Esempio (manca)

giovedì 4 maggio 2023 16:46

ESEMPIO FIFO

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

int main (int argc, char *argv[]) {

  int fd;
  int ret;

  if (argc!= 2) {
    printf("Syntax: write_on_fifo <fifo_name>\n");
    exit(-1);
  }

  ret = mkfifo(argv[1], S_IRUSR|S_IWUSR);

  if (ret == -1) {
    perror("fifo creation error: \n");
    exit(EXIT_FAILURE);
  }

  fd = open(argv[1], O_WRONLY);

  if (fd == -1) {
    perror("fifo open error: \n");
    exit(EXIT_FAILURE);
  }

  printf("fd=%d\n",fd);
  fflush(stdout);
  close (1);
  dup(fd);
  execve("./writer", NULL, NULL);
  perror("Exec error: ");
  exit(EXIT_FAILURE);
}
```

Questo redirector non fa altro che, creare una FIFO e poi ridirezionare lo standard output sulla FIFO stessa. Quindi se noi andiamo a scrivere su std:out, in realtà la scrittura va sulla FIFO.
All'interno del main creiamo questa FIFO - la fifo deve avere un nome che viene indicata in questa applicazione in argv[1], verifichiamo il valore di ritorno che ci viene dato da mkfifo, nel caso terminiamo l'applicazione se c'è un problema nella creazione, e poi tramite la open andiamo ad aprire questa fifo che abbiamo creato e ci facciamo ritornare il codice del canale per andare ad operare su questa FIFO, attenzione, questa lettura è write_only, dopo chiudiamo lo stad:out e duplichiamo fd, quindi se poi lanciamo un'applicazione con una exec un'applicazione che utilizza la scrittura in std:out, in realtà i dati verranno inseriti in questa PIPE con nome.

REDIRECTOR.C

```c
#include <unistd.h>

#define MAXBUF 4096

int main () {
    char buffer[MAXBUF];
    int res_r,res_w,prev_w;

    res_r = read(0, buffer, MAXBUF);

    while(res_r) {
        prev_w = 0;
        res_w = 0;
        do {
            prev_w =prev_w +res_w;
            res_w = write(1, &buffer[prev_w],res_r -prev_w);
        } while (res_w+prev_w <res_r);

        res_r=read(0, buffer, MAXBUF);
    }

    return 0;

}
```

In questo writer, in un ciclo, non fa altro che leggere da standard input e rileggere poi nel ciclo e scrivere su std:out!
CONSIDERANDO ANCHE I RESIDUI.

WRITER.C

```
/FIFO> ./write_on_fifo pippo
```

PIPPO è il nome della FIFO.

