

Progetto SO 2020/21

The Taxicab game

provvisoria

Bini, Radicioni, Schifanella

30 novembre 2020

Indice

1	Descrizione	1
1.1	Mappa della città	1
1.2	Richieste	2
1.3	Situazione di stallo	2
1.4	Terminazione	2
1.5	Stampa	2
2	Configurazione	3
3	Requisiti implementativi	3
4	Composizione gruppo di studenti	3
5	Consegna	3
6	Valutazione e validità	4

1 Descrizione

Si intende realizzare un sistema in cui sono presenti vari taxi (simulati da processi) che si spostano all'interno di una città.

1.1 Mappa della città

Le strade della città sono rappresentate da una griglia di larghezza `SO_WIDTH` e altezza `SO_HEIGHT`. La griglia ha `SO_HOLES` celle inaccessibili in posizione casuale. Non ci possono essere “barriere” di celle inaccessibili: per ogni cella inaccessibile le otto celle adiacenti, non possono essere inaccessibili. Figura 1 illustra un esempio di posizionamento legale di celle inaccessibili, mentre Figura 2 mostra alcuni casi di celle inaccessibili.

Ogni cella è caratterizzata da

- un tempo di attraversamento estratto casualmente fra `SO_TIMENSEC_MIN` e `SO_TIMENSEC_MAX` (in nanosecondi)
- una capacità estratta casualmente fra `SO_CAP_MIN`, `SO_CAP_MAX` che rappresenta il numero massimo di taxi che possono essere ospitati nella cella.

Le caratteristiche della cella di cui sopra sono generate casualmente all'atto della creazione della mappa e non variano nel tempo.

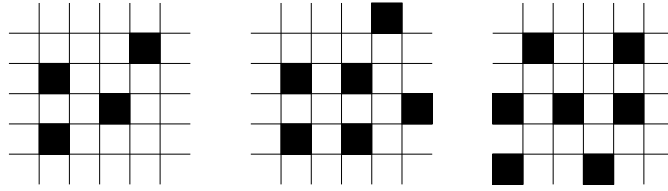


Figura 1: Esempio di posizione **consentita** delle celle inaccessibili.

1.2 Richieste

Le richieste di taxi sono originate da `SO_SOURCES` celle della mappa e sono caratterizzate da:

- la cella di partenza (la cella fra le `SO_SOURCES` ove la richiesta è generata);
- la cella di destinazione determinata in modo casuale fra una qualunque delle celle della mappa.

Le richieste originatesi in ognuna delle `SO_SOURCES` sorgenti sono generate da `SO_SOURCES` processi che immettono tali richieste in una coda di messaggi oppure una pipe. I processi taxi prelevano la richiesta a seconda della loro convenienza. Le richieste possono essere servite soltanto da un taxi che si trova nella cella di origine della richiesta.

Le richieste sono create da `SO_SOURCES` processi secondo un pattern configurabile a propria scelta (per esempio attraverso un alarm periodico oppure con un intervallo variabile). Inoltre deve poter essere possibile generare una richiesta da terminale, mediante invio di segnale al processo “generatore”.

1.3 Situazione di stallo

Se un taxi non si muove entro `SO_TIMEOUT` secondi termina rilasciando eventuali risorse possedute. Se stava effettuando una corsa per un cliente, la richiesta viene marcata come “aborted”.

1.4 Terminazione

La simulazione ha una durata `SO_DURATION` secondi che sarà un alarm per il processo master.

1.5 Stampa

È presente un processo “master” che raccoglie le statistiche delle varie richieste eseguite.

Ogni secondo viene stampato a terminale lo stato di occupazione delle varie celle.

Alla fine della simulazione vengono stampate:

- numero viaggi (eseguiti con successo, inevasi e abortiti)
- la mappa con evidenziate le `SO_SOURCES` sorgenti e le `SO_TOP_CELLS` celle più attraversate
- il processo taxi che
 1. ha fatto più strada (numero di celle) di tutti
 2. ha fatto il viaggio più lungo (come tempo)
 3. ha raccolto più richieste/clienti

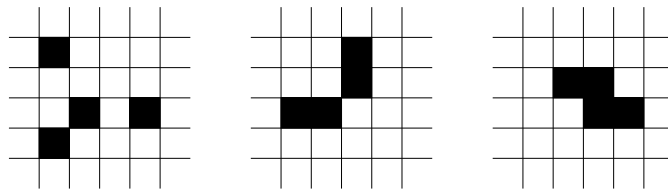


Figura 2: Esempio di posizione **non consentita** delle celle inaccessibili.

