Corso di Sistemi Operativi e Reti

Corso di Sistemi Operativi

Prova scritta 28 Giugno 2022

ISTRUZIONI PER CHI È IN PRESENZA:

1. **Rinomina** la cartella chiamata "Cognome-Nome-Matricola" che hai trovato sul Desktop e in cui hai trovato questa traccia, sostituendo "Cognome" "Nome" e "Matricola" con i tuoi dati personali e **lasciando i trattini**; se hai un doppio nome oppure un doppio cognome dovrai chiamare la cartella come in questo esempio:
    a. DeLuca-MarcoGiovanni-199999
2. **Carica** tutto il materiale didattico che vorrai usare sul Desktop; puoi farlo solo nei primi 5 minuti della prova;
3. **Svolgi** il compito; lascia tutto il sorgente che hai prodotto nella cartella di cui al punto 1;
4. Quando hai finito lascia la postazione facendo logout,

senza spegnere il PC.

SALVA SPESSO

ISTRUZIONI PER CHI SI TROVA ONLINE:

1. **Questo file contiene il testo che ti è stato dato ieri, incluso il codice;**
2. **Mantieni a tutto schermo** questo file per tutta la durata della prova; puoi scorrere liberamente tra le sue pagine, ma non puoi cambiare applicazione;
3. **Firma** preliminarmente il foglio che userai per la consegna con nome cognome e matricola;
4. **Svolgi** il compito; puoi usare solo carta, penna e il tuo cervello;
5. **Aiutati** con i numeri di linea per indicare le eventuali modifiche che vorresti fare al codice che ti è stato dato.
6. **Alla scadenza** termina *immediatamente* di scrivere, e attendi di essere chiamato, pena l'esclusione dalla prova;
7. **Quando è il tuo turno** mostra il foglio ben visibile in webcam, e poi metti una foto dello stesso foglio in una chat privata Microsoft Teams con il prof.

CI SONO DEI PUNTI AMBIGUI NELLA TRACCIA? COMPLETA TU

È parte integrante di questo esercizio completare le specifiche date nei punti non esplicitamente definiti, introducendo nuove strutture dati, o estendendo quelle preesistenti laddove si ritenga necessario, risolvendo eventuali ambiguità. Si può cambiare il codice dei metodi esistenti dove serve.

POSSO CAMBIARE IL PROTOTIPO DEI METODI RICHIESTI O DI QUELLI ESISTENTI? NO

Non è consentito modificare il prototipo dei metodi se questo è stato fornito. Potete aggiungere qualsivoglia campo e metodo di servizio, e qualsivoglia classe ausiliaria, ma NON variare l'interfaccia dei metodi pubblici già specificati. Analogamente, i metodi esistenti possono essere modificati nel loro codice, ma non se ne deve cambiare il risultato finale o il significato.

CHE LINGUAGGIO POSSO USARE? PYTHON 3.X

Il linguaggio da utilizzare per l'implementazione è Python 3.6 o successivo. Ricorda che l'operatore di formattazione `f` (esempio, `f"Ciao sono la stringa {testo}"`) è disponibile solo dalla versione 3.6 di Python in poi, ma può essere sostituito con `"Ciao sono la stringa %s" % testo`

POSSO CONSENTIRE SITUAZIONI DI RACE CONDITION NEL MIO CODICE? NO
POSSO CONSENTIRE SITUAZIONI DI DEADLOCK NEL MIO CODICE? NO
POSSO CONSENTIRE ALTRE SITUAZIONI DI BLOCCO TOTALE NEL MIO CODICE, TIPO NESTED LOCKOUT, LIVELOCK O ALTRO? NO
POSSO CONSENTIRE SITUAZIONI DI STARVATION NEL MIO CODICE? SI, tranne quando ti viene chiesto esplicitamente di rimuoverle

MA IL MAIN() LO DEVO AGGIORNARE? E I THREAD DI PROVA? SI

E' obbligatorio implementare esplicitamente del codice di prova oppure modificare il codice di prova pre-esistente, e accertarsi che giri senza errori prima della consegna.

