Relazione del Progetto “The Real Paroliere”

Candidato:Luca Valente
Corso A
Matricola N° 616750

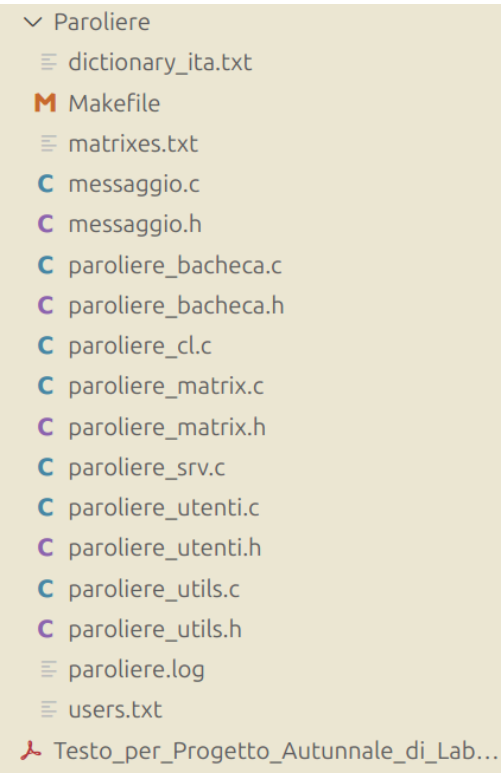
Introduzione:

Il progetto “The Real Paroliere online” è un sistema di gioco multithreading che include funzionalità di registrazione, login, cancellazione degli utenti registrati, generazione di matrici, inserimento messaggi su una bacheca condivisa e display degli stessi fino ad 8 messaggi, gestione temporale delle partite (durata partita + pausa).

Scopo del gioco:

- I giocatori devono trovare il maggior numero possibile di parole in una griglia 4x4 di lettere.
- Le parole devono essere formate collegando lettere adiacenti (in orizzontale, verticale o diagonale).
- Le parole devono esistere nel dizionario italiano (o in un dizionario che viene specificato all’avvio del server per passaggio di parametro.
- Le parole devono essere di almeno 4 lettere.

Organizzazione dei file:



Il progetto si sviluppa principalmente su 4 tipi di file:

File .c (Sorgenti)

paroliere_srv.c	→	File principale del server
paroliere_cl.c	→	File principale del client
messaggio.c	→	Gestione messaggi
paroliere_matrix.c	→	Gestione matrice di gioco
Paroliere_utenti.c	→	Gestione utenti
paroliere_bacheca.c	→	Gestione bacheca
paroliere_utils.c	→	Utilità varie

File .h (Header)

messaggio.h	→	Definizioni messaggi
paroliere_matrix.h	→	Definizioni matrice
Paroliere_utenti.h	→	Definizioni utenti
paroliere_bacheca.h	→	Definizioni bacheca
paroliere_utils.h	→	Definizioni utilità

File .txt/.log (Dati)

- dictionary_ita.txt → Dizionario italiano
- users.txt → Database utenti
- matrixes.txt → Matrici di test
- paroliere.log → File di log

File di Build:

- Makefile → Script di compilazione

Scelte progettuali:

- ***Modularità del codice***: Il codice è suddiviso in diversi moduli, ciascuno responsabile di una funzionalità specifica, come la gestione degli utenti (paroliere_utenti.c), la gestione della matrice di gioco (paroliere_matrix.c) e la gestione della bacheca dei messaggi (paroliere_bacheca.c).
- ***Strutture dati principali:***
    - **Trie**: Utilizzato per il dizionario delle parole, consentendo una ricerca efficiente delle parole valide. Implementato in paroliere_matrix.c.
    - **Liste concatenate**: Usate per gestire la lista dei client connessi e la coda dei punteggi, come definito in paroliere_utils.h.
    - **Array**: Impiegati per gestire la matrice di gioco e per memorizzare gli utenti registrati e loggati.
- ***Algoritmi fondamentali:***
    - **Ricerca parole nella matrice**: Un algoritmo ricorsivo di backtracking verifica se una parola può essere formata nella matrice di gioco. Questo è implementato nelle funzioni di paroliere_matrix.c.
    - **Gestione concorrente dei client**: L'uso di thread permette al server di gestire molteplici client simultaneamente. La sincronizzazione è garantita tramite mutex e semafori, come visto in paroliere_srv.c.
    - **Ordinamento dei punteggi**: Viene utilizzato un algoritmo di bubble sort per ordinare i punteggi dei giocatori alla fine di ogni partita, come mostrato in paroliere_srv.c alle linee in cui viene gestita la classifica.
- **Motivazioni alla base**:
    - **Efficienza**: L'uso del trie per il dizionario come descritto da traccia del progetto. parole.
    - **Scalabilità**: La struttura multithread e l'uso di semafori permettono al server di gestire efficacemente un numero elevato di client e di cambiare poche cose all’interno del codice per poterne gestire ancora di più.
    - **Affidabilità**: La sincronizzazione delle risorse condivise tramite mutex assicura l'integrità dei dati, prevenendo race conditions.

Strutturazione del codice (logica di divisione):

Il codice è strutturato in modo modulare, con una divisione logica in più file, ciascuno responsabile di una specifica funzionalità.

