TP N°: 3 : Correction

Exercice 1

Écrire un programme en C qui crée deux processus fils. Le processus père doit attendre la terminaison des deux processus fils.

→ Chaque processus fils doit afficher un message avec son identifiant de processus (PID) et envoyer un signal **SIGUSR1** au processus père.

→ Le processus père doit afficher le message de réception du signal pour chaque processus fils.

```c
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#include <unistd.h>
#include <signal.h>
void handleSignal(int signal) {
if(signal==SIGUSR1)
    printf("Signal reçu : %d \n", signal);
}

int main() {
    pid_t pid1, pid2;
    signal(SIGUSR1, handleSignal);
    while(1){}
    pid1 = fork();
    if (pid1 == -1) {
        printf("Erreur lors de la création du premier processus fils");
        exit(1);
    } else if (pid1 == 0) {
        // Code du premier processus fils
        printf("Je suis le premier processus fils (PID : %d)\n", getpid());
        kill(getppid(), SIGUSR1);
        exit(0);
    }
    pid2 = fork();
    if (pid2 == -1) {
        printf("Erreur lors de la création du deuxième processus fils");
        exit(1);
    } else if (pid2 == 0) {
        // Code du deuxième processus fils
        printf("Je suis le deuxième processus fils (PID : %d)\n", getpid());
        kill(getppid(), SIGUSR1);
        exit(0);
    }
    // Code du processus père
    waitpid(pid1,NULL,0);
    waitpid(pid2,NULL,0);

    return 0;
}
```

Exercice 2

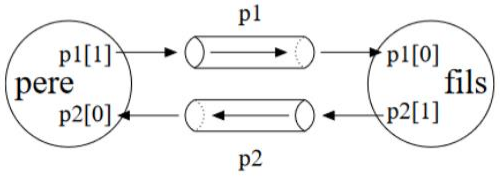
Soit une application de transmission bidirectionnelle d'informations entre un processus père et l'un de ses fils via des tubes.

Les spécifications de l'application sont les suivantes :

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

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.



Écrire ensuite le programme en langage C correspondant.

Solution 1 : en utilisant les tableaux.

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#define NB_ENTIERS 5

int main() {
    int i, nombre[5], nbr_dble[5],buffer[5];
    pid_t pid1;
    int p1[2], p2[2];
    pipe(p1);
    pipe(p2);
     pid1 = fork();
    if (pid1 == 0) {
        close(p1[1]);
        close(p2[0]);
       read(p1[0], buffer, sizeof(buffer));
        for (i = 0; i < NB_ENTIERS; i++) {
                buffer[i] = buffer[i] * 2;
            }
                    write(p2[1], buffer, sizeof(buffer));
        close(p1[0]);
        close(p2[1]);
        exit(0);
    } else {
        close(p1[0]);
        close(p2[1]);
        for (i = 0; i < NB_ENTIERS; i++) {
            printf("Entrez le (%d) entier :  \n",(i+1));
            scanf("%d", &nombre[i]);
        }
         write(p1[1], nombre, sizeof(nombre));

        close(p1[1]);

        read(p2[0], nbr_dble, sizeof(nbr_dble));
        printf("Les doubles sont :\n");
        for (i = 0; i < NB_ENTIERS; i++) {

                printf("%d ", nbr_dble[i]);
        }
        printf("\n");
        close(p2[0]);
    }

    return 0;
}
```

Solution 2 :

```c
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/types.h>
#include <sys/wait.h>
#define NB_ENTIERS 5

int main() {
    int i, nombre;
    pid_t pid1;
    int p1[2], p2[2]; /* Descripteurs du tube p1 et p2. */
    pipe(p1); /* Création du tube p1. */
    pipe(p2); /* Création du tube p2. */
     pid1 = fork();
    if (pid1 == 0) { /* Création du fils. */
        close(p1[1]); /* Fermeture du tube p1 en écriture pour le fils. */
        close(p2[0]); /* Fermeture du tube p2 en lecture pour le fils. */

        for (i = 0; i < NB_ENTIERS; i++) {
           read(p1[0], &nombre, sizeof(int)); /* Lecture nombre sur p1. */
                nombre = nombre * 2; /* Calcul du double. */
                write(p2[1], &nombre, sizeof(float)); /* Écriture double sur p2. */
        }
        close(p1[0]); /* Fermeture du tube p1 en lecture pour le fils. */
        close(p2[1]); /* Fermeture du tube p2 en écriture pour le fils. */
        exit(0);
    } else {
        close(p1[0]); /* Fermeture du tube p1 en lecture pour le père. */
        close(p2[1]); /* Fermeture du tube p2 en écriture pour le père. */
        for (i = 0; i < NB_ENTIERS; i++) {
            printf("Entrez un entier: \n "); /* Demande d'un entier. */
            scanf("%d", &nombre); /* Saisie d'un entier. */
            write(p1[1], &nombre, sizeof(int)); /* Écriture de l'entier sur p1. */
        }
        close(p1[1]); /* Fermeture du tube p1 en écriture pour le père. */


        for (i = 0; i < NB_ENTIERS; i++) {
           read(p2[0], &nombre, sizeof(int)); /* Lecture d'un entier sur p2. */
                printf("Le (%d) entier multipie est :", (i+1));
                printf("%d \n", nombre); /* Affichage de l'entier. */
        }

        printf("\n"); /* Important pour forcer l'affichage. */
        close(p2[0]); /* Fermeture du tube p2 en lecture pour le père. */
    }

    return 0;
}
```