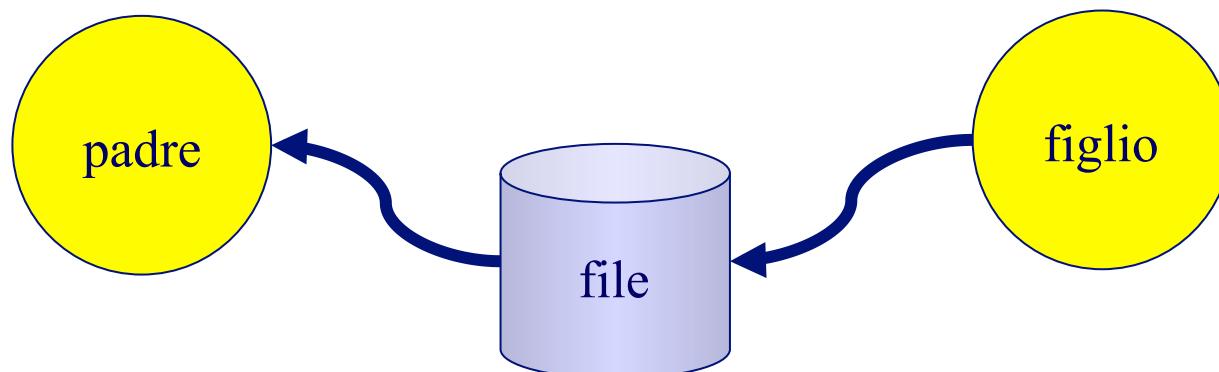


Esercizio sulla gestione di file in Unix

Esercizio

Si vuole realizzare un programma C che, utilizzando le system call di Unix, realizzi uno schema di comunicazione tra due processi (padre e figlio) mediante l'accesso a un file condiviso. In particolare, tenere conto delle seguenti specifiche:

- ❑ il figlio assume il ruolo di **mittente** dei messaggi;
- ❑ il padre e` il **destinatario**.
- ❑ Assumere un numero prefissato di messaggi (ad esempio, 10) a lunghezza fissa (ad es., 15 bytes).



Impostazione

Hp: uso del file come **canale** di comunicazione

- Generazione del processo figlio → uso di **fork**
- Necessita` di **sincronizzare** gli accessi al file:
 - **scrittura** di un messaggio (figlio): **write**
 - **lettura** del messaggio (padre): **read**
 - utilizzo di **segnali**
 - **send** e **receive** realizzate mediante **handler** di segnali.
- **creazione/cancellazione** di un file temporaneo: **creat, unlink**
- interazione stretta tra padre e figlio: condivisione di I/O pointer
 - apertura del file prima della **fork**

Esempio di soluzione

```
#include <signal.h>      /* uso i segnali*/
#include <fcntl.h>        /* uso i file*/
#include <string.h>        /* per manipolare i messaggi*/
#define max 10 /* numero di messaggi */

/* variabili globali (devono essere visibili
   da send e receive): */
int fd;                  /* file descriptor*/
int pid, ppid;           /* id figlio e padre */
int cont=0;               /* contatore dei messaggi */
/* prototipi handlers: */
void send(int signo);
void receive(int signo);
void fine(int signo);
```

```

main()
{ fd=creat("mailbox.tmp", 0777); /*creaz. File temp. */
  close(fd);
  fd=open("mailbox.tmp",O_RDWR); /*apertura file */
  signal(SIGUSR1, receive); /*SIGUSR1: ric. nuovo
messaggio */
  pid=fork(); /*figlio eredita fd: I/O pointer condiviso*/
  if (pid==0) /*codice figlio: mittente*/
  { ppid=getppid();
    signal(SIGUSR1, send); /* SIGUSR1: invio mess. */
    signal(SIGUSR2, fine); /* SIGUSR2: fine comunic. */
    send(-1); /* il figlio prende l'iniziativa:
primo messaggio */
    for(;;) pause(); /*ciclo di attesa*/
  } /* continua...*/
}

```

```
/* ... continua */

else if (pid>0) /* codice padre */
    for(;;) pause(); /*ciclo di attesa del
padre*/
}
} /* fine main */
```

Realizzazione di *send*

```
void send(int signo) /* interfaccia fissa: handler!*/
{   char buff[80]="Inizio!\n";
    int n;
    cont++;
    lseek(fd,0,0); /*reset dell'I/O pointer*/
    if (signo==-1) /* normale chiamata di procedura*/
        n=write(1, buff, strlen(buff)); /*stampa*/
    sprintf(buff, "Ciao babbo %2d\n",cont );
    n=write(fd, buff,15 ); /* scrittura nel file */
    kill(ppid, SIGUSR1); /* risveglio del padre*/
    return;
}
```

Realizzazione di *receive*

```
void receive(int signo)
{   char buff[80];
    int status;
    cont++;
    lseek(fd, 0,0); /* reset dell'I/O pointer */
    read(fd, buff, 15); /* lettura dal file */
    write(1, buff, 15); /* stampa su stdout il msg.*/
    if (cont!=max)
        kill(pid,SIGUSR1); /* ancora messaggi*/
    else
    {   close(fd); /*chiusura file*/
        kill(pid, SIGUSR2); /* ultimo messaggio*/
        wait(&status);
        printf("stato figlio:%d\n", status>>8);
        exit(0);
    }   return;
}
```

Gestore della terminazione

```
void fine(int signo) /* terminazione del figlio */
{
    char buff[25]= "FINE\n";
    close(fd); /* chiusura file*/
    write(1,buff , strlen(buff)); /* stampa */
    unlink("mailbox.tmp"); /* cancellazione file
*/
    exit(0);
}
```

Esecuzione

```
[aciampolini@ccib48 esercizi]$ gcc -o esefile file.c
```

```
[aciampolini@ccib48 esercizi]$ esefile
```

Inizio!

Ciao babbo 1

Ciao babbo 2

Ciao babbo 3

Ciao babbo 4

Ciao babbo 5

Ciao babbo 6

Ciao babbo 7

Ciao babbo 8

Ciao babbo 9

Ciao babbo 10

FINE

stato figlio:0

```
[aciampolini@ccib48 esercizi]$
```

Osservazioni sulla soluzione

- File come ***mailbox***
- modello di comunicazione **a *rendez-vous***: un messaggio alla volta con sincronizzazione stretta (ed esplicita) tra mittente e destinatario.

Spunti per modifiche ed estensioni:

- E` necessaria la condivisione dell'I/O pointer?
- Se i messaggi avessero lunghezza variabile ?
- Se il numero dei messaggi non fosse prefissato ?
- Come fare per una sincronizzazione diversa?
- Come fare per aumentare la capacita` del canale (possibilita` di gestire piu` messaggi all'interno del file) ?