Progetto di Sistemi Operativi
Modulo Unix

Elisa Solinas
Matricola: 811737 Email: elisa.solinas@unito.it

Simone Stella
Matricola: 813905 - Email: simone.stella298@edu.unito.it

Lorenzo Tabasso
Matricola: 812499 - Email: lorenzo.tabasso@edu.unito.it

16 Gennaio 2017

1 Organizzazione del codice

Il programma calcetto è organizzato in 5 file.c che contengono le differenti funzioni, raggruppate a seconda della tipologia del processo che le esegue. Il nucleo del programma è la funzione `main()`, contenuta nel file arbitro.c ed eseguita dal processo arbitro.

2 Preparazione

Una volta avviato il programma vengono create le differenti IPC:

1. Una zona di memoria condivisa che contiene il punteggio della partita, espresso tramite un array di interi,
2. Un semaforo palla,
3. Due semafori squadra, ciascuno inizializzato a 5, che rappresenta il numero di posti disponibili all'interno della squadra (ciascun giocatore si riserverà una risorsa),
4. Una coda messaggi, che verrà utilizzata per lo scambio di messaggi tra il processo fato e il processo giocatore.

A questo punto, l'arbitro crea i seguenti processi:

1. Fato, che apre il file `log.txt` (su cui verra salvato un resoconto di ciò che avverrà nella partita) e si mette in pausa, attendendo il segnale da parte dell'arbitro che lo avvertirà dell'effettiva creazione di entrambe le squadre,

2. Le Squadre, che vengono create in maniera ricorsiva. In ciascuna di queste vengono successivamente creati i 5 giocatori, i quali si mettono subito in pausa attendendo di poter iniziare la partita. Una volta creati tutti i giocatori, le squadre inviano un segnale all'arbitro, dopodiché si metteranno in pausa, attendendo che l'arbitro gli invii il segnale di inizio partita.

3 Avvio della partita

Quando tutti i processi sono stati creati, il fato avvia un timer, la cui durata è impostata in base al parametro inserito nel file di configurazione (letto da quest'ultimo), e successivamente l'arbitro invierà un segnale per "svegliare" le squadre, le quali sveglieranno a loro volta i rispettivi giocatori. Per scegliere quale delle due squadre darà il calcio d'inizio, l'arbitro invoca la funzione `lanciaMonetina()` che restituisce un numero casuale tra 0 e 1.

In questo momento, i primi processi ad agire sono i giocatori, che tentano di riservarsi il `semaforoPalla`, che gli permetterà di fare la giocata (rappresentata dall'omonima funzione `giocata()`). Il fato si mette in attesa di un eventuale messaggio da parte di un giocatore, l'arbitro continua a ciclare in loop, in attesa che il Fato gli comunichi un eventuale aggiornamento del punteggio. La squadra, tramite una `wait()` si mette in attesa di un'eventuale morte di un suo processo figlio, causata da un infortunio.

4 Giocata

Una volta iniziata la partita, i giocatori, usando la funzione `tentativo()`, entrano in un loop infinito, cercando di riservarsi il `semaforoPalla`. Quando un giocatore prende la palla, invoca la funzione `giocata()`, che determinerà il tipo di azione, inoltre, scriverà la scelta nella coda di messaggi, dove il fato la riceverà ed elaborerà una risposta casuale, pilotata dalle percentuali inserite nel file `config.txt`.

Grazie alla funzione `sleep(1)`, lo stesso giocatore sarà obbligato a non concorrere immediatamente per il prossimo tentativo, in modo tale da far giocare anche gli altri giocatori.

Giocata può generare un numero casuale tra 1 e 3 (tramite l'assegnamento `scelta = sceltaGiocata()`), i quali corrispondono ai seguenti eventi:

1. **Infortunio**
   Se il giocatore si infortuna, rilascia una risorsa del semaforo squadra ed esce dal campo. La squadra del giocatore, si ritrova quindi con una risorsa disponibile e crea un nuovo giocatore, il cui pid viene inserito nella matrice dei giocatori. Dopodiché il nuovo giocatore riserva un'ulteriore risorsa

(occupando l'ultimo posto disponibile in squadra) e tenta, mediante un loop infinito, di accedere al `semaforoPalla`.

2. **Tiro**
   Il giocatore tira, e se il tiro ha successo, il fato invia un segnale tra `SIGUSR1` e `SIGUSR2` all'arbitro, per distinguere la squadra che ha segnato ed aggiornare il relativo punteggio. Altrimenti, se il tiro fallisce, il giocatore perde il possesso palla.

3. **Dribbling**
   Se il giocatore effettua un dribbling, e quest'ultimo ha successo, non rilascia il `semaforoPalla`, ed effettua un'ulteriore giocata tramite la chiamata ricorsiva alla funzione `giocata()`. Altrimenti, perde nuovamente il possesso palla.

5 Terminazione della partita

Quando il timer impostato dal fato scade, viene inviato un segnale allo stesso fato, che a sua volta lo invia all'arbitro, il quale termina innanzitutto il suddetto Fato, e comunica alle Squadre di terminare i propri Giocatori, grazie alla funzione `terminazioneGiocatori()`. Le Squadre a loro volta terminano con una `exit()`, l'arbitro, le attende con una `wait()`, infine, dealloca tutte le IPC create e termina sé stesso.

6 Note aggiuntive

1. **La macro "mosseOgni"**
   Abbiamo deciso di impostare tramite la macro `mosseOgni` presente nel file `libraries.h`, la velocità della partita. Dopo vari Test, abbiamo notato che le velocità adeguate variano da 2 (velocità massima) a infinito (velocità minima), infatti una volta impostata correttamente `mosseOgni`, ogni giocata avverrà ogni intervallo di secondi specificato.

2. **Formattazione del testo nel Terminale**
   Le funzioni `printf()` che compaiono nel terminale sono state formattate in modo da consentire una visualizzazione più chiara possibile di quello che accade durante la partita. Abbiamo perciò pensato sia creativamente e sia scopo di debug durante il testing, di utilizzare dei `Tab` e una gestione dei colori a video, per evidenziare le diverse risposte alle azioni.

3. **Formattazione del testo nel Log**
   abbiamo infine, concepito il `Log` come una sorta di registro, in maniera tale che le azioni vengano riportate di seguito, senza formattazioni visive particolari. Unica eccezione: l'aggiornamento del punteggio, nuovamente a scopo di chiarezza visiva.