

Corso di Sistemi Operativi e Reti

Prova scritta 18 GENNAIO 2021

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. **Alla scadenza** termina *immediatamente* di scrivere, e attendi di essere chiamato, pena l'esclusione dalla prova;
6. **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.

ESERCIZIO 1, TURNO 2 - PROGRAMMAZIONE MULTITHREADED

Si rammenti che il codice del metodo `set(i,v,d)` prevede di azzerare `In[i]` e `Out[_om(i)]` qualora questi diventassero uguali a seguito dell'operazione di `set`.

Si scriva invece una versione del metodo `set` denominata `wset` in cui, se l'operazione `wset` dovesse rendere uguali `In[i]` e `Out[_om(i)]`, anziché azzerare i due elementi, l'operazione viene sospesa con una attesa bloccante fintantoché l'operazione di `wset` stessa si possa effettuare senza che, a seguito di questa, si verifichi `In[i] == Out[_om(i)]`.

Si modifichino le altre parti del codice preesistente laddove lo si ritenga necessario.

ESERCIZIO 2, TURNO 1 - PERL

Si scriva uno script Perl dal nome `findWithGrep.pl` che riceve come argomenti di input il `path` ad una cartella del sistema, un parametro che può essere `-g` oppure `-u` e una stringa `S`. Lo script dovrà ricercare all'interno della directory specificata (e tutte le sue subdirectory) **tutti e solo i file** che contengono nel loro nome la sottostringa `S` raggruppati per utente se è specificata l'opzione `-u`, per gruppo se è specificato il parametro `-g`.

Per ogni utente (o gruppo) bisognerà calcolare la size totale tra i file selezionati allo step precedente; successivamente, lo script dovrà stampare su un `FILE` dal nome `results.out`, per ogni utente (rispettivamente per gruppo) la size totale dei file trovati. La stampa dovrà essere **ordinata in ordine decrescente di size TOTALE raggrupata per utente (rispettivamente per gruppo) e, a parità di dimensione, in ordine lessicografico**. Infine, bisognerà stampare la somma totale di spazio occupato dai file selezionati. Si veda esempio di stampa a pagina successiva.

Tutti i parametri sopra specificati (`path`, parametro `-u` oppure `-g` e stringa `S`) sono **obbligatori**. Non devono essere presenti ulteriori parametri.

Tutti i controlli sulle eventuali stringhe dovranno essere effettuati facendo opportuno uso di espressioni regolari (**REGEXP**).

Lo script sarà eseguito, quindi, con la seguente sintassi:

```
./findWithGrep.pl path/to/directory[-g|-u] string_S
```

ESEMPIO:

1. Contenuto della cartella GennaioTest :

```
.:
total 40
drwxrwxrwx 1 francesco admin      4096 Jan 16 10:49 .
drwxrwxrwx 1 francesco admin      4096 Jan 16 10:49 ..
drwxrwxrwx 1 root      root      4096 Sep 12 17:26 'Sotto cartella'
-rwxrwxrwx 1 francesco admin     5056 Sep 12 15:21 esempio.txt
-rwxrwxrwx 1 michele   francesco    0   Sep 12 14:03 main.pdf
-rwxrwxrwx 1 root      root         0   Sep 12 14:03 output.a
-rwxrwxrwx 1 michele   francesco    1709 Sep 12 17:31 soluzioneGennaio.pl
-rwxrwxrwx 1 francesco admin    13114 Sep 12 14:24 'test - Copia.docx'
-rwxrwxrwx 1 michele   francesco   12223 Sep 12 14:24 test.docx
-rwxrwxrwx 1 francesco admin      0 Sep 12 14:15 txt.ciao
```

'./Sotto cartella':

total 6784

drwxrwxrwx	1	root	root	4096	Sep 12 17:26	.
drwxrwxrwx	1	francesco	admin	4096	Jan 16 10:49	..
-rwxrwxrwx	1	michele	francesco	5563392	Sep 3 11:42	documento_1.pdf
-rwxrwxrwx	1	giovanni	isabella	741583	Sep 4 14:53	documento_2.pdf
-rwxrwxrwx	1	francesco	admin	631277	Sep 4 14:52	documento_3.pdf

2. Esecuzione Script (raggruppamento per gruppo di utenti, vale la stessa cosa per il raggruppamento per nome utente):

```
./findWithGrep.pl ./GennaioTest -g doc
```

