Università degli Studi di Cagliari

Sistemi Operativi

Modulo di Laboratorio 13E

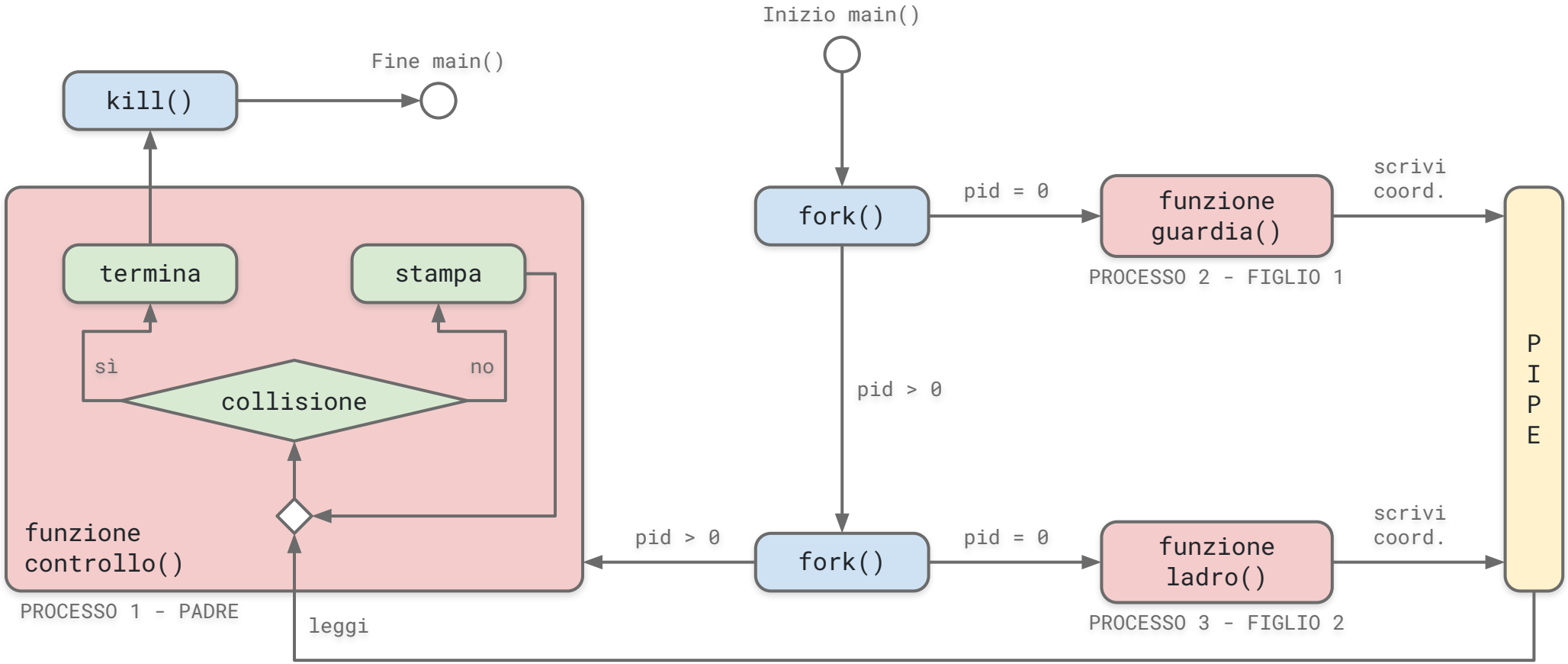
Esercizio 01: regole di gioco

- Scrivere in C con l'utilizzo delle ncurses un **programma multiprocesso** che implementi il gioco "**guardia e ladro**"
- Il gioco si compone di **due oggetti sullo schermo**:
  - La **guardia** (carattere **#**), controllata dall'**utente** tramite le **frecce direzionali**, inizia la partita dall'**angolo inferiore destro** dello schermo
  - Il **ladro** (carattere **$**), controllato dalla **logica di gioco** e che si muove in **maniera casuale**, inizia la partita dall'**angolo superiore sinistro** dello schermo
- Il gioco **termina quando la guardia** "**cattura**" **il ladro**
  - Ovvero quando il carattere # controllato dall'utente si sovrappone al carattere $

