Corso di Sistemi Operativi e Reti

Prova scritta telematica del 30 LUGLIO 2020

ESERCIZI 1 e 2 - MATERIALE PRELIMINARE E ISTRUZIONI

ISTRUZIONI

1. **Questo documento è diviso in tre parti.**
2. **Nella prima parte trovi la traccia di un esercizio insieme con la sua soluzione commentata.** Fino al momento dell'esame puoi analizzare questo codice da solo, in compagnia, facendo uso di internet o di qualsiasi altro materiale. Puoi fare girare il codice, puoi modificarlo, fino a che non lo hai capito a fondo. Per comodità, a questo file è allegato anche il sorgente in file di testo separato.
3. **Nella seconda parte di questo documento** trovi una domanda tipo, molto simile a quella che effettivamente ti verrà posta il giorno dell'esame. Puoi provare a svolgere questa domanda tipo per capire se sei in grado di sostenere la prova reale, ricordando che avrai a disposizione mezz'ora di tempo.
4. **Nella terza parte del documento** troverai una possibile soluzione alla domanda tipo.
5. **Allegato a questo file** trovi lo stesso codice Python dell'esercizio sulla programmazione multi-threaded presentato nella parte 1, e un file dal nome `auth.log`. Il file di log rappresenta il formato di input sul quale verterà l'esercizio da svolgere in Perl e contiene più righe che indicano tutti i tentativi di accesso (falliti e non) registrati da un certo server.

PRIMA PARTE - MATERIALE DIDATTICO

Il codice fornito implementa una versione del gioco “BassoNumero”, che consente la presenza simultanea di N thread giocatori in parallelo.

*Nel gioco del “BassoNumero”, ogni giocatore punta segretamente (cioè senza essere a conoscenza delle puntate altrui) un **qualsiasi** numero intero a partire da 1. Una volta che tutti i giocatori hanno effettuato le proprie puntate, queste vengono esaminate. Vince il giocatore che ha puntato il numero più basso che non sia stato puntato da nessun altro giocatore. Ad esempio, supponiamo di avere una partita in cui i giocatori G1, G2, G3, G4 e G5 effettuano rispettivamente le puntate:*

G1 : 1
G2: 1
G3: 2
G4: 3
G5: 13

In questo caso i numeri puntati da un solo giocatore sono 2, 3 e 13, mentre il numero 1 è stato puntato da due giocatori. Il giocatore vincente è G3, poichè “2” è la puntata unica più bassa.

La classe **`NumeroBasso`** è preposta allo scopo di gestire una o più partite. La classe contiene il metodo **`puntaNumero(num : int)`** che consente a un certo thread/giocatore di puntare il numero voluto, e del metodo **`gioca(int N) -> Thread,`** che avvia e gestisce una partita tra N giocatori simulati, restituendo infine il Thread ID (TID) del thread vincitore.

```
01: from threading import Thread, RLock, Condition, current_thread
02: from random import randint
03:
04: class Player(Thread):
05:
06:     def __init__(self,nb):
07:         super().__init__()
08:         self.nb = nb
09:
10:     def run(self):
11:         self.nb.puntaNumero(randint(1,10))
12:
13: class NumeroBasso:
14:
15:
16:     def __init__(self):
17:         self.giocate = []
18:         self.lock = RLock()
19:         self.threadGioca = Condition(self.lock)
20:         self.partitaInCorso = False
21:         self.nGiocate = 0
22:         self.barrier = None
23:
24:     def gioca(self,N : int) -> int:
25:         with self.lock:
26:             self.giocate = {}
27:             self.nGiocate = 0
28:             self.partitaInCorso = True
29:             for _ in range(0,N):
30:                 Player(self).start()
31:             while(self.nGiocate < N):
32:                 self.threadGioca.wait()
33:             self.partitaInCorso = False
34:             for k in sorted(self.giocate):
35:                 if len(self.giocate[k]) == 1:
36:                     print(f"Il vincitore è il thread {self.giocate[k][0]} che ha puntato il numero {k}")
37:                     return self.giocate[k][0]
```

```
38:             print("Non ci sono vincitori")
39:             return 0
40:
41:     def puntaNumero(self,n : int):
42:         with self.lock:
43:             self.giocate.setdefault(n,[]).append(current_thread().ident)
44:             self.nGiocate += 1
45:             self.threadGioca.notify()
46:
47: if __name__ == '__main__':
48:     gameManager = NumeroBasso()
49:     v = 1
50:     while v != 0:
51:         v = gameManager.gioca(10)
```

SECONDA PARTE - ESEMPIO DI DOMANDA

Aggiungi alla classe `NumeroBasso` un metodo `monitoraPartita()`. Tale metodo deve produrre una stampa ogni qualvolta avviene una giocata e infine terminare quando la partita termina. La stampa deve indicare il TID del thread che ha appena giocato e nulla altro.

Si scriva su carta il codice del metodo, e si indichi, aiutandosi con i numeri di riga, quali modifiche andrebbero apportate al codice pre-esistente.

TERZA PARTE - ESEMPIO DI SOLUZIONE

Alla classe NumeroBasso aggiungo il metodo:

```
def monitoraPartita(self):
        with self.lock:
            while len(self.ultimeGiocate) == 0 and self.partitaInCorso:
                self.threadGioca.wait()
                while len(self.ultimeGiocate) > 0:
                    print(f"{self.ultimeGiocate.pop()}")
                if not self.partitaInCorso:
                    return
```

Aggiungo subito dopo la linea 19:

```
            self.ultimeGiocate = []
```

Aggiungo subito dopo la linea 33:

```
            self.threadGioca.notify()
```

Aggiungo subito dopo la linea 45:

```
            self.ultimeGiocate.append(current_thread().ident)
```