3. Azioni effettuate internamente dallo script

Lo script prenderà in considerazione i seguenti file presenti nella cartella GennaioTest (e nelle sue sottocartelle):

- test - Copia.docx la cui size è 13114 e il cui gruppo proprietario è admin
- test.docx la cui size è 12223 e il cui gruppo proprietario è francesco
- documento_1.pdf la cui size è 5563392 e il cui gruppo proprietario è francesco
- documento_2.pdf la cui size è 741583 e il cui gruppo proprietario è isabella
- documento_3.pdf la cui size è 631277 e il cui gruppo proprietario è admin

Successivamente, lo script farà eseguire la somma delle size raggruppando per gruppo e stamperà ordinatamente sul file `results.out` il risultato calcolato

4.1. Output scritto su file `result.out`:

francesco	5575615
isabella	741583
admin	644391

Spazio totale occupato:	6961589

MATERIALE PER LA PROVA SULLA PROGRAMMAZIONE MULTI-THREADED

Il codice fornito implementa una struttura dati thread-safe chiamata `DischiConcentrici`. Tale struttura dati è composta di due array circolari (detti “dischi”), che chiameremo `In` e `Out`, ciascuno di N valori interi. Ogni elemento di `In` ha un suo omologo in `Out`. Le coppie di elementi omologhi sono inizialmente impostate considerando gli elementi di `In` e `Out` di pari indice (e cioè, per un dato valore i , l’omologo di `In[i]` è proprio `Out[i]`), ma è possibile che questa configurazione venga variata ruotando virtualmente `Out` rispetto ad `In`. Ad esempio, ruotando `Out` di 3 posizioni in avanti, avremmo che `In[i]` diventa omologo rispetto all’elemento `Out[(i+3)%N]`.

Nel seguito, dato i , inteso come indice da applicare sul vettore `In`, chiameremo $om(i)$ l’indice omologo da applicare sul vettore `Out`.

Le operazioni che si devono poter compiere su una istanza di `DischiConcentrici` sono:

`shift(m)`. Sposta il disco `Out` di m posizioni in avanti (o indietro se m è negativo), aggiornando dunque la corrispondenza tra indici omologhi in accordo.

`set(i, v, d)`. Se $d=1$, imposta l’elemento i -esimo di `In` a v . Se $d=0$, imposta l’elemento attualmente omologo dell’indice i in `Out` al valore v . Se a seguito di questa operazione dovesse risultare `In[i] == Out[om(i)]`, bisognerà porre `In[i] = Out[om(i)] = 0`.

Ad esempio, se $om(i) = 3$, l’operazione `set(4, 7, 0)` dovrà impostare `Out[4+3] = 7`.

`get(i, d)`. Se $d=1$, restituisce il valore di `In[i]`. Se $d=0$, restituisce il valore di `Out[om(i)]`. Tuttavia se il valore che si sta per restituire dovesse risultare pari a 0, ci si deve porre in attesa bloccante fino a che il valore corrispondente non diventa diverso da 0, restituendo infine il nuovo valore.

```

1  from threading import Thread, Lock, RLock, Condition
2  from random import random, randint
3  from time import sleep
4
5  debug = True
6
7  #
8  # Stampa sincronizzata
9  #
10 plock = Lock()
11 def sprint(s):
12     with plock:
13         print(s)
14 #
15 # Stampa solo in debug mode
16 #
17 def dprint(s):
18     with plock:
19         if debug:
20             print(s)
21
22
23 class DischiConcentrici():
24
25     def __init__(self, size : int):
26         #
27         # Lock interno per la gestione della struttura dati
28         #
29         self.lock = RLock()
30         self.waitCondition = Condition(self.lock)
31         #
32         # Tiene traccia della corrispondenza In e Out
33         #
34         self.shiftAttuale = 0
35         #
36         # I due array interni
37         #

