Progettino Lab2 Corso A

Valutazione in itinere

Processi, thread, socket, segnali

Si consideri una cartella con al suo interno files ed altre cartelle contenenti, tra gli altri, file con estensione ".dat". Tali file sono file testuali che contengono numeri interi o decimali, uno per riga, eventualmente separati da righe vuote. I numeri possono essere preceduti o seguiti da spazi vuoti e/o caratteri di tabulazione. Per ogni file ".dat" vogliamo calcolare la media aritmetica e la deviazione standard dei numeri in esso contenuti.

Supponendo che la directory corrente '.' contenga i seguenti file (-) e directory (d):

- ./prova1.dat
- ./prova2.dat
- ./script.sh
d ./provadir
- ./provadir/provadir.dat
d ./provadir/provadir1
- ./provadir/provadir1/provadir1.dat
d ./provadir/provadir2
- ./provadir/provadir2/provadir2-1.dat
- ./provadir/provadir2/provadir2-2.dat
d ./provadir/provadir2/provadir3
- ./provadir/provadir2/provadir3/provadir3-1.dat
- ./provadir/provadir2/provadir3/provadir3-2.dat
- ./provadir/provadir2/provadir3/provadir3-3.dat

Riquadro1: cat ./provadir/provadir.dat

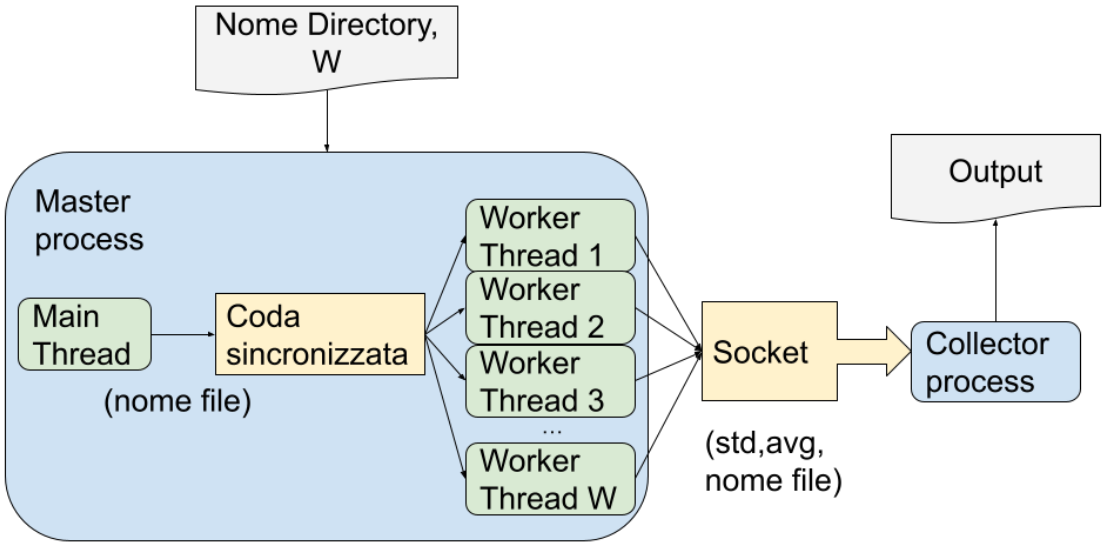
```
123.4

-32.8
      77.54
1234.7869
          -21.123
33.1234
11
2.8
```

Vogliamo ottenere un output tabellare come il seguente:

n	avg	std	file
3	3.00	.81	./prova1.dat
7	30.71	29.81	./provadir/provadir2/provadir3/provadir3-3.dat
7	30.71	29.81	./provadir/provadir2/provadir2-1.dat
13	4.61	2.55	./provadir/provadir1/provadir1.dat
1	1.00	0	./prova2.dat
8	178.59	402.10	./provadir/provadir.dat
5	32.00	3.08	./provadir/provadir2/provadir2-2.dat
13	18.92	10.57	./provadir/provadir2/provadir3/provadir3-1.dat
6	50.83	67.58	./provadir/provadir2/provadir3/provadir3-2.dat

Dove 'n' è il totale dei numeri validi nel file considerati per il calcolo, 'avg' è la media aritmetica, 'std' è la deviazione standard, e 'file' è il nome del file considerato. Nel *Riquadro1* è mostrato il contenuto del file *provadir.dat* contenuto nella directory *provadir*, avente *n=8* righe valide (su un totale di 11), una media aritmetica pari a *avg=178.59* ed una deviazione standard pari a *std=402.10*.

Si chiede di realizzare **programma scritto in C** che utilizzi *due processi* e che produca l'output descritto sopra. La struttura del programma deve seguire lo schema della figura:



In particolare, il processo ***Master*** esegue i calcoli per ogni file, usando **W** thread *Worker*, e manda i risultati ad un processo ***Collector*****,** tramite socket. Il processo Collector può essere creato dal processo Master, o può essere un programma da eseguire separatamente. Il *Master* prende come parametro dalla linea di comando il nome della directory iniziale, e deve percorrere ricorsivamente tutti i sottodirectory, ottenendo i nomi dei file .dat da leggere. I nomi vengono comunicati dal thread *main* ai thread *worker* tramite un buffer illimitato (potete usare l'implementazione di unboundedequeue.h). Un thread *Worker*, dopo aver letto il nome di un file dal buffer condiviso, lo apre e calcola i valori necessari, poi comunica al processo *Collector* il risultato (avg, std e nome file), tramite un socket. Potete utilizzare sia i socket AF_UNIX che AF_INET. Nella comunicazione con socket, il Collector fa da server, e i worker sono dei client, che, per ogni file processato mandano un messaggio con il risultato per il file. Si può utilizzare una solo connessione, condivisa dai vari worker thread, o una connessione per worker thread. La connessione deve essere permanente (una connessione con messaggi multipli). Il processo *Collector* legge i dati dal socket e stampa la tabella come nell'esempio. I dati vengono stampati in ordine di arrivo nel socket (non vanno riordinati per directory). Quando i worker finiscono il loro lavoro il Collector legge gli ultimi risultati poi si spegne.

Si può considerare che il nome del file abbia al massimo 255 caratteri. Il parametro W viene anche lui passati al Master alla linea di comando, insieme al nome della directory.

Lo studente dovrà consegnare un file in formato ".tar.gz" o ".zip" contenente:

- Il codice sorgente del programma C (e gli eventuali include file .h)
- Un Makefile per compilare il programma C e per eseguire i test
- Una o più directory contenenti i file ".dat" per eseguire i test e dimostrare la funzionalità dello script e del programma C. **Limitare la dimensione di tali directory a non più di 2MB e massimo 200 file**.

Il comando *make* se lanciato senza argomenti, deve compilare il programma C (senza warning né errori). **Il codice C che non compila sulla macchina del corso (laboratorio2.di.unipi.it) non verrà valutato**. Il Makefile deve avere almeno tre target fittizi (PHONY): *test1* che manda in esecuzione *il programma C* su una delle directory di test fornite e con W=1, *test2* che manda in esecuzione *il programma C* su una delle directory di test fornite W=5 e *test3* che manda in esecuzione *valgrind sul programma C* su una delle directory di test fornite e con W=5. Il codice deve eseguire correttamente sulla macchina del corso. **Il codice che non esegue sulla macchina di laboratorio non verrà valutato.**