Esempi di Comunicazione Interprocesso (IPC) tramite FIFO

November 23, 2025

Contents

1 Esempio 1: Produttore e Consumatore Semplice

Uso di una FIFO (Named Pipe) per una comunicazione unidirezionale semplice tra un processo "Produttore" e un processo "Consumatore". Il Produttore crea la FIFO, scrive un numero fisso di messaggi e poi la elimina. Il Consumatore apre la FIFO, legge i messaggi e si chiude.

1.1 Codice del Produttore (`produttore.c`)

Il Produttore crea la FIFO e la apre in scrittura (`O_WRONLY`). Se non c'è un Consumatore aperto in lettura, l'operazione di `open` si blocca.

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <fcntl.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <unistd.h>
#include <errno.h>

//Definizione costanti per dimensione buffer, numero messaggi e percorso FIFO
#define MAX_SIZE 50
#define N_MES 10
#define PATH_FIFO "/tmp/dati_fifo"

int main(){
    int fd; //Inizializzo file descriptor per la FIFO
    char buffer[MAX_SIZE];

    //Creazione della FIFO con permessi 0666 (lettura e scrittura per tutti)
    if (mkfifo(PATH_FIFO, 0666) == -1){
            perror("Errore nella creazione della fifo");
            return 1;
            //Drastico, ritorna anche se esiste
    }

    //Apertura della FIFO in modalit  scrittura (O_WRONLY)
    //Modalit  bloccante: la chiamata si blocca finch un processo non apre la FIFO in lettura.
    printf("Produttore: In attesa dell'apertura del Consumatore...\n");
    fd = open(PATH_FIFO, O_WRONLY);

    if (fd == -1){
```

```
32          perror("Errore nell'apertura della FIFO in
                scrittura");
33          return 1;
34      }
35      printf("Produttore: Connesso. Inizio scrittura.");
36
37      //Invio dei messaggi
38      for (int i = 0; i < N_MES; i ++){
39          //snprintf permette di formattare il messaggio in
                una dimensione precisa, previene buffer
                overflow
40          snprintf(buffer, MAX_SIZE, "Messaggio %i del
                produttore.", i + 1);
41
42          //Scrive il messaggio nella FIFO
43          //Aggiungo + 1 perch  includo il terminatore
                nullo '\0'
44          if(write(fd, buffer, strlen(buffer) + 1) == -1){
45              perror("Errore scrittura sulla FIFO");
46              return 1;
47          }
48      }
49
50
51      close(fd);
52      printf("Produttore: Tutti i messaggi inviati. Chiudo
            la FIFO.\n");
53      //Elimina il file FIFO dal filesystem
54      unlink(PATH_FIFO) //Potremmo aggiungere un controllo
            sulla chiusura
55
56      return 0;
57  }
```

Listing 1: Codice C del Produttore (`produttore.c`)

1.2 Codice del Consumatore (`consumatore.c`)

Il Consumatore si aspetta che la FIFO sia già stata creata (dal Produttore). Apre la FIFO in lettura (`O_RDONLY`) e si blocca finché il Produttore non la apre in scrittura.

```
1  #include <stdio.h>
2  #include <stdlib.h>
3  #include <string.h>
4  #include <fcntl.h>
5  #include <sys/stat.h>
6  #include <sys/types.h>
7  #include <unistd.h>
```

```c
#include <errno.h>

//Come il produttore
#define MAX_SIZE 50
#define N_MES 10
#define PATH_FIFO "/tmp/dati_fifo"

int main(){
    int fd;
    char buffer[MAX_SIZE];
    ssize_t bytes_input; //Numero di byte letti
    int m_read = 0; //Contatore messaggi letti

    //Apertura della FIFO in modalit  lettura (O_RDONLY)
    //Questa chiamata si blocca finch  il Produttore non apre la FIFO in scrittura.
    printf("Consumatore: In attesa che il Produttore apra la FIFO...\n");
    fd = open(PATH_FIFO, O_RDONLY);

    if (fd == -1){
        perror("Errore nell'apertura della FIFO in lettura");
        return 1;
    }
    printf("Consumatore: Connesso. Inizio lettura.");


    while(m_read < N_MES){
        //Legge dalla FIFO e salva nel buffer, ritorna i bytes coinvolti
        bytes_input = read(fd, buffer, MAX_SIZE);

        if(bytes_input == -1){
            perror("Errore lettura dalla FIFO");
            return 1;
        }

