L'UNIVERSITÉ INNOVANTE	*Module : Système d'exploitation et programmation système* ***TP N°4 : Les tubes*** *Intervenants : Pr M.BAKHOUYA, A. Kharbouch, H. El Khoukhi*	**Année universitaire :** 2019/2020

Rapport

Réaliser par : EL HANAFI Maha

Objectif :

Création et communication entre processus

Exercice1 :

1.1

```c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/wait.h>
int main(int argc, char **argv){
    int p[2],pid;
    char buffer[16];

    if(pipe(p)<0)
        exit(1);

    if((pid = fork())<0)
        exit(1);

    if(pid==0)
    {
        close(p[0]);
        strcpy(buffer,"bonjour\n");
        write(p[1],buffer,(strlen(buffer)+1));
    }
    else
    {
        close(p[1]);
        read(p[0],buffer,sizeof(buffer));
        printf("J'ai lu : %s\n",buffer);
        waitpid(pid,0,0);
    }
    return 0;
}
```

Cela fonctionne :

```
uir_student@ubuntu:~$ gedit ex1tp4.c
uir_student@ubuntu:~$ gcc -o ex1tp4  ex1tp4.c
uir_student@ubuntu:~$ ./ex1tp4
J'ai lu : bonjour
```

1.2

```c
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <sys/wait.h>
int main(int argc, char const *argv[]) {

  //Declaration de 2 pipe p1 et p2 et un pid du proc fils
  int p1[2],p2[2],pid;

  //Pour la reception ou l'envoi d'un message sur le pipe
  char buffer[23]; //23 est la taille du buffer, vous pouvez l'augmenter

  //La taille maximum d'un message qu'on peut lire
  const int max = 100;

  //Creation du premier pipe
  if(pipe(p1) < 0)
  {
    perror("Erreur de creation du premier pipe\n");
    exit(1);
  }
  //Creation du deuxieme pipe : type de retour ( -1 : error, sinon créé avec succes)
  if (pipe(p2) < 0) {
    perror("Erreur de creation du deuxieme pipe\n");
    exit(1);
  }

  //Creation du processus fils. le processus père et fils peuvent communiquer
  // a traver  p1 (père->fils) et p2 (fils -> père)
  pid = fork();

  //Erreur du fork
```

```c
    if (pid < 0) {
      perror("Erreur de creation du fils");
      exit(1);
    }

    if (pid > 0)
    {
          strcpy(buffer,"bonjour\n");
          write(p1[1],buffer,(strlen(buffer)+1));
          close(p1[1]);

  int taille;
          read(p2[0],&taille,sizeof(int));
          printf("la taille est : %d\n",taille);
          close(p2[0]);
          wait(NULL);
    }
    else
    {
           read(p1[0],buffer,(strlen(buffer)+1));
        printf("J'ai lu : %s\n",buffer);
        close(p1[0]);

      int taille = strlen(buffer);
      //char tailleM[5];
      //itoa(taille, tailleM, 10);
      //strcpy(buffer,tailleM);
          write(p2[1],&taille,(sizeof(int)));
          close(p2[1]);
    }

    return 0;
  }
```

Cela fonctionne :

```
uir_student@ubuntu:~/Desktop$ gcc -o ex12tp4  ex12tp4.c
uir_student@ubuntu:~/Desktop$ ./ex12tp4
J'ai lu : H
la taille est : 1
```

Exercice2 :

1.2

```c
#include <unistd.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/wait.h>
/*
    TP4 : Les tubes
    Intervenants : Profs. M.Bakhouya, A.Kharbouch, H.El Khoukhi
*/

/*
Création de deux processus qui échangent des entiers de la manière suivante (voire Chapitre 2) :
• Le père crée un tube
• Le père crée un fils
• Le fils lit une suite d'entiers au clavier terminée par Ctrl-D
• Il en fait la somme
• Il l'écrit dans le tube au père
• Le père lit la somme dans le tube puis l'affiche
*/
int main()
{
    pid_t pid;
    int n,somme=0, p[2], status;
    char chaine[80];

    /*Le père crée un tube*/
    if(pipe(p) < 0)
    {
        perror("Création du premier pipe a echouée \n");
        exit(1);
    }

    /*Le Père crée un fils*/
    pid = fork();
```

```
33      pid = fork();
34
35      /*Erreur de création de fils*/
36      if(pid < 0)
37      {
38          perror("Création du fils a échouée \n");
39          exit(1);
40      }
41
42      /*Processus Fils */
43      if(pid == 0)
44      {
45          /*############ COMPLETER ############*/
46          /*Le fils ferme la lecture sur le pipe pour pouvoir ecrire*/
47          //??????
48          close(p[0]);
49          printf("Processus Fils -- Entiers (fin ctrl + D) \n");
50          /*NB : EOF = Ctrl + D, End of File*/
51
52          /*############ COMPLETER ############*/
53          /*Le fils lit une suite d'entiers au clavier terminée par Ctrl-D*/
54          //???????
55
56
57          printf("Entrer une suite d'entiers: \n");
58          while ( scanf ("%d", &status) == EOF)
59  {
60              read(p[0], &status, sizeof(status));
61          }
62          /*############ COMPLETER ############*/
63          /*Le fils écrit la somme dans le tube au père*/
64          //????????
65          close(p[1]);
```

```
65          close(p[1]);
66            read(p[0], &chaine , 80);
67            printf ("%s", chaine);
68
69
70          /*############ COMPLETER ############*/
71          /*Le fils ferme l'ecriture sur le pipe pour que d'autres peuvent lire*/
72          //????????
73           close(p[1]);
74          /*Le fils termine*/
75          exit(0);
76      }
77      /*Processus Père*/
78      else
79      {
80           /*############ COMPLETER ############*/
81          /*Le père ferme l'ecriture sur le tube pour pouvoir lire*/
82          //???????
83          close(p[1]);
84         while (read(p[0], &status, sizeof(status)) !=0)
85             { n++;
86               somme +=status;
87             }
88          close (p[0]);
89
90
91           printf("la somme des %d entiers reçus : %d \n" ,n ,somme);
92
93           write(p[1], chaine, sizeof(chaine));
94           close(p[1]);
95          /*############ COMPLETER ############*/
96          /*Le père lit la somme dans le tube puis l'affiche*/
97          read (p[0],chaine, somme);
```

```
94            close(p[1]);
95           /*############ COMPLETER ############*/
96           /*Le père lit la somme dans le tube puis l'affiche*/
97           read (p[0],chaine, somme);
98           printf("%d" , somme);
99           /*Processus père en attente de l'ecture*/
100          printf("Processus Père bloqué en Lecture\n");
101          //???????????
102          printf("Processus père -- somme = %d \n",somme);
103
104           /*############ COMPLETER ############*/
105          /*Le père ferme la lecture sur le pipe*/
106          //???????
107          close(p[0]);
108          /*############ COMPLETER ############*/
109          /*Le père ne doit pas quiter avant son fils*/
110          //????????
111      }
112      return 0;
113  }
```

Cela fonctionne :

```
uir_student@ubuntu:~$ gedit test.c
uir_student@ubuntu:~$ ./test
Processus Fils -- Entiers (fin ctrl + D)
Entrer une suite d'entiers:
14
#◆8▯▯◆la somme des 62914623 entiers reçus : 0
0Processus Père bloqué en Lecture
Processus père -- somme = 0
uir_student@ubuntu:~$
```