Sistemi Operativi – a.a. 2023/2024
prova di laboratorio
– 31 gennaio 2025 –

Creare un programma **`calc-verifier-4.c`** in linguaggio C che accetti invocazioni sulla riga di comando del tipo:

`calc-verifier-4 <first-operands> <second-operands> <operations>`

Il programma dovrà essere in grado di verificare la correttezza di somma di operazioni minori descritte a cavallo di 3 file testuali. Ogni operazione minore è composta da due operandi e da una tra le seguenti operazioni: somma, sottrazione o moltiplicazione. Esempio: (4 x 3) + (7 - 4) + ... + (10 + 7) + (3 + 15).

I tre file forniti sulla riga di comando contengono, in ogni riga, rispettivamente:
- il primo operando;
- il secondo operando;
- l'operazione da applicare (+, -, x).

Il risultato atteso della sommatoria finale è fornito come ultima riga nel file delle operazioni.

Alcuni esempi sono forniti a corredo: esempio A (`A.op1`, `A.op2`, `A.ops`) e esempio B (`B.op1`, `B.op2`, `B.ops`).

Al suo avvio il programma creerà 4 thread ausiliari oltre a quello principale (main):
- un thread OP1 che si occuperà di leggere i primi operandi delle operazioni minori dal primo file fornito;
- un thread OP2 che si occuperà di leggere i secondi operandi delle operazioni minori dal secondo file fornito;
- un thread OPS che si occuperà di leggere il tipo di operazione da applicazione nelle varie operazioni minori dal terzo file fornito e il risultato finale atteso;
- un thread CALC che si occuperà di eseguire le operazioni minori una volta che gli altri thread avranno, di volta in volta, fornito le specifiche (operandi e tipo di operazione).

I thread si coordineranno tramite esattamente due semafori POSIX: non si devono usare altri strumenti di coordinamento. L'unica struttura dati condivisa dai thread conterrà (oltre ai due semafori):
- `operando_1`: intero di tipo `long long`;
- `operando_2`: intero di tipo `long long`;
- `operazione`: identificativo dell'operazione da calcolare;
- `risultato (dell'operazione minore corrente)`: intero di tipo `long long`;
- eventuali flag per segnalare il termine delle operazioni.

I 3 thread OP1, OP2 e OPS, per ogni elemento letto dal rispettivo file, depositeranno lo stesso nella struttura condivisa e ne segnaleranno la disponibilità al thread CALC che si occuperà di calcolare l'operazione minore attuale depositandone il risultato nella struttura condivisa. Questo per ogni operazione minore disponibile all'interno dei file: la sommatoria dei vari risultati intermedi sarà gestita dal thread OPS con una sua variabile locale; questo si occuperà anche di verificare la correttezza del valore finale.

Il programma dovrà funzionare con triplette di file che descrivano una sommatoria con un qualunque numero di operazioni minori. Tutti i thread dovranno terminare spontaneamente alla fine dei lavori e non si dovranno usare strutture dati con visibilità

globale. E' necessario rispettare fedelmente la struttura dell'output riportato nell'esempio a seguire.

Tempo: 1 ora e 35 minuti

L'output atteso sui file di esempio è il seguente:

```
$ ./calc-verifier-4 A.op1 A.op2 A.ops

[MAIN] creo i thread ausiliari
[OP1] leggo gli operandi dal file 'A.op1'
[OP2] leggo gli operandi dal file 'A.op2'
[OP1] primo operando n.1: 30
[OPS] leggo le operazioni e il risultato atteso dal file 'A.ops'
[OPS] operazione n.1: +
[OP2] secondo operando n.1: 3
[CALC] operazione minore n.1: 30 + 3 = 33
[OP1] primo operando n.2: 4
[OPS] sommatoria dei risultati parziali dopo 1 operazione/i: 33
[OPS] operazione n.2: x
[OP2] secondo operando n.2: 17
[CALC] operazione minore n.2: 4 x 17 = 68
[OPS] sommatoria dei risultati parziali dopo 2 operazione/i: 101

  ...

[OPS] operazione n.100: -
[OP1] primo operando n.100: 2
[OP1] termino
[OP2] secondo operando n.100: 14
[OP2] termino
[CALC] operazione minore n.100: 2 - 14 = -12
[CALC] termino
[OPS] sommatoria dei risultati parziali dopo 100 operazione/i: 2119
[OPS] risultato finale atteso: 2119 (corretto)
[OPS] termino
[MAIN] termino il processo
```