Lucrarea de laborator Nr. 06

Procese. Mecanisme de comunicare între procese: pipe cu nume (fișier FIFO), redirectare I/O, mecanisme pipe cu nume (fișier FIFO)

Aplicații demonstrative:

1. Conectarea pe prompter prin PIPE cu Nume (fișier FIFO).
2. Conectarea a două procese printr-un PIPE cu Nume (fișier FIFO) cu redirectarea fișierelor standard de intrare respectiv ieșire. Procesele sunt în relație de părinte-copil

Canale de comunicaţie. (Pipes)

Una dintre modalitățile de comunicare între *procese* (*proces* = instanța unui program) în Unix este cea prin intermediul *canalelor de comunicaţie* (*pipes*, în limba engleză). Practic este vorba despre o "conductă" (un bufer/fișier FIFO) prin care pe la un capat se scriu mesajele, iar pe la celălalt capat se citesc - deci este vorba despre o structură de tip coadă, adică o lista FIFO (First-In,First-Out).

Aceste canale de comunicație sunt de două categorii:

- *pipe*-uri anonime (interne): aceste "conducte" sunt create în memoria internă a sistemului Unix sub forma unor bufere de comunicare (aceste bufere se mai numesc şi pipe-uri interne);
- *pipe*-uri cu nume (externe): aceste "conducte" sunt fişiere de un tip special, de tip *FIFO (**F**irst **I**nput **F**irst **O**utput)*, deci sunt păstrate în sistemul de fişiere (aceste fişiere *fifo* se mai numesc şi pipe-uri externe).

Canale de comunicaţie cu nume (pipe-uri cu nume).

Un *pipe cu nume* este un fișier special de tip FIFO în care un proces asincron poate să scrie respectiv altul, sau acelaşi proces, poate să citească. *Pipe*-ul reflectă natura asincronă a comunicării între procese. FIFO-urile (pipe-uri cu nume) pot fi create cu apelul sistem ***mkfifo()*** sau cu o comandă *mkfifo* din shell.

Comanda *mkfifo* – crează, pe prompter, un *pipe cu nume* (un fișier de comunicare de tip *FIFO)*

Sintaxă simplificată

```
mkfifo pipe_cu_nume
```

Întreaga documentație se găsește la

https://www.gnu.org/software/coreutils/mkfifo

Funcţia *mkfifo()* - crează *pipe cu nume* (un fișier de comunicare de tip *FIFO)*

Apelul sistem utilizat în programele C pentru a realiza o conectare inter-proces prin *pipe cu nume* are următorul prototip:

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

int mkfifo(const char *path, mode_t mode);
```

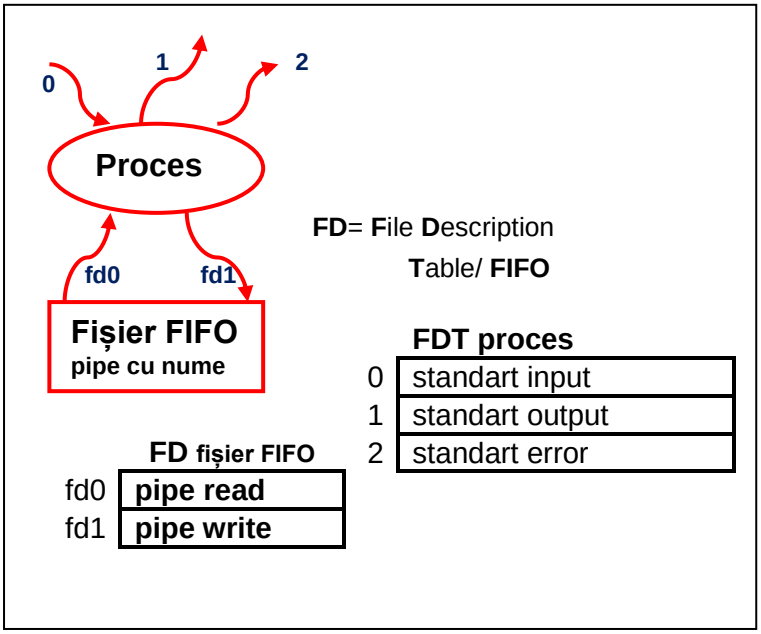
Apelul funcției *mkfifo()* asumă indicatorii *O_CREATE*|*O_EXCL* la crearea fişierului special FIFO (de tip p) ce reprezintă un *pipe cu nume* şi returnează eroare, *EEXIST*, dacă fişierul există deja.

O_CREAT Dacă *path* nu există atunci crează fișierul FIFO.
O_EXCL Dacă O_CREAT setat și există *path* atunci apelul *mkfifo()* eșuează cu eroarea EEXIST.

Următorul segment de cod crează *pipe-ul cu nume: pipe_cu_nume*, care poate fi citit în orice proces dar poate fi scris doar de proprietar (permisiuni de acces *rw-r--r--*).