MATERIALE PER ESERCIZIO 1 (Programmazione multithread. Punti: 0-20)

Il codice fornito implementa una classe `ReadWriteLockEvoluto` secondo le specifiche di seguito riportate.

La classe `ReadWriteLockEvoluto` si comporta come un Read/Write Lock tradizionale ma aggiunge i seguenti metodi:

`setReaders(self, max_readers :int).` Il metodo imposta un limite sul numero massimo di lock in lettura possibili sul `ReadWriteLockEvoluto` in esame e termina immediatamente. L'impostazione del numero massimo di lettori ha effetto sulle chiamate ad `acquireReadLock`. Le richieste di locking in modalità lettura che eccedono il numero precedentemente impostato di lettori (*readers*) vengono poste in attesa. Quando si reimposta `max_readers`, non c'è nessun impatto sui lettori già in possesso del lock (ad esempio, se 20 lettori possiedono il read lock, e improvvisamente `max_readers` scende a 10, non si agisce sui 20 lettori già in possesso del read lock, ma solo sui futuri lettori). Il limite di default per il numero massimo di lettori su un `ReadWriteLockEvoluto` è 10.

Esempi:

1. Si assuma che il numero massimo di lettori sia stato precedentemente impostato a 5 tramite la funzione `setReaders.` Nel caso in cui ci siano 5 thread "lettori" che possiedono già il lock in lettura non sarà possibile per un ipotetico sesto thread acquisire il read_lock. Tale thread dovrà quindi rimanere in attesa finchè almeno 1 degli altri thread non rilasci il lock in lettura.

2. Si supponga di effettuare la chiamata `L.setReaders(5)` su un certo lock L, mentre sono già presenti 7 lock in lettura; in tal caso i 7 lock in lettura vengono mantenuti, ma non sarà possibile acquisire futuri lock in lettura, finchè i lock attivi non siano almeno 4.

`enableWriters(self, enable: bool)`. Tramite questo metodo è possibile abilitare e disabilitare l'acquisizione di lock in scrittura sul `ReadWriteLockEvoluto` (se `enable = true` → la scrittura sulla risorsa sarà abilitata, se `enable = false` → la scrittura sarà disabilitata).

N.B. Quando si disabilita la possibilità di acquisire il lock in scrittura, eventuali write lock precedentemente acquisiti vengono mantenuti. Tutti i successivi thread che dovessero tentare di acquisire il lock in scrittura entrano in uno stato di attesa, finché la possibilità di acquisire il lock in scrittura non viene riabilitata.

ESERCIZIO 1 - PROGRAMMAZIONE MULTITHREADED

Punto 1

Si modifichi il codice di prova introducendo un secondo valore condiviso, chiamato `dc2`, di tipo `ReadWriteLockEvoluto`. Si lancino più istanze di thread di tipo `Copiatore`. Un thread `Copiatore` periodicamente sceglie a caso un valore tra `dc` e `dc2` e lo copia sull'altro elemento non sorteggiato. Ad esempio, se viene scelto `dc`, allora bisogna effettuare l'operazione `dc2.dato=dc.dato`. Altrimenti bisognerà fare l'operazione `dc.dato = dc2.dato`.

Punto 2

Modificare `getDato` e `setDato` in maniera tale da provocare l'eccezione `NoLockAcquired` allorquando questi metodi vengono invocati senza che si possieda il lock corretto. Si noti che il possesso del write lock deve consentire di invocare sia `getDato` che `setDato`, mentre il possesso del read lock deve consentire di invocare solamente `getDato`.

Punto 3

Si noti che se lo stesso thread T invoca per due volte consecutive acquireWriteLock, T si blocca *in attesa di sè stesso*. Lo stesso problema si verifica se uno stesso thread invoca tante volte acquireReadLock, fino a saturare il numero di lettori disponibili, oppure quando uno scrittore, già in possesso del lock in scrittura, prova ad acquisire il lock in lettura.
Si modifichi il `readwritelockevoluto` in maniera tale da ignorare eventuali invocazioni consecutive di *acquireReadLock o acquireWriteLock*, così rendendo il `readwritelockevoluto` *rientrante*.

