Elaborato Sistemi Operativi - Forza 4

Davide Cerullo, Edoardo Bazzotti

Sommario

Funzionamento del programma

1.1 Server

Per una corretta esecuzione del programma F4Server, quest'ultimo dovrà essere eseguito con alcuni parametri:

```
./F4Server <righe> <colonne> <Segno1> <Segno2>
```

Nel caso in cui uno dei **parametri** inseriti sia errato verrà ritornato un messaggio con una guida per l'esecuzione del programma.

All'avvio il Server genera 2 **chiavi**, una per la creazione della memoria condivisa (shmid), l'altra per la creazione dei semafori (semid); una volta creata la **memoria condivisa** verrà inizializzata una tabella con le dimensioni inserite come parametri, inoltre dopo la tabella sarà presente 1 byte di spazio extra, il quale verrà utilizzato come flag per la condivisione di informazioni tra Client e Server.

Per quanto riguarda i **semafori** abbiamo deciso di crearne 3:

- mutex: semaforo binario inizializzato a 0, permette al Server di gestire l'arbitraggio del gioco, in quanto questo semaforo viene messo ad 1 ogni qualvolta un Client effettua una mossa.
- giocatore1, giocatore2: semafori binari entrambi inizializzati a 0, servono per bloccare il Client dopo che ha fatto una mossa e vengono sbloccati dal Server al loro turno.

Per il passaggio di: shmid, semid, numero e simbolo giocatore, numero righe/colonne, abbiamo utilizzato una **FIFO** pipeline. Il file comune che viene creato nella posizione /tmp/share ci consente di mettere in comunicazione Client e Server senza che questi abbiano relazioni precedenti; alla connessione della FIFO da parte del Client quest'ultimo legge tutte le informazioni (sopra riportate) e le utilizza per collegare la memoria condivisa ed i semafori, così facendo il Server è in grado di visualizzare il PID del Client guardando la struttura shmid_ds (fornita tramite shmctl).

I **segnali** inviati dal Server sono:

- SIGUSR1: segnale inviato al Client per comunicare la fine della partita in caso di vittoria, sconfitta o pareggio.
- SIGUSR2: segnale inviato al Client per comunicare la chiusura del Server.
- SIGTERM: segnali inviato al figlio e/o al bot per terminarli.

Dopo aver inviato uno dei sopracitati segnali, il Server si occuperà di rimuovere: la memoria condivisa, il file condiviso con la FIFO ed i semafori creati, prima di terminare. Oltre a questi segnali gestiamo anche SIGALARM, il quale viene catturato dal Server dopo un **Timeout** di tempo che grazie alla System Call alarm comincerà a decorre dal momento del passaggio di turno al giocatore, che dovrà fare una mossa entro il tempo prestabilito, pena la perdita della partita.

La funzione **checkWin** serve per il compito di arbitraggio del Server ed in particolare si occupa di controllare se la partita non si è conclusa, controllando se ci sono sequenze di 4 gettoni uguali in riga, colonna o diagonale, controlla inoltre se non ci siano più posti disponibili nella matrice.

L'ultima implementazione delle funzionalità aggiuntive consiste nella modalità SinglePlayer, nella quale il Client (previa comunicazione tramite parametro "*") giocherà contro un **BOT** generato da parte di un figlio del Server eseguendo il comando:

```
execl("./F4ClientBot", "./F4ClientBot", NULL);
```

Il quale eseguirà una versione del Client modificata in modo da generare mosse casuali.

1.2 Client

Per una corretta esecuzione del programma F4Client, quest'ultimo richiede l'esecuzione con alcuni parametri:

```
Multiplayer --> ./F4Client <nomeUtente>
Singleplayer --> ./F4Client <nomeUtente> "*"
```

Nel caso in cui uno dei **parametri** inseriti sia errato verrà ritornato un messaggio d'errore con la relativa guida per un esecuzione corretta del programma.