```
#include <sys/stat.h>
#include <sys/types.h>
...
mode_t fifo_perms = S_IRUSR | S_IWUSR | S_IRGRP| S_IROTH; // permisiuni de acces
                                                          // rw- r-- r--
...
if (mkfifo("pipe_cu_nume", fifo_perms) == -1)
        perror("Nu pot crea pipe_cu_nume ");
...
```

În figura de mai jos apelul **`mkfifo()`** crează un fișier de comunicare de tip *FIFO (Pipe cu nume) la care* apelantul are acces prin descriptorii de fişier (file descriptors) **`fd0`** şi **`fd1`**.



Exemplu:

```
#define LMSG 16        //  - lungime mesaj
int fd0, fd1;          //  - descriptori fișier FIFO
//- deschidere fișier FIFO în RD (pipe cu nume)
mkfifo("pipe_cu_nume",S_IRUSR | S_IWUSR | S_IXUSR);
fd0=open("pipe_cu_nume",
        O_RDONLY|O_NONBLOCK)
//- deschidere fișier FIFO în WR (pipe cu nume)
fd1=open("pipe_cu_nume",
        O_WRONLY);

//  - scrie date în  fișier FIFO (pipe cu nume
write (fd1, m1, LMSG);
.....
//  - citeste date din  fișier FIFO (pipe cu nume)
read (fd0, buffer, LMSG);
```

Datele trimise (scrise) către **`fd1`** sunt accesate (citite) prin **`fd0`** pe baza primul-sosit-primul-servit (**`FIFO`**). Un *pipe cu nume* are un nume extern sau permanet, deci un proces poate accesa un *pipe cu nume* prin orice nume de descriptor definit în cadrul procesului. În cazul de față s-au utilizat *fd0* și *fd1* ca nume de descriptori de fișier FIFO. Din aceasta cauză un *pipe cu nume* poate fi utilizat de orice proces prin descriptorul de fișier rezultat în urma unui apel *open*() pe fișierul respectiv. Se observă că apelul *open()* pentru fișierul FIFO în citire se face în modul de blocare O_NONBLOCK, adică fără blocare. În acest caz, apelul *open()* pentru citire reușește, chiar dacă fișierul nu a fost deschis printr-un apel *open()* în scriere. Un fișier FIFO se deschide (*open()*) ca orice fișier normal și se utilizează apeluri *read()* sau *write()* pentru a citi/scrie în el. În consecință un apel *open()* pe un fișier FIFO poate conduce, în anumite condiții, la o blocare a procesului apelant pe *open().* Se aplică următoarele regului:

- Daca apelul *open()* s-a făcut atât în citire cât și în scriere (O_RDWR), atunci apelul *open()* nu va fi blocat în apelant.
- Daca apelul *open()* s-a făcut în citire (O_RDONLY), atunci apelul *open()* va fi blocat în apelant până când alt proces deschide fișierul FIFO pentru scriere, cu excepția cazului în care este specificat O_NONBLOCK, caz în care deschiderea se termină cu succes.

- Daca apelul *open()* s-a făcut în scriere (O_WRONLY), atunci apelul *open()* va fi blocat în apelant până când alt proces deschide fișierul FIFO pentru citire, cu excepția cazului în care este specificat O_NONBLOCK, caz în care deschiderea eșuează.
- Un proces care utilizează ambele capete ale conexiunii în scopul de a comunica cu sine, ca în exemplul de mai sus, trebuie să fie foarte atent la posibilitățile de blocare pe *open()* (O_NONBLOCK) pentru a evita impasuri.
  În exemplul analizat mai mai sus, pentru evitarea impasului (blocării), trebuie să avem următoarea secvență de apeluri *open():*

