Progetto SO 2019/20

Bini, Radicioni, Schifanella

28 novembre 2019

Indice

1 Descrizione

Si intende realizzare un gioco in cui dei giocatori competono tra di loro per la conquista di alcune bandierine poste all'interno di un campo di gioco modellato come una scacchiera. Per conquistare le bandierine i giocatori possono spostare le loro pedine.

Sono presenti 3 tipi di processo:

1. un processo `master` che gestisce il gioco,
2. `SO_NUM_G` processi giocatore,
3. `SO_NUM_P` processi pedina per ogni giocatore.

1.1 Round

Tutta la durata del gioco è suddivisa in *round*. Ogni round:

1. il processo master posiziona un numero di bandierine a sua scelta, ognuna su una cella libera da pedine. Il processo master decide il punteggio assegnato ad ogni bandierina;
2. il processo master stampa lo stato secondo quanto descritto in Sezione 1.6;

3. il processo master segnala ai giocatori l'inizio del round, rendendo loro noti posizione e punteggio delle bandierine piazzate;
4. i giocatori forniscono indicazioni alle pedine in modo che le pedine si possano muovere al fine di occupare le celle con le bandierine e ottenere il punteggio relativo;
5. dopo che **tutti** giocatori hanno fornito le indicazioni date all'inizio round alle loro pedine, le pedine iniziano a mouversi;
6. ogni volta che una bandierina è conquistata tutte le pedine, tutti i giocatori e il master vengono informati;
7. quando tutte le bandierine piazzate nel round sono state conquistate, il processo master:
   (a) termina il round
   (b) stampa lo stato secondo quanto descritto in Sezione 1.6;
   (c) piazza le nuove bandierine e avvia un nuovo round.

Se un round dura più di `SO_MAX_TIME` secondi, il processo master ne è informato e termina il gioco, stampando lo stato (come descritto in Sezione 1.6).

1.2 Processo master

Inizio gioco Il master genera una scacchiera rettangolare di dimensioni `SO_BASE` e `SO_ALTEZZA`. Poi genera `SO_NUM_G` processi giocatore e attende che i processi giocatore abbiano piazzato le loro pedine. Dopo che i processi giocatore hanno piazzato tutte le loro pedine, si avvia il primo round.

Svolgimento del round All'inizio di ogni round, il master distribuisce le bandierine sulla scacchiera su celle non occupate da pedine. Il numero di bandierine distribuite per ogni round è un numero casuale compreso fra `SO_FLAG_MIN` e `SO_FLAG_MAX`. La somma totale dei punti assegnati alle bandierine è uguale a `SO_ROUND_SCORE`.

Dopo aver piazzato le bandierine, il master stampa lo stato secondo quanto descritto in Sezione 1.6. Permette quindi ai giocatori di fornire le indicazioni di inizio round alle pedine. Dopo che questo è avvenuto, avvia il timer e permette alle pedine di iniziare a muoversi.

Fine gioco Il gioco termina quando dall'inizio di un round sono trascorsi `SO_MAX_TIME` secondi senza che il round sia terminato.

1.3 Processo giocatore

Il processo giocatore ha il compito di coordinare le pedine e prende le proprie decisioni senza l'intervento di giocatori umani.

Inizio gioco All'inizio del gioco, i processi giocatore piazzano **a turno** una pedina alla volta in una cella disponibile, secondo la strategia che ritengono più opportuna. Una pedina è un processo. Si tenga presente che in questa fase, le bandierine non sono state piazzate. Dopo aver piazzato tutte le proprie pedine, tutti i processi giocatore rimangono in attesa dell'inizio del primo round.

Svolgimento del round All'inizio del round, il master piazza le bandierine. I giocatori danno quindi le prime indicazioni alle proprie pedine su come spostarsi. Una volta terminata la comunicazione delle indicazioni del giocatore alle proprie pedine, il master viene informato. Il master quindi, avvia il round (e il timer) e permette alle pedine di muoversi. Maggiori dettagli sulle indicazioni di movimento sono in Sezione 1.4.

