- Gestione dei processi -

- *Comunicazione tra processi: "la funzione* **pipe()"**

```
#include <unistd.h>

int pipe ( int filedes[2] );
```

- l'argomento *filedes* è costituito da due descrittori di file:
  - *filedes[0]* è aperto in lettura e rappresenta il lato in lettura della pipe ;
  - *filedes[1]* è aperto in scrittura e rappresenta il lato in scrittura della pipe;
  - l'output di *filedes[1]* è l'input per *filedes[0]*;
- restituisce 0 in caso di successo, -1 altrimenti.

Esercizio n° 1 – fork + pipe -

- *Scrivere un programma C in cui il processo di comunicazione tra padre e figlio è gestito da una **pipe**. Il programma dovrà quindi creare la pipe, generare il processo figlio **PF** che scriverà nella pipe una stringa inserita da tastiera. Il processo padre, visualizzerà sul video il contenuto della pipe.*

*Esecuzione: $ **./a.out***

Introduci una stringa da scrivere nella pipe e premi [Enter]

Output:

*Il padre ha letto nella pipe la stringa <**aaaaaaaaaa**>*

Soluzione Esercizio n° 1

```
#include <stdio.h> /* perror */
#include <errno.h> /* perror */
#include <unistd.h> /* write, lseek, close, exit */
#include <sys/types.h> /*open, lseek */
#include <sys/stat.h> /*open */
#include <fcntl.h> /*open*/
#define MAXLINE 256
int main (void)
{     int pid,fd[2]; /*contiene i due descrittori per accedere alla pipe in
                                lettura ed in scrittura*/
      int nread;
      char line[MAXLINE];
      if (pipe(fd)<0) { /*restituisce in fd due descrittori fd[0]: per la lettura
e                          fd[1]: per la scrittura*/
        perror("pipe");
        exit(1);}
      if((pid=fork())<0) /*Creazione di un nuovo processo */
      {perror("fork");
       exit(1);
      } /*CONTINUA NELLA SLIDE SUCCESSIVA*/
```

Soluzione Esercizio n° 1

```
else if (pid==0) /* pid=0 : Processo figlio */
      {
            printf("Introduci una stringa da scrivere nella pipe e premi
[Enter]\n");
            scanf("%s",line);
            close(fd[0]); /*chiude il descrittore per leggere dalla pipe*/
             /*Rimane bloccato nel write finchè la pipe è piena*/
            write(fd[1],line,strlen(line)); /*Scrive la stringa nella pipe*/
      }
      else  /* pid > 0 : Processo padre */
      {
            close(fd[1]); /*chiude il descrittore per scrivere nella pipe*/
            /*Rimane bloccato nel read finchè la pipe è vuota*/
            nread = read(fd[0],line,MAXLINE); /*Legge la stringa nella
pipe*/
            line [nread]='\0'; /*terminatore di null*/
            printf("Il padre ha letto nella pipe la stringa %s\n",line);
      }
exit(0);
}
```

Esercizio n° 2 – fork + pipe -

- *Scrivere un programma C che prenda come parametro un file **inputFile** (controllare che il numero di argomenti passati sia corretto). Il programma dovrà aprire tale file, creare un processo figlio, **PF** che comunica con il processo padre mediante una **pipe**. Il processo figlio, leggerà i primi 10 caratteri di tale file, scrivendoli nella pipe, il padre, visualizzerà sul video i caratteri scritti dal figlio leggendoli dalla pipe*

*Esempio **inputFile***

***1111111111**2222222222333333333344444444445555555555666666666*

Output:

Il padre ha letto nella pipe la stringa 1111111111

Soluzione Esercizio n° 2

```
#include <stdio.h> /* perror */
#include <errno.h> /* perror */
#include <unistd.h> /* write, lseek, close, exit */
#include <sys/types.h> /*open, lseek */
#include <sys/stat.h> /*open */
#include <fcntl.h> /*open*/
#define READLENGTH 10
main(int argc, char **argv)
{     int infile, fd[2], pid, nread;
      char buf[10];
      if (argc != 2)
      { printf ("uso: nomeprogramma <inputFile> \n"); exit (1); }
      /* Apertura file  inputFile*/
      if ((infile=open(argv[1], O_RDONLY)) <0)
        { printf("Errore apertura inputFile %s\n:",argv[1]);
          exit(1); }
      if (pipe(fd)<0) {perror("pipe");
         exit(1);
      }/*CONTINUA NELLA SLIDE SUCCESSIVA*/
```

