## Elaborato 2: Processi e IPC.

**Consegna entro**: 25 Giugno 2017, ore: 23:59:59

## Testo dell'elaborato

Si vuole realizzare un programma in C che utilizzi le system call (IPC), ove possibile, per implementare alcune operazioni su Matrici intere in maniera concorrente, distribuite su un certo numero di processi. Il progetto deve essere commentato in formato Doxygen, e corredato da uno script configurazione per tale tool e da uno script Makefile per la sua compilazione. Inoltre si devono allegare al progetto anche eventuali file di supporto.

Nel dettaglio: il programma dovrà calcolare la matrice quadrata C di ordine N risultante dal prodotto tra due matrici quadrate A e B di ordine N. Successivamente deve calcolare la somma degli elementi della matrice C. Alla fine dell'esecuzione la matrice C, dovrà essere memorizzata in un file. La somma degli elementi di C dovrà essere stampata a video. Queste operazioni dovranno essere svolte mediante la cooperazione tra P processi concorrenti.

Il programma dovrà prendere 5 parametri:

1. Il path del file contenente la matrice A.
2. Il path del file contenente la matrice B.
3. Il path del file in cui salvare la matrice C.
4. L'Ordine N (dimensione) delle due matrici.
5. Il numero P di processi concorrenti da utilizzare.

*E.g.,*

```
./calcola matA Matb MATc 6 4
```

La matrice A è contenuta nel file matA, la matrice B è contenuta nel file Matb, le matrici sono di ordine 6 (matrici quadrate 6x6) e devono essere allocati 4 processi. La matrice C verrà salvata in MATc.

Il processo principale (PADRE) dovrà:

1. Leggere le matrici A e B dai file.
2. Memorizzare le due matrici in due aree di Memoria Condivisa.
3. Creare una terza area di Memoria Condivisa che conterrà la matrice C risultante della moltiplicazione tra matrici.
4. Creare una quarta area di Memoria Condivisa che conterrà il risultato della somma degli elementi di C: questa area di Memoria Condivisa dovrà contenere un unico intero.
5. Creare i P processi che coopereranno concorrentemente per svolgere la computazione, e creare ed inizializzare tutte le strutture di supporto quali: semafori, pipe, fifo, etc.
6. Coordinare i P processi figli per "distribuire" le operazioni da compiere.

    a. Operazioni per calcolare la matrice C, prodotto di A e B.
    b. Operazioni per calcolare la somma degli elementi di C.
7. Scrive su file la matrice C.
8. Stampa a video il valore della somma degli elementi di C.

Ogni processo figlio eseguirà una routine di questo tipo:

1. Attesa della ricezione di una richiesta di operazione da parte del padre
2. Ricezione dei dati e dell'operazione da eseguire, da parte del processo padre.
3. Esecuzione dell'operazione.
4. Scrittura del risultato nella corretta locazione di memoria condivisa.
5. Invio di un messaggio al padre per comunicare il completamento dell'operazione.
6. Attesa del comando successivo.

Ogni figlio ha due modalità di esecuzione:

Modalità "Moltiplicazione":

Il padre richiede l'operazione di Moltiplicazione e manda al figlio due indici: i, j. Il figlio deve calcolare il valore dell'elemento i,j della matrice C. Il figlio dunque:

- deve leggere la riga i-esima della matrice A;
- deve leggere la colonna j-esima della matrice B;
- effettuare le operazioni aritmetiche necessarie;
- scrivere il risulato nella cella i,j della matrice C;
- comunicare al padre se l'operazione è andata a buon fine.

Modalità "Somma":

Il padre richiede l'operazione di Somma e comunica al figlio un indice k. Il figlio deve calcolare la somma della riga k-esima della matrice C, e sommarla al valore parziale della somma degli elementi di C. Il figlio dunque:

- legge la riga k-esima di C;
- somma gli elementi letti;
- legge il valore parziale della somma degli elementi di C;
- somma gli elementi della k-esima riga di C e la somma parziale degli elementi di C;
- aggiorna il valore parziale degli elementi di C.