1.4 Processo pedina

Movimento pedina Ogni pedina occupa una cella della scacchiera. La posizione delle pedine è nota a tutte le altre pedine (ed ai processi giocatore) in ogni fase del gioco (compresa la fase iniziale di posizionamento delle pedine). Le pedine si possono muovere nelle quattro direzioni possibili di una cella alla volta, ma non oltre il bordo della scacchiera. Prima di spostarsi, provano ad accedere alla cella verso cui intendono dirigersi. Se una pedina riesce ad accedere ad una nuova cella:

1. rilascia la risorsa della cella precedentemente occupata
2. esegue una `nanosleep(...)` di `SO_MIN_HOLD_NSEC` nanosecondi
3. prosegue con l'esecuzione del resto del codice.

(la `nanosleep(...)` serve a rallentare la simulazione che altrimenti sarebbe troppo veloce).

Ogni pedina ha un numero totale `SO_N_MOVES` di mosse a sua disposizione per tutto il gioco. Il numero di mosse residue di ogni pedina è noto all'interno della propria squarda (giocatore e pedine). Quando il numero di mosse residue raggiunge zero, il processo pedina rimane fermo nella cella occupata.

I processi pedina eseguono in parallelo, ciascuno seguendo la propria strategia.

Pedine e giocatore della stessa squadra, se lo ritengono utile, possono sincronizzarsi. Pedine di giocatori diversi **non possono** sincronizzarsi.

Indicazioni di movimento Un processo pedina riceve dal proprio giocatore l'indicazione su come spostarsi. All'inizio del round, le pedine di ogni giocatore attendono che **tutti** i giocatori abbiano dato alle proprie pedine le indicazioni sui primi spostamenti da fare. Invece, nelle fasi successive del gioco tale sincronizzazione non è presente. La pedina è libera di seguire le indicazioni del giocatore oppure di fare altre scelte se questo è ritenuto vantaggioso. Se una pedina ritiene di non poter raggiungere il proprio obiettivo (per esempio, perché la cella verso cui si vuole spostare è occupata, perché non ha mosse a sufficienza, etc.), ne può dare comunicazione al proprio giocatore. Il giocatore, se vuole, imposta nuove indicazioni di spostamento. Alternativamente, la pedina può decidere in autonomia come spostarsi se lo ritiene vantaggioso per la propria squadra.

Indicazioni di spostamento Le indicazioni del giocatore alle sue pedine possono essere fornite in due momenti:

1. prima dell'inizio del round: in questa fase le pedine non si stanno spostando;
2. durante lo svolgimento del round: in questa fase le pedine si stanno spostando.

Il tipo di indicazione che il giocatore fornisce alla pedina è a discrezione degli sviluppatori del progetto. I giocatori non devono conoscere le indicazioni date dagli altri. L'obiettivo di ogni giocatore è quello di raggiungere il punteggio più alto ottenuto avendo una propria pedina nella casella della bandierina in modo da guadagnare i punti corrispondenti. Se ritenuto utile, durante lo svolgimento del round, il giocatore può cambiare le indicazioni date alla pedina.

Conquista di una bandierina Quando una pedina conquista una bandierina, tutte le altre pedine, tutti i giocatori e il master sono informati. Il processo master ha tra i suoi compiti anche quello di tenere il punteggio del gioco.

1.5 Scacchiera

La scacchiera è realizzata con una matrice in memoria condivisa. Ogni cella è protetta da un semaforo distinto. Il processo pedina che vuole entrare in una cella chiede l'accesso al semaforo. Attenzione: ci possono essere deadlock con grande facilità!! Per questo motivo, la pedina che non riesce ad accedere ad una cella può anche decidere di desistere (per esempio attraverso `semtimedop(...)` oppure invocando la `semop(...)` con flag il `IPC_NOWAIT`).