```
fd0=open("pipe_cu_nume", O_RDONLY|O_NONBLOCK);  // deschidere în citire fără blocare
fd1= open("pipe_cu_nume", O_WRONLY)   // dechidere în scriere cu blocare
                                      // până are loc o deschidere în citire
```

  Abordarea de mai jos ar produce un impas (blocare) pe *open()* în citire. Efect: nu se mai ajunge la open() în scriere.

```
fd0=open("pipe_cu_nume", O_RDONLY);   // deschidere în citire cu blocare
                                      // până are loc o deschidere în scriere
fd1= open("pipe_cu_nume", O_WRONLY)   // dechidere în scriere
```

  Abordarea de mai jos ar produce un impas (blocare) pe *open()* în scriere. Efect: nu se mai ajunge la open() în citire.

```
fd1= open("pipe_cu_nume", O_WRONLY)   // dechidere în scriere cu blocare
                                      // până are loc o deschidere în citire
fd0=open("pipe_cu_nume", O_RDONLY);   // deschidere în citire
```

Funcţia *dup2()* - crează o copie a unui descriptor de fișier utilizând un număr de descriptor furnizat de utilizator.

Apelul sistem `dup2()` utilizat în programele C realizează o redirectare și are următorul prototip:

```
#include<fcntl.h>

int dup2(oldfd, newfd);
```

unde

oldfd -este vechiul descriptor de fișier
newfd -este noul descriptor de fișier utilizat de dup2() pentru a crea copia

Mai jos este o implementare C în care newfd este descriptorul fișierului de ieșire standard (stdout) care are valoarea 1. Se realizează o redirectare a fișierului standard de iesire către fișierul fisier.txt