Essendo questa la prima esperienza con la programmazione C potrebbe essere leggermente difficile leggere con facilità questi file, in quanto la separazione è stata fatta per questioni di logica, ma al loro interno l'ordine è carente. Ho tuttavia provveduto a commentare adeguatamente il codice per rendere la lettura più scorrevole e comprensibile:

- paroliere_srv.c: Implementa il server. Gestisce le connessioni dei client, la comunicazione e coordina le varie componenti del gioco. Si occupa di avviare il server, gestire i thread per ogni client e implementare la logica principale del gioco.
- paroliere_cl.c: Implementa il client. Gestisce la comunicazione con il server, l'interfaccia utente e l'elaborazione dei comandi inseriti dall'utente.
- paroliere_utils.c e paroliere_utils.h: Contengono funzioni di utilità condivise tra server e client. Includono funzionalità per la gestione dei client connessi, la sincronizzazione tramite mutex e semafori, e operazioni comuni come il logging.
- paroliere_utenti.c e paroliere_utenti.h: Gestiscono la registrazione, l'autenticazione e la gestione degli utenti. Implementano funzioni per caricare gli utenti da users.txt, aggiungere nuovi utenti, verificare le credenziali e aggiornare i punteggi totali.
- paroliere_matrix.c e paroliere_matrix.h: Si occupano della generazione e gestione della matrice di gioco. Forniscono funzionalità per generare matrici casuali o caricarle da matrixes.txt, verificare se una parola è presente nella matrice e gestire il dizionario tramite un trie caricato da dictionary_ita.txt.
- paroliere_bacheca.c e paroliere_bacheca.h: Implementano la bacheca dei messaggi, permettendo ai giocatori di pubblicare e visualizzare messaggi. Gestiscono l'aggiunta di nuovi messaggi, la sincronizzazione dell'accesso tramite mutex e il recupero dei messaggi in formato CSV.
- messaggio.c e messaggio.h: Definiscono la struttura dei messaggi scambiati tra client e server e implementano le funzioni per l'invio e la ricezione dei messaggi, standardizzando la comunicazione.
- `Makefile`: Contiene le istruzioni per compilare il progetto, automatizzando la costruzione sia del client che del server.
- paroliere.log: File di log dove vengono registrati gli eventi significativi arrivati al server, come le operazioni degli utenti.

Struttura dei programmi:

La struttura dei programmi è basata su un'architettura client-server, dove il server gestisce la logica di gioco e la comunicazione con i client attraverso socket TCP. L'uso di thread permette al server di gestire contemporaneamente più client, e le meccaniche di sincronizzazione assicurano la coerenza dei dati condivisi.

La divisione del codice in file specifici per ciascuna funzionalità facilita la manutenzione e l'espansione futura del progetto. Ogni componente è ben isolato, permettendo modifiche indipendenti senza influire sul resto del sistema.

Breve overview dei Thread presenti nel progetto:

Come citato più volte di sopra, il progetto è stato sviluppato per supportare un gioco in multithreading.

Un breve schema dei thread che sono presenti lato server e lato client, è il seguente:

- Un thread principale che gestisce l'accettazione delle nuove connessioni.
- Un thread per ciascun client connesso, quindi il numero varia dinamicamente in base al numero di client che stanno giocando contemporaneamente. Per consigli da traccia l'attuale limite è stato settato a 32, la variabile globale BACKLOG presente in
- Un "scorer_thread" che gestisce i punteggi e le classifiche.
- Un "tempo_thread" che gestisce il timer delle partite.

Il client, invece, funziona interamente su **un singolo Thread** senza creare thread aggiuntivi.

Struttura dei test:

Il sistema è stato testato ripetutamente con più combinazioni. Il file Makefile presenta delle strutture di test che ho scritto per mostrare qual era la forma designata da un makefile. La mancanza di uso dei test tramite makefile è dovuta alla limitazione dove tester doveva poter prendere delle decisioni sul flusso delle partite, testando possibili combinazioni differenti. Il programma è stato testato ripetutamente giocando normalmente con client connessi contemporaneamente (alcune partite sono anche registrate nei file di log se si desidera dare un'occhiata).

Sono state testate tutte le possibili combinazioni di avvio del server per garantire che i passaggi di parametro all'avvio dello stesso funzionassero.

Compilazione ed esecuzione del codice:

Compilazione tramite il comando:

make

Oppure se si preferisce ricompilare:

make clean && make

Per avviare il codice da terminale (controllare si è nella folder corretta Paroliere/) per server:

(Avvio base)

./paroliere_srv --port 12345

(Avvio con configurazione alternativa)

./paroliere_srv --port 12345 --matrici matrixes.txt --durata 3 --seed 42 --disconnetti-dopo 30

Per avviare il codice da terminale (controllare si è nella folder corretta Paroliere/) per client:

./paroliere_cl --server localhost --port 12345

Una volta connessi sono disponibili i seguenti comandi:

```
- registra_utente <nome>    # Registra nuovo utente (max 10 caratteri)
- login_utente <nome>       # Accedi con utente esistente
- matrice                   # Visualizza matrice corrente
- p <parola>               # Proponi una parola
- msg <testo>              # Invia messaggio in bacheca
- show-msg                 # Visualizza bacheca
- punti_finali             # Visualizza punteggi
- cancella_utente <nome>   # Rimuovi un utente
- fine                     # Chiudi il client
```