Programmazione di sistema
Esercitazione 4

Esercizio 1

Realizzare una funzione

template <typename T>
T runInChild(std::function<T(void)> fun)

che esegua “fun” in processo figlio, aspetti la terminazione e ne restituisca il risultato.
Nota bene:

- il processo figlio può prendere i dati su cui lavorare utilizzando una closure, ma deve utilizzare una pipe per restituire il risultato
- sulle pipe si opera come su un “file” con accesso sequenziale, quindi occorre stare attenti a serializzare i dati in modo corretto; per i tipi primitivi e quelli definiti con struct o array di lunghezza nota si può scrivere il valore binario o una serializzazione custom
- se si vogliono gestire strutture arbitrarie meglio utilizzare qualche schema noto (es. JSON, una possibile libreria: https://www.boost.org/doc/libs/1_76_0/libs/json/doc/html/index.html)
- Suggerimento: il tipo T potrebbe avere due funzioni friend in grado di gestire la serializzazione e la deserializzazione da un buffer di caratteri o leggendo e scrivendo direttamente da un file descriptor

Esercizio 2

Realizzare un sistema che permetta di eseguire in parallelo con N figli dei task in modo da velocizzare l’esecuzione, tenendo conto della priorità dei task.
Il task in questione è calcolare tutti i numeri primi presenti all’interno di un range indicato.
La struttura del programma è la seguente:

- all’avvio il padre legge come argomento quanti figli deve avviare e una cartella dove vengono inseriti i task.
    - i task sono descritti in file con n range composti da tre interi, ogni linea è un task
    - formato della linea: priority start end
- dopo l’avvio ogni figlio rimane in attesa che gli si passi un range composto da due interi
- il padre esegue questo loop (in pseudo codice)
  finché è attivo

- controlla se ci sono nuovi file di task, li legge tutti e li accoda in una struttura opportuna
- se ci sono figli liberi passa loro un task da eseguire in ordine di priorità (numero più basso corrisponde a priorità più alta)
- si mette in attesa di una risposta da un task qualsiasi e salva i risultati in un file di output

La condizione di terminazione è un range con valori 0 0 0

Note:

- i figli gestiscono al massimo un task per volta
- per comunicare con i figli occorre una pipe separata per figlio
- per ricevere i risultati occorre una pipe condivisa fra tutti i figli, dove il padre legge e i figli possono scrivere.
  I sistemi posix garantiscono che se più processi scrivono su un file o una pipe, la write è atomica (cioè non è mischiata con le altre) se vengono scritti meno di PIPE_BUF byte
  (https://unix.stackexchange.com/questions/68146/what-are-guarantees-for-concurrent-writes-into-a-named-pipe)
- gestire anche il caso in cui vi possono essere messaggi di più lunghi di PIPE_BUF
- per la funzione di calcolo dei numeri primi si può fare in mondo naive (provando tutti i divisori minori di radice di N) o cercare funzioni ottimizzare con google
- questo scheduler di task non è ottimale in quanto può rimanere a lungo bloccato nella read dei risultati, in attesa che un figlio abbia finito: se nel frattempo arrivano altri task non li si può schedulare, anche se ci sono figli liberi
- si può considerare che non ci siano conflitti nei file con i task se li si crea in una directory differente e li si sposta in quella dei task dopo averli scritti completamente: la move di un file è atomica, quando viene creata la entry nella cartella tutto il contenuto è disponibile all'istante
- attenzione alla corretta chiusura di padre e figli!

Varianti

- rendere lo scheduler più veloce per i task di priorità più alta (0) riservando un numero di figli per loro
- è possibile evitare di bloccarsi in lettura testando se vi sono dati disponibili utilizzando la primitiva **select**
  https://man7.org/linux/man-pages/man2/select.2.html
  Le select è una funzione a cui si passano dei set di file descriptor e questa ritorna immediatamente se in qualche file è possibile scrivere / leggere senza bloccarsi o entro un timeout se nessun file raggiunge la condizione.
  Modificare il codice fermandosi al massimo 1 centesimo di secondo in attesa di risultati.