RELAZIONE PROGETTO FARM

Vito Indelicato 583577

link github repo: https://github.com/vitoindelicato/SolProject/tree/master

Contenuto dell'archivio:

Nell'archivio consegnato saranno presenti varie cartelle e vari file, la seguente schematizzazione è usata per favorire l'ordine e la leggibilità del progetto.

Saranno presenti 3 cartelle:

- Documents → conterrà la traccia con le specifiche del progetto, e questa relazione.

- lib → conterrà al suo interno coppie di file (source e headers) che conterranno strutture dati e funzioni.

- src → conterrà al suo interno i programmi (e gli eventuali header) che definiscono il comportamento delle principali componenti del progetto (quindi: **master worker** - **worker** - **collector**).

Andremo ora ad esaminare più a fondo il contenuto delle directories (si prendono in esame solo lib e src per ovvi motivi).

lib:

Al suo interno conterrà 4 coppie di source/header.

- enhanced_sc : All'interno di enhanced_sc.c vi sarà un'implementazione di tutte le SYSTEM CALL utilizzate nel progetto, che controlla sempre l'effettivo successo della SC, riportando sempre se si è verificato un errore durante la chiamata.

- tools : tools conterrà al suo interno i metodi che ci servono per interagire con i file; vi saranno infatti funzioni che verificano che un file sia regolare, e anche la funzione che si occupa dell'esplorazione delle directories.

- queue : fornisce l'implementazione della coda di task che comunicherà con i thread worker. (Si spiegherà il funzionamento della coda più nel dettaglio avanti).

- node : questi file definiscono la struttura del nodo che andrà a comporre la linkedlist ed i metodi di cui si servirà il processo Collector per salvare, ordinare e stampare i dati ricevuti dai thread worker.

src:

- farm.c :

  è l'entry point che fa partire tutto il progetto.
  Si occupa di mascherare i segnali e della partenze del thread che andrà a gestirli, del parsing degli argomenti della linea di comando e crea i processi master worker e collector.

- master_worker :

  in breve, il master worker eseguirà le seguenti azioni nel seguente ordine:

  1. Inizializzazione della coda concorrente dei task
  2. Creazione ed avvio dei thread worker
  3. Inserimento dei filename sulla coda
  4. Join dei thread e liberazione memoria allocata

- worker :

  Al suo interno conterrà 3 funzioni che definiscono il comportamento

del worker:

1. worker_function: È la funzione che viene passata alla pthread_create.
   Una volta partito, il thread crea delle sue strutture interne con le quali andrà a lavorare ed interagire con la coda.

2. calculator : È la funzione che si occupa del calcolo del risultato.
   Prende il filename, lo apre, calcola, ritorna il risultato.

3. connect_wrapper : È la routine che permette la connessione al server.

- collector : questo file implementa la funzione collector ed altri metodi di cui si serve appunto il collector per il suo funzionamento. (Spiegherò più nel dettaglio il funzionamento del collector nelle prossime sezioni)

Dettagli implementativi:

In questa sezione andrò a spiegare più in profondità gli aspetti cardine del progetto.

Queue:

Il tipo di coda che ho deciso di implementare è una coda circolare.
Il motivo della scelta è quello di evitare lo shrinking della stessa a seguito di ripetute operazioni di enqueue e dequeue.

La struttura della coda è definita in /lib/queue.h
Tutti i metodi sono sviluppati in /lib/queue.c

La coda contiene:

1. rear - front: interi che gestiscono rispettivamente la coda e la testa della queue

2. size : è l'intero che definisce la lunghezza della coda. viene utilizzato come modulo nelle operazioni di incremento e decremento degli indici.

3. done: è un intero (usato come un booleano) che indica se la coda può ancora ricevere task o meno.

4. char **items : È l'array che contiene tutti i puntatori ai filename che vengono inseriti in coda.

5. pthread_mutex_t q_lock : È la variabile mutex che serve acquisire prima di fare qualsiasi operazione sulla coda.

Le operazioni di enqueue() e di dequeue() vengono eseguite da due componenti diverse:

- l' **enqueue**() è operata dal master worker, ogni volta che effettua il parsing di un filename dalla cmdline oppure nella visita delle directory.

  Per eseguire questa operazione bisogna sempre fare l'acquire della lock.

  Una volta controllato se l'operazione di inserimento è permessa, il filename viene inserito e viene inviato - ai threads in attesa - un segnale che indica che la coda non è più vuota.

- la **dequeue**() invece viene eseguita dal worker che acquisisce la lock, la funzione ritorna il filename, e prima di farlo manda un segnale al master worker che indica che la coda non è più piena.

Worker:

Il generico worker crea - una sola volta - le strutture dati che gli servono ed inizia ad aspettare che la coda venga riempita prima di iniziare a lavorare.

Se il worker riesce ad estrarre un filename dalla coda, rilascia la lock della coda ed inizia a lavorare con quel filename.

Tramite la funzione calculator esegue i calcoli e, una volta arrivato il risultato procede a creare un buffer in formato csv di questo tipo :

result;filename

Una volta creato questo buffer, verrà creata una connessione alla socket definita dal collector, e verrà scritto (tramite funzione writen) sulla stessa.

Il worker si comporta quindi da client.

Dopo la scrittura il worker chiuderà la connessione, eseguirà una sleep - se definita da cmdline - e ripeterà il ciclo.
Il worker ritornerà solo nel momento in cui la coda sarà vuota e non ci sono più task da inserire.

Collector:

La componente collector è quella che farà da master nella connessione con i workers.

Una volta avviata infatti andrà subito a creare una server socket sulla quale leggere tutti i messaggi inviati dai workers.

Una volta letto il messaggio in formato csv, mandato dal worker, il collector procederà ad estrarne i campi result e filename, creando una variabile di tipo _node.

Una volta creata questa variabile si procede all'inserimento - ordinato - in una linked list.

Oltre al buffer descritto prima, al collector arrivano altri due tipi di messaggi:

1. messaggio 'DONE' inviato da master worker, indica che tutti i worker hanno finito il loro lavoro, quindi il server può chiudere la connessione.

2. messaggio 'PRINT' viene inviato direttamente da farm qualora ricevesse il segnale SIGUSR1 ricevuto questo messaggio, il server provvederà a stampare la linkedlist contenente i dati inseriti fino a quel momento, per poi continuare il suo normale funzionamento.