resource\_type \*resource;

int nb\_read = 0;

mutex nb\_reader = 1

mutex lock\_writer = 1;

reader() {

P(nb\_reader) // a single reader changes the number at a time

nb\_read = nb\_read + 1;

if(nb\_read == 1) // only the first lock the access to writers

P(lock\_writer);

V(nb\_reader)

// no waiting for other readers

reading(resource); // lasts a random time

P(nb\_reader)

nb\_read = nb\_read - 1;

if(nb\_read == 0)

V(lock writer);

V(nb\_reader);

}

writer() {

P(lock\_writer); // competition with the first reader

writing(resource);

V(lock\_writer); // release the first reader

}

resource\_type \*resource;

int nb\_read = 0;

mutex nb\_reader= 1

mutex lock\_writer = 1;

mutex wrt\_wait = 1;

reader() {

P(nb\_reader) // a single reader changes the number at a time

nb\_read = nb\_read + 1;

if(nb\_read == 1) // only the first lock the access to writers

P(lock\_writer);

V(nb\_reader)

// no waiting for other readers

reading(resource); // lasts a random time

P(nb\_reader)

nb\_read = nb\_read - 1;

if(nb\_read == 0)

V(lock\_writer);

V(nb\_reader);

}

writer() {

P(wrt\_wait); // the other writers remain stuck here

P(lock\_writer); // competition with the first reader

writing(resource);

V(lock\_writer); // release the first reader

V(wrt\_wait); // a reader has passed before giving way to the next writer

}

The end of a writer releases the lock\_writer mutex thus releasing a possible reader before releasing the semaphore wrt\_wait.

resource type \*resource;

int nb\_read = 0;

int nb\_wrt = 0;

mutex nb\_reader = 1

mutex nb\_writer = 1

mutex lock\_reader = 1;

mutex lock\_writer = 1;

mutex rdr\_wait = 1;

writer() {

P(nb\_writer);

nb\_writer = nb\_writer + 1;

if(nb\_writer == 1)

P(lock\_reader); // stop the next readers

V(nb\_writer);

P(lock\_writer); // can create a queue of writers

writing(resource);

V(lock\_writer);

P(nb\_writer);

nb\_writer = nb\_writer - 1;

if(nb\_writer == 0)

V(lock­\_reader);

V(nb\_writer); // it's the last writer who unblocks the reader (s)

}

reader() {

// filter the readers one by one

// competition between a reader and writers

P(rdr\_wait); // prevents other readers from passing

P(lock\_reader); // the lock is obtained when the last writer release it

P(nb\_reader);

nb\_lect = nb\_lect + 1;

if(nb\_lect == 1)

P(lock\_writer); // competition with the writers (the first wins)

V(nb\_reader);

V(lock\_reader);

// a writer can obtain the lock while the other readers wait (rdr\_wait)

V(rdr\_wait);

reading(resource);

P(nb\_reader);

nb\_reader = nb\_reader - 1;

if(nb\_reader == 0)

V(lock\_writer);

//the last release the writers

V(nb\_reader);

}

resource\_type \*resource;

int nb\_read = 0;

mutex lock\_reader= 1

mutex lock\_writer = 1;

mutex fifo = 1;

writer() {

P(fifo); // queue of readers and writers

P(lock\_writer);

V(fifo);

writing(resource);

V(lock\_writer);

}

reader() {

P(fifo); // queue of readers and writers

P(lock\_reader);

nb\_read = nb\_read + 1;

if(nb\_read == 1)

P(lock\_writer);//competition with the writers

V(lock\_reader);

V(fifo);

reading(resource);

P(lock\_reader);

nb\_read = nb\_read - 1;

if(nb\_read == 0)

V(lock\_writer);

V(lock\_reader);

}