```

```

38         self.In = [1] * size
39         self.Out = [1] * size
40         self.size = size
41
42     #
43     # Data in input una posizione in In, restituisce la posizione omologa in Out
44     #
45     def _om(self, i : int):
46         with self.lock:
47             dprint("I:%d" % i)
48             return (i + self.shiftAttuale) % self.size
49
50     #
51     # Esempio, con len(In) = len(Out) = 10:
52     #   shiftAttuale = 0, dunque _om(i) = i
53     #
54     # Corrispondenza tra In e Out:
55     #
56     #   In: 0 1 2 3 4 5 6 7 8 9
57     #   Out: 0 1 2 3 4 5 6 7 8 9
58     #
59     # Dopo aver invocato shift(2) ==> shiftAttuale = 2, _om(i) = (i+2) % 10
60     #
61     # Corrispondenza tra In e Out:
62     #
63     #   In: 0 1 2 3 4 5 6 7 8 9
64     #   Out: 2 3 4 5 6 7 8 9 0 1
65     #
66
67     def shift(self, m : int):
68         with self.lock:
69             self.shiftAttuale += m
70
71     def set(self, i : int, v : int, d : int):
72         with self.lock:
73             if d == 0:
74                 self.Out[self._om(i)] = v

```



```

75         else:
76             self.In[i] = v
77         if self.In[i] == self.Out[self._om(i)]:
78             self.In[i] = 0
79             self.Out[self._om(i)] = 0
80         elif v != 0:
81             self.waitCondition.notifyAll()
82
83     def get(self, i : int, d : int):
84         with self.lock:
85             while (d == 0 and self.Out[self._om(i)] == 0) or (d == 1 and self.In[i] == 0):
86                 dprint("In attesa")
87                 self.waitCondition.wait()
88                 dprint("Risvegliato")
89             if d == 0:
90                 return self.Out[self._om(i)]
91             elif d == 1:
92                 return self.In[i]
93
94     class ManipolatoreDischi(Thread):
95
96         def __init__(self, d : DischiConcentrici):
97             super().__init__()
98             self.iterazioni = 1000
99             self.d = d
100
101     def run(self):
102         while(self.iterazioni > 0):
103             self.iterazioni -= 1
104             r = random()
105             i = randint(0, self.d.size-1)
106             v = randint(0,10)
107             d = randint(0,1)
108             if r < 0.5:
109                 sprint("get(%d,%d) = %d" % (i,d,self.d.get(i,d)))
110             else:
111                 sprint("set(%d,%d,%d)" % (i,v,d))

```

```
112         self.d.set(i,v,d)
113     if r < 0.1:
114         self.d.shift(i)
115         sleep(random()/100)
116
117
118 D = DischiConcentrici(10)
119 for i in range(0,100):
120     ManipolatoreDischi(D).start()
```

PROGRAMMAZIONE IN PERL - MATERIALE PRELIMINARE

All'interno dell'esercizio verrà utilizzato il comando shell `ls` studiato a lezione, normalmente utilizzato per listare tutti i file presenti all'interno di una directory. Un possibile output del comando `ls`, seguito da alcuni dei suoi parametri, è il seguente:

```
.:
total 40
drwxrwxrwx 1 francesco francesco      4096 Jan 16 10:49 .
drwxrwxrwx 1 francesco francesco      4096 Jan 16 10:49 ..
drwxrwxrwx 1 root      root          4096 Sep 12 17:26 'Sotto cartella'
-rwxrwxrwx 1 francesco francesco     5056 Sep 12 15:21 esempio.txt
-rwxrwxrwx 1 francesco francesco       0 Sep 12 14:03 main.pdf
-rwxrwxrwx 1 root      root           0 Sep 12 14:03 output.a
-rwxrwxrwx 1 francesco francesco     1709 Sep 12 17:31 soluzioneGennaio.pl
-rwxrwxrwx 1 francesco francesco    13114 Sep 12 14:24 'test - Copia.docx'
-rwxrwxrwx 1 francesco francesco    12223 Sep 12 14:24 test.docx
-rwxrwxrwx 1 francesco francesco       0 Sep 12 14:15 txt.ciao

'./Sotto cartella':
total 6784
drwxrwxrwx 1 root root      4096 Sep 12 17:26 .
drwxrwxrwx 1 francesco francesco 4096 Jan 16 10:49 ..
-rwxrwxrwx 1 francesco francesco 5563392 Sep  3 11:42 documento_1.pdf
-rwxrwxrwx 1 francesco francesco 741583 Sep  4 14:53 documento_2.pdf
-rwxrwxrwx 1 francesco francesco 631277 Sep  4 14:52 documento_3.pdf
```