Pipe

Leonardo Bizzoni

October 23, 2023

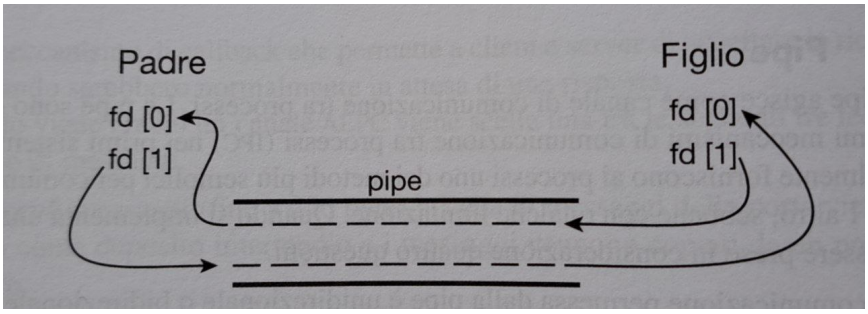
Una **pipe** agisce come un canale di comunicazione tra processi e sono state uno dei primi meccanismi di comunicazione tra processi nei primi sistemi UNIX. In sistemi UNIX-based una pipe è un tipo speciale di file e quindi viene ereditata da processi figli tramite chiamate *fork()*.

Non è possibile accedere ad una pipe da processi esterni.

1 Pipe convenzionali/anonime

Le pipe convenzionali permettono a 2+ processi di comunicare **unidirezionalmente** secondo una modalità detta **del produttore-consumatore**. Il produttore scrive ad un'estremità(**write-end**) del canale mentre il consumatore legge dall'altra(**read-end**).

Per instanziare una pipe convenzionale si utilizza la funzione *pipe(int fd[2])*. Essa crea una pipe alla quale si può accedere tramite i descrittori dei file(*fd[0]: read-end, fd[1]: write-end*). Inoltre non è possibile scrivere un altro valore nel write-end prima che il valore precedente sia letto.



1.1 Esempio codice

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

```
#include <stdlib.h>
#include <unistd.h>
#include <sys/wait.h>

int main(void) {
  int fd[2];

  if (pipe(fd) == -1) {
    printf("Errore nella creazione della pipe.\n");
    exit(EXIT_FAILURE);
  }

  if (fork() == 0) {
    close(fd[0]); // Il processo figlio non leggerà mai dalle pipe
  // tanto vale chiuderla subito
    int x;
    printf("[Child] Scrivi un numero: ");
    scanf("%d", &x);

    write(fd[1], &x, sizeof(int));
    close(fd[1]); // Quando ho scritto devo chiudere il write-end
  } else {
    close(fd[1]);

    printf("[Parent] waiting...\n");
    int y, cpid = wait(NULL);

    read(fd[0], &y, sizeof(int));
    close(fd[0]);

    printf("[Parent] Ho ricevuto dall'anonymous-pipe: %d\n", y);
  }

  exit(EXIT_SUCCESS);
}
```

2 Named Pipes

La versione **epica** delle pipe anonime:

- La comunicazione può essere bidirezionale.
- Relazione padre-figlio non necessaria.

In UNIX le named pipes sono dette **FIFO** ed una volta create si comportano come normali file all'interno del FS.

Nonostante la comunicazione bidirezionale sia permessa l'unica tipologia di trasmissione è quella *half-duplex*. Inoltre la comunicazione è limitata a processi(*2+*) sulla stessa macchina, se fosse necessaria la comunicazione tra più macchine vengono utilizzati i **socket**.

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

int main(void) {
  if (mkfifo("pipe", 0640) == -1) { // Puoi utilizzare anche "mkfifo" dalla CLI
    printf("Errore nel creare la pipe.\n");
    exit(EXIT_FAILURE);
  } // e skippare questa parte

  if (fork() == 0) {
    char buff[256] = "Hello, World!";
    int pipe_fd = open("pipe", O_WRONLY);

    write(pipe_fd, &buff, sizeof(buff));
    close(pipe_fd);
  } else {
    char buff[256];
    int pipe_fd = open("pipe", O_RDONLY);

    printf("[Parent] In attesa di dati dalla pipe...\n");
    read(pipe_fd, &buff, sizeof(buff));
    close(pipe_fd);

    unlink("pipe");

    printf("[Parent] Ho letto dalla pipe: %s\n", buff);
```

```
  }

  exit(EXIT_SUCCESS);
}
```