PROGETTO LABORATORIO 2

BRIGLIA MARCO
617320

STRUTTURA PROGETTO

- bibData: Cartella contenente i file.txt che contengono i record dei libri.
- new_bibData: Cartella contenente i file di record dei libri aggiornati a fine sessione.
- libs: Cartella contenente le librerie utilizzate per sviluppare il progetto.
- bibaccess.sh: script bash che calcola quante richieste di LOAN e QUERY sono state fatte durante la sessione.
- bibclient.c: codice del client.
- bibserver.c: codice del server.

STRUTTURE DATI

All'interno del mio progetto ho utilizzato strutture dati semplici e lineari per garantire efficienza e stabilità nel programma.

Qua di seguito elencate:

- Libro_t: struct che contiene tutti i parametri possibili che un libro può avere basandoci su quelli presenti nei file.txt di bibData. Sono tutti puntatori a stringhe fatta eccezione per il campo "anno" che invece è un int.
- richiesta_client_t: struct semplice composto da 2 puntatori a stringhe dove immagazzinare i valori del parsing della richiesta del client.
- msg_client_t: struct che serve alla comunicazione del client/server. Al suo interno troviamo il campo "type" (char) che rappresenta il tipo di messaggio inviato a seconda del protocollo di comunicazione stabilito, "length" (unsigned int)

che contiene la dimensione del messaggio inviato, e per finire "data" (char[]) che contiene il messaggio vero e proprio.
- param_worker_t: struct di argomenti da passare al thread worker fondamentali alla sua esecuzione. "coda" (queue_t), coda condivisa da tutti i thread worker da dove attingono alle richieste dei client per poi gestirle singolarmente; "biblioteca" (linked_list_t), lista dei libri non ripetuti all'interno del file.txt dal quale il server ha letto; lock (pthread_mutex_t), mutex per la sincronizzazione di lettura e scrittura; log (FILE), file di log da aggiornare a fine gestione richiesta.

Librerie

- aux_function.h: contiene MACRO, struct, # include di librerie esterne e tutte le funzioni che mi sono servite alla realizzazione dei client e server. (mi perdoni il disordine).
- linked_list.h: contiene gli struct node_t e linked_list_t per la gestione dei dati della lista, insieme a funzioni base per interagire con essa.
- queue.h: come linked_list_t, contiene gli struct node_queue_t e queue_t, insieme alle funzioni base di una coda.

Di seguito le librerie che contengono funzioni wrapper per il controllo degli errori:

- file.h: contiene funzioni wrapper di fopen, fclose e close.
- mutx.h: contiene funzioni wrapper di init, destroy, lock, unlock e wait.
- sig.h: contiene funzioni wrapper di emptyset e addset.
- socket.h: contiene funzioni wrapper di socket, bind, listen, accept e connect.
- thread.h: contiene funzioni wrapper di create, join, detach e sigmask.

PROBLEMI E DIFFICOLTA'

Considerando la mia totale inesperienza con C prima di interfacciarmi a questo progetto, la uniche difficoltà che ho riscontrato (che non si sono rivelate poi particolarmente problematiche) sono state:

- Nella fase iniziale: varie gestioni dei puntatori e dell'inizializzazione della memoria. Alcune volte mi sono imbattuto in "segmentation fault" che mi hanno fatto perdere qualche mezz'ora a rileggermi il codice da capo, ma una volta appreso il problema non si è più ripresentato nelle fasi più avanzate del progetto.
- Nella fase finale: essendo la prima volta che lavoravo con un progetto client/server, la gestione dei file descriptor dei client attraverso select() mi ha causato qualche grattacapo.
-

README

Per compilare il programma basterà aprire la cartella Biblio sul terminale e lanciare il comando make.

- make: compilerà tutti i file oggetto delle librerie, di bibserver.c e di bibclient.c
- make clean: pulirà al directory da tutti i file creati a tempo di compilazione.
- make test: lancerà la serie di test richiesta composta da 5 server e 40 client, infine eseguirà lo script bash che restituirà la somma delle richieste effettuate a seconda del tipo più quelle per ogni server.
- make kill: un'aggiunta che semplicemente lancia pkill bibserver chiudendo tutti i processi qualora ce ne dovessero essere di aperti.