Soluzione Esercizio n° 2

```
 /*Esecuzione fork*/
if ((pid = fork()) == -1)
      {     perror("fork");
            exit(1);}
      else if (pid == 0) /*Processo Figlio*/
      {     close(fd[0]); /*chiude il descrittore di lettura della pipe*/
            /*legge i primi 10 caratteri dal file*/
            if ( (nread = read(infile, buf, READLENGTH)) <= 0) {
                  close(infile);
                  exit(1);
            }
            /*scrive i primi 10 nella pipe*/
            if ( (write(fd[1], buf, READLENGTH)) <= 0) {
                  close(fd[1]);
                  exit(1);
            }
            close(fd[1]);
            exit(0);
      } /*CONTINUA NELLA SLIDE SUCCESSIVA*/
```

Soluzione Esercizio n° 2

```
else {   /*Processo Padre*/
            close(fd[1]); /*chiude il descrittore  per la scrittura sulla pipe*/
            /*legge i primi 10 caratteri dalla pipe*/
            if ( (nread = read(fd[0], buf, READLENGTH)) <= 0) {
                  close(fd[0]);
                  exit(1);
            }
            /*scrive a video primi 10 caratteri*/
            if ( (write(STDOUT_FILENO, buf, READLENGTH)) <= 0) {
                  close(fd[0]);
                  exit(1);
            }
            close(fd[0]);
      }
      close(infile);
      exit(0);
}
```

Esercizio n° 3 – fork + 2 pipe –
(comunicazione bidirezionale)

- *Scrivere un programma C che realizzi una **comunicazione bidirezionale** tra il processo padre ed il processo figlio mediante la creazione di due pipe distinte "**pipe1**" , "**pipe2**" Il programma dovrà creare le due pipe ed il processo figlio. Il processo padre, scriverà sulla **pipe1 (pdf1)** un messaggio immesso da tastiera, il processo figlio leggerà tale messaggio dalla **pipe1 (pdf1)**, lo stamperà a video e scriverà la notifica di lettura sulla **pipe2 (pdf2)** che verrà poi letta dal padre e stampata a video. (Utilizzare la funzione **fgets** per leggere un messaggio con spazi inserito da tastiera)*

Esecuzione: ./a.out

Inserire il messaggio e premere [ENTER]

Output:

Il processo figlio ha ricevuto il messaggio: <xxxxxx>

Il processo padre ha ricevuto la notifica di lettura del messaggio <xxxxxx>

Soluzione Esercizio n° 3

```
#include <stdio.h> /* perror */
#include <errno.h> /* perror */
#include <unistd.h> /* write, lseek, close, exit */
#include <sys/types.h> /*open, lseek */
#include <sys/stat.h> /*open */
#include <fcntl.h> /*open*/
#define SIZE 128
int main(int argc, char **argv) {
  int pfd1[2], pfd2[2]; /*descrittori di pipe*/
  int nread, pid;
  char inBuf[SIZE], outBuf[SIZE];
  if (pipe(pfd1)!=0 || pipe(pfd2)!=0) { /*creazione delle due pipe*/
    perror("pipe() fallita)");
    exit(-1);
  }
  if ((pid=fork()) < 0 ) { /*creazione del processo figlio*/
    perror("fork() fallita");
    exit(-2);
}/*CONTINUA NELLA SLIDE SUCCESSIVA*/
```

Soluzione Esercizio n° 3

```
if (pid==0) { /* figlio */
    close(pfd1[1]); /*descrittore di lettura pipe1 */
    close(pfd2[0]); /*descittore di scrittura pipe2 */
     /* legge un messaggio dal padre */
    nread=read(pfd1[0], inBuf, SIZE); /*legge il messaggio da
pipe1*/
    inBuf[nread]='\0'; /*terminatore di null */
    close(pfd1[0]);
    printf("Il figlio ha ricevuto il messaggio: %s\n", inBuf);
   /* manda un messaggio indietro di notifica lettura su
pipe2*/
    write(pfd2[1],inBuf,strlen(inBuf));
    close(pfd2[1]);
  }/*CONTINUA NELLA SLIDE SUCCESSIVA*/
```

Soluzione Esercizio n° 3

```
else { /* padre */
close(pfd1[0]); /*descrittore di scrittura pipe1 */
close(pfd2[1]); /*descrittore di lettura pipe2 */

   printf("Inserire il Messaggio e premere [ENTER]");  // legge un
msg
   fgets(outBuf,SIZE,stdin);                              // da tastiera
   write(pfd1[1], outBuf, strlen(outBuf)); /*Scrittura su pipe1 il
mess*/
   close(pfd1[1]);
   /* leggi il messaggio di notifica lettura del figlio */
   nread = read(pfd2[0],inBuf,SIZE); /*Lettura su pipe2*/
   inBuf[nread]='\0'; /*terminatore di null */
   close(pfd2[0]);
   printf("Il padre ha ricevuto la notifica di lettura del messaggio:
                %s\n",inBuf);
  }
  exit(0);
}
```