Readers/Writers Problem Solution

- This program ensures mutual exclusion between writers, and between the 1st reader and any writers, but not between multiple readers.
- A semaphore named mutex is used to ensure mutual exclusion when readers update a shared counter called readcount, which tracks the number of active readers. Another semaphore named wrt is used to control access to the shared resource. It is acquired by writers and by the first reader.
- First Reader Behavior: If the reader finds that it is the first one to enter (i.e., readcount increments from 0 to 1), it calls sem_wait(&wrt) to acquire the lock wrt. This prevents any writer from entering the critical section while at least one reader is present.
- Last Reader Behavior: If the reader finds that it has been the last to exit (i.e., readcount becomes 0), it calls sem_post(&wrt) to allow a writer (if any are waiting) to acquire the lock wrt and enter the critical section.
- Writer Behavior: A writer begins by calling sem_wait(&wrt) to acquire the lock wrt and enter the critical section to write data. Since a writer must have exclusive access, it will block until wrt is available—that is, until no reader holds it (because the first reader acquired it) and no other writer is active. Upon exiting the critical section, it calls sem_post(&wrt) to allow waiting readers or writers to continue.
- Readers-Preference and Its Consequences: Because the first reader blocks any writer until all readers have exited, if new readers continuously arrive, a writer may starve. This readers-preference model is efficient for systems primarily performing read operations but might cause fairness issues when writes are necessary.

```
/* shared memory */
semaphore mutex;
semaphore wrt;
int readcount;
```

```
/* initialization.*/
mutex = 1;
wrt   = 1;
readcount = 0;
```

```
/* writer */
sem_wait(&wrt);


… critical section
to write data …


sem_post(&wrt);
```

```
/* reader */
sem_wait(&mutex);
readcount++;
if(readcount==1)
   sem_wait(&wrt);
sem_post(&mutex);

… read data …

sem_wait(&mutex);
readcount--;
if(readcount==0)
   sem_post(&wrt);
sem_post(&mutex);
```