```
#include...
...
int main()
{
  int oldfd = open("fisier.txt",O_WRONLY | O_APPEND);
  dup2(oldfd, 1) ;                  //  newfd este descriptorul de fisier al lui stdout (= 1)
                                    //  oldfd este descriptorul de fisier pentru fisier.txt
// dup2(oldfd,STDOUT_FILENO);
  printf("scriu in fisier.txt \n");  // Toate instructiunile printf vor scrie in fisier.txt
  printf("scriu in fisier.txt \n");  // Toate instructiunile printf vor scrie in fisier.txt
...
  return 0;
}
```

1. Conectarea pe promter prin pipe cu nume

Exemplu de creare din shell prin comanda *mkfifo* a unui fişier special FIFO având numele: *pipe_cu_nume* precum și introducerea/extragerea de date din el:

$ *mkfifo pipe_cu_nume* ⇒crează un pipe cu nume (fișierul FIFO pipe_cu_nume)
$ *ls -l pipe_cu_nume* ⇒afișează informații extinse despre fișierul *pipe_cu_nume*
prw-r--r-- 1 virgiliu.streian domain users **0** Mar 08 01:10 **pipe_cu_nume**
⇒în informațiile extinse **p** indică faptul că este un fișier de tip FIFO (pipe), adică un pipe cu nume
⇒ atributul "*size*" este întotdeauna egal cu **0** dacă fișierul pipe este vid.

$ *rm pipe_cu_nume* ⇒sterge *pipe cu nume* (fișierul FIFO pipe_cu_nume)
$ *mkfifo -m 654 pipe_cu_nume* ⇒[re]crează fișier un pipe cu nume (fișierul FIFO *pipe_cu_nume*) cu drepturile de acces
rw- r-x r--
6 5 4
$ *ls -l pipe_cu_nume* ⇒afișează informații extinse despre fișierul *pipe_cu_nume*
prw-r-xr-- 1 virgiliu.streian domain users **0** Mar 20 08:05 *pipe_cu_nume*

$ echo -e "articol1\n" > *pipe_cu_nume* & ⇒Se introduc în background …
[1] 27802
$ echo -e "articol2\n" > *pipe_cu_nume* & ⇒ +3 articole
[2] 27809
$ echo -e "articol3\n" > *pipe_cu_nume* & ⇒ ++ în în fișierul FIFO *pipe_cu_nume*
[3] 27810

$ cat *pipe_cu_nume* ⇒Se extrag articole din fișierul FIFO *pipe_cu_nume*
⇒ (se remarcă ordinea inversă de extragere)
articol3

articol2

articol1

[1] Done echo -e "articol1\n" > *pipe_cu_nume*
[2]- Done echo -e "articol2\n" > *pipe_cu_nume*
[3]+ Done echo -e "articol3\n" > *pipe_cu_nume*

$ *ls -l pipe_cu_nume* ⇒afișează informații extinse despre fișierul *pipe_cu_nume*
⇒ se remarcă lipsa informațiilor – nr. octeți=**0**
prw-r-xr-- 1 virgiliu.streian domain users **0** Mar 20 08:15 *pipe_cu_nume*

$ cat *pipe_cu_nume* ⇒Se încearcă extragere de articole din fișierul FIFO *pipe_cu_nume*
⇒ (se remarcă apariția unui deadlock
Fiind un fișier pipe se încearcă citirea din el de date
dar datele nu au intrat în el, deci așteptare infinită)
^C ⇒Ieșire din deadlock prin ctrl/c

$ touch fisvid ⇒Se crează un fișier vid
$ ls -l fisvid ⇒ Afișare informații extinse despre fișiers
Se remarcă tip fișier= **–** (normal) și lipsa informațiilor – nr. octeți=**0**
-rw-rw-r-- 1 streian streian **0** mar 24 18:34 fisvid

$ cat fisvid ⇒Se afișează fișierul vid

$ ⇒Apare imediat prompterul (afișarea unui fișier normal vid nu crează un deadlock)

2. Conectarea a două procese printr-un PIPE cu Nume (fișier FIFO) cu redirectarea fișierelor standard de intrare respectiv ieșire. Procesele sunt în relație de părinte-copil

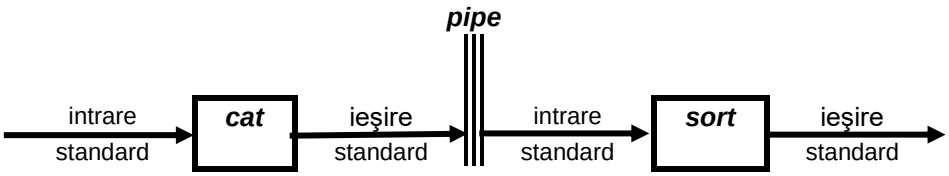
Redirectarea prin pipe cu nume a ieșirii standard a comenzii *cat /etc/passwd* către intrarea comenzii *sort –d*

- **Program pipen_.c**

Programul C (cu numele ***pipen_.c***) tratat în această aplicație implementează startarea comenzilor din *shell* `cat /etc/passwd|sort -d` n printr-un PIPE cu Nume (fișier FIFO) .
Iată mai jos ceea ce urmează a fi implementat:

```
$cat /etc/passwd | sort –d↵
      admin:x:65535:0:admin:/export/home/admin:/bin/bash
      adm:x:4:4:Admin:/var/adm:
      bin:x:2:2::/usr/bin:
      cl2-2011:x:65547:98:Chindris Luian:/export/home/cl2-
      2011:/bin/bash
      daemon:x:1:1::/:
      dladm:x:15:65:Datalink Admin:/:
      dm2-2011:x:65546:98:Dima Maria:/export/home/dm2-2011:/bin/bash
      . . . etc . . .
```

În exemplul de mai sus ieşirea standard a comenzii (*procesului*) ***cat*** (afişează fişierul */etc/passwd*) este conectată la intrarea comenzii (*procesului*) ***sort*** (afișează sortat alfabetic crescător- opţiune *–d* fișierul de intrare) printr-un bufer de comunicare ***pipe***. Efectul va fi că se afişează lista rezultată în urma comenzii **cat** sortată alfabetic.



Notă

Pentru ca exemplul să fie reluabil se va da o comandă ***rm*** de ștergere a tuturor fișierelor cu numele *pipen_* din directorul curent, fișiere care vor fi create în cadrul prezentei lucrări. Semnificația opțiunilor -fr este următoarea:

- *-f* efect: anularea mesajelor de eroare de *fișier negăsit* care pot să apară.
- *-r* efect: ștergerea directoarelor „not empty” (au conținut- nu sunt goale -).

