Università di Napoli Federico II
Corso di Laurea in Ingegneria Informatica e Elettronica
Esame di Sistemi Operativi
Proff. De Carlini, Cotroneo, Cinque

Prova pratica del 23/06/2010
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** per la simulazione di un servizio per gli aggiornamenti in tempo reale dei risultati di una partita di calcio. L'applicazione è costituita da 12 thread, di cui due thread, TA e TB, aggiornano i risultati per le due squadre, denominate "A" e "B", e gli altri 10 thread rappresentano ipotetici utenti del servizio.

I thread condividono i risultati nella seguente struttura dati **partita**:

```
typedef struct {
      unsigned int goal_A;
      unsigned int goal_B;
} partita;
```

I thread TA e TB aggiornano la struttura 6 volte ogni 15 secondi (la partita simulata dura complessivamente 90 secondi), ognuno per la propria squadra. L'aggiornamento consiste nell'incrementare in maniera casuale la variabile `goal_A` (o `goal_B`) di un'unità, estraendo a caso un intero compreso tra 0 e 1 con la funzione `rand()`[1]. L'aggiornamento deve avvenire in mutua esclusione tra tutti i thread.

Ognuno dei 10 thread utente effettua un'operazione di lettura ogni 5 secondi dalla struttura "partita", stampando a video le informazioni lette. I thread utente terminano dopo aver effettuato 18 letture. Due o più thread utente possono leggere contemporaneamente dalla struttura, se non occupata da TA o TB.

La struttura "partita" deve essere visibile da tutti i thread, e l'accesso a tale struttura da parte dei thread deve essere disciplinato attraverso gli strumenti di sincronizzazione offerti dalla libreria **PThreads**.

I thread TA, TB e i 10 thread utente sono generati dal programma principale attraverso le primitiva pthread_create(). Una volta generati i thread (come joinable), il programma principale ne attende la terminazione, stampa a video il risultato finale e la squadra vincente, e termina a sua volta.

---

[1] La generazione casuale può essere implementata con la funzione rand() di stdlib.h; ad esempio: int goal = rand() % 2
rand() richiede che venga generato un seme dei numeri casuali attraverso la funzione srand(time(NULL)).