Université de Sousse **Ecole Supérieure des Sciences** **et des Technologies de Hammam Sousse** **Département Electronique et Informatique**	ESST Hammam Sousse	**Matière : Système d'exploitation 2** **Niveau : LI 1** **A.U. : 2021 - 2022** **Responsable : O. FAKHFAKH**

Correction - TD3 : Les Tubes de communication

Exercice 1 — Producteur – Consommateur

Ecrire un programme C comportant un processus producteur (lecture de l'entrée standard et écriture dans un tube) et un processus consommateur qui met en majuscule les minuscules (lecture du tube et écriture sur la sortie standard).

```
#include <ctype.h>
#include <stdio.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <stdlib.h>
#include <unistd.h>
#define BUFSZ 1024
/* Note : La fonction suivante n'est pas "propre".
   Il faut toujours tester la valeur de retour de read et write
*/
void minmaj(char *buf, int sz)
{  int i;
   for (i = 0; i < sz; i++)
      buf[i] = toupper(buf[i]);
}

void copy(int in, int out, void *buf, int sz)
{  while (write(out, buf, read(in, buf, sz)));
}

void copymm(int in, int out, void *buf, int sz)
{  int n;

   while ((n = read(in, buf, sz)))
   {  minmaj(buf, n);
      write(out, buf, n);
   }
}
int main(void)
{  int p[2];
   pid_t pid;
   static char buf[BUFSZ];

   if (pipe(p))
   {  perror("pipe"); exit(1);
   }
   if ((pid = fork()) == -1)
   {  perror("fork"); exit(1);
   }
```



```
   if (pid)
   {  close(p[0]);
      copy(STDIN_FILENO, p[1], buf, BUFSZ);
      close(p[1]);
      wait(NULL);
   }
   else
   {  close(p[1]);
      copymm(p[0], STDOUT_FILENO, buf, BUFSZ);
      close(p[0]);
   }
   return 0;
}
```

STDIN_FILENO :Standard input value, *stdin*. Its value is 0.

Exercice 2 : Les tubes nommés

Utiliser les tubes nommés pour assurer la communication entre deux processus.

Ecrire le programme C du processus Ecrivain qui lit des caractères sur l'entrée standard et envoie les lettres et les chiffres à processus Lecteur.

Ecrire le programme C du processus Lecteur qui compte les lettres et les chiffres et affiche les résultats à la fin.

Le nom de pipe est donné en argument

Processus écrivain

```
int main (int argc, char ** argv) {

int fd_write;

if ( mkfifo(argv[1], 0666,0) {

        perror ("mkfifo");

        exit (1);

}

if (( fd_write = open (argv[1], O_WRONLY)) == -1) {

      perror ("open");

      exit (1);

}

while (read (STDIN_FILENO, &k, 1) >0)

      if (isalnum(k)) write (fd_write ,&k, 1) ;

wait(0);

}
```

Processus lecteur

```
int main (int argc, char ** argv) {
int fd_read;
if (( fd_read = open (argv[1], O_RDONLY)) == -1) {
    perror ("open");
    exit (1);
}
while (read (fd_read], &k, 1) >0)
          if (isdigit (k)) chiffre++ ;
           else lettre++ ;
       printf ("%d chiffres recus\n", chiffre) ;
       printf ("%d lettres recues\n", lettre) ;
     exit (0) ;
}
```

Exercice 3 — Tube et fichier

Ecrire un programme C qui crée deux processus A et B communiquant par tube de communication. Le processus A ouvre un fichier donné en argument du programme et transmet le contenu de ce fichier au processus B via un tube de communication. Le processus B écrit le contenu du tube dans le deuxième fichier passé en argument.

Exercice 4 — Communication bidirectionnelle

Donner l'organisation d'une application de transmission bidirectionnelle d'informations entre un processus père et un processus fils via des tubes.

- Le père envoie 5 entiers au fils.
- Le fils affiche ces entiers envoyés et les renvoie multipliés par 2.
- Le père affiche ces doubles.

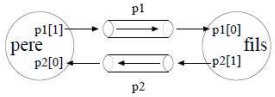
Écrire ensuite le programme C correspondant.

Solution

Dans ce problème, on crée deux tubes p1 et p2 pour faire communiquer les deux processus :

– le père a accès en écriture sur p1 et en lecture sur p2,

– le fils a accès en écriture sur p2 et en lecture sur p1.




```
#include <stdio.h>
#include <unistd.h>
#define NB_ENTIERS 5
void fils ( int p1[2] , int p2[2] )
{
   int i , nombre , nb_lus ;
   close (p1[1])
   close (p2[0]) ;
   nb_lus = read ( p1[0] , &nombre , sizeof (int) )
   while( nb_lus == sizeof (int) )
   {
     nombre = nombre * 2 ;
     printf ( " Les doubles sont :%d \n " , nombre ) ;
     write (p2[1] , &nombre , sizeof (float)) ;
     nb_lus = read ( p1[0] , &nombre, sizeof(int)) ;
   }
   close (p1[0])
   close (p2[1]) ;
   exit(0) ;
}
int main( )
{
   int i , nombre ,p1 [2] , p2 [2] ;
   pipe ( p1 ) ;
   pipe ( p2 ) ;
   if ( fork () == 0)
     fils ( p1 , p2 ) ;
   else
   {
     close (p1[0]) ;
     close (p2[1]) ;
     for ( i =0; i <NB_ENTIERS ; i ++)
     {
       printf ( " Entrez un entier \ n " ) ;
       scanf ( " %d " , &nombre ) ;
       write ( p1[1] , &nombre , sizeof(int)) ;
     }
     close(p1[1]) ;
     printf ( " Les doubles sont: \ n " ) ;
     for ( i =0; i <NB_ENTIERS ; i ++)
     {
       read ( p2[0] , &nombre , sizeof (int)) ;
       printf ( "%d ", nombre) ;
      }
       printf ( "\n" ) ;
       close(p2[0]) ;
   }
   return 0 ;
}
```