Il padre invierà i comandi ai figli mediante delle pipe.

Il padre riceverà le comunicazioni da parte di tutti figli mediate una singola coda di messaggi: tutti i processi invieranno le informazioni utilizzando questa coda di messaggi, il processo padre dovrà filtrarli per capire, per ogni messaggio, quale figlio si è liberato.

**Il padre non potrà compiere NESSUNA operazione aritmetica sulle matrici. Le uniche operazioni permesse sono quelle implementate dalle modalità "Moltiplicazione" e "Somma" dei processi figli. Il padre dovrà limitarsi a coordinare/schedulare le operazioni dei figli.**

Note:

- Il formato dei file di input contenenti le matrici è libero.
- I "formati" di dati utilizzati nella comunicazione tra padre e figli (e viceversa), ossia il formato delle stringe passanti per la pipe e dei messaggi, sono libera scelta dello studente.
- La corretta sincronizzazione tra letture e scritture su aree di memoria condivisa deve essere garantita, dove necessario, utilizzando semafori.
- Alla fine dell'esecuzione, il padre libera tutte le risorse.
- Si aggiungano ai processi delle stampe a video per poter seguirne l'esecuzione.
- Per ogni chiamata ad una system call, si deve controllare che tale funzione abbia successo.
- I dettagli su come utilizzare la memoria condivisa o quanta allocarne, o quanti semafori, sono lasciati liberi.
- Quando i processi devono attendere, non devono fare attese attive: si devono bloccare.
- Ove possibile, si devono usare le system call al posto delle equivalenti chiamate a funzioni di libreria.
- **Tutte le stampe a video, le letture e le scritture su file devono avvenire tramite system call** (quindi ad esempio non si possono utilizzare *printf*, *fprintf*, *scanf*, *fscanf*, *perror*).
- Il programma deve gestire correttamente a prescindere dal numero N di processi (1, 2, uguale, minore o maggiore dell'ordine delle matrici).
- Il programma deve essere in grado di gestire matrici quadrate di qualsiasi Ordine.

**FACOLTATIVO: Implementazione mediante thread**

Si implementi una SECONDA VERSIONE del programma che si avvalga di thread anziché processi per risolvere il medesimo problema.

In questo caso P non viene passato da linea di comando: il programma è libero di utilizzare liberamente le thread, oltre che di decidere dinamicamente il numero di thread da utilizzare.

Tutte le operazioni aritmetiche che utilizzano le matrici dovranno essere svolte da thread.

**Nota:**

Le thread saranno materia di esame orale. Dunque, per chi svolgerà la parte facoltativa, la parte di esame orale riguardante le thread si baserà principalmente sulla presentazione di questa.

A chi non svolgerà la parte facoltativa verrà chiesto di rispondere ad alcune domande in merito alle thread e di svolgere durante l'interrogazione alcuni esercizi che utilizzano le thread.

## FAQ

1. *E' possibile inserire il codice su file separati?*

   Questa decisione e' lasciata allo studente, che puo' scegliere il modo piu' opportuno. L'importante e' che il codice sia ben strutturato e leggibile.

2. *Si possono usare funzioni quali la printf, fscanf, etc.?*

   **No**. L'elaborato richiede di usare le system call ove possibile, e tali funzioni sono rimpiazzabili tramite le system call *open*, *write*, etc. In generale, si cerchi di utilizzare il piu' possibile le system call. Invece, si possono usare funzioni quali la *sprintf*, perche' non ha una system call equivalente.

3. *Alcune cose non sono ben specificate nell'elaborato. Cosa faccio?*

   Alcune cose non sono specificate apposta per lasciare liberta' agli studenti di implementarle come preferiscono. In caso di dubbi, si possono comunque contattare i docenti per eventuali chiarimenti.

4. *Quale dei due stili presentati a lezione bisogna usare per il Makefile?*

   Lo studente può usare quello che preferisce.

N.B.: Tutto quanto non esplicitato in questo documento può essere implementato liberamente.