# Relazione del Progetto di Sistemi Operativi
# A.A. 2020/2021

## The Taxicab Game

Tarquini Alice 861373 Corso B Turno T3
Luciani Fabio 863367 Corso B Turno T3
Marino Giuseppe 814025 Corso B Turno T3

Il progetto consiste di 7 file .c , 2 file .h e un Makefile ed è stato così pensato per sfruttare il più possibile le tecniche di divisione in moduli del codice, viene compilato tramite l'utility **make** con le opzioni di compilazione "gcc -std=c89 -pedantic" e il processo da avviare tramite make è master.c, mentre tutti gli altri file sono da esso dipendenti.

### Makefile

L'intero codice viene avviato tramite Makefile con la quale si deve determinare la tipologia di mappa ( o Città ) su cui si vogliono far viaggiare i propri taxi. Il makefile è stato dotato dei seguenti comandi:

- "make dense" e "make large" che creano una mappa con parametri predefiniti sulla base dei dati forniti nella consegna del progetto e visibili qua sotto:

| parametro | "dense" | "large" |
|---|---|---|
| SO_WIDTH | 20 | 60 |
| SO_HEIGHT | 10 | 20 |
| SO_HOLES | 10 | 50 |
| SO_TOP_CELLS | 40 | 40 |
| SO_SOURCES | SO_WIDTH×SO_HEIGHT−SO_HOLES | 10 |
| SO_CAP_MIN | 1 | 3 |
| SO_CAP_MAX | 1 | 5 |
| SO_TAXI | SO_SOURCES/2 | 1000 |
| SO_TIMENSEC_MIN [nsec] | 100000000 | 10000000 |
| SO_TIMENSEC_MAX [nsec] | 300000000 | 100000000 |
| SO_TIMEOUT [sec] | 1 | 3 |
| SO_DURATION [sec] | 20 | 20 |

Tabella 1: Esempio di valori di configurazione.

- "make custom", crea una mappa di dimensioni 10*10 in cui tutti i valori vengono inseriti tramite input da tastiera;
- "make clean", da utilizzare in caso di modifica dei valori SO_WIDTH ed SO_HEIGHT definiti nel file header.h ( utile in caso di modifiche delle dimensioni della mappa di tipo custom );
- "make run", da utilizzare successivamente a tutti gli altri comandi per avviare il processo master.c e, quindi, il programma;

### header.h

Il file header.h contiene al suo interno:
- macro
- chiavi usate dalle memorie condivise
- definizione delle sole dimensioni delle mappe custom, large e dense
- struct cell, map, param, message, record, stats
- metodi vari tra cui generazione di numeri casuali, stampa della mappa, ottenimento dei parametri, ecc.

## header.c

Il file header.c si occupa della definizione di tutti i metodi precedentemente dichiarati nel file header.h

## semaphore.h

Il file semaphore.h contiene al suo interno:
- chiave usata dai semafori
- id dei semafori usati
- dichiarazione dei metodi per l'uso dei semafori

Utilizziamo 5 semafori, uno per l'accesso alla memoria condivisa, uno per l'accesso alla coda di messaggi, un semaforo per la gestione dei taxi, un semaforo per la gestione delle source e un semaforo dove aspettiamo la creazione e inizializzazione di tutti i taxi e le sources per far partire la simulazione.

## semaphore.c

Il file semaphore.c si occupa della definizione di tutti i metodi per l'uso dei semafori dichiarati nel file semaphore.h.

## holes.c

Il file holes.c sostanzialmente crea i buchi da inserire nella mappa del gioco.
Nel caso in cui si stia usando la mappa "custom", se il numero di buchi è maggiore del valore SO_WIDTH*SO_HEIGHT viene richiesto l'inserimento di un numero compatibile grazie al quale il processo prova ad inserire tutti i buchi richiesti nella mappa.
Se però tale richiesta di inserimento non viene eseguita entro un secondo, scatta un alarm che, tramite un segnale SIGALRM, ne interrompe l'inserimento e attraverso un segnale SIGTERM restituisce il numero di buchi effettivamente inseriti.

## taxi.c

Inizialmente vengono posizionati i taxi sulla mappa, facendo attenzione ad evitare i buchi e a rispettare le capacità massime delle celle. Una volta partita la simulazione i taxi iniziano a muoversi alla ricerca di una cella source, trovata questa accedono alla coda di messaggi, prelevano il messaggio contenente la destinazione da raggiungere e iniziano il loro tragitto verso la destinazione.
Una volta raggiunta la destinazione vengono aggiornate le statistiche e il taxi ricomincia la ricerca di una source.
Nel caso in cui un taxi sia fermo per un tempo >= SO_timeout, questo taxi viene marcato come aborted, rilascia le risorse e il suo processo viene terminato.
Nel momento in cui viene terminato, verrà creato un nuovo processo taxi tramite “taxi_reborn".

## taxi_reborn.c

Si occupa essenzialmente delle stesse funzionalità di "taxi", se non per la mancanza dei semafori relativi all'attesa prima della partenza della simulazione.

## sources.c

Crea le source, quando parte la simulazione ogni source, con un intervallo di 4 secondi, genera delle richieste che hanno e le carica in coda di messaggi.

## master.c

Il file master.c è il processo che si occupa della gestione della simulazione. Inizializza le strutture dati, si occupa di prendere i parametri per la simulazione ed esegue in sequenza:
- fork per inserimento buchi (chiamata di "holes" tramite execve); -- holes.c

- fork di n processi taxi (chiamata di "taxi" tramite execve); -- taxi.c
- fork di m processi source (chiamata di "source" tramite execve); -- sources.c
- attesa che i taxi e le source siano create e inizializzate;
- partenza della simulazione.

Una volta iniziata la simulazione, ogni secondo, attraverso un alarm, viene stampata la mappa con lo stato di occupazione delle celle.
Alla fine della simulazione, stamperà una mappa con evidenziate le sources e i buchi, l'elenco delle SO_top_cells e le statistiche finali.