Esercizio 01: dettagli implementativi

- Il programma richiesto deve far uso di **3 processi distinti**:
  - Un primo processo **figlio acquisisce l'input utente e gestisce la posizione della guardia**, comunicandola al padre ogni volta che la aggiorna
  - Un secondo processo **figlio gestisce la posizione del ladro**, **generando uno spostamento casuale** e comunicando la posizione aggiornata al padre
  - Il processo padre **riceve le comunicazioni** sulle nuove posizioni degli oggetti di gioco, **stampa gli oggetti** stessi, **verifica le collisioni** e **controlla la terminazione**
- Definire delle **funzioni dedicate per la logica del padre e dei processi figlio**
- Utilizzare le **pipe per le comunicazioni** tra processi padre/figlio
- Gli eventi di gioco devono essere adeguatamente **temporizzati**

Esercizio 01: diagramma di flusso



Esercizio 02: regole di gioco

- Scrivere in C con l'utilizzo delle ncurses un **programma multiprocesso** che implementi il gioco "**guardia, ladro e avvocato**"
- Il gioco si compone di **tre oggetti sullo schermo**:
  - La **guardia** (carattere **#**), controllata dall'**utente** tramite le **frecce direzionali**, inizia la partita dall'**angolo inferiore destro** dello schermo
  - Il **ladro** (carattere **$**), controllato dalla **logica di gioco** e che si muove in **maniera casuale**, inizia la partita dall'**angolo superiore sinistro** dello schermo
  - L'**avvocato** (carattere **A**), controllato dalla **logica di gioco** e che si muove anch'esso in **maniera casuale**, inizia la partita dall'**angolo superiore destro** dello schermo
- Il **gioco termina in due condizioni** distinte e dagli esiti diversi:
  - Con **vittoria** del giocatore se la **guardia #** "**cattura**" **il ladro $** prima dell'avvocato A
  - Con **sconfitta** del giocatore se l'**avvocato A** "**cattura**" **il ladro $** prima della guardia #

