The Taxicab game

## Comandi TRamite MakeFILE:

Compilazione: make all

Esecuzione: make run

Eliminazione: make clean

## STruttura

Il codice è strutturato nel seguente modo:

Cartella bin: contiene i file binari.

Cartella build: contiene i file oggetto.

Cartella src: contiene i file sorgente.

Parameters: file in formato .txt, contenente I parametri utilizzati per configurare l’esecuzione del programma.

Makefile: file utilizzato per compilare e avviare il programma.

## DeScrizione DEL Codice

All’interno della cartella src sono presenti quattro file:

Master.c che da il via alla simulazione degli spostamenti dei taxi all’interno di una città. Esso si occupa della lettura dei parametri dal file Parameters.txt, successivamente crea la mappa di gioco e genera tramite processo padre i processi figli chiamati Taxi e Source utilizzati per rappresentare l’iter lavorativo del tassista (ovvero prendere il passeggero e portarlo a destinazione). La comunicazione dei processi avviene attraverso appositi meccanismi IPC, come coda di messaggi, memoria condivisa e semafori, permettendo quindi la sincronizzazione dei processi ed il corretto accesso alle sezioni critiche di codice. Il processo Master si occupa poi di raccogliere le statistiche in uscita dei processi Taxi e Source e ricavarne un output da stampare insieme a mappa di gioco e statistiche dettagliate dei Taxi ottenute attraverso l’apposita memoria condivisa.

All’interno di Source.c vi è invece la gestione dei processi figli Source i quali segnaleranno ai Taxi tramite code di messaggi la posizione di partenza del passeggero, inserita come tipo di messaggio, e la loro destinazione, inserita come contenuto del messaggio. Inoltre, ogni 2 secondi i processi Source, attraverso un apposito segnale di alarm, controllano se siano ancora presenti delle richieste da loro inoltrate e in caso contrario si occupano di chiudere il processo.

Taxi.c i raccoglie ed esegue le richieste dei processi Source attraverso il prelievo di un messaggio dall’apposita coda. Il Taxi viene generato in una cella casuale e controlla immediatamente se si trova su una cella Source. In caso di riscontro positivo, cercherà un messaggio e ne analizzerà il contenuto, iniziando nel caso un viaggio verso la destinazione. Come da specifiche progettuali si muoverà solamente in orizzontale e in verticale all’interno della mappa passando da cella a cella utilizzando l’apposito semaforo tramite un’operazione temporizzata, al fine di non superare il tempo di attesa imposto dal parametro di timeout, caso nel quale si considererebbe il viaggio “abortito” ed in seguito rigenerato. Diversamente, il Taxi verrà eliminato e riposizionato se non trova una richiesta sulla cella in cui è localizzato.

Il Common infine è utilizzato come libreria nella quale sono presenti: valori di uscita dei processi, struct utilizzate per il passaggio di dati tramite memoria condivisa, metodi per la segnalazione degli errori e funzioni per la sincronizzazione dei processi tramite semafori.

I meccanismi IPC utilizzati consistono in:

* Una coda di messaggi, in cui principalmente i Source sono gli scrittori ed i Taxi i lettori.
* Due set di semafori; il primo contenente il semaforo di sincronizzazione di partenza dei processi Master, Source e Taxi ed il mutex dedicato alla scrittura sulla memoria condivisa. Il secondo set invece è dedicato al cap massimo di taxi per cella e contiene quindi un semaforo per ogni cella.
* Tre memorie condivise, due delle quali contengono la mappa di gioco sotto forma di array anziché di matrice e vengono passate a Source e Taxi rispettivamente, per un utilizzo diretto. La terza invece è utilizzata da Master e Taxi per la scrittura e lettura delle statistiche di viaggio.