readers++;                    // Note that there is now one more

  Signal(orderMutex);           // Release order of arrival semapho
  Signal(readersMutex);         // We are done accessing the number

  ReadResource();               // Here the reader can read the res

  Wait(readersMutex);           // We will manipulate the readers c
  readers--;                    // We are leaving, there is one les
  if (readers == 0)             // If there are no more readers cur
    Signal(accessMutex);        // ...release exclusive access to t
  Signal(readersMutex);         // We are done accessing the number
}
```

Writers Part:

- `Wait()`: Decrements the value of a semaphore by 1.
- `Signal()`: Increments the value of a semaphore by 1.

These are same as P() or V() which are generally used with semaphores.

```
void writer(){
  Wait(orderMutex);             // Remember our order of arrival
  Wait(accessMutex);            // Request exclusive access to the
  Signal(orderMutex);           // Release order of arrival semapho

  WriteResource();              // Here the writer can modify the r

  Signal(accessMutex);          // Release exclusive access to the
}
```