```
$rm -fr pipen_.*↵          ⇒şterge fişierele pipen_.* aflate în directorul curent
$rm -f pipe_cu_nume↵       ⇒şterge fişier FIFO aflat în directorul curent

$which cat↵                ⇒afişează calea absolută a comenzii cat
/bin/cat                   ⇒ se va pune în programul pipen_.c în funcția execl()

$which sort↵               ⇒afişează calea absolută a comenzii sort
/usr/bin/sort              ⇒ se va pune în programul pipen_.c în funcția execl()
```

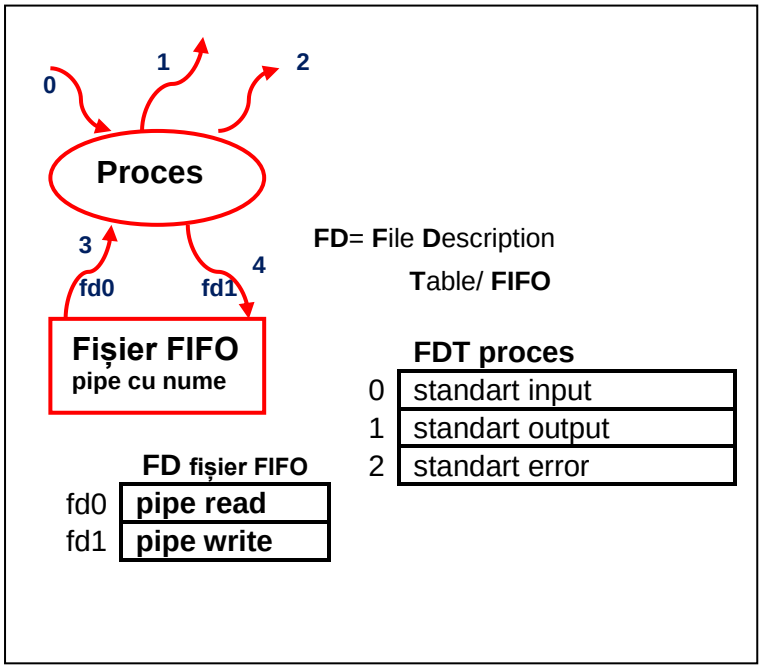
- **Program pipen_.c**

```
/* pipen_.c*/
#include <stdio.h>
#include <stdlib.h>                                    // pentru functia exit()
#include <unistd.h>
#include <fcntl.h>
#include <errno.h>
#include <sys/types.h>
#include<sys/stat.h>

int main(void)
{
        int fd0, fd1;
        pid_t childpid;
//      [1]
                /*
                -creaza un pipe cu nume cu rwx pentru user
                -se asuma indicatorii O_CREATE|O_EXCL
                - daca pipe-ul cu nume exita deja (EEXIST) nu se semnaleaza eroare
                */
                // permisiuni de acces:  rwx------
if ((mkfifo("pipe_cu_nume",S_IRUSR | S_IWUSR | S_IXUSR ) == -1) && (errno != EEXIST)) {
        fprintf(stderr, "Nu pot crea pipe-ul cu numele: %s\n", "pipe_cu_nume");
        perror("mkfifo esuat");
exit(1);
}
fd0= open("pipe_cu_nume", O_RDONLY|O_NONBLOCK);        // deschidere în citire fără blocare
fd1= open("pipe_cu_nume", O_WRONLY);                   // deschidere în scriere cu blocare
//      ----[1]
if (( childpid=fork())==0)                             //      [2]
{
                /*cod copil: comanda cat este startata din copil*/
                // sleep(10); // test pt. a forta ca c-da sort sa astepte ca lista sa fie disponibila in pipe
        fprintf(stderr,"COPIL:: startez comanda cat /etc/passwd \n");
//      [3.1]
        dup2(fd1,STDOUT_FILENO);            //  fd1 (pipe cu nume) si  stdout sunt acum echivalente
        /*merge si asa
                dup2(fd1, 1)
        deoarece STDOUT_FILENO = 1
        */

