Esercizio 1 - «forker»

- un processo lancia M thread, ciascun thread effettua N fork (il processo "forkato" dorme per un secondo e poi termina).
- fork_counter è un contatore visibile soltanto nel processo principale
- Per ogni fork, salva il pid in un array (accesso concorrente) e incrementa il contatore fork_counter. [gestire il resize dell'array all'occorrenza].

- Ogni volta che un processo figlio termina, il suo pid viene tirato via dall'array (assegnare -1 all'elemento dell'array corrispondente) e fork_counter viene decrementato.
- Quando fork_counter == 0, il processo termina.
- Il thread principale del processo principale, ogni secondo, mostra il valore corrente di fork_counter.

Esercizio 1.1 – variante di «forker»

- Come esercizio 1 ma con le seguenti aggiunte:
- ogni processo figlio restituisce un valore random (valore compreso tra 0 e 9).
- Il processo padre conta i risultati di ciascun tipo e alla fine quando fork_counter == 0, scrive un istogramma dei risultati del tipo:
- 0: n volte, 1: m volte, 2: .....

Esempio da sviluppare in classe

- processo padre, processo figlio: canale di comunicazione bidirezionale (es. 2 pipe, socket, …) per implementare messaggi ping/pong tra i due processi.

Esercizio 2 – «instradatore»

- un processo lancia 10 fork, con ciascun processo figlio il processo padre mantiene una coppia di pipe (oppure: socket? signals?) per avere un canale bidirezionale di comunicazione con ciascun figlio.
- Ad ogni processo figlio viene assegnato un identificativo interno da 0 a 9
- protocollo di comunicazione, ogni messaggio ha questo formato: il primo byte indica il processo di destinazione del messaggio (0-9), seguono poi 15 bytes che contengono il payload (lunghezza totale: 16 bytes)
- Il processo padre fa "routing": ovvero instrada i pacchetti provenienti da un processo figlio ad un altro.
- Ogni processo figlio, invia 10 messaggi ping ad altrettanti processi (id destinazione determinato in modo random), poi termina.
- Messaggi supportati: ping/pong