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

Prova pratica del 29/5/2013
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 processo servente **multithread** basato su **code di messaggi UNIX**. Il processo servente, denominato **Server**, riceve richieste di elaborazione provenienti da un gruppo di 3 processi denominati **Client**. Ogni client invia 5 richieste di elaborazione. Ogni messaggio inviato dovrà contenere il PID del Client e due valori interi, selezionati casualmente tra 0 e 10 con la funzione `rand()`. I processi Client devono inoltre attendere e stampare a video il messaggio di risposta del Server proveniente da un'apposita coda dei messaggi di risposta, prima di mandare la richiesta successiva. Il codice dei Client e del Server deve risiedere in **due eseguibili distinti**, che sono invocati da un **programma principale** attraverso una delle varianti della primitiva `exec`. Quando tutti i Client terminano, il programma principale attende 3 secondi e poi invia un messaggio speciale al Server, contenente la coppia di valori {-1, -1}, che causa la terminazione del Server come illustrato di seguito.

Il Server istanzia 2 tipi di thread, rispettivamente un **Manager** e 2 **Worker**. Il Manager verifica periodicamente la disponibilità di un messaggio **senza bloccarsi**, attendendo con una `sleep` di un secondo tra un controllo e l'altro. Quando riceve un messaggio, lo pone in un array di buffer condiviso con i thread Worker. Il messaggio viene prelevato da un thread Worker, che calcola il prodotto della coppia di valori, ed ha il compito di inviare al Client (identificato dal PID) un messaggio contenente il prodotto risultante, tramite la coda di messaggi di risposta. Quando il Manager riceve il messaggio con la coppia di valori {-1,-1}, forza la terminazione dei Worker con `pthread_cancel` e termina a sua volta. Il processo Server attende la terminazione del thread Manager per poi terminare.