        close(fd0);
        close(fd1);
        execl("/bin/cat","se ignora","/etc/passwd",NULL);          //  [4] se execută c-da bin/cat /etc/passwd
        perror("execl: comanda cat esuata");
} else
{
                //cod parinte: comanda sort este startata din parinte*/
                /* Comanda sort sorteaza alphabetic sau numeric o lista.
                 -d  -- in ordine alfabetica
                */
        // sleep(10); // test pt. a forta ca c-da cat sa livreze lista in pipe
        fprintf(stderr,"PARINTE:: startez comanda sort -d\n");
//      [3.2]
        dup2(fd0, STDIN_FILENO);            //  fd0 (pipe cu nume) si  stdin sunt acum echivalente
        /*merge si asa
                dup2(fd0, 0)
        deoarece STDIN_FILENO = 0
        */
        close(fd0);
        close(fd1);
```

```
            execl("/usr/bin/sort", "se ignora","-d",NULL);        //  [5] se execută c-da bin/sort -d
            perror("execl: comanda sort esuata");
    }
                        /* secventa comuna parinte / copil  */
     exit(0);
    }
```

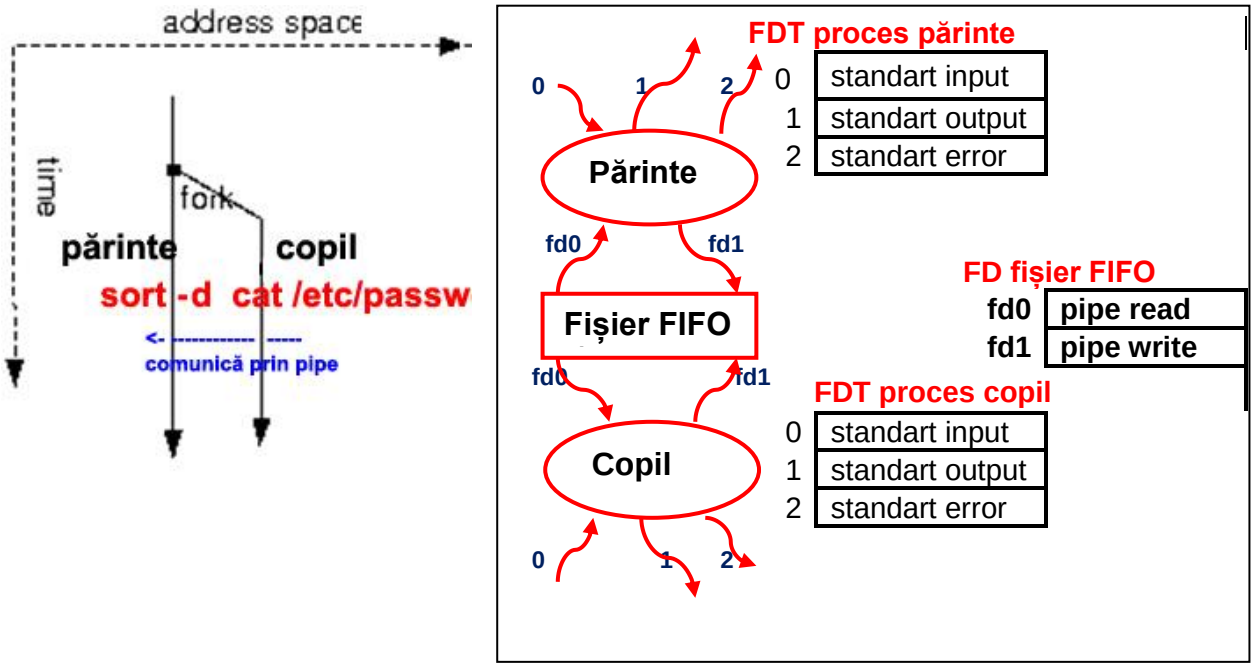
- **Explicații funcționare program**

[1] Apelul funcției *mkfifo()* crează un fișier FIFO (pipe cu nume) la care apelantul are acces prin descriptorii fd0 și fd1. Datele scrise în fd1 sunt citite din fd0 dupa principiul de baza first-in-first-out (FIFO).



