FIFO in C su Linux: Teoria ed Esempi Pratici

Roberto Marino

13 maggio 2025

1 Introduzione alle FIFO

Le **FIFO** (First-In First-Out), conosciute anche come **named pipe**, sono un meccanismo di comunicazione inter-processo (IPC) che consente a processi diversi di scambiarsi dati attraverso un file speciale. A differenza delle *pipe anonime*, le FIFO sono identificate da un nome nel file system. La due grandi differenze tra pipe e fifo (entrambi unidirezionali) sono la persistenza e la non correlazione padre-figlio.

2 Caratteristiche delle FIFO

- Le FIFO sono file speciali che si comportano come canali di comunicazione unidirezionali.
- Possono essere utilizzate da processi che non hanno una relazione genitore-figlio.
- Il file FIFO rimane nel file system fino a quando non viene rimosso.
- Le operazioni su FIFO sono **bloccanti** per default.

3 Creazione di una FIFO

3.1 Da terminale

```
$ mkfifo mia_fifo
```

3.2 Da codice C

```c
#include <sys/types.h>
#include <sys/stat.h>
#include <stdio.h>

int main() {
    if (mkfifo("mia_fifo", 0666) == -1) {
        perror("Errore nella creazione della FIFO");
    }
    return 0;
}
```

Il secondo parametro (0666) rappresenta i permessi di lettura/scrittura per tutti gli utenti.

4 Utilizzo delle FIFO in C

4.1 Scrittura su FIFO

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>

int main() {
    int fd = open("mia_fifo", O_WRONLY); // La fifo deve essere gi
        creata
    if (fd == -1) {
        perror("Errore nell'apertura della FIFO");
        return 1;
    }
    write(fd, "Ciao dal writer!\n", 17);
    close(fd);
    return 0;
}
```

4.2 Lettura da FIFO

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>

int main() {
    char buffer[128];
    int fd = open("mia_fifo", O_RDONLY);
    if (fd == -1) {
        perror("Errore nell'apertura della FIFO");
        return 1;
    }
    int n = read(fd, buffer, sizeof(buffer));
    buffer[n] = '\0';
    printf("Letto: %s", buffer);
    close(fd);
    return 0;
}
```

4.3 Compilazione e test

```
$ gcc writer.c -o writer
$ gcc reader.c -o reader
```

Aprire due terminali:

- **Terminale 1**: `./reader`
- **Terminale 2**: `./writer`

5 Modalità non bloccante

È possibile aprire una FIFO in modalità **non bloccante** utilizzando il flag `O_NONBLOCK`:

```
int fd = open("mia_fifo", O_RDONLY | O_NONBLOCK);
```

In questo caso, se nessun processo scrive, la `read()` restituirà subito `0` o `-1` con errno impostato.

6 Pulizia

La FIFO rimane nel file system fino a quando non viene rimossa manualmente:

```
$ rm mia_fifo
```

7 Uso con `select()` o `poll()`

È possibile utilizzare `select()` o `poll()` per monitorare più FIFO contemporaneamente o per implementare timeout.

Esempio base con `select()`:

```
#include <stdio.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/select.h>

int main() {
    int fd = open("mia_fifo", O_RDONLY | O_NONBLOCK);
    fd_set rfds;
    FD_ZERO(&rfds);
    FD_SET(fd, &rfds);

    struct timeval tv = {5, 0}; // 5 secondi di timeout
    int retval = select(fd + 1, &rfds, NULL, NULL, &tv);

    if (retval == -1) {
        perror("select()");
    } else if (retval) {
        char buf[128];
        int n = read(fd, buf, sizeof(buf));
        buf[n] = '\0';
        printf("Dati disponibili: %s\n", buf);
    } else {
        printf("Timeout: nessun dato ricevuto.\n");
    }

    close(fd);
    return 0;
}
```

Per approfondimenti: `man 7 fifo`