Elaborato System Call

Eva Viesi

VR414213

Università Degli Studi Di Verona,

16/06/2018

Indice:

1. Struttura Del Progetto

L'elaborato è suddiviso su 8 file: `padre.c`, `padre.h`, `figlio.c`, `figlio.h`, `defines.h`, `helper.c`, `helper.h`, `mytypes.h`.

I file sono strutturati secondo questo schema:

```
elaborato-system-call/
├── src/
│   ├── figlio.c
│   ├── padre.c
│   └── helper.c
├── include/
│   ├── figlio.h
│   ├── padre.h
│   ├── helper.h
│   ├── mytypes.h
│   └── defines.h
├── run.sh
└── elaborato
```

2. Documentazione Files .c

`padre.c`:

E' il processo padre che inizializza le risorse necessarie per la lettura del file, le code di messaggi per l'invio delle operazioni lette ai processi figli e la pipe per la ricezione dei risultati computati dai figli. Si occupa inoltre di inviare i segnali di terminazione ai processi figli e di liberare le risorse a fine esecuzione.

`figlio.c`:

E' il processo figlio che si prenota per ricevere il segnale di terminazione dal processo padre, si mette in ascolto sulla coda di messaggi per leggere le operazioni a lui destinate e invia i risultati tramite pipe al processo padre.

`helper.c`:

Il file contiene le funzioni ausiliarie per i file `padre.c` e `figlio.c`.

3. Documentazione Files .h

padre.h: Contiene le definizioni degli header del file `padre.c`, ovvero:

1. **`void sigalarm_handler(int signal):`**

   @descr : stampa a video "tempo scaduto" dopo 10s di attesa del figlio
   @param `[int signal]` : Corrisponde al segnale ricevuto

2. **`void wait_child_ended():`**

   @descr : attende che tutti i figli abbiano terminato il loro compito

3. **`void send_message_to_childs(int n_op_per_figlio[MAX_CHILD]):`**

   @descr : invia le operazioni ai figli tramite la coda di messaggi
   @param `[int n_op_per_figlio[MAX_CHILD]]` : array che calcola quante operazioni sono destinate a ciascun figlio

4. **`void terminate():`**

   @descr : invia il segnale di terminazione "SIGTERM" a tutti i figli, attende che essi abbiano terminato la loro esecuzione e libera le risorse utilizzate

5. **`void save_output(char* f_path):`**

   @descr : salva i risultati in un file di output
   @param `[char* f_path]` : nome del file di output

6. **`void read_file_and_fill_matrix(int data_readed, int fd, int *n_figli_da_generare, int n_op_per_figlio[]):`**

   @descr : legge il file di input e genera una matrice contenente i dati letti
   @param `[int data_readed]` : bytes letti dal buffer
   @param `[int fd]` : file descriptor
   @param `[int *n_figli_da_generare]` : quantià di figli da generare
   @param `[int n_op_per_figlio[]):]` : array contenente il numero di operazioni destinate a ciascun figlio

figlio.h: Contiene le definizioni degli header del file `figlio.c`, ovvero:

1. **`void figlio(int p[], int n_figlio, int msg_id_rcv):`**

   @descr : il processo figlio si prenota per ricevere il segnale di terminazione, legge dalla coda di messaggi le operazioni che deve svolgere, le salva, le computa e restituisce i risultati al processo padre attraverso la pipe.
   @param `[int p[]]` : pipe di comunicazione tra padre e figlio
   @param `[int n_figlio]` : identificatore del figlio i^esimo
   @param `[int msg_id_rcv]` : id della coda dei messaggi

2. **`void sigterm_handler(int sig):`**

   @descr: funzione invocata quando il processo figlio intercetta il segnale `SIGTERM`
   @param `[int sig]` : segnale ricevuto

helper.h: Contiene le definizioni degli header del file `helper.c`, ovvero:

1. **`int negative_integer(int value):`**

   @descr : sistema il problema dei numeri negativi nel calcolo della loro lunghezza
   @param `[int value]` : valore da controllare se è negativo

2. **`int intergerLength(int value):`**

   @descr : restituisce il numero di cifre di un intero.
   @param `[int value]` : numero intero

3. **`int char_to_operation(char value):`**

   @descr : prende come parametro il carattere corrispondente ad un'operazione e lo converte in un intero.
   @param `[char value]` : carattere da convertire

4. **`int operation_to_char(int value):`**

   @descr : converte il valore intero nel corrisponente carattere in char
   @param `[int value]` : intero da convertire

mytypes.h: contiene la struttura di messaggi utilizzata all'interno del codice

defines.h: contiene le costanti utilizzate all'interno del codice.