Relazione Sistemi Operativi 2019/20

Vannella Alessio

Indice

Sommario

Questa breve relazione spiega le scelte progettuali attuate per la realizzazione del progetto didattico del corso di Sistemi Operativi dell'università di Torino.

1 Esecuzione e compilazione del codice

Per la compilazione del progetto è stato predisposto un makefile nella directory principale che compila tutte le dipendenze. In particolare, questi sono i modi di utilizzo:

- "make all": Compila senza eseguire nulla;
- "make run": Compila ed esegue automaticamente il codice;
- "make clean": Pulisce tutti i file generati;

Il file di output eseguibile è denominato "game".
Per impostare la modalità di gioco tra "easy" e "hard", nella sotto cartella config è presente un file bash "set_cfg" che può essere richiamato con la sintassi:

```
./set_cfg mode
```

dove mode può assumere il valore di "easy" o "hard".

2 Struttura del codice

Il progetto è diviso diviso in moduli. Ogni modulo ha una funzione specifica per il progetto.
Dal punto di vista strutturale, nella cartella "lib" sono presenti tutti gli header dei moduli, mentre nella cartella "src" sono presenti le definizioni di tutte le funzioni, diviso per moduli.
I moduli sono divisi in due macro categorie: **librerie di gioco** e **librerie generiche**.

2.1 Position

La libreria position definisce l'omonimo tipo che serve ad identificare univocamente una posizione sulla scacchiera.
Gli unici campi della struttura sono due interi x ed y, che indicano le due coordinate della posizione nella scacchiera.

2.2 Cella

La libreria cella definisce e si occupa di gestire le celle della scacchiera. Per farlo, definisce più tipi di dato:

- flag: utilizzata per rappresentare le bandiere sulla scacchiera;
- pawn: utilizzata per rappresentare le pedina sulla scacchiera;
- cella: la struttura che incapsula le due precedenti. Ogni cella può essere sia una pedina che una bandiera a seconda di come è inizializzata;

2.3 Player

La libreria player racchiude i comportamenti dei giocatori. Definisce una struttura player che ha come campi un intero che segna il numero del giocatore e un puntatore ad array di celle che salva un riferimento per ogni pedina di quel giocatore.
L'unica funzione definita serve a calcolare ed assegnare le indicazioni dal player alle proprie pedine.

2.4 Scacchiera

La libreria scacchiera raggruppa tutti gli strumenti utili per la gestione della scacchiera. Al suo interno è presente, oltre che una matrice di celle per rappresentare la scacchiera stessa, un vettore di celle per salvare le bandierine piazzate nella scacchiera e un intero che tiene traccia del numero delle bandierine rimaste in gioco.

2.5 Statistics

La libreria statistics è un contenitore logico per salvare le statistiche di gioco round dopo round.

2.6 masterStruct

La struttura masterStruct incapsula tutte le altre strutture definite precedentemente. Attraverso una variabile di tipo masterStruct si ha accesso a tutti gli elementi di gioco.

2.7 myIPC

La libreria myIPC racchiude tutti gli strumenti per la sincronizzazione tra processi utilizzati nel progetto: semafori, code di messaggi e memoria condivisa. Inoltre, questa libreria definisce una struttura nella quale vengono salvati tutti gli id delle ipc utilizzate nel programma.

2.8 genericLib

La libreria genericLib definisce dei metodi che non hanno uno scopo preciso e serve anche ad interfacciare il codice principale con le varie librerie.

3 Strategie adottate

3.1 Posizionamento delle pedine

In fase di posizionamento delle pedine, la scacchiera è idealmente divisa in 8 sezioni isolate tra di loro. All'interno di ogni sezione c'è un numero definito di pedine per ogni giocatore. Ogni giocatore avrà almeno una pedina all'interno di ogni sezione.
Questo meccanismo permette di avere un numero equo di pedine sparse per tutta la scacchiera.

3.2 Movimento delle pedine

Prima dell'inizio del round, i giocatori calcolano la posizione della propria pedina più vicina ad ogni bandiera e ne impostano il target.
Una volta avviato il round, le pedine che hanno un target si muoveranno verso il loro l'obiettivo.
Le pedine restanti, invece, calcoleranno in modo dinamico la bandiera più vicina a loro e, se esiste, imposteranno il target su di essa e si muoveranno per conquistarla.

Le pedine con un target, una volta conquistato il loro target (da parte loro o di pedine di altre squadre), si comporteranno come delle pedine senza target e cercheranno la bandiera più vicina alla loro posizione.
I movimenti delle pedine sono limitati al quadrante nel quale sono state create.

Requisito fondamentale trovare un target è che la distanza tra la bandierina e la pedina sia minore o uguale a MAX_DISTANCE, variabile calcolata in base ai parametri del gioco. Se questo requisito non è soddisfatto, le pedine resteranno ferme nella loro posizione per il round attuale.

3.3 Comunicazione tra processi

La comunicazione tra processi è realizzata utilizzando diversi strumenti di sincronizzazione.
Lo stato della partita e i dati di gioco, sono salvati in memoria condivisa. Per realizzare questo meccanismo, sono state create diverse aree di memoria condivisa, una per un'intera struttura masterStruct e una per ogni array dinamico utilizzato nell'esecuzione del codice.
Per segnalare al master che una bandiera è stata catturata, si utilizza una coda di messaggi che invia un messaggio al master con la posizione della bandiera da rimuovere e l'indice del giocatore che ha conquistato la bandiera.
Per quanto riguarda la sincronizzazione tra i vari gruppi di processi, si utilizzano dei semafori.
I semafori invece sono utilizzati per eseguire l'accesso in mutua esclusione ai dati presenti nella memoria condivisa, per evitare l'accesso a dati inconsistenti. L'utilizzo dei semafori è fatto con il flag IPC_NOWAIT.

Sincronizzazione