Problema dei lettori-scrittori

Problema e possibili soluzioni

Francesco Lucia

Università degli studi della Basilicata

A.A. 2021/2022

Sommario

Il Problema

Il problema dei lettori-scrittori è un classico problema di concorrenza e sincronizzazione che analizza uno scenario che si verifica in molti programmi applicativi.
Nel problema si ha un dato condiviso (che può essere un file, una variabile, un database, un'area di memoria) e una serie di processi che vogliono accedervi in modalità di scrittura o lettura. I processi che chiameremo "lettori" intendono leggere il contenuto condiviso, i processi "scrittori" intendono modificarlo.
La differenza con un problema di sincronizzazione classico è nel fatto che è possibile differenziare le strategie da adottare in base al tipo di processo che accede alla risorsa ottimizzando la soluzione.

Analisi del problema

Infatti se i processi lettori leggessero contemporaneamente a un processo scrittore, potrebbero leggere solo una parte del contenuto o non riuscire ad accedervi affatto. Se due scrittori scrivessero contemporaneamente potrebbero sovrascrivere uno il lavoro dell'altro o generare dati corrotti. Mentre se due lettori dovessero leggere contemporaneamente non si verificherebbe alcun problema.
Analizzando il problema si possono quindi riassumere una serie di considerazioni:

- Più lettori possono leggere contemporaneamente i dati.
- Se uno scrittore sta scrivendo, nessun processo può leggere.
- Solo uno scrittore per volta può scrivere.

Creazione di lettori e scrittori

Nell'implementare la soluzione sono stati creati 12 threads lettori e 7 threads scrittori con l'utilizzo della libreria pthreads. Questa permette di avviare un thread con la sintassi

```
int pthread_create(pthread_t *thread,
const pthread_attr_t* attr, void *(*start_routine)(void*)
, void* arg );
```

Che prende come parametri il riferimento al descrittore del thread (che contiene il thread_id), un puntatore alla struttura (attr) con le caratteristiche del thread, la funzione eseguita dal thread e il puntatore ai parametri da passare alla funzione.
In questo caso per la creazione degli scrittori è sufficiente l'istruzione

```
pthread_create(&scrittori[i], NULL, scrittore, (void*)i);
```

Dove l'array scrittori è stato creato precedentemente con l'istruzione

```
pthread_t scrittori[NUMERO_SCRITTORI];
```

Soluzione con semafori e lock mutex

Per semplicità nella soluzione proposta la memoria condivisa è rappresentata da una semplice variabile con visibilità globale, inizialmente con valore 1.
I lettori non fanno altro che stampare il contenuto della variabile sullo schermo, gli scrittori invece modificano la variabile moltiplicandone il contenuto per un numero casuale tra 1 e 10, con l'istruzione

```
variabile_condivisa *= rand()%10+1;
```

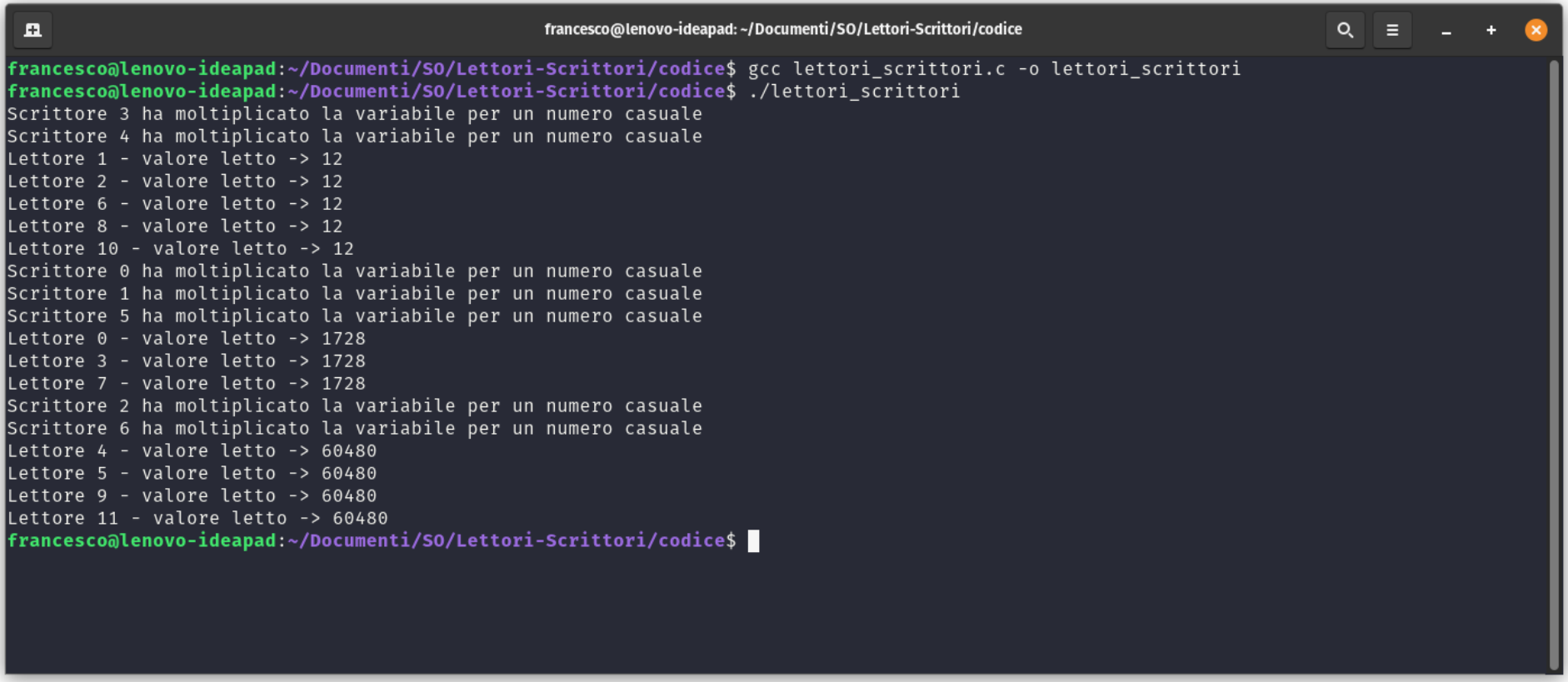
A causa della semplicità delle operazioni di lettura e scrittura (trattandosi di una sola variabile), le operazioni vengono effettuate istantaneamente dai threads alla loro creazione. Per simulare una casualità la prima istruzione di un thread è una sleep con un parametro generato casualmente tra 0 e 3 secondi dalla funzione *float randomFloat(float min, float max)*