```
OPERATIVI/CURRENT/SOFTWARE-EXAMPLES/PIPES-MESSAGES/UNIX/FIFO> /TEACHING/SISTEMIO
francesco@linux-mxb5:~/git-web-site/FrancescoQuaglia.github.io/TEACHING/SISTEMI-
OPERATIVI/CURRENT/SOFTWARE-EXAMPLES/PIPES-MESSAGES/UNIX/FIFO> less -f pippo
```

→

CONTENUTO DELLA FIFO

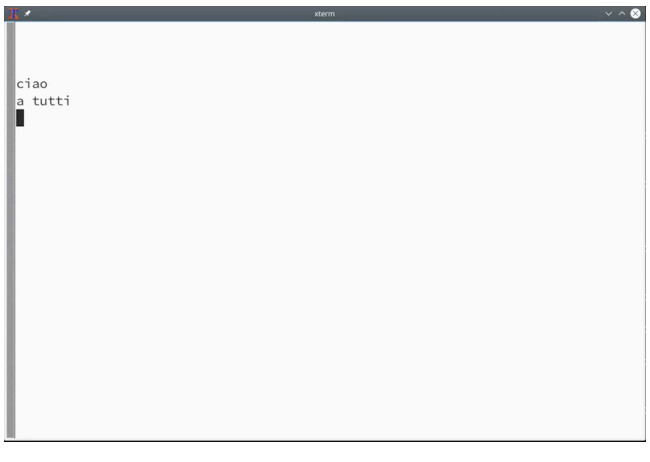
```
/FIFO> ./write_on_fifo pippo
fd=3
```

A questo punto la FIFO è stata aperta in lettura. Quindi siamo stati sbloccati, e abbiamo l'FD = 3. Rimanevamo bloccati nella riga 25. Prima la fifo non era ancora stata aperta, se non fossimo stati bloccati sulla OPEN ci sarebbe

stato fornito l'output di fd, ho provato ad aprire in O_WRONLY senza mettere il flag O_NONBLOCK questa fifo, E SE NON C'È nessuno interessato a leggere i dati da questa FIFO io vengo fermato sull'apertura.

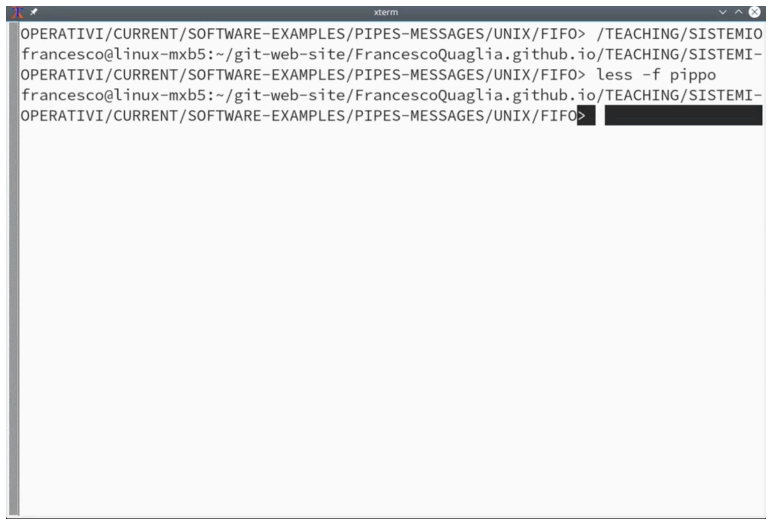
Ora invece siamo pronti, abbiamo lanciato anche la exec dell'altro programma e tutti i dati che scriviamo li ritroviamo nella fifo.

```
/FIFO> ./write_on_fifo pippo
fd=3
ciao
a tutti
```



Se noi la fifo la mandiamo in chiusura, l'applicazione in nero ha ancora la FIFO aperta perché abbiamo il codice 3 che poi avevamo duplicato sul codice di canale 1, se però adesso proviamo ad immettere qualsiasi informazione, quindi viene scritta sullo stad:out e quindi sulla FIFO dove c'eravamo ridirezionati, ovviamente otterremo il completamento della nostra applicazione.

```
/FIFO> ./write_on_fifo pippo
fd=3
ciao
a tutti
lkndjwdbkwjb
francesco@linux-mxb5:~/git-web-site/FrancescoQuaglia.github.io/TEACHING/SISTEMI-OPERATIVI/CURRENT/SOFTWARE-EXAMPLES/PIPES-MESSAGES/UNIX
/FIFO>
```



Inizio pipe sistemi windows 26_04_2021

Abbiamo introdotto il concetto che, una PIPE o una named pipe è un oggetto dove possiamo inserire delle informazioni usando una regola FIFO, e quindi estrarle con lo stesso ordine rispetto all'ordine inserimento.
La named pipe è un oggetto che ha un nome e quindi è identificabile all'interno del VFS, ci sarà un hard-link verso l'I-NODE di questo oggetto, questo ci permette di avere processi che non sono correlati tra di loro, quindi P e P' per esempio, che possono tranquillamente utilizzare la named pipe per eventualmente comunicare informazioni.
Altrimenti invece se vogliamo usare una PIPE tradizionale che è un oggetto senza nome, dobbiamo avere una relazione del tipo P, che è il creatore di questa PIPE, e poi eventualmente P genera P' o altri processi facendo ereditare a P' appunto i canali per arrivare su questa PIPE. Noi sappiamo che quando creiamo nuovi processi abbiamo che la nostra tabella dei File Descriptors viene ad essere ereditata dai processi che vengono ad essere generati.

Vediamo gli stessi concetti per quanto riguarda i sistemi operativi windows. P10 00:03:00