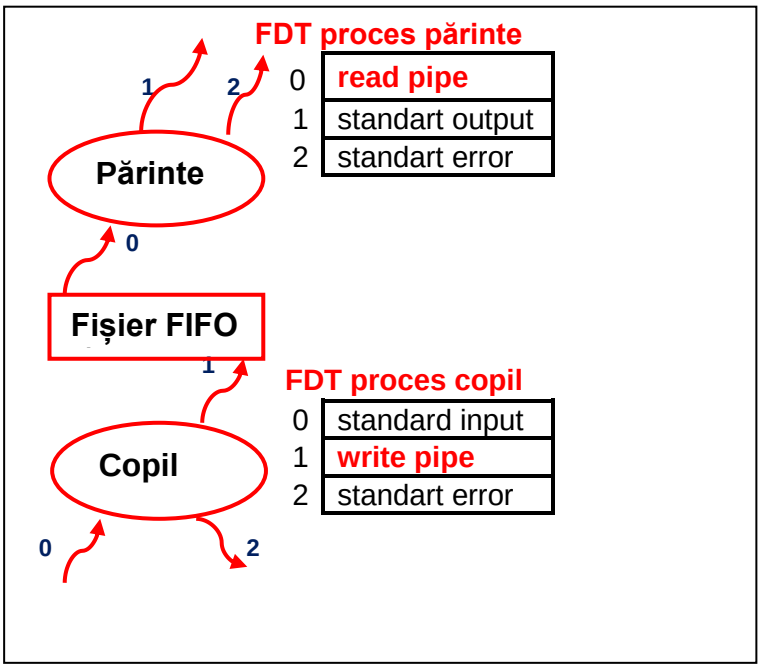
Un *pipe cu nume (fișier FIFO)* este un nume extern sau permanent, deci un process poate sa-l acceseze prin orice descriptori de fișier. Din această cauză un *pipe cu nume* poate fi utilizat de către orice proces, ne având restricția că poate fi utilizat numai de procesul care l-a creat și de descendenții acestuia.

[2] Apelul funcției *fork()* crează un *proces copil* conform figurii de mai jos, realizându-se apoi datorită moștenirii comunicarea *părinte copil* prin *pipe cu nume*:



[3.1/.2] în *copil(.1) / părinte(.2)*, realizează conectarea celor două procese printr-un *pipe cu nume* (fișier FIFO) cu redirectarea fişierelor standard de intrare respectiv ieşire către *pipe-ul cu nume*.

Starea FDT în momentul apelului `execl`



[4] în *copil*, se execută */bin/cat /etc/passwd*

[5] în *părinte*, se execută */usr/bin/sort -d*

- **Compilare linkeditare şi rulare program *pipe_.c***

```
$gcc -o pipen_ pipen_.c↵
$./pipen_↵
```

- **Mesaje afişate în urma rulării**

```
PARINTE:: startez comanda sort -d
COPIL:: startez comanda cat /etc/passwd
admin:x:65535:0:admin:/export/home/admin:/bin/bash
adm:x:4:4:Admin:/var/adm:
bin:x:2:2::/usr/bin:
cl2-2011:x:65547:98:Chindris Lucian:/export/home/cl2-2011:/bin/bash
daemon:x:1:1::/:
dladm:x:15:65:Datalink Admin:/:
dm2-2011:x:65546:98:Dima Maria:/export/home/dm2-2011:/bin/bash
. . . etc . . .
```