Soluzione con semafori e lock mutex

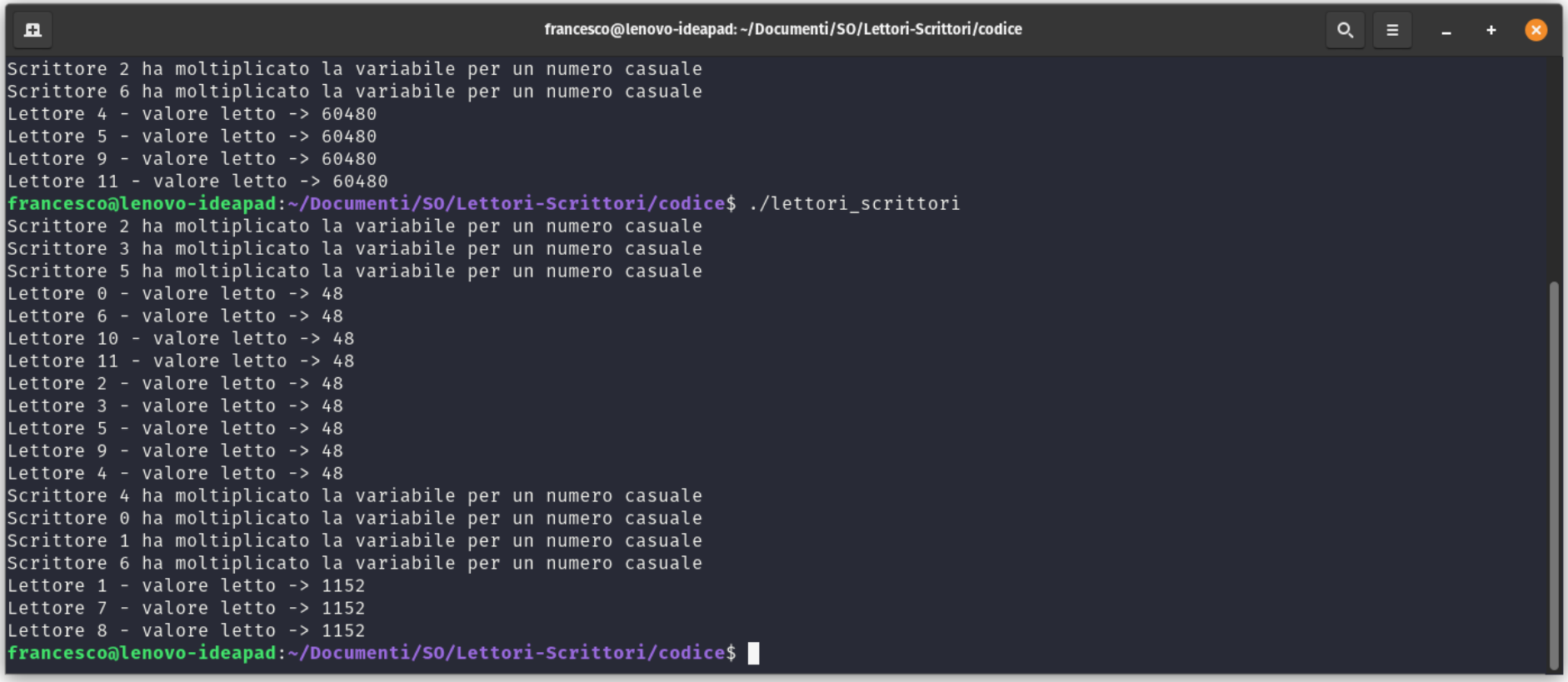
La soluzione privilegia i lettori. La variabile *num_lettori* conta i lettori attualmente attivi. La sincronizzazione è gestita attraverso l'utilizzo in un lock mutex (*mutex*) per la sezione critica relativa all'aggiornamento della variabile contatore dei lettori, e un semaforo (*semaforo_scrittura*) per regolare l'autorizzazione a scrivere per gli scrittori.