Esercizio 02: dettagli implementativi

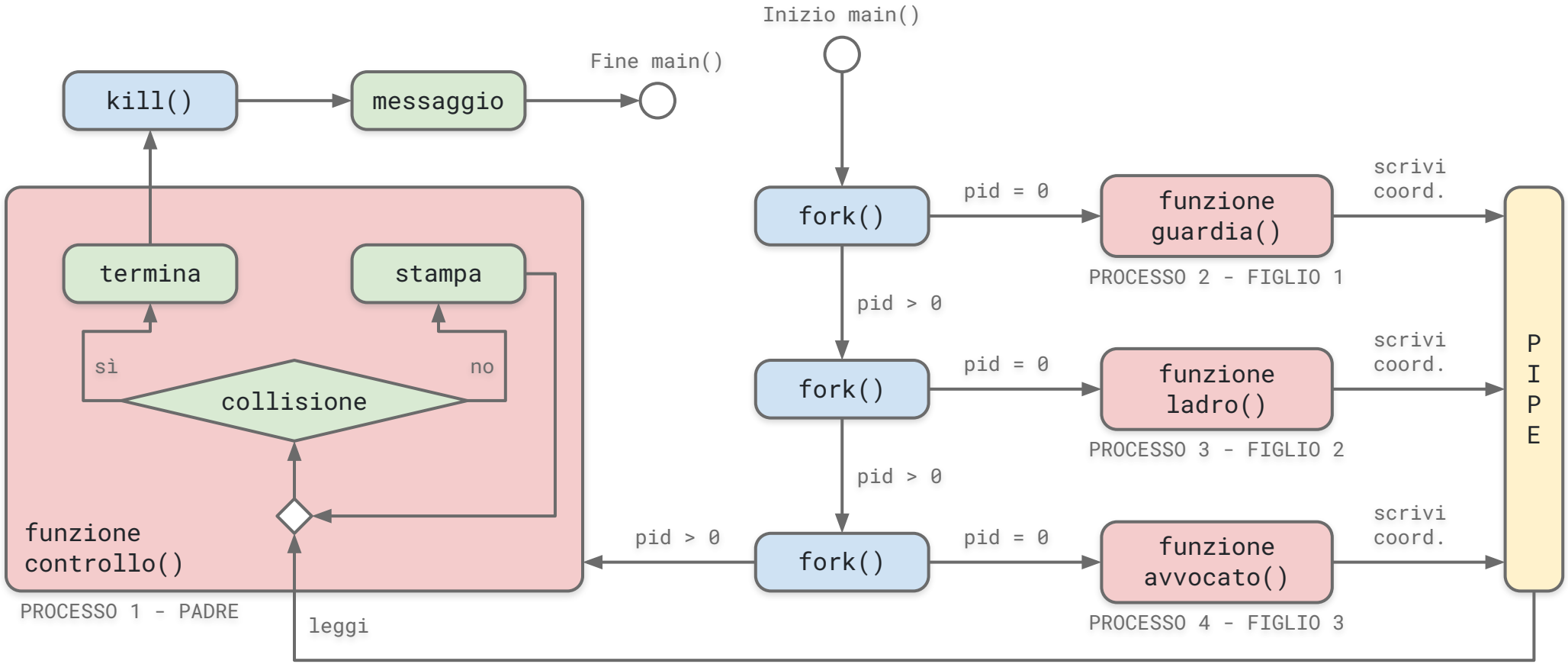
- Il programma richiesto deve far uso di **4 processi distinti**:
  - Un primo processo **figlio acquisisce l'input utente e gestisce la posizione della guardia**, comunicandola al padre ogni volta che la aggiorna
  - Un secondo processo **figlio gestisce la posizione del ladro**, **generando uno spostamento casuale** e comunicando la posizione aggiornata al padre
  - Un terzo processo **figlio gestisce la posizione dell'avvocato**, **generando uno spostamento casuale** e comunicando la posizione aggiornata al padre
  - Il processo padre **riceve le comunicazioni** sulle nuove posizioni degli oggetti di gioco, **stampa gli oggetti** stessi, **verifica le collisioni** e **controlla la terminazione**
    - A fine gioco stampa un messaggio adeguato per il giocatore ("vittoria"/"sconfitta")
- Definire delle **funzioni dedicate per la logica del padre e dei processi figlio**
- Utilizzare le **pipe per le comunicazioni** tra processi padre/figlio
- Gli eventi di gioco devono essere adeguatamente **temporizzati**

Esercizio 02: pseudocodice

```
#include <stdio.h>
#include <curses.h>
#include <stdlib.h>
#include <unistd.h>

int main() {
    Inizializzo_pipe();
    Creo_processo_figlio(Guardia);
    Creo_processo_figlio(Ladro);
    Creo_processo_figlio(Avvocato);
    Funzione_Controllo();
    Termino_processi(); // Usare kill() e waitpid()
    Stampo_messaggio(TipoCollisione);
    Exit();
}

Funzione_Controllo() {
    while(!Collisione) {
        Leggo_coordinate_dalla_pipe();
        Rilevo_tipo_oggetto();
        Pulisco_posizione_precedente_oggetto();
        Stampo_nuova_posizione_oggetto();
        if(Collisione)
            TipoCollisione = Definisci_collisione();
    }
}
```

```
Guardia() {
    while(1) {
        Leggo_tasti_cursore();
        Aggiorno_coordinate_oggetto();
        Comunico_coordinate_con_pipe();
    }
}


Ladro() {
    while(1) {
        Genero_movimento_casuale();
        Aggiorno_coordinate_oggetto();
        Comunico_coordinate_con_pipe();
        Sleep(Temporizzazione);
    }
}

Avvocato() {
    while(1) {
        Genero_movimento_casuale();
        Aggiorno_coordinate_oggetto();
        Comunico_coordinate_con_pipe();
        Sleep(Temporizzazione);
    }
}
```

Esercizio 02: diagramma di flusso



Fine Modulo 13E