**EXERCICES DIRIGES 4**
**Communication par tubes anonymes**
**CORRECTION**

*Préambule : rappel du format des primitives permettant l'utilisation des tubes*

## Exercice 1

```
/*************************************/
/*   communication entre deux pipes,    */
/*************************************/
#include <stdio.h>

int pip1[2];        /*  descripteurs pipe 1  */
int pip2[2];        /*  descripteurs pipe 2  */

main()
{
int idfils;

/*  ouverture pipes */
if(pipe(pip1))
  {
  perror("pipe 1");
  exit(1);
  }

if(pipe(pip2))
  {
  perror("pipe 2");
  exit(2);
  }

/* creation processus */
if((idfils=fork())==-1)
  {
  perror("fork");
  exit();
  }

if(idfils)  pere();
  else   fils();

exit();

}


/*processus pere*/
pere()
{
char rep[8];

/* le premier pipe sert dans le sens pere vers fils on le ferme en lecture
*/
close(pip1[0]);

/*le second pipe sert dans le sens fils vers pere on le ferme en ecriture
*/
close(pip2[1]);
/* on envoie un message au fils par le pipe 1*/
```

```
if(write(pip1[1],"hello",5)!=5)
  {
  fprintf(stderr,"pere: erreur en ecriture\n");
  exit();
  }

/* on attend la reponse du fils par le pipe 2 */
if(read(pip2[0],rep,7)!=7)
  {
  fprintf(stderr,"fils: erreur lecture\n");
  exit();
  }

rep[7]=0;
printf("message du fils: %s\n",rep);
exit();
}


/* processus fils */
fils()
{
char mesg[6];

/*fermeture du pipe 1 en ecriture */
close(pip1[1]);

/* fermeture du pipe 2 en lecture */
close(pip2[0]);

/* attente d'un message du pere */
if(read(pip1[0],mesg,5)!=5)
  {
  fprintf(stderr,"fils: erreur lecture\n");
  exit();
  }

mesg[5]=0;
printf("la chaine recue par le filsest : %s\n",mesg);

/* envoi d'un message au pere */
if(write(pip2[1],"bonjour",7)!=7)
  {
  fprintf(stderr,"fils: erreur ecriture\n");
  exit();
  }
}
```

## Exercice 3

```
/*****************************************/
/* commande ps -e | wc -l                */
/*****************************************/

#include <stdio.h>
#include <unistd.h>


main()
{
int p[2];

/*  ouverture d'un pipe */
```

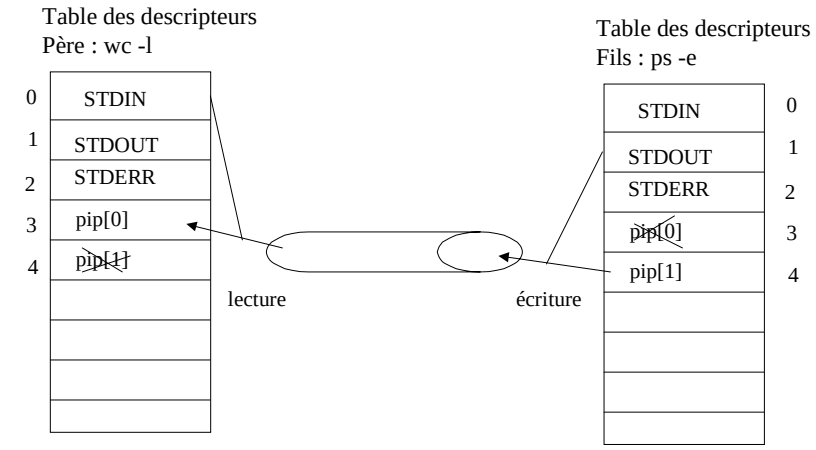
```
if(pipe(p))
  {
  perror("pipe");
  exit();
  }

switch (fork()) {
case -1 : /* erreur */
           perror (" pb fork ");
           exit();
case 0 : /* le processus fils execute la commande ps -e */
         /* la sortie standard du processus est redirigee sur le tube */
           close (STDOUT_FILENO);
          (void)dup(p[1]); /* p[1] sortie standard du processus */
           close (p[1]);
           close (p[0]); /* le processus ne lit pas dans le tube */
           execlp ("ps", "ps", "-e", NULL);
           perror("pb execlp(ps)");
           exit();
default : /* le processus pere execute la commande wc -l */
         /* l'entree standard du processus est redirigee sur le tube */
           close (STDIN_FILENO);
          (void)dup(p[0]); /* p[1] sortie standard du processus */
           close (p[0]);
           close (p[1]);
           execlp ("wc", "wc", "-l", NULL);
           perror("pb execlp(wc)");
           exit();
           }
}
```

**Fonctionnement de la primitive DUP**

La commande ps –e affiche les caractéristiques de tous les processus. la destination naturelle des informations est la sortie standard STDOUT. La commande wc –l compte les lignes de l'entrée qui lui est fournie, naturellement l'entrée standard STDIN.
Lors de la création d'un processus, 3 fichiers d'entrées sorties sont automatiquement crées : l'entrée standard STDIN (0), la sortie standard STDOUT(1) et la sortie d'erreur STDERR (2).
La primitive DUP (int dup(int desc)) associe le plus petit descripteur disponible du processus appelant à la même entrée dans la table des fichiers ouverts que le descripteur desc. Pour rediriger un descripteur standard sur un descripteur de tube, il faut :

- fermer le descripteur standard
- faire un dup avec comme paramètre le descripteur du tube concerné
- fermer le descripteur du tube.




wc -l lit sur STDIN
il faut rediriger l'entrée de wc -l sur pip[0]
<=> STDIN = pip[0]
close (STDIN)
dup (pip[0])
close (pip[0])

ps -e écrit sur STDOUT
il faut rediriger la sortie sur pip[1]
<=> STDOUT = pip[1]
close (STDOUT)
dup (pip[1])
close (pip[1])