Relazione progetto SO (Sistemi operativi)

Giorgio Mecca
880847
Turno T3
2019/2020

Il progetto SO 2019/2020 pone la creazione di un “gioco” utilizzando il linguaggio c con lo standard c89.
Il “gioco” si sviluppa su un campo modellato come una scacchiera resa condivisibile tra tutti i processi in gioco; questa sarà suddivisa in tre scacchiere equivalenti, una di char per il posizionamento delle pedine dei vai giocatori(A,B,C,…) e delle bandiere rappresentate dal carattere ‘1’;un’altra è una scacchiera di interi che conterrà i punteggi delle pedine : un punteggio nella scacchiera di int sarà in corrispondenza dell’indirizzo della bandiera corrispondente; l’ultima è una “scacchiera”di semafori per cui ogni semaforo coprirà una posizione della scacchiera generale. Il tutto è reso condivisibile grazie ad una struttura(struct)”map” in memoria condivisa che conterrà 3 interi: i 3 id delle memorie condivise di semafori(idSam), di char (idMap), e di interi (idScore).
Il gioco si svolgerà su tre tipi di processi: il primo sarà il processo master che ha il compito di gestire il gioco e dirigere i player che saranno dei processi figlio di master, i player hanno invece il compito di gestire le varie pedine che gli appartengono e che sono realizzate come processi figlio di ogni player.
Il progetto è composto di 6 file principali:

-config: un file che contiene i dati necessari al game(è obbligatorio rispettare l’ordine e la tipologia di scrittura dei dati)

-data.h: file header che contiene i dati generali per tutti i processi come varie enumerazioni, strutture di messaggi utilizzati per la comunicazione tra processi, strutture per immagazzinare dati delle pedine(piece) utilizzate dai player, strutture per i dati dei player,delle bandiere e le metriche utilizzate dal master.

-function.h,function.c: libreria contenete metodi generali comuni ai processi

-piece.c: file sorgente delle pedine, verrà utilizzato grazie ad una execve() eseguita dopo una fork() del player

-player.c: file sorgente dei player, verrà utilizzato grazie ad una execve() eseguita dopo una fork() del master

-master.c: file sorgente del master che verrà lanciato all’esecuzione del Game

/* tutti i metodi sono commentati con la loro funzione e una descrizione degli attributi*/
Nel progetto è presente anche un file “makefile” utilizzato per la compilatione e l’esecuzione del progetto con i comandi:

→make
→make run

La comunicazione tra i vari processi avviene tramite l’utilizzo di code di messaggi, una tra il master e i player e una per ogni player e le pedine appartenenti quel player. Queste verranno utilizzate gestendo il type del messaggio come il PID del destinatario e il campo ‘id’(presente in ogni struct) come il PID del mittente

Il Game si avvia con il processo master che crea i vari id (memoria condivisa, semafori, coda di messaggi) e crea con una fork() i processi player che verranno reindirizzati al sorgente player.c con una execve. I player da poco avviati creano la coda di messaggi da utilizzare con le proprie pedine e attendono il loro turno per creare e posizionare le pedine. Il turno è gestito dal master con una coda di messaggi, il master in via un messaggio che decide il turno ad un giocatore e attende una sua risposta prima di inviare il messaggio ad un altro giocatore, così per tutte le pedine per effettuare dei turni.Il posizionamento delle pedine avviene secondo la funzione PosixPiece che suddivide la

scacchiera in diversi blocchi e piazza una pedina in ognuno di essi. La divisione in blocchi avviene grazie alla funzione findBA che cerca i fattori della moltiplicazione che genera un numero(numero pedine) con il miglior rapporto tra base e altezza(fattore1, fattore2). Dopo aver effettuato le operazioni iniziali(set dei propri dati, gestione dei signal(handler)) sia la pedina che il giocatore attendono il messaggio di inizio round. I round sono gestiti con dei while(true) e degli switch nel caso di giocatori e pedine, i case dello switch si suddividono in Tag(caratteri singoli che saranno inviai tramite messaggi), questi sono: F,R,P,T sia per il player che per la pedina.Il round comincia con il tag 'F' che viene inviato dal master al player dopo aver piazzato le bandiere e stampato lo stato iniziale della scacchiera. Il tag 'F':