Per l'inizializzazione di shmid e semid, abbiamo usato una **FIFO** pipeline, sulla quale il Server scriverà questi valori ed il Client andrà a leggerli per poi successivamente connettersi alla shared memory ed ai semafori creati.

Una volta connesso ai **semafori** il Client si bloccherà in attesa che il Server sia pronto per arbitrare la partita. Arrivato il turno del Client, quest'ultimo verrà sbloccato e chiederà all'utente la colonna nella quale inserire il gettone ed attraverso la funzione insertPlay farà la sua mossa, successivamente sbloccherà il Server (semop) che procederà al controllo della partita.

Per una corretta gestione del gioco abbiamo implementato il controllo di diversi tipi di **segnali** come:

- SIGINT: Ctrl-C premuto dal Client
- SIGUSR1: Il gioco è finito perché si è arrivati ad una delle possibili situazioni di Win/Lose/Spare.
- SIGUSR2: Il Server è terminato.
- SIGUP: Il Client è terminato per via della chiusura del terminale.

Alla ricezione di uno dei segnali il Client terminerà tramite la funzione closeAll, la quale provvederà a disconnettere tutti i semafori e la memoria condivisa.

Screenshot

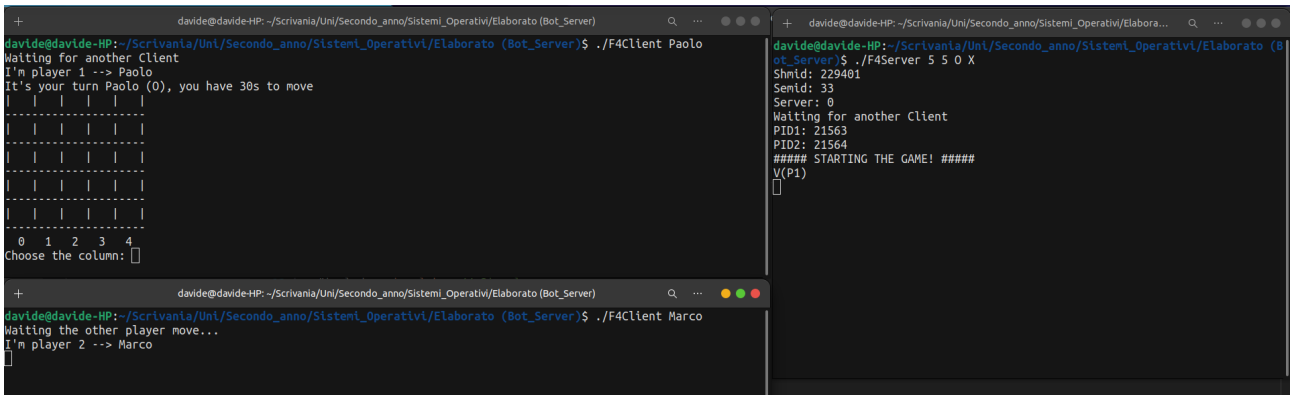

Figura 2.1: Avvio del programma

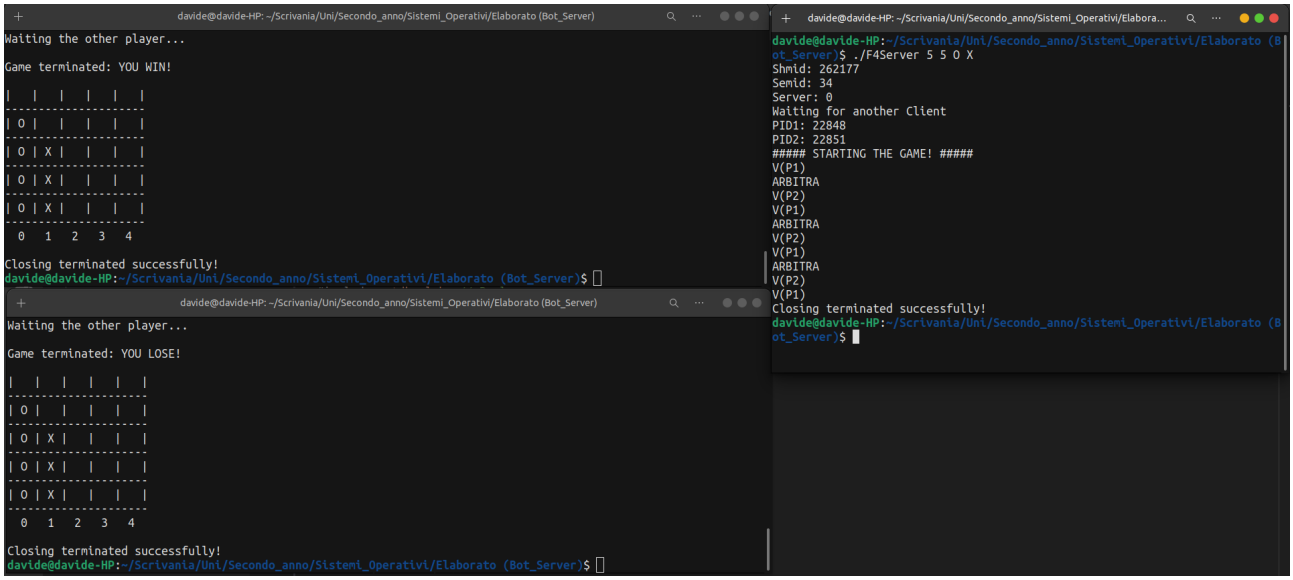

Figura 2.2: Terminazione del programma

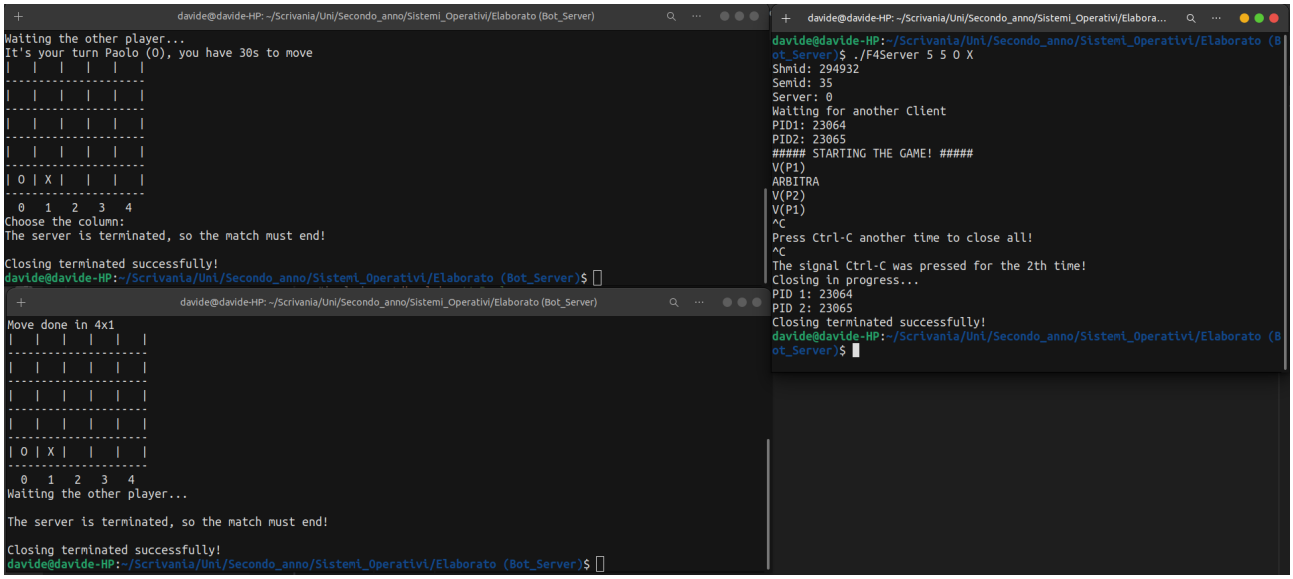

Figura 2.3: Chiusura forzata del Server