1.6 Stampa stato

Quando necessario, viene stampato lo stato del gioco che consiste in:

- una rappresentazione visuale della scacchiera in ASCII che illustra posizione delle pedine e delle bandierine;
- il punteggio attuale dei giocatori e le mosse residue a disposizione.

1.7 Fine del gioco

Il gioco termina quando un round dura più di `SO_MAX_TIME` secondi. Quando ciò succede, il master stampa lo stato come descritto in Sezione 1.6 e le metriche come descritte in Sezione 1.8.

1.8 Metriche

La qualità delle soluzioni adottate è valutata con le seguenti metriche:

- numero di round giocati,
- rapporto di mosse utilizzate e mosse totali per ciascun giocatore,
- rapporto di punti ottenuti e mosse utilizzate,
- rapporto di punti totali (sommati fra tutti i giocatori) e tempo totale di gioco.

2 Configurazione

Il gioco legge i parametri di configurazione a tempo di esecuzione (dalle variabili di ambiente preventivamente impostate o da file). Un cambiamento dei parametri non deve determinare una nuova compilazione dei sorgenti. I parametri di configurazione sono i seguenti:

- `SO_NUM_G`: numero di processi giocatore,
- `SO_NUM_P`: processi pedina per ogni giocatore,
- `SO_MAX_TIME`: durata massima di un round
- `SO_BASE`, `SO_ALTEZZA`: dimensioni del campo di gioco
- `SO_FLAG_MIN`, `SO_FLAG_MAX`: numero minimo e massimo di bandierine posizionate per round
- `SO_ROUND_SCORE`: punteggio totale assegnato per round alle varie bandierine
- `SO_N_MOVES`: numero totale di mosse a disposizione delle pedine (vale per tutto il gioco, non per ogni round)
- `SO_MIN_HOLD_NSEC`: numero minimo di nanosecondi di occupazione di una cella da parte di una pedina

La Tabella 1 elenca valori “easy” e “hard” per le configurazione da testare.

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
```

parametro	"easy"	"hard"
SO_NUM_G	2	4
SO_NUM_P	10	400
SO_MAX_TIME	3	1
SO_BASE	60	120
SO_ALTEZZA	20	40
SO_FLAG_MIN	5	5
SO_FLAG_MAX	5	40
SO_ROUND_SCORE	10	200
SO_N_MOVES	20	200
SO_MIN_HOLD_NSEC	(0.1 sec) 100000000	(0.1 sec) 100000000

Tabella 1: Esempio di valori di configurazione.

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:

1. un'unica email di richiesta di un appuntamento per la discussione viene inviata a **tutti** i docenti di laboratorio Unix coinvolti:
   - turno T1: `daniele.radicioni@unito.it`
   - turni T2 e T3: `enrico.bini@unito.it`
   - turno T4: `claudio.schifanella@unito.it`

   Tale email **deve** allegare il progetto che verrà discusso;
2. i docenti si consulteranno e comunicheranno al gruppo con quale docente **tutti** gli studenti del gruppo dovranno effettuare la discussione;
3. **tutti** gli studenti del gruppo si presentano **dal docente comunicato**, il giorno prefissato per la discussione, e riceverranno una valutazione individuale.

5 Consegna

Il progetto è costituito da:

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

Il progetto si consegna inviandolo per email al vostro docente (si veda Sezione 4 in caso di gruppo composto da studenti di turni diversi). Verrà discusso il progetto inviato per email, **non** sue eventuali modifiche successive. Tale email deve

- elencare **nome**, **cognome**, **matricola** di ogni componente del gruppo
- avere in **allegato il progetto** (.zip, .tgz)

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

- 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

6 Valutazione e validità

Il progetto descritto nel presente documento potrà essere discusso inviando l'email al docente entro Novembre 2020. Da Dicembre 2020 sarà discusso il progetto assegnato durante l'anno accademico 2020/21.

La discussione avviene **individualmente**. Durante la discussione

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

La valutazione del progetto è **individuale** ed espressa in 30-esimi.

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 **365 giorni 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.