Esempio:

```
1.dc.acquireWriteLock()
2.dc.acquireWriteLock()
3.prints("Che voglia di stampare che ho")
```

Con il sorgente fornito, un thread si bloccherebbe per sempre sul rigo 2. Questo non deve succedere, la seconda chiamata ad acquireWriteLock deve terminare immediatamente senza attese, poiché il thread corrente già possiede lo stesso lock, che è stato acquisito sul rigo 1.

SALVA SPESSO

ESERCIZIO 2, TURNO 1 - PERL

Si scriva uno script Perl dal nome **`shopper.pl`** che consenta di esplorare i prodotti disponibili presso un certo negozio. I prodotti disponibili sono listati, insieme al prezzo di vendita, in un insieme di file contenuti nella cartella `PRODOTTI`. Tali file sono denominati come `nome_categoria.txt` e contengono l'elenco dei prodotti appartenenti alla corrispondente categoria:

Esempio:

Il file denominato `alimenti.txt` conterrà i seguenti prodotti

```
pasta|0.99
tonno|3.49
pesto|3.49
```

Il file denominato `igiene.txt` conterrà i seguenti prodotti

```
deodorante|3.15
spazzolino|2.00
```

Lo script sarà invocato nel seguente modo

```
shopper.pl OPZIONI path [prezzo_massimo]
```

I parametri `OPZIONI` e `path` sono obbligatori, mentre il parametro `prezzo_massimo` è in genere opzionale, ma in base all'opzione utilizzata può diventare obbligatorio (**si veda sotto**).

I possibili valori del parametro `OPZIONI` sono i seguenti:

- **`-a`**
  Quando si usa l'opzione **`-a`** il parametro **`prezzo_massimo`** **NON VA INDICATO** . Bisogna produrre in output `STDOUT` l'elenco dei prodotti divisi per categoria, ordinando la stampa in ordine alfabetico per categoria e prodotto. Il parametro **`path`** conterrà il path alla cartella `PRODOTTI`
  `Esempio:`
  Lo script potrà essere lanciato nel seguente modo:

  ```
  ./shopper.pl -a path_alla_cartella
  ```

  e dovrà produrre in output:

  ```
  alimenti
  - pasta
  - pesto
  - tonno
  igiene
  - deodorante
  - spazzolino
  ```

- **`-c`**
  Quando si usa l'opzione **`-c`** il parametro **`prezzo_massimo`** diventa obbligatorio. Bisogna produrre in output STDOUT l'elenco dei prodotti con prezzo inferiore a **`prezzo_massimo`** presenti nel file indicato dal parametro **`path.`** La stampa conterrà i nomi dei prodotti e i relativi prezzi: sarà ordinata per prezzo *crescente* e, a parità di prezzo, in ordine alfabetico per prodotto.

Esempio:

Lo script potrà essere lanciato nel seguente modo:

```
./shopper.pl -c path_a_alimenti.txt "4.0"
```

e dovrà produrre in output:

```
pasta : 0.99
pesto : 3.49
tonno : 3.49
```

- **-cd**

  Lo script si comporta come quando invocato con l'opzione **-c.** La differenza è nella produzione dell'output: la stampa verrà infatti fatta su un `FILE` dal nome **out.log** e sarà ordinata per prezzo *decrescente* e, a parità di prezzo, si manterrà l'ordine alfabetico per prodotto.

  Esempio:

  Lo script potrà essere eseguito nel seguente modo:

  ```
  ./shopper.pl -cd path_a_alimenti.txt "4.0"
  ```

  e dovrà produrre in output un file dal nome **out.log** con il seguente contenuto al suo interno:

  ```
  tonno : 3.49
  pesto : 3.49
  pasta : 0.99
  ```