Foglio Esercizi 1
Consegna su Aulaweb entro Lunedi 19 marzo 2018

Consegnate codice con commenti agli esperimenti o soluzione proposta (foto/testo/pdf)

Esercizio 0 Concurrency at work
Usando la libreria pthread scrivere un programma C con N>1 thread,
Ogni thread deve stampare la sequenza ptid_1 … ptid_5 con ptid=identificatore del thread.
Analizzare i possibili risultati del programma inserendo se necessario delle chiamate a sleep per simulare ritardi tra due stampe successive nello stesso thread.

Esercizio 1 Memory Fence at work
Considerate il programma C multithreaded visto nella prima lezione (https://dibris.aulaweb.unige.it/mod/resource/view.php?id=35445).
Provate ad eseguire il programma sulla vostra macchina o in laboratorio.
Utilizzare le memory fence disponibili come direttive ASM del C per eliminare l'eventuale anomalia dovuta a delay nelle scritture in memoria (vedi https://bartoszmilewski.com/2008/11/05/who-ordered-memory-fences-on-an-x86/).

Esercizio 2 Asynchronous Functions with Pthreads
Consideriamo una funzione chiamata "asynch" con le seguenti caratteristiche:
- quando chiamata dal main viene eseguita su un thread separato.
- l'esecuzione effettua un task costoso in termini di tempo (es simulato anche solo con una "sleep" di alcuni secondi)
- la chiamata restituisce prima della terminazione un valore (es. un intero) al chiamante.

Consideriamo ora una funzione "get" con le seguenti caratteristiche:
- permette al main di poter recuperare il risultato di una determinata chiamata di asych in un qualsiasi passo della sua esecuzione.
- "get" deve essere bloccante per il main quando il risultato relativo alla chiamata di asynch non è ancora disponibile.

Es.
main invoca async (eseguita in maniera asincrona su un thread separato)
main effettua una serie di elaborazioni qualsiasi (es. sleep di K secondi)
main invoca get e si blocca fino a che il risultato è pronto
main stampa il risultato

Scrivere un programma C con pthread, lock e variabili condition per implementare "asynch" e "get" stampando alla fine il valore del risultato dal main.

Esercizio 3 Monitor with priority condition variables
Scrivere un'implementazione in pseudo codice delle primitive dei monitor (down, up, wait, signal) assumendo che la semantica dia effettivamente priorità maggiore ai thread in attesa su una condizione rispetto a quelli in attesa di entrare nel monitor.