        //Il Produttore invia strlen(buffer) + 1, quindi include il terminatore nullo.
        //In teoria il carattere nullo c'   gi  : buffer[bytes_input - 1] = '\0';
        m_read++;
        printf("Consumatore: Ricevuto [%d/%d]: %s\n", m_read, N_MES, buffer);
    }

    close(fd);
    printf("Consumatore: Lettura completata e terminata.\
```

```
        n");
    return 0;
}
```

Listing 2: Codice C del Consumatore (`consumatore.c`)

2 Esempio 2: Semplice Chat Server e Client

Chat in cui un processo Server riceve uno stream di messaggi inviati da un processo Client tramite una FIFO. (Client → Server).

2.1 Codice del Server (`Server.c`)

Il Server crea la FIFO e si mette in ascolto permanente.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>
#include <errno.h>

#define SIZE_BUFFER 50
#define PATH_INFO "/tmp/chat_server"

int main(){
    ssize_t bytes_read;
    int fd;
    char buffer[SIZE_BUFFER];
    int is_running = 1; // Flag di controllo per il ciclo

    //Crea la FIFO
    if(mkfifo(PATH_INFO, 0666) == -1){
            perror("Server: Errore nella creazione della
               FIFO");
            return 1;
    }

    //Apre la FIFO in lettura (O_RDONLY). Si blocca in
       attesa del Client.
    printf("Server: In attesa di un Client sulla FIFO '%s
       '...\n", PATH_INFO);
    fd = open(PATH_INFO, O_RDONLY);

    if(fd == -1){
        perror("Server: Errore nell'apertura della FIFO")
           ;
        return 1;
    }
    printf("Server: Connesso. In attesa di messaggi.\n");

    //Lettura continua, controllata dal flag is_running
    while(is_running){
```

```c
        //Legge fino a SIZE_BUFFER - 1 byte (lasciando
           spazio per il terminatore nullo)
        bytes_read = read(fd, buffer, SIZE_BUFFER - 1);

        if(bytes_read == -1){
            perror("Server: Errore lettura");
            is_running = 0; // Uscita dal ciclo
        }
        else if(bytes_read == 0){
            //Il Client ha chiuso l'accesso in scrittura.
            printf("Server: Client disconnesso.\n");
            close(fd);
            is_running = 0; //Uscita dal ciclo
        }
        else{
            //Termina la stringa letta. Il Client invia
               il carattere '\n' da fgets.
            buffer[bytes_read] = '\0';
            printf("Client: %s", buffer);
        }
    }


    if (is_running == 0 && bytes_read != 0) {
        close(fd);
    }

    unlink(PATH_INFO);
    printf("Server: Terminazione.\n");
    return 0;
}
```

Listing 3: Codice C del Server (`Server.c`)

2.2 Codice del Client (`client.c`)

Il Client tenta di aprire la FIFO creata dal Server in modalità scrittura.

```c
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <fcntl.h>
#include <string.h>

#define SIZE_BUFFER 50
#define PATH_INFO "/tmp/chat_server"

```

```c
int main() {
    int fd;
    char buffer[SIZE_BUFFER];
    int is_running = 1; //Flag di controllo per il ciclo

    //Apertura della FIFO in scrittura (O_WRONLY)
    //Attende l'apertura in lettura da parte del Server.
    printf("Client: Tentativo di apertura FIFO '%s' in scrittura.\n", PATH_INFO);

    fd = open(PATH_INFO, O_WRONLY);
    if (fd == -1) {

        perror("Client: Errore nell'apertura della FIFO. Assicurati che il Server sia in esecuzione");
        return 1;
    }
    printf("Client: Connesso. Invia i messaggi (digita 'esci' per terminare).\n");

    //Ciclo di scrittura: legge da stdin e scrive sulla FIFO
    while (is_running) {
        //Legge l'input da tastiera (stdin), incluso il carattere '\n'
        if (fgets(buffer, SIZE_BUFFER, stdin) == NULL) {
            is_running = 0; //Imposta il flag per uscire
        }
        //Condizione di uscita
        else if (strncmp(buffer, "esci\n", 5) == 0) {
            is_running = 0; //Imposta il flag per uscire
        }
        //Scrive solo se non    stata impostata l'uscita
        else if (write(fd, buffer, strlen(buffer)) == -1) {
            perror("Client: Errore durante la scrittura sulla FIFO");
            is_running = 0; //Imposta il flag per uscire
        }
    }

    close(fd);
    printf("Client: Disconnesso e terminato.\n");

    return 0;
}
```

Listing 4: Codice C del Client (`client.c`)