2 Configurazione

I seguenti parametri sono letti a **tempo di esecuzione**:

- `SO_TAXI`, numero di taxi presenti
- `SO_SOURCES`, numero di punti sulla mappa di origine dei clienti (si immagini che siano le stazioni, gli aeroporti, etc.)
- `SO_HOLES`, numero di celle della mappa inaccessibili. Si ricordi: ogni cella inaccessibile non può averne altre nelle otto celle che la circondano
- `SO_CAP_MIN`, `SO_CAP_MAX`, capacità minima e massima di ogni cella
- `SO_TIMENSEC_MIN`, `SO_TIMENSEC_MAX`, tempo minimo e massimo necessario per l'attraversamento di ogni cella (in nanosecondi).

Un cambiamento dei precedenti parametri non deve determinare una nuova compilazione dei sorgenti. Inoltre, i seguenti parametri sono letti a **tempo di compilazione**, da file o da variabili di ambiente:

- `SO_WIDTH`, larghezza della mappa;
- `SO_HEIGHT`, altezza della mappa.

3 Requisiti implementativi

Il progetto deve essere realizzato sfruttando le tecniche di divisione in moduli del codice, compreso l'utilizzo dell'utility `make`.

Al termine del gioco, le risorse IPC che sono state allocate dai processi devono essere chiuse. Le opzioni di compilazione del codice devono contenere almeno:

```
gcc -std=c89 -pedantic
```

Il codice del progetto deve poter eseguire su una macchina (virtuale o fisica) che presenta parallelismo (due o più processori).

4 Composizione gruppo di studenti

Il progetto si deve svolgere in gruppo composto al **massimo 3 da componenti**. È possibile anche svolgere il progetto da soli.

Si raccomanda che il gruppo sia composto da studenti dello **stesso turno**, i quali discuteranno con il docente del proprio turno. È consentita anche la realizzazione del progetto di laboratorio da parte di un gruppo composto da studenti di turni diversi alle seguenti condizioni. In questo caso, **tutti** gli studenti del gruppo discuteranno con **lo stesso docente**. Esempio: Tizio (turno T1) e Caio (turno T2) decidono di fare il progetto insieme. Lo consegnano e vengono convocati dal prof. Radicioni il giorno X. Tale giorno Tizio e Caio si presentano e ricevono entrambi una valutazione dal Prof. Radicioni (anche se Caio fa parte del turno T2 il cui docente di riferimento è il prof. Bini).

5 Consegna

Il progetto è costituito da:

1. il codice sorgente
2. una breve relazione che sintetizzi le scelte progettuali compiute

Il progetto si consegna compilando la seguente Google Form

- <https://forms.gle/mQGURwe89StBvB818>

la quale richiederà, oltre al caricamento del progetto stesso (un unico file in formato .zip o .tgz), anche i seguenti dati per ciascun componente del gruppo: cognome, nome, matricola, email. Dopo il caricamento del progetto, verrete convocati dal docente con cui discuterete (si veda Sezione 4 in caso di gruppo composto da studenti di turni diversi). Attenzione: **consegnare il progetto una volta sola**. Un eventuale ulteriore consegna prima dell'appuntamento **annullerà la data dell'appuntamento**.

La consegna deve avvenire almeno **10 giorni prima** degli appelli scritti per dare modo al docente di pianificare i colloqui:

- se spedite 10 giorni prima di un appello, il docente propone una data per la discussione entro l'appello seguente
- altrimenti, la data sarà dopo l'appello seguente.
- È inoltre necessario che la mail contenente il progetto sia inviata in copia a tutti gli eventuali co-autori, e corredata dei rispettivi dati (nome, cognome, matricola).

6 Valutazione e validità

Il progetto descritto nel presente documento potrà essere discusso inviando l'email al docente entro Novembre 2021. Da Dicembre 2021 sarà discusso il progetto assegnato durante l'anno accademico 2021/22.

Tutti i membri del gruppo devono essere presenti alla discussione. La valutazione del progetto è **individuale** ed espressa in 30-esimi. Durante la discussione

- verrà chiesto di illustrare il progetto
- verranno proposti quesiti sul programma "Unix" del corso

È necessario ottenere una votazione di almeno **18** su 30 per poter essere ammessi allo scritto. In caso di superamento della discussione del progetto, la votazione conseguita consentirà di partecipare allo scritto per i **cinque appelli successivi** alla data di superamento.

In caso di mancato superamento, lo studente si potrà ripresentare soltanto dopo almeno **un mese** dalla data del mancato superamento

Si ricorda che il voto del progetto ha un peso di $\frac{1}{4}$ sulla votazione finale di Sistemi Operativi.