Colonne

Prima esecuzione

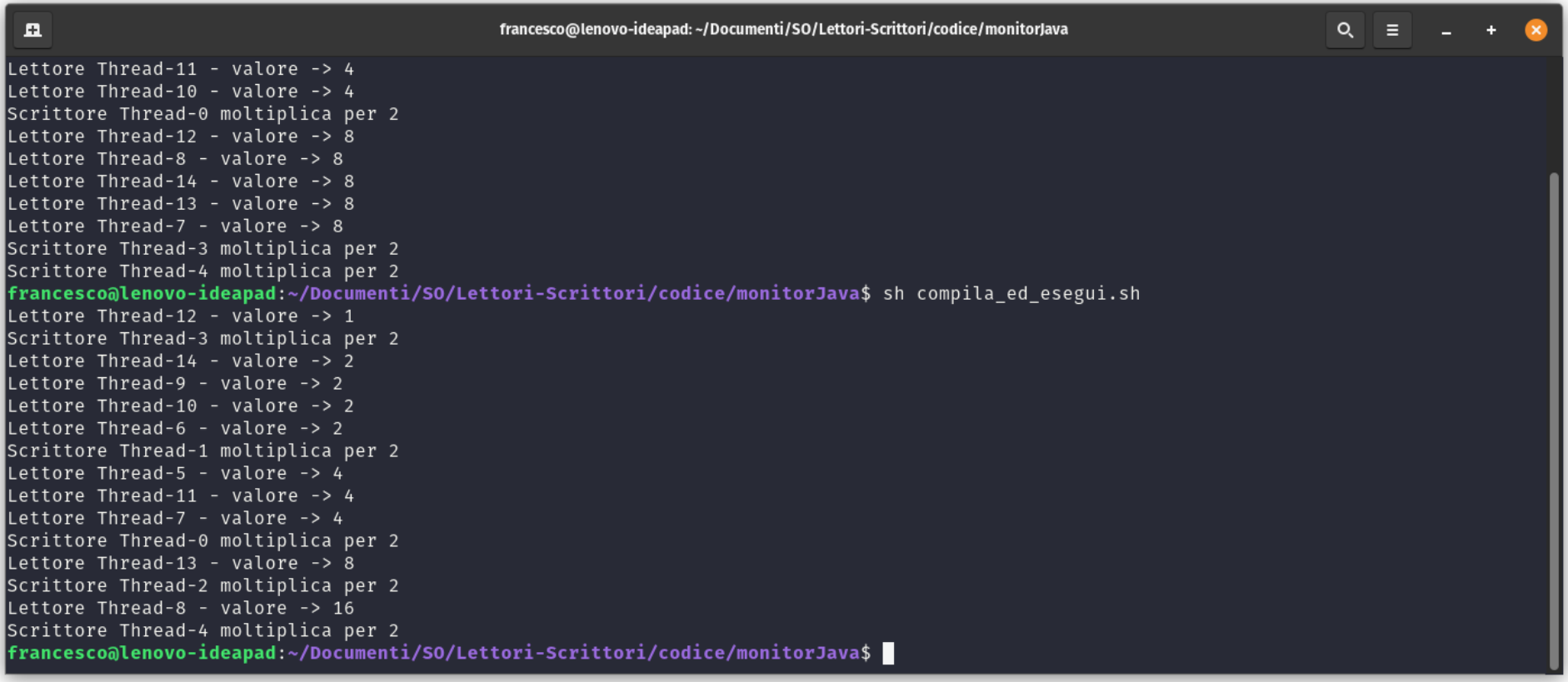


Seconda esecuzione



Prima esecuzione

Seconda esecuzione



Link al repository con i sorgenti delle soluzioni viste e delle slides.
https://github.com/FrancescoLucia/Lettori-Scrittori.git