Università degli Studi di Napoli Federico II
Corso di Laurea in Ingegneria Informatica
Esame di Sistemi Operativi
Proff. Cotroneo, Natella

Prova pratica del 12/02/2015

Durata della prova: 150 minuti

Cognome .... Nome .... Matr. ....

°°°

*Lo studente legga attentamente il testo e produca il programma, il makefile, ed i casi di test necessari per dimostrarne il funzionamento. La mancata compilazione dell'elaborato, la compilazione con errori o l'esecuzione errata del programma daranno luogo alla valutazione come **prova non superata**. Ricordarsi di indicare Nome, Cognome e matricola su questo stesso foglio, che dovrà essere in ogni caso consegnato alla Commissione. Al termine della prova lo studente dovrà fare verificare il funzionamento del programma ad un membro della Commissione.*

°°°

Testo della prova

Si realizzi in linguaggio C/C++ un'applicazione **multithread** che svolga delle elaborazioni aritmetiche in concorrenza, utilizzando il costrutto **Monitor** ed un vettore di buffer per lo scambio degli operandi. Il vettore dovrà essere definito nel seguente modo:

```
typedef struct {
      int operandi[4];        // il buffer contiene un array di operandi (da 2 a 4)
      int totale_operandi;    // il numero di operandi presenti nell'array
} buffer;
typedef struct {
      buffer elaborazioni[5];
      // ... inserire qui le variabili per la sincronizzazione ...
} MonitorElaborazioni;
```

I thread di tipo **Richiedenti** fungeranno da produttori, e inseriranno nel vettore una quantità casuale di operandi (mediante la primitiva `rand()`): 2, 3, oppure 4 operandi, da collocare a partire dalla posizione 0 dell'array "*operandi*". I valori degli operandi sono anche essi casuali tra 0 e 10. Inoltre, i thread Richiedenti dovranno scrivere nel campo "*totale_operandi*" il numero di operandi che sono stati inseriti nel buffer. I thread di tipo **Elaboratori** fungeranno da consumatori, prelevando dal vettore. Gli Elaboratori dovranno calcolare la somma degli operandi nel buffer. Per simulare l'elaborazione, gli Elaboratori dovranno porsi in attesa con `sleep()` per un numero di secondi pari al numero di operandi (ad esempio, 2 secondi nel caso di 2 operandi), e stampare a video sia gli operandi sia il risultato della somma.

I thread dovranno sincronizzarsi utilizzando la soluzione del **problema dei produttori/consumatori con vettore di stato**. È richiesta la seguente variazione dello schema: i thread Elaboratori, al momento della scelta dell'elemento da consumare, dovranno selezionare il buffer con il minor numero di operandi, effettuando una ricerca sull'intero vettore di buffer. Ad esempio, se il vettore contiene 2 buffer occupati, uno con 2 operandi ed uno con 4 operandi, il thread dovrà consumare dal buffer con 2 operandi.

Per esercitare il programma, andranno creati 4 thread Richiedenti (che faranno 3 produzioni ciascuno) e 2 thread Elaboratori (che faranno 6 consumazioni ciascuno).