RELAZIONE PROGETTO LAB 2

Il progetto è diviso in 5 FILE:

1. Server.c -> contiene la funzione del server chiamata nel main
2. Macro.h -> libreria che contiene tutte le strutture, le macro , le librerie esterne , le variabili globali e le funzioni utilizzabili
3. Matrice.c -> contiene il main del paroliere_srv
4. Macro.c -> implementa le funzioni del dizionario
5. Client.c -> contiene il main del paroliere_cl

Il processo server viene lanciato mediante il seguente comando:

./paroliere_srv nome_server porta_server [--matrici data_filename] [--durata durata_in_minuti] [--seed rnd_seed] [--diz dizionario] [--disconnetti tempo in minuti]

ES -> ./paroliere_srv 127.0.0.1 1071 --durata 1 --diz dictionary_ita.txt --disconetti 2 --matrici crea.txt

Il processo client viene avviato con:

./paroliere_cl nome_server porta_server

ES-> ./paroliere_cl 127.0.0.1 1071

Il file Server.c contiene :

- **Strutture Dati Principali**:
    - client_r e client_a: Due array usati per tenere traccia dei client registrati e attivi.
    - ultimo_messaggio: Un array per memorizzare i tempi dell'ultimo messaggio di ogni client.
    - bacheca: Spazio condiviso per i messaggi, creato dinamicamente per gestire messaggi di lunghezza variabile.
    - file_matrix_buffer: Buffer per caricare il contenuto di un file matrice, utile per leggere tutto in memoria e accedervi più velocemente.
- **Algoritmi Principali**:
    - Un ciclo cerca il primo posto libero negli array dei client attivi, garantendo che non si superi il numero massimo di connessioni usando la mappa dei client attivi.
    - Vengono usati thread per gestire diverse funzionalità (es. timer, gestione delle connessioni). (specificati in macro.h)

3. Funzionamento del Server

- **Fasi Principali**:
    1. **Inizializzazione**:
        - Configura segnale SIGINT per una chiusura sicura.
        - Alloca memoria per array e buffer.
    2. **Avvio del Server**:
        - Crea un socket e lo collego a una porta (bind).

        - Si mette in ascolto delle connessioni con listen.
    3. **Gestione dei Client**:
        - Accetta nuove connessioni con accept e assegna un thread a ogni client.
        - Controlla il numero massimo di connessioni attive usando una mappa (client_a).
    4. **Crea file di log**
- **Gestione delle Matrici**:
    - Se viene specificato un file matrice, viene caricato in memoria per essere usato dal server. Al termine del file, la matrice viene creata casualmente da seed di default o specificato.

Il file Macro.h contiene:

1. Strutture Dati

- **TrieNode**: Struttura che rappresenta un nodo in una trie, una struttura ad albero utilizzata per gestire un dizionario di parole.
- **Messaggio**: La struttura Messaggio definisce il formato di un messaggio.
- **Code**: Sono definite due code:
    - **Coda di punteggi**: Gestisce i punteggi dei giocatori.
    - **Coda di utenti**: Gestisce i nomi degli utenti registrati.
- **Argomenti**: La struttura ArgomentiT viene utilizzata per passare parametri al thread collegato al corrispettivo client.

2. Dichiarazioni di Funzioni

- **Gestione del Server e Client**:
    - server(): Funzione principale del server per gestire le connessioni.
    - client(): Funzione del client per connettersi al server.
    - client_close(): Funzione per chiudere una connessione client.
- **Funzioni di Gestione della Matrice**: Gestiscono la creazione e la visualizzazione della matrice.
- **Funzioni di Gestione dei Messaggi**: Funzioni come genera_messaggio() e ricevi_messaggio() per inviare e ricevere messaggi tra client e server.
- **Funzioni di Gestione delle Code**: Come push_p() e pop_p() per aggiungere e rimuovere punteggi dalla coda.
- **Funzioni di Controllo**: Funzioni come check_word() per validare le parole e string_in_array() per verificare la presenza di una stringa in un array.
- **Funzioni dei Thread**: Scorrer, timer_function, server_function...

- **Algoritmo dfs usato per cercare parole nella matrice**

3. Variabili Globali

- **Mutex**: Sono definiti vari mutex per la sincronizzazione tra i thread, come mutex_coda_p, mutex_coda_u, e mutex_client_a.
- **Altre Variabili Globali**: Include variabili per gestire la durata del gioco, la matrice, i client connessi, e la coda dei punteggi.

4. Gestione dei Segnali

- **Signal Handlers**: Funzioni come sigint_handler() e sigusr1_handler() gestiscono i segnali di sistema, per esempio per terminare il server o per gestire timeout.

5. Funzioni di Pulizia

- **Pulizia della Memoria**: Funzioni come freeTrie() e free_coda_punteggi() sono utilizzate per liberare la memoria occupata da strutture dati dinamiche, come la trie e le code.

Il file matrice.c contiene:

1. **Gestione Parametri di Configurazione**:
    - Il programma inizia leggendo i parametri passati tramite la riga di comando usando la funzione getopt_long().
    - Questi parametri includono:
        - **--matrici**: Nome del file della matrice.
        - **--durata**: Durata della partita in minuti.
        - **--seed**: Seed per la generazione di numeri casuali.
        - **--diz**: File contenente il dizionario di parole.
        - **--disconetti**: Tempo di disconnessione in minuti.
2. **Caricamento del Dizionario**:
    - Il dizionario di parole viene letto da un file specificato con l'opzione --diz.
    - Ogni parola viene inserita in una struttura di tipo TrieNode (una trie) tramite la funzione insert().
3. **Generazione della Matrice**:
    - La matrice di gioco viene generata usando un seed casuale per creare una disposizione casuale delle lettere. La matrice viene letta dal file se l'opzione --matrici è specificata.
4. **Funzione Server**:
    - La funzione server() viene chiamata alla fine per avviare il server con i parametri configurati.

Il file macro.c contiene:

1. **Implementazioni funzioni** con relativa spiegazione
2. **Struttura thread**
    a. Thread server_function , thread associato ad ogni nuovo client.
    b. Thread scorrer, sta in attesa di un segnale SIGUSR1 (fine scrittura nella lista dei punteggi da parte dei client) per poi scrivere la classifica in una variabile globale
    c. Thread gestore matrice, si occupa della gestione del tempo e della creazione della matrice dopo ogni pausa. Invia ai client registrati il segnale SIGURSR1 e allo scorrer il segnale SIGUSR per gestire la classifica.
    d. Thread timer, si occupa della gestione dei tempi di disconnessione dei client attivi, attraverso la gestione di un array con tutti i tempi di ultimo messaggio.
3. **Gestione segnali,** con relativi handler : Segnalano semplicemente la ricezione del messaggio, escluso l'handler del SIGINT. Quest'ultimo permette l'arresto del server e dei client in modo controllato e pulito.

Il file client.c contiene:

Comunicazione Client-Server:

- Il client stabilisce una connessione TCP con il server.
- Il client invia messaggi al server con tipi diversi (ad esempio, proposta di parola, registrazione utente, ecc.).
- Inoltre, ascolta le risposte dal server utilizzando un thread dedicato CONTROLLORE che riceve i messaggi e li elabora di conseguenza.
- Il thread controlla continuamente il tipo di messaggio ricevuto e lo elabora in base al tipo

5. Gestione degli Errori:

- Vengono stampati messaggi di errore per casi specifici (ad esempio, input non valido, utente non registrato, ecc.).

6. Validazione dell'Input:

- Il client valida l'input dell'utente per vari comandi. Verificando la correttezza del prompt.