-player: indica la possibilità di dare le indicazioni prima del timer alle pedine. Per far ciò viene scelto un obiettivo per ogni pedina, ci si assicura che ogni bandiera sia segnata come obiettivo di una pedina se questa ne ha la possibilità(numero di mosse minori della distanza) e per le pedine rimanenti viene scelto l'obiettivo(bandiera) più vicina sempre seguendo la possibilità(la distanza non sia eccessiva), per tutte le altre verrà assegnato l'obj [-1,-1] che andrà ad influire sul round della pedina.

-pedine: il tag F indica la invio di un messaggio da parte del player contenente il proprio obiettivo che verrà letto e si posiziona in attesa di un altro messaggio.

Alla fine del case il player invia un messaggio al master con cui conferma la la fine delle operazioni e aspetta il messaggio per cambiare case(si andrà all'inizio del round).

Il round comincia con il master che avvia il timer gestito con la funzione alarm(), avverte i giocatori con un messaggio di inizio round e infine controlla in ciclo se ci sono ancora bandiere sulla scacchiera o se l'allarme è scattato. Il player riceve il messaggio di inizio round indicato con il tag 'R' e si sposta nel case opportuno in cui avverte anche le pedine dell'inizio del round con un messaggio e attende il segnale di fine round del master con una msgrcv(). Le pedine quando ricevono il messaggio con il tag di inizio round 'R' iniziano un ciclo con cui fanno una mossa per ogni iterazione del ciclo, questo ciclo si ferma se il proprio obiettivo è già stato catturato quindi il flag 'imp'=TRUE oppure si è arrivati all'obiettivo e quindi si conquista la bandiera allora si modifica il flag ARRIVO=TRUE, oppure nel caso il round deve terminare il player invia un segnale SIGUSR1 che tramite un handler modifica il flag ROUND=FALSE, alla fine del ciclo la pedina rimane in attesa di un messaggio con il successivo tag. Per ogni mossa la pedina effettua una semop() cercando di ottenere il permesso sulla casella scelta per posizionarsi, se andata a buon fine libera il semaforo precedente e si ferma con una nanosleep(), se non ottiene il l'accesso al semaforo non effettua il movimento e aspetta la prossima iterazione del ciclo per riprovarci.

Al termine del i-esimo round il maste resetta il timer aggiornando il tempo totale di gioco e invia il tag='P' ai player richiedendo le informazioni di conquista e di fine round. I giocatori quindi con il tag 'P' inviano un segnale alle pedine per uscire dal loro ciclo e inviano anche un messaggio con il tag 'P', le pedine quindi inviano un messaggio contenente i dati modificati dal round come posizione corrente, numero mosse rimanenti e un tag di conquista settato su 'C' se hanno conquistato, 'L' altrimenti. I player ricevono il messaggio e controllando i tag inviano un messaggio al master per ogni conquista e effettuando un'altra sincronizzazione con il master e tag 'p' inviano al master un messaggio contenente il totale delle mosse rimanenti. Il master riceve un messaggio per ogni bandierina conquistata e utilizzando il campo 'id' nei messaggi,che indica il pid del mittente, aggiorna il punteggio dei player, in seguito richiede con il tag 'p' il messaggio di fine round con cui aggiorna le mosse e stampa lo stato del Game(scacchiera, giocatori, mosse e punteggio). Infine il master effettua un controllo sull'allarme e se questo è scattato invia in tag 'T' di termine Game ai giocatori, stampa lo stato finale con le metriche e attende la chiusura dei player per poi chiudere la memorie condivise, i semafori e la coda messaggi e terminare. I player nel case 'T' effettuano la stessa operazione inviando il medesimo tag alle pedine e aspettando la loro chiusura, le pedine nel case 'T' terminano (exit(0)). Nel caso l'allarme non fosse scattato il master riposiziona le bandiere stampa lo stato iniziale e reinvia il tag 'F' ricominciando il ciclo.