Université Internationale de Rabat	**Module :** Système d'Exploitation et Programmation Système *1ère Année Cycle Ingénieur* Mohamed BAKHOUYA, Abdelhak KHARBOUCH **Projets**	**Année Universitaire** 2019/2020

Projet 4

Un réseau en anneau 'Token Ring'

Fait par : EL HANAFI Maha

Objectif :
La communication entre Processus et échange de données

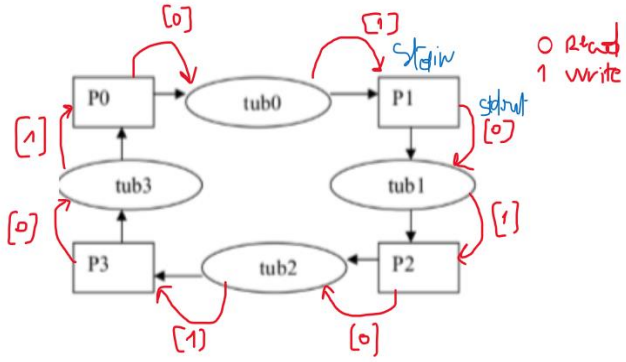
Communication entre processus par tubes :

Un tube est un moyen de transmission de données d'un processus à un autre. Comme il est implanté par fichier, il sera désigné par des descripteurs et manipulé par les primitives read() et write().
Un tube se comporte comme une file fifo (first in, first out) : les premières données entrées dans le tube seront les premières à être lues. Toute lecture est destructive : si un processus lit une donnée dans le tube, celle-ci n'y sera plus présente, même pour les autres processus

Enoncée :

Considérez N processus qui communiquent au moyen de tubes de communication non nommés. Chaque processus partage deux tubes (un avec le processus de droite et un autre avec le processus de gauche). Par exemple pour N=4, les processus communiquent selon le schéma suivant :

Schéma explicatif :



Programmes et données :

*Soit le programme C suivant : Projet4_Code-Source1.c

```c
#include <stdlib.h>
#include <unistd.h>
#include <stdio.h>
#include <string.h>
#include <sys/wait.h>
#include<sys/types.h>

/*Projet4.a/b*/
/*Implémenter le schema explicatif de communication des n processus créés.
Ecrire un programme qui met en place cette structure de processus et les connexions correspondantes.
Ce programme recevra un unique argument : le nombre n de processus de l'anneau*/
/*@mahahanafi*/
int main(int argc, char const *argv[])
{
pid_t pid;
int nbprocessus = 4;
int p[nbprocessus][2];
int i,j;
//*****
printf("On a %d Processes \n", nbprocessus);
//creation du pipe
     for(i=0; i<nbprocessus; i++)
{
       pipe(p[i]);
}
     for(i=0; i<nbprocessus; i++)
{    //creation de processus après la création du pipe
     pid = fork();

     if (pid<0)
     {
          perror("error fork");
          exit(2);
```

```
    }
    /*D'après le schema explicatif l'entrée standard (stdin) et la sortie standard (stdout) de chaque processus Pi sont
    redirigées vers les tubes appropriés.Par exemple, pour le processus P0, l'entrée et la sortie
    standards deviennent respectivement les tubes tub3 et tub0. */
    else if (pid ==0) //fils
        {
        printf("PID de processus fils: %d son père est : %d \n",  getpid(), getppid());
        dup2(p[i][1],1); //L'entrée est vers pipe du gauche
        dup2(p[(nbprocessus+i-1)%nbprocessus][0],0); //La sortie est vers pipe du droit
        for(j=0; j<nbprocessus; j++)
        {
            close(p[j][0]); //fermeture du pipe de lecture
            close(p[j][1]); //fermeture du pipe d'ecriture
        }
           exit(0);
        } exit(0);
}
  return 0;
}
```

Soit l'affichage provoqué par l'exécution de ce programme : ./ Projet4_Code-Source1

```
uir_student@ubuntu:~$ gcc -o projet4a projet4a.c
uir_student@ubuntu:~$ ./projet4a
On a 4 Processes
PID du processus fils: 4542  son père est : 3650
 PID du processus fils: 4543  son père est : 4542
```

