The Readers/Writers Problem (Courtois, 1971)

- Models access to a database, such as an airline reservation system
  - Multiple readers (customers) want to read the database — access schedules, flight availability, etc.
  - Multiple writers (the airline) want to write the database — update the schedules
- It is acceptable to have multiple readers at the same time
  - But while a writer is writing to (updating) the database, no one else can access the database — either to read or to write
- Two versions:
  - Readers have priority over writers
  - Writers have priority over readers
  - In most solutions, the non-priority group can starve

Readers/Writers w/ Readers Priority (Using Semaphores)

Initialization:

```
semaphore mutex = 1;  /* for mutual exclusion */
semaphore write = 1;  /* for mut. ex. & synch. */
int readers = 0;      /* number of readers */
```

Reader:

```
P(mutex);
readers++;
if (readers == 1)
    P(write);
V(mutex);

read database

P(mutex);
readers– –;
if (readers == 0)
    V(write);
V(mutex);
```

Writer:

```
P(write);
write database
V(write);
```

Notes on R/W w/ Readers Priority (Using Semaphores)

- Reader:
  - Needs mutually exclusive access while manipulating “readers” variable
  - Does not need mutually exclusive access while reading database
  - If this reader is the first reader, it has to wait if there is an active writer (which has exclusive access to the database)
    - First reader did a “P(write)”
  - If other readers come along while the first one is waiting, they wait at the “P(mutex)”
  - If other readers come along while the first one is actively reading the database, they can also get in to read the database
  - When the last reader finishes, if there are waiting writers, it must wake one up

Notes on R/W w/ Readers Priority (Using Semaphores) (cont.)

- Writer:
  - If there is an active writer, this writer has to wait (the active writer has exclusive access to database)
  - If there are active readers, this writer has to wait (readers have priority)
    - First reader did a “P(write)”
  - The writer only gets in to write to the database when there are no other active readers or writers
  - When the writer finishes, it wakes up someone (either a reader or a writer — it’s up to the CPU scheduler)
  - If a reader gets to go next, then once it goes through the “V(mutex)” and starts reading the database, then all other readers waiting at the top “P(mutex)” get to get in and read the database as well

Readers/Writers w/ Writers Priority (Using Semaphores)

Reader:

```
P(mutex);
if (AW+WW > 0)
    WR++;
else {
    V(OKToRead);
    AR++;
}
V(mutex);
P(OKToRead);

read database

P(mutex);
AR– –;
if (AR == 0 &&
    WW > 0) {
    V(OKToWrite);
    AW++; WW– –;
}
V(mutex);
```

Writer:

```
P(mutex);
if (AW+AR > 0)
    WW++;
else {
    V(OKToWrite);
    AW++;
}
V(mutex);
P(OKToWrite);

write database

P(mutex);
AW– –;
if (WW > 0) {
    V(OKToWrite);
    AW++; WW– –;
} else if (WR > 0) {
    V(OKToRead);
    AR++;  WR– –;
}
V(mutex);
```

Notes on R/W w/ Writers Priority (Using Semaphores)

- Reader:
  - If there are active or waiting writers, this reader has to wait (writers have priority)
  - Otherwise, this reader can read (possibly along with other readers)
  - When the last reader finishes, if there are waiting writers, it must wake one up
- Writer:
  - If there are active readers or writers, this writer has to wait (everyone has to finish before writer can update database)
  - Otherwise, this writer can write (and has exclusive access to database)
  - When the writer finishes,
    - (first choice) if there are waiting writers, it must wake one up (writers have priority)
    - (second choice) if there are waiting readers, it must wake one up

Readers/Writers w/ Writers Priority (Using Locks and CVs)

Reader:

```
acquire(mutex);
while (AW+WW > 0) {
    WR++;
    wait(OKToRead);
    WR– –;
}
AR++;
release(mutex);

read database

acquire(mutex);
AR– –;
if (AR == 0 &&
    WW > 0)
    signal(OKToWrite);
release(mutex);
```

Writer:

```
acquire(mutex);
while (AW+AR > 0) {
    WW++;
    wait(OKToWrite);
    WW– –;
}
AW++;
release(mutex);

write database

acquire(mutex);
AW– –;
if (WW > 0)
    signal(OKToWrite);
else
    br’cast(OKToRead);
release(mutex);
```

Readers/Writers w/ Writers Priority (Using Locks and CVs, Solution 2)

Reader:

```
acquire(mutex);
while (i <0)
    wait(access);
i++;
release(mutex);

read database

acquire(mutex);
i– –;
if (i == 0)
    signal(access);
release(mutex);
```

Writer:

```
acquire(mutex);
while (i != 0)
    wait(access);
i– –;
release(mutex);

write database

acquire(mutex);
i = 0;
br’cast(access);
release(mutex);
```

- Notes:
  - “access” conveys right to access
  - If i > 0, i counts number of active readers
  - If i == 0, no one is accessing the data
  - If i < 0, there is an active writer