*Soit le programme C suivant : Projet4_Code-Source2.c :

```
/*Projet4.*/
/*Ecrire un programme permettant aux processus de faire circuler entre eux un message.
En fait, le processus principal initialise la variable message à 10 puis il fait passer le message au processus suivant.
Ensuite, chaque processus fait passer le message au processus d'après de manière à ce que le message fasse le tour de l'anneau.
À chaque passage du message, au niveau du processus principal, celui-ci décrémente la valeur du message.
Lorsque le message atteint la valeur 0, les processus transmettent une dernière fois le message avant de s'arrêter.*/
/*@mahahanafi*/
#include<sys/types.h>
#include<stdio.h>
#include<stdlib.h>
#include<unistd.h>
#include <sys/wait.h>
#include <stdarg.h>


////debugging information to a file
void printLog(const char *fp, ...)
{

  FILE* pFile = fopen("file1.txt", "a");
  if(pFile != NULL)
  {
    va_list ap;
    va_start(ap, fp);
    vfprintf(pFile, fp, ap);
    va_end(ap);
    fclose(pFile);
  }
}
```

```c
int main(int argc, char const *argv[]) {

//initialisation des variables
  pid_t pid;
  int n,i,j,status;
  int message;



  int nbProcessus = 4;
  int p[nbProcessus][2]; //initialisation de la pipe

  //le programme commence avec processus
  printLog("On a %d Processus \n", nbProcessus);
  //creation des pipes
  for(j = 0; j < nbProcessus; j++)
    pipe(p[j]);

  printLog("les pipes sont crées \n");

  for ( i = 0; i < nbProcessus; i++)
  {

    printLog("Processus: %d \n",i);

        //creation des processus
        pid =fork();

        if(pid == -1)
        {
            printLog("Error Fork\n");
            exit(2);
```

```c
            exit(2);
        }
      //fils
        else if (pid == 0)
        {
      printLog("Processus: %d : Fils: %d , Père: %d \n",i,getpid(),getppid());

      int left = i;//pipe de gauche
      int right = (i+1)%nbProcessus;//pipe du droit

      printLog("Processus Fils %d :%d , pipe du gauche: %d et pipe du droit: %d \n",i,getpid(),left,right);

      dup2(p[left][0],0); //redirection de l'entrée vers la pipe de gauche
      dup2(p[right][1],1); //redirection de la sortie vers la pipe de droit

      //le processus 0 initialise la circulation du message /***
      if(i==0)
      {
        message = 10;
        printLog("Processus %d: %d commence avec le message: %d \n",i,getpid(),message);
        write(1,&message, sizeof(int)); //l'ecriture du message
        printLog("Processus %d :%d envoie le message: %d\n",i,getpid(),message);
      }

      while(1)
      {
        //blockage de l'ecriture
        printLog("Processus %d : %d en train d attendre le message\n",i,getpid());
        //la la lecture de message
        read(0, &message, sizeof(int));
        printLog("Processus %d : %d recoit ce message: %d\n",i,getpid(),message);
```

```c
        read(0, &message, sizeof(int));
        printLog("Processus %d : %d recoit ce message: %d\n",i,getpid(),message);


        if(message <= 0)
        {
          printLog("Processus %d : %d termine avec ce message; %d\n",i,getpid(),message);
          write(1, &message, sizeof(int));
          printLog("Processus %d : %d termine et envoie ce dernier message: %d\n",i,getpid(),message);

            exit(0);
        }

        //incrementation si le message <=0
        message--;


        write(1, &message, sizeof(int));
        printLog("Processus %d: %d en voie ce message: %d\n",i,getpid(),message);
      }
    }

  }

  printLog("Processus pere %d en train d attendre le fils a termine\n",i,getpid());
//terminaison des processus
  for ( i = 0; i < n; i++)
    wait(&status);

  printLog("Processus pere %d termine ce programme \n",i,getpid());

  return 0;
}
```

Soit l'affichage provoqué par l'exécution de ce programme : ./ Projet4_Code-Source2

```
uir_student@ubuntu:~/Desktop/projetmaha$ gedit projet11.c
uir_student@ubuntu:~/Desktop/projetmaha$ gcc -o projet11  projet11.c
uir_student@ubuntu:~/Desktop/projetmaha$ ./projet11
uir_student@ubuntu:~/Desktop/projetmaha$ ls
file11.txt  file.txt    projet11    projet11.c~  projetc.c   tests
file1.txt   myfile.txt  projet11.c  projetc      projetc.c~
uir_student@ubuntu:~/Desktop/projetmaha$ gedit file1.txt
```

*L'exemple d'affichage de file1.txt

```
On a 4 Processus
les pipes sont crees
Processus: 0
Processus: 1
Processus: 2
Processus: 3
Processus: 0 : Fils: 3944 , Pere: 3943
Processus Fils 0 :3944 , pipe du gauche: 0 et pipe du droit: 1
Processus 0: 3944 commence avec le message: 10
Processus 0 :3944 envoie le message: 10
Processus 0 : 3944 en train d attendre le message
Processus: 1 : Fils: 3945 , Pere: 3943
Processus pere 4 en train d attendre le fils a termine
Processus Fils 1 :3945 , pipe du gauche: 1 et pipe du droit: 2
Processus pere 0 termine ce programe
Processus 1 : 3945 en train d attendre le message
Processus 1 : 3945 recoit ce message: 10
Processus 1: 3945 en voie ce message: 9
Processus: 2 : Fils: 3946 , Pere: 3943
Processus 1 : 3945 en train d attendre le message
Processus Fils 2 :3946 , pipe du gauche: 2 et pipe du droit: 3
```

```
44 Processus 0 : 3944 recoit ce message: 3
45 Processus 0: 3944 en voie ce message: 2
46 Processus 1 : 3945 recoit ce message: 2
47 Processus 0 : 3944 en train d attendre le message
48 Processus 1: 3945 en voie ce message: 1
49 Processus 1 : 3945 en train d attendre le message
50 Processus 2 : 3946 recoit ce message: 1
51 Processus 2: 3946 en voie ce message: 0
52 Processus 2 : 3946 en train d attendre le message
53 Processus 3 : 3947 recoit ce message: 0
54 Processus 3 : 3947 termine avec ce message; 0
55 Processus 3 : 3947 termine et envoie ce dernier message: 0
56 Processus 0 : 3944 recoit ce message: 0
57 Processus 0 : 3944 termine avec ce message; 0
58 Processus 0 : 3944 termine et envoie ce dernier message: 0
59 Processus 1 : 3945 recoit ce message: 0
60 Processus 1 : 3945 termine avec ce message; 0
```

Sources

http://cours.polymtl.ca/inf2610/ExercicesPDFs/PipesCorrige.pdf
http://www.iro.umontreal.ca/~dift3820/.old/demos/pipe.pdf
https://zestedesavoir.com/tutoriels/755/le-langage-c-1/notions-avancees/les-fonctions-a-nombre-variable-darguments/
http://30minparjour.la-bnbox.fr/blog/2010/05/15/creer-plusieurs-processus-avec-fork/
https://mtodorovic.developpez.com/